ADOBE® DIRECTOR® 11
Dictionnaire de script

Guide de prise en main du logiciel Adobe® Director® 11 pour Windows®

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, Etats-Unis.

Table des matières

Chapitre 9 : Constantes

Chapitre 10 : Evénements et messages

Chapitre 11 : Mots-clés

Chapitre 12 : Méthodes

Chapitre 13 : Opérateurs

Chapitre 14 : Propriétés

Chapitre 15 : Moteur physique

Chapitre 1 : Introduction

Cette référence fournit des informations conceptuelles et pratiques sur la programmation dans Adobe® Director® 11 ainsi que des descriptions et des exemples concernant les API de programmation que vous utilisez pour rédiger des scripts.

Les API de programmation représentent les outils par lesquels vous accédez aux fonctionnalités de Director via un script pour ajouter une dimension interactive à une animation. En utilisant ces API, vous pouvez non seulement créer des fonctionnalités interactives identiques à celles fournies par les comportements prédéfinis qui sont livrés avec Director, mais aussi en créer d'autres plus puissantes et variées.

Les comportements prédéfinis vous permettent d'ajouter des fonctionnalités interactives de base à une animation, comme par exemple déplacer la tête de lecture sur un numéro d'image ou un repère ou faire un zoom avant lorsque l'utilisateur clique sur une image-objet. Ils permettent également d'utiliser des fonctionnalités non interactives telles que l'animation d'images-objets, le chargement de médias et la navigation dans une image. Les API de programmation vous permettent de développer et de personnaliser ces types de fonctionnalités.

Public visé

Cette référence vous concerne si vous souhaitez effectuer une ou plusieurs des opérations suivantes :

- développer les fonctionnalités existantes des comportements prédéfinis à l'aide de scripts ;
- ajouter des fonctionnalités à une animation en utilisant des scripts au lieu des comportements prédéfinis ;
- ajouter des fonctionnalités plus puissantes, variées et personnalisées à une animation que celles proposées par les comportements prédéfinis.

Cette référence a pour objectif de fournir toutes les informations, de base et avancées, dont vous avez besoin pour ajouter une dimension interactive à vos animations en utilisant des scripts. Ainsi, il n'est pas nécessaire d'avoir une grande expérience dans la programmation pour rédiger des scripts efficaces dans Director.

Quel que soit votre niveau d'expérience avec Director, Lingo ou la syntaxe JavaScript™, prenez un moment pour consulter « Principes de base de la programmation dans Director », page 4 et « Rédaction de scripts dans Director », page 43 avant de commencer à rédiger des scripts. Comme tous les produits, Director possède ses propres et uniques conventions de programmation, types de données, etc. Pour pouvoir rédiger des scripts efficaces, vous devez vous familiariser avec ces caractéristiques uniques de Director.

Nouveautés concernant la programmation dans Director

Si vous avez déjà écrit des scripts avec les versions précédentes de Director, vous devez prendre note des innovations et modifications importantes concernant la programmation dans cette nouvelle version.

Adobe Director prend en charge la programmation en Unicode.

Limites de la prise en charge d'Unicode dans Director

- Les langues qui s'écrivent de la droite vers la gauche ne sont pas prises en charge.
- Les fonctions de l'Xtra File I/O telles que readchar(), getLength() et getPosition() ne fonctionnent qu'avec une entrée de caractères à un octet. Pour pouvoir lire des caractères Unicode à deux ou trois octets, lisez le fichier en entier dans un objet de chaîne à l'aide de la méthode readFile(). Utilisez ensuite la méthode char...of pour lire le caractère Unicode.
- Les composants de Director Physics ne prennent pas en charge Unicode.
- Les noms Unicode des chemins d'accès HTTP ne sont pas pris en charge.
- Vous ne pouvez pas nommer un Xtra de programmation comme une chaîne Unicode en utilisant la clé de registre « kMoaMmDictType_MessageTable ». Vous ne pouvez pas non plus exposer les fonctions Lingo nommées avec Unicode en utilisant les Xtras de programmation.
- Les Xtras de programmation exposent les fonctions Lingo. Les noms de ces fonctions exposés via les Xtras ne sont pas pris en charge avec Unicode.
- Les noms de modèles 3D ne sont pas pris en charge avec Unicode.

La fenêtre Script comporte un panneau de l'explorateur en plus de l'Editeur de script. Par défaut, le panneau de l'explorateur apparaît à gauche de l'Editeur de script. Vous pouvez voir le panneau de l'explorateur en mode dictionnaire ou navigateur de scripts.

Mode dictionnaire

Le mode dictionnaire affiche une liste de fonctions de script Lingo/JavaScript intégrées organisées sous la forme d'une arborescence. Les fonctions sont classées selon leur catégorie et par ordre alphabétique comme un index.

Utilisez le mode dictionnaire pour effectuer les opérations suivantes :

- parcourir les fonctions intégrées des scripts Lingo et JavaScript ;
- utiliser les fonctions intégrées pour créer des scripts.

Mode navigateur de scripts

Le mode navigateur de scripts affiche les scripts et les gestionnaires associés qui ont été utilisés dans l'animation. Dans ce mode, vous pouvez créer de nouveaux scripts et gestionnaires.

Utilisez le mode navigateur de scripts pour effectuer les opérations suivantes :

- parcourir les scripts et les gestionnaires de l'animation actuelle comme une arborescence ou une liste ;
- trier les scripts en fonction du nom du script, du nom de l'acteur, du numéro de l'acteur ou du type de script ;
- localiser un gestionnaire dans l'Editeur de script ;
- créer des scripts pour chaque type de script.

Nouveautés de cette documentation

Si vous avez appris à rédiger des scripts dans les versions précédentes de Director, vous devez prendre connaissance des modifications apportées à cette nouvelle documentation. Le manuel *Référence de scripting de Director* remplace le *Dictionnaire Lingo* livré avec les versions précédentes de Director. Cette référence n'est pas organisée de la même manière que le *Dictionnaire Lingo*.

Dans le *Dictionnaire Lingo*, les informations concernant le modèle de programmation étaient classées par fonction. Par exemple, si vous vouliez apprendre à utiliser des images-objets dans un script, vous deviez consulter l'une des sections référencées sous l'en-tête Images-objets, telles que Glissement d'images-objets, Dimensions des images-objets, etc. En outre, toutes les API de programmation étaient répertoriées par ordre alphabétique dans une seule liste, c'est-à-dire que les fonctions, les propriétés, les événements, etc. étaient tous réunis dans la même liste par ordre alphabétique.

Dans le manuel *Référence de scripting de Director*, les informations concernant le modèle de programmation sont classées par objet. Cette classification reflète fidèlement l'organisation des objets de programmation actuels que vous utilisez dans vos scripts. Par exemple, si vous voulez savoir comment utiliser les images-objets dans un script, vous devez consulter la section Images-objets du chapitre Objets principaux de Director.

Les API de programmation sont toujours répertoriées par ordre alphabétique mais elles sont également classées en fonction du type d'API. Par exemple, toutes les méthodes sont listées par ordre alphabétique sous l'en-tête Méthodes, toutes les propriétés sont listées par ordre alphabétique sous l'en-tête Propriétés, etc.

Recherche d'informations en matière de programmation dans Director

Le nouveau manuel *Référence de scripting de Director* contient les rubriques suivantes :

Principes de base de la programmation dans Director fournit des informations sur les concepts et composants de base que vous utilisez lors de la rédaction de scripts dans Director.

Rédaction de scripts dans Director fournit des informations sur l'environnement de programmation de Director, en plus de celles concernant les concepts et techniques de programmation avancés.

Débogage de scripts dans Director fournit des informations sur la manière de résoudre les problèmes liés à vos scripts lorsqu'ils ne fonctionnent pas correctement.

Objets principaux de Director fournit une liste des objets et API utilisés pour accéder aux principales fonctionnalités et fonctions de Director, telles que le moteur du lecteur de Director, les fenêtres d'animation, les images-objets, les sons, etc.

Types de médias fournit une liste des types de médias et API que vous utilisez pour accéder aux fonctionnalités des divers types de médias (RealMedia, DVD, GIF animé, etc.) et qui sont ajoutés aux animations en tant qu'acteurs.

Objets de programmation fournit une liste des objets de programmation, aussi appelés Xtras, et des API que vous utilisez pour développer les fonctionnalités de base de Director. Les Xtras fournissent certaines fonctions telles que l'importation de filtres et la connexion à Internet.

Objets 3D fournit une liste des objets utilisés pour ajouter des fonctionnalités 3D à une animation.

Constantes fournit une liste des constantes disponibles dans Director.

Evénements et messages fournit une liste des événements disponibles dans Director.

Mots-clés fournit une liste des mots clés disponibles dans Director.

Méthodes fournit une liste des méthodes disponibles dans Director.

Opérateurs fournit une liste des opérateurs disponibles dans Director.

Propriétés fournit une liste des propriétés disponibles dans Director.

Chapitre 2 : Principes de base de la programmation dans Director

Si vous n'avez jamais créé de scripts dans Director®, prenez le temps d'apprendre les principaux concepts de programmation qui vous aident à créer des scripts dans Director avant de commencer. Certains de ces principes de base comprennent des définitions de termes importants, les règles de syntaxe, les types de données disponibles et des informations sur les éléments de programmation de base dans Director, tels que les variables, les tableaux, les opérateurs, etc.

Types de scripts

Une animation de Director peut contenir quatre types de scripts : les comportements, les scripts d'animation, les scripts parents et les scripts associés aux acteurs. Les comportements, les scripts d'animation et les scripts parents figurent tous dans la fenêtre Distribution sous la forme d'acteurs indépendants. Un script associé à un acteur figure dans la fenêtre Distribution et n'apparaît pas indépendamment.

- **Comportements** : ces scripts sont associés à des images-objets ou images dans le scénario et sont appelés comportements d'image-objet ou comportements d'image. La miniature de la fenêtre Distribution de chaque comportement contient une icône de comportement dans l'angle inférieur droit.

  Lorsqu'il est utilisé dans le manuel *Référence de scripting de Director*, le terme *comportement* se rapporte à n'importe quel script associé à une image-objet ou une image. Cette définition diffère de celle des comportements figurant dans la Palette des bibliothèques de Director. Pour plus d'informations sur ces comportements de Director, consultez les rubriques du manuel Utilisation de Director dans l'Aide de Director.

  Tous les comportements ajoutés à une bibliothèque de distribution figurent dans le menu local Comportements de l'Inspecteur de comportement. Ce menu ne contient pas les autres types de scripts.

  Vous pouvez placer le même comportement à plusieurs endroits du scénario. Lorsque vous modifiez un comportement, la nouvelle version de ce comportement est automatiquement appliquée à tous les endroits auxquels il est associé dans le scénario.

- **Scripts d'animation** : ces scripts contiennent des gestionnaires disponibles globalement ou au niveau d'une animation. Les gestionnaires d'événements d'un script d'animation peuvent être appelés depuis n'importe quel script de l'animation pendant la lecture.

  Une icône de script d'animation apparaît dans l'angle inférieur droit de la miniature correspondante dans la fenêtre Distribution.

  Les scripts d'animation sont disponibles pour l'animation entière, quelle que soit l'image où se trouve la tête de lecture ou l'image-objet que l'utilisateur manipule. Lors de la lecture d'une animation dans une fenêtre ou comme animation liée, un script d'animation n'est disponible que pour sa propre animation.

- **Scripts parents** : ces scripts spéciaux contiennent les éléments Lingo utilisés pour créer des objets enfants. Vous pouvez utiliser des scripts parents pour générer des objets scripts qui ont une réponse et un comportement similaires tout en se comportant indépendamment les uns des autres. Une icône de script parent apparaît dans l'angle inférieur droit de la miniature correspondante dans la fenêtre Distribution.

Pour plus d'informations sur l'utilisation des scripts parents et des objets enfants, consultez « Programmation orientée objet avec la syntaxe Lingo », page 49.

La syntaxe JavaScript n'utilise pas de scripts parents ou d'objets enfants. Elle utilise des techniques de programmation orientée objet de style JavaScript normales. Pour plus d'informations sur la programmation orientée objet dans la syntaxe JavaScript, consultez « Programmation orientée objet avec la syntaxe JavaScript », page 58.

- **Composants d'acteurs Flash®** : vous ne pouvez accéder à ces composants placés sur la scène (sous la forme d'images-objets Flash) lorsqu'ils sont invisibles qu'en utilisant l'objet Acteur. L'utilisation de l'objet Image-objet pour une image-objet Flash avec une propriété d'invisibilité renverra un message d'erreur.

- **Scripts** associés aux acteurs : scripts directement associés à un acteur, indépendamment du scénario. Lorsque ces acteurs sont affectés à des images-objets, leurs scripts sont mis à la disposition de ces dernières.

  Contrairement aux comportements, aux scripts parents et aux scripts d'animation, les scripts d'acteurs n'apparaissent pas dans la fenêtre Distribution. Cependant, si l'option Afficher les icônes de script des acteurs est sélectionnée dans la boîte de dialogue Préférences de la fenêtre Distribution, les acteurs auxquels sont associés des scripts sont identifiés par une petite icône de script dans l'angle inférieur gauche de leur miniature dans la fenêtre Distribution.

Terminologie de programmation

La syntaxe Lingo et la syntaxe JavaScript utilisent toutes deux des termes qui sont propres à chaque langage, en plus de certains termes qui sont partagés par ces langages.

Les termes importants de programmation sont présentés dans l'ordre alphabétique. Ces termes sont utilisés tout au long du manuel *Référence de scripting de Director*, il est donc nécessaire de bien les assimiler avant de continuer.

- **Constantes** : éléments dont les valeurs ne changent jamais. Par exemple, dans Lingo, les constantes telles que `TAB`, `EMPTY` et `RETURN` ont toujours les mêmes valeurs et ne peuvent être modifiées. Dans la syntaxe JavaScript, les constantes telles que `Math.PI` et `Number.MAX_VALUE` ont toujours les mêmes valeurs et ne peuvent être modifiées. Vous pouvez également créer vos propres constantes dans la syntaxe JavaScript à l'aide du mot-clé `const`.

  Pour plus d'informations sur les constantes, consultez « Constantes », page 14.

- **Evénements** : actions qui se produisent pendant la lecture d'une animation. Les événements se produisent à l'arrêt d'une animation, au démarrage d'une image-objet, à l'entrée de la tête de lecture dans une image, lors de l'emploi du clavier par l'utilisateur, etc. Les événements de Director sont tous prédéfinis et ont toujours le même sens.

  Pour plus d'informations sur les événements, consultez « Evénements », page 28.

- **Expressions** : parties d'instruction produisant une valeur. Par exemple, `2 + 2` est une expression.

- **Fonctions** : fonctions de haut niveau ou types spécifiques de codes de la syntaxe JavaScript.

  Une *fonction de haut niveau* ordonne à une animation de faire quelque chose pendant la lecture de l'animation ou renvoie une valeur, mais elle n'est pas appelée à partir d'un objet spécifique. Par exemple, vous appelez la fonction de haut niveau `list()` en utilisant la syntaxe `list()`. Comme c'est le cas pour une fonction, une *méthode* ordonne également à une animation de faire quelque chose pendant la lecture de l'animation ou renvoie une valeur, mais elle est toujours appelée à partir d'un objet.

Une fonction est utilisée dans la syntaxe JavaScript pour représenter un gestionnaire d'événement, un objet personnalisé, une méthode personnalisée, etc. L'utilisation des fonctions JavaScript dans de tels cas est expliquée dans les rubriques appropriées, plus loin dans cette référence.

- **Gestionnaires** ou gestionnaires d'événements : ensemble d'instructions placées dans un script et exécutées en réponse à un événement déterminé et à un message subséquent. Lorsqu'un événement se produit, Director génère et envoie un message correspondant aux scripts et un gestionnaire approprié est exécuté en réponse au message. Les noms des gestionnaires sont toujours les mêmes que les événements et messages auxquels ils répondent.

  Remarque : *bien qu'un événement soit géré par une fonction dans la syntaxe JavaScript, le terme* gestionnaire *désigne, dans cette référence, à la fois les gestionnaires Lingo et les fonctions de la syntaxe JavaScript qui gèrent les événements.*

  Pour plus d'informations sur les gestionnaires, consultez « Gestionnaires », page 30.

- **Mots-clés** : mots réservés ayant un sens particulier. Par exemple, dans Lingo, le mot-clé `end` indique la fin d'un gestionnaire. Dans la syntaxe JavaScript, le mot-clé `var` indique que le terme suivant est une variable.

- **Listes** (Lingo) ou **Tableaux** (syntaxe JavaScript) : ensembles ordonnés de valeurs qui permettent le suivi et la mise à jour d'un ensemble de données, tels qu'une série de noms ou de valeurs affectées à un ensemble de variables. Un exemple simple de liste est une liste de nombres, telle que `[1, 4, 2]`.

  Pour plus d'informations sur l'utilisation des listes dans Lingo et dans la syntaxe JavaScript, consultez « Listes linéaires et listes de propriétés », page 33.

  Pour plus d'informations sur l'utilisation des tableaux de la syntaxe JavaScript, consultez « Tableaux de la syntaxe JavaScript », page 40.

- **Messages** : avertissements envoyés aux scripts par Director lorsque des événements déterminés se produisent dans une animation. Par exemple, lorsque la tête de lecture entre dans une image donnée, l'événement `enterFrame` se produit et Director envoie un message `enterFrame`. Si un script contient un gestionnaire `enterFrame`, les instructions de ce gestionnaire sont exécutées puisque le gestionnaire a reçu le message `enterFrame`. Si aucun script ne contient de gestionnaire de message, le message est ignoré dans le script.

  Pour plus d'informations sur les messages, consultez « Messages », page 29.

- **Méthodes** : termes entraînant une action pendant la lecture de l'animation ou le renvoi d'une valeur ; elles sont appelées à partir d'un objet. Par exemple, vous appelez la méthode `insertFrame()` de l'objet Animation en utilisant la syntaxe `_movie.insertFrame()`. Bien que leurs fonctionnalités soient similaires aux fonctions de haut niveau, les méthodes sont toujours appelées à partir d'un objet, contrairement aux fonctions de haut niveau.

- **Opérateurs** : calculent une nouvelle valeur à partir d'une ou de plusieurs valeurs. Par exemple, l'opérateur d'addition (+) additionne deux valeurs ou plus pour produire une nouvelle valeur.

  Pour plus d'informations sur les opérateurs, consultez « Opérateurs », page 19.

- **Paramètres** : repères permettant de transmettre des valeurs aux scripts. Les paramètres s'appliquent aux méthodes et aux gestionnaires d'événements mais pas aux propriétés. Ils sont exigés par certaines méthodes, mais ne sont pas obligatoires pour d'autres.

  Par exemple, la méthode `go()` de l'objet Animation envoie la tête de lecture à une image précise, et indique éventuellement le nom de l'animation dans laquelle se trouve l'image. Pour effectuer cette tâche, la méthode `go()` nécessite au moins un paramètre et en accepte un autre. Le premier paramètre requis précise l'image à laquelle doit être envoyée la tête de lecture, le second paramètre facultatif indique l'animation dans laquelle se trouve l'image. Vu que le premier paramètre est obligatoire, une erreur de script est renvoyée s'il n'est pas présent au moment où la méthode `go()` est invoquée. Étant donné que le second paramètre est facultatif, la méthode effectue sa tâche même si ce paramètre n'est pas présent.

- **Propriétés** : attributs définissant un objet. Par exemple, l'image-objet d'une animation possède des attributs précis tels que sa largeur, sa hauteur, la couleur de son arrière-plan, etc. Pour accéder aux valeurs de ces trois attributs, vous devez utiliser les propriétés `width`, `height` et `backColor` de l'objet de l'image-objet.

  Pour plus d'informations sur l'attribution de propriétés aux variables, consultez « Stockage et mise à jour de valeurs dans des variables », page 16.

- **Instructions** : instructions valides que Director peut exécuter. Les scripts sont tous composés d'ensembles d'instructions. L'instruction Lingo suivante est une instruction complète.

  ```
  _movie.go("Author")
  ```

  Pour plus d'informations sur la rédaction d'instructions, consultez « Programmation avec la syntaxe à point », page 44.

- **Variables** : éléments servant à stocker et à mettre à jour des valeurs. Les variables doivent commencer par une lettre, un trait de soulignement (_) ou le signe dollar ($). Les caractères suivants du nom d'une variable peuvent également être des chiffres (0 à 9). Pour attribuer des valeurs aux variables ou modifier les valeurs de plusieurs propriétés, utilisez l'opérateur égal à (=).

  Pour plus d'informations sur l'utilisation des variables, consultez « Variables », page 15.

Syntaxe de programmation

Les règles générales suivantes s'appliquent à Lingo et à la syntaxe JavaScript :

- Les repères de commentaires varient de Lingo à la syntaxe JavaScript.

  Les commentaires Lingo sont tous précédés de deux traits d'union (--). Chaque ligne d'un commentaire constitué de plusieurs lignes doit être précédée de deux traits d'union.

  ```
  -- This is a single-line Lingo comment

  -- This is a
  -- multiple-line Lingo comment
  ```

  Les commentaires d'une seule ligne utilisant la syntaxe JavaScript sont précédés de deux barres obliques (//). Les commentaires à plusieurs lignes sont précédés du signe /* et suivis du signe */.

  ```
  // This is a single-line JavaScript syntax comment

  /* This is a
  multiple-line JavaScript syntax comment */
  ```

  Vous pouvez placer un commentaire sur une ligne séparée ou après une instruction. Le texte qui suit les repères de commentaires sur la même ligne est ignoré.

  Les commentaires peuvent être variés : il peut s'agir de notes concernant un script ou un gestionnaire particulier ou une instruction dont l'objectif n'est pas évident. Ces commentaires vous permettent de vous familiariser avec une procédure que vous n'avez pas utilisée depuis longtemps.

  L'ajout d'un grand nombre de commentaires n'augmente pas la taille de votre fichier d'animation lorsqu'il est enregistré sous forme d'un fichier compressé DCR ou DXR. Les commentaires sont supprimés du fichier lors du processus de décompression.

Vous pouvez également utiliser des repères de commentaires pour ignorer les sections de code que vous souhaitez désactiver à des fins de test ou de débogage. En ajoutant des repères de commentaires au code au lieu de le supprimer, vous transformez provisoirement une section en commentaires. Sélectionnez le code à activer ou à désactiver et utilisez les boutons Insérer une marque de commentaire ou Supprimer la marque de commentaire de la fenêtre Script pour ajouter ou retirer rapidement les repères de commentaires.

• Les parenthèses sont nécessaires après chaque nom de méthode et de fonction. Par exemple, lorsque vous appelez la méthode `beep()` de l'objet Son, vous devez insérer des parenthèses après le mot beep. Sinon, une erreur de script se produira.

```
// JavaScript syntax
_sound.beep(); // this statement will work properly
_sound.beep; // this statement will result in a script error
```

Lorsque vous appelez une méthode, une fonction ou un gestionnaire à partir d'une autre méthode ou fonction ou d'un autre gestionnaire, vous devez inclure des parenthèses dans l'instruction d'appel. Dans l'exemple suivant, la méthode `modifySprite()` contient un appel de gestionnaire `spriteClicked`. L'appel du gestionnaire `spriteClicked` doit inclure des parenthèses, sinon, vous obtenez une erreur de script :

```
// JavaScript syntax
function modifySprite() {
    spriteClicked(); // this call to the handler will work properly
    spriteClicked; // this call to the handler results in a script error
}
function spriteClicked() {
    // handler code here
}
```

Vous pouvez également utiliser des parenthèses pour annuler l'ordre de priorité des opérations mathématiques ou pour faciliter la lecture de vos instructions. Par exemple, la première expression mathématique ci-dessous renverra le résultat 13, tandis que la seconde expression renverra le résultat 5 :

```
5 * 3 - 2 -- yields 13
5 * (3 - 2) -- yields 5
```

• La syntaxe des gestionnaires d'événements varie de Lingo à JavaScript. Avec Lingo, les gestionnaires utilisent la syntaxe *`on handlerName`*. Avec la syntaxe JavaScript, les gestionnaires sont implémentés comme des fonctions et utilisent la syntaxe `function handlerName()`. Par exemple, les instructions suivantes contiennent un gestionnaire qui émet un signal sonore lorsque l'utilisateur clique sur la souris :

```
-- Lingo syntax
on mouseDown
    _sound.beep()
end
// JavaScript syntax
function mouseDown() {
    _sound.beep();
}
```

• La syntaxe des paramètres des gestionnaires d'événements peut varier de Lingo à la syntaxe JavaScript. Lingo et JavaScript prennent en charge les paramètres transmis à un gestionnaire, placés entre parenthèses. Si plusieurs paramètres sont transmis, chaque paramètre est séparé par une virgule. Dans Lingo, vous pouvez également transmettre des paramètres qui ne sont pas placés entre parenthèses. Par exemple, le gestionnaire `addThem` reçoit les deux paramètres `a` et `b` :

```
-- Lingo syntax
on addThem a, b -- without parentheses
    c = a + b
end
```

```
on addThem(a, b) -- with parentheses
    c = a + b
end

// JavaScript syntax
function addThem(a, b) {
    c = a + b;
}
```

• Le mot-clé `const` peut être utilisé dans la syntaxe JavaScript pour préciser une constante dont la valeur ne change pas. Lingo possède son propre ensemble de constantes prédéfinies (`TAB`, `EMPTY`, etc.), le mot-clé `const` ne s'applique donc pas à Lingo.

Par exemple, l'instruction suivante indique une constante appelée `intAuthors` et définit sa valeur sur 12. Cette valeur est toujours 12 et ne pourra pas être modifiée par un script :

```
// JavaScript syntax
const intAuthors = 12;
```

• Le mot-clé `var` dans la syntaxe JavaScript peut être placé devant un mot pour préciser que ce terme est une variable. L'instruction suivante crée une variable appelée `startValue` :

```
// JavaScript syntax
var startValue = 0;
```

Remarque : *bien que l'utilisation de* `var` *dans la syntaxe JavaScript soit facultative, il est recommandé de toujours déclarer les variables JavaScript locales ou celles se trouvant au sein d'une fonction à l'aide de* `var`*. Pour plus d'informations sur l'utilisation des variables, consultez* *« Variables », page 15**.*

• Le symbole de continuation (\) de Lingo indique qu'une longue ligne d'exemples de codes s'étale sur deux ou plusieurs lignes. Les lignes Lingo divisées de cette manière ne représentent pas des lignes de codes séparées. Par exemple, le code suivant est valide :

```
-- Lingo syntax
tTexture = member("3D").model("box") \
    .shader.texture
```

La syntaxe JavaScript ne comporte pas de symbole de continuation de ligne. Pour diviser une longue chaîne de codes JavaScript en plusieurs lignes, il vous suffit d'ajouter un retour à la ligne à la fin d'une ligne, puis de continuer le code sur la ligne suivante.

• Le point-virgule peut être utilisé pour indiquer la fin d'une instruction en code JavaScript. Les points-virgules ne s'appliquent pas à Lingo.

L'utilisation du point-virgule est facultative. Lorsqu'il est utilisé, il doit être placé à la fin d'une instruction complète. Par exemple, les deux instructions suivantes créent une variable appelée `startValue` :

```
// JavaScript syntax
var startValue = 0
var startValue = 0;
```

Un point-virgule n'indique pas nécessairement la fin d'une ligne de code JavaScript, et plusieurs instructions peuvent être placées sur une ligne. Cependant, pour des raisons de clarté, il est recommandé de placer des instructions distinctes sur des lignes séparées. Par exemple, les trois instructions suivantes occupent une seule ligne de code et fonctionnent correctement, mais la lecture du code n'est pas facile.

```
// JavaScript syntax
_movie.go("Author"); var startValue = 0; _sound.beep();
```

- Les espaces entre les caractères au sein des expressions et instructions sont ignorés dans Lingo et la syntaxe JavaScript. Dans les chaînes de caractères entre guillemets droits, les espaces sont considérés comme des caractères. Si vous souhaitez insérer des espaces dans une chaîne, vous devez les y placer explicitement. Par exemple, la première instruction ci-dessous ignore les espaces entre les éléments de la liste, et la seconde inclut les espaces.

```
-- Lingo syntax
myList1 = ["1",      "2",      "3"] -- yields ["1", "2", "3"]
myList2 = ["  1  ", "  2  ", "  3  "] -- yields ["  1  ", "  2  ", "  3  "]
```

- La sensibilité à la casse peut varier de Lingo à la syntaxe JavaScript.

  Lingo ne différencie pas les majuscules des minuscules, ce qui vous permet d'utiliser les majuscules et les minuscules comme bon vous semble. Par exemple, les quatre instructions suivantes sont équivalentes :

```
-- Lingo syntax
member("Cat").hilite = true
member("cat").hiLite = True
MEMBER("CAT").HILITE = TRUE
Member("Cat").Hilite = true
```

  Bien que Lingo ne soit pas sensible à la casse, il est recommandé de choisir une casse et de l'utiliser de manière cohérente dans l'ensemble de vos scripts. Cette habitude permet d'identifier plus facilement les noms de gestionnaires, variables, acteurs, etc.

  La syntaxe JavaScript est sensible à la casse lorsqu'elle fait référence à des objets, aux propriétés de haut niveau ou méthodes faisant référence à des objets ou aux variables définies par l'utilisateur. Par exemple, la méthode de haut niveau `sprite()` renvoie une référence à une image-objet précise et est implémentée dans Director avec toutes les lettres minuscules. La première instruction ci-dessous se rapporte au nom de la première image-objet d'une animation, tandis que les deux instructions suivantes entraînent une erreur de script.

```
// JavaScript syntax
sprite(1).name // This statement functions normally
Sprite(1).name // This statement results in a script error
SPRITE(1).name // This statement results in a script error
```

  Les chaînes littérales sont toujours sensibles à la casse dans Lingo et la syntaxe JavaScript.

  Pour plus d'informations sur l'utilisation des chaînes, consultez « Chaînes », page 13.

Types de données

Un *type de données* est un ensemble de données dont les valeurs possèdent des caractéristiques similaires et prédéfinies. Chaque valeur de variable et de propriété dans Director est d'un type de données précis, et les valeurs renvoyées par les méthodes sont d'un type de données précis.

Par exemple, considérons les deux instructions suivantes. Dans la première instruction, il a été attribué à la variable `intX` le nombre entier 14. La variable `intX` est donc du type de données Nombre entier. Dans la seconde instruction, il a été attribué à la variable `stringX` une séquence de valeurs de caractères, c'est-à-dire une chaîne. La variable `stringX` est donc du type de données Chaîne.

```
-- Lingo syntax
intX = 14
stringX = "News Headlines"

// JavaScript syntax
var intX = 14;
var stringX = "News Headlines";
```

Les valeurs renvoyées par les méthodes ou fonctions appartiennent également à un type de données. Par exemple, la méthode `windowPresent()` de l'objet Lecteur renvoie une valeur indiquant si une fenêtre est présente. La valeur renvoyée est `TRUE` (1) ou `FALSE` (0).

Certains types de données sont partagés par Lingo et la syntaxe JavaScript, d'autres se rapportent à l'un ou l'autre des langages. L'ensemble de types de données pris en charge par Director est fixe et ne peut être modifié, ce qui signifie qu'il n'est pas possible d'ajouter de nouveaux types de données ou de supprimer les types de données existants. Director prend en charge les types de données suivants.

Type de données	**Description**
# (symbole)	Unité autonome pouvant représenter une condition ou un indicateur. Par exemple, `#list` ou `#word`.
Tableau	(syntaxe JavaScript seulement) Bien qu'il ne s'agisse pas exactement d'un type de données, un objet Tableau peut être utilisé avec des listes linéaires de valeurs. La fonctionnalité d'un objet Tableau est identique à celle du type de données Liste de Lingo.
Valeur booléenne	Une valeur `TRUE` (1) ou `FALSE` (0). Dans Lingo, toutes les valeurs `TRUE` ou `FALSE` sont de simples constantes de nombres entiers, 1 pour `TRUE` et 0 pour `FALSE`. Dans la syntaxe JavaScript, toutes les valeurs `true` ou `false` sont par défaut les vraies valeurs booléennes `true` ou `false` mais sont automatiquement converties, le cas échéant, en simples constantes de nombres entiers dans Director. Dans Lingo, `TRUE` et `FALSE` peuvent être en majuscules ou en minuscules. Dans la syntaxe JavaScript, `true` et `false` doivent toujours être en minuscules.
Couleur	Représente la couleur d'un objet.
Constante	Une donnée dont la valeur ne change pas.
Date	Bien qu'il ne s'agisse pas exactement d'un type de données, un objet Date permet d'utiliser des dates dans la syntaxe JavaScript. Dans Lingo, utilisez la méthode date() pour créer un objet Date et utiliser des dates.
Valeur à virgule flottante	(Lingo seulement) Nombre à virgule flottante. Par exemple, 2,345 ou 45,43.
Fonction	(syntaxe JavaScript seulement) Bien qu'il ne s'agisse pas exactement d'un type de données, un objet Fonction peut être utilisé pour préciser une chaîne de codes à exécuter.
Nombre entier	(Lingo seulement) Nombre entier. Par exemple, 5 ou 298.
Liste	Une liste linéaire ou de propriétés composée respectivement de noms de valeurs ou de propriétés et de valeurs.
Mathématique	(syntaxe JavaScript seulement) Bien qu'il ne s'agisse pas exactement d'un type de données, un objet Mathématique peut être utilisé pour effectuer des opérations mathématiques.
Nul	(syntaxe JavaScript seulement) Indique une variable dont la valeur correspond à 0 dans les contextes numériques et à `false` dans les contextes booléens.
Nombre	(syntaxe JavaScript seulement) Bien qu'il ne s'agisse pas exactement d'un type de données, un objet Nombre peut être utilisé pour représenter des constantes numériques telles qu'une valeur maximale, une valeur not-a-number (NaN) et une infinité.
Objet	Bien qu'il ne s'agisse pas exactement d'un type de données, un objet Objet peut être utilisé pour créer un conteneur personnalisé de données et des méthodes agissant sur les données.
Point	Point sur l'espace de coordonnées ayant une coordonnée horizontale et une coordonnée verticale.
Rectangle	Rectangle dans l'espace de coordonnées.

Type de données (Suite)	Description (Suite)
Expression régulière	(JavaScript seulement) Modèle d'expression régulière utilisé pour la correspondance avec les combinaisons de caractères dans les chaînes.
Chaîne	Chaîne de symboles clavier ou de valeurs de caractères. Par exemple, « Director » ou « $5,00 ».
Non défini	(syntaxe JavaScript seulement) Indique une variable qui n'a pas de valeur.
Vecteur	Point dans l'espace 3D.
VOID	(Lingo seulement) Indique une valeur vide.

Remarque : *plusieurs types de données et d'objets de la syntaxe JavaScript possèdent leur propre ensemble de méthodes et propriétés qui peuvent être utilisées pour une manipulation plus approfondie de ces types. Bien que le manuel* Référence de scripting de Director *donne parfois des informations sur certaines de ces méthodes et propriétés, ces informations ne sont pas complètes. Pour plus d'informations sur ces types de données et objets et leurs méthodes et propriétés, consultez l'une des nombreuses documentations d'autres éditeurs sur le sujet.*

Les propriétés intégrées de Director, telles que la propriété `name` de l'objet Acteur, ne peuvent recevoir que des valeurs qui sont du même type de données que celui de la propriété. Par exemple, le type de données de la propriété `name` de l'objet Acteur est une chaîne, et la valeur doit donc être une chaîne telle que `News Headlines`. Si vous essayez d'attribuer une valeur d'un type de données différent à cette propriété, telle que le nombre entier 20, vous obtenez une erreur de script.

Si vous créez vos propres propriétés, leurs valeurs peuvent être de n'importe quel type de données, quel que soit le type de données de la valeur initiale.

Lingo et la syntaxe JavaScript sont tous deux dynamiques. Cela signifie que vous n'avez pas à préciser le type de données d'une variable lorsque vous la déclarez et que les types de données sont automatiquement convertis, le cas échéant, lors de l'exécution d'un script.

Par exemple, la syntaxe JavaScript suivante définit la variable `myMovie` comme étant un nombre entier, et plus tard dans le script, elle est définie comme étant une chaîne. Lors de l'exécution du script, le type de données de `myMovie` est converti automatiquement :

```
-- Lingo syntax
myMovie = 15 -- myMovie is initially set to an integer
...
myMovie = "Animations" -- myMovie is later set to a string

// JavaScript syntax
var myMovie = 15; // myMovie is initially set to an integer
...
myMovie = "Animations"; // myMovie is later set to a string
```

Valeurs littérales

Une *valeur littérale* est une partie d'instruction ou d'expression qui doit être traitée telle quelle, et non comme une variable ou un élément de script. Les valeurs littérales que vous pouvez rencontrer dans un script sont les chaînes de caractères, les entiers, les nombres décimaux, les noms et numéros d'acteurs, les noms et numéros d'images et d'animations, les symboles et les constantes.

Chaque type de valeur littérale est régi par ses propres règles.

Chaînes

Les *chaînes* sont des mots ou des groupes de caractères que le script traite en tant que mots standard, et non en tant que variables. Elles doivent être encadrées de guillemets droits. Par exemple, vous pouvez utiliser des chaînes pour transmettre des messages aux utilisateurs de votre animation ou pour attribuer des noms aux acteurs. Dans l'instruction suivante, `Hello` et `Greeting` sont des chaînes. `Hello` est le texte littéral placé dans l'acteur texte et `Greeting` est le nom de ce dernier.

```
-- Lingo syntax
member("Greeting").text = "Hello"
```

De même, lorsque vous testez une chaîne, vous devez l'entourer de guillemets droits, comme dans l'exemple suivant :

```
-- Lingo syntax
if "Hello Mr. Jones" contains "Hello" then soundHandler
```

Lingo et la syntaxe JavaScript considèrent les espaces figurant au début ou à la fin d'une chaîne comme une partie littérale de la chaîne. L'expression suivante comprend un espace après le mot *à* :

```
// JavaScript syntax
trace("My thoughts amount to ");
```

Bien que Lingo ne distingue pas les majuscules des minuscules lorsqu'il fait référence aux acteurs, aux variables, etc., les chaînes littérales sont sensibles à la casse. Par exemple, les deux instructions suivantes placent un texte différent dans l'acteur indiqué, car `Hello` et `HELLO` sont des chaînes littérales :

```
-- Lingo syntax
member("Greeting").text = "Hello"
member("Greeting").text = "HELLO"
```

Dans Lingo, la fonction `string()` peut convertir une valeur numérique en une chaîne. Dans la syntaxe JavaScript, la méthode `toString()` peut convertir une valeur numérique en une chaîne.

Remarque : *si vous essayez d'utiliser la méthode `toString()` dans la syntaxe JavaScript sur une valeur `null` ou `undefined`, vous obtenez une erreur de script. Ceci n'est pas le cas avec Lingo, dont la fonction `string()` s'applique à toutes les valeurs, y compris celles qui sont `VOID`.*

Nombres

Dans Lingo, il existe deux types de nombres : les entiers et les décimaux.

Un *nombre entier* ne comporte ni fraction ni décimale, dans les plages -2 147 483 648 et +2 147 483 647. Entrez des nombres entiers sans utiliser de virgule. Utilisez le signe moins (-) pour les nombres négatifs.

Un *nombre décimal*, également appelé *nombre à virgule flottante*ou *valeur à virgule flottante*, est un nombre qui inclut une virgule décimale. Dans Lingo, la propriété `floatPrecision` détermine le nombre de décimales utilisées pour l'affichage de ces nombres. Director utilise toujours le nombre entier, jusqu'à 15 chiffres utiles, dans ses calculs ; il arrondit tout nombre comportant plus de 15 chiffres utiles.

La syntaxe JavaScript ne fait pas de distinction entre les nombres entiers et les nombres à virgule flottante, et n'utilise que des nombres. Par exemple, les instructions suivantes montrent que le nombre 1 est un nombre entier dans Lingo et un nombre dans la syntaxe JavaScript, et que le nombre décimal 1,05 est un nombre à virgule flottante dans Lingo et un nombre dans la syntaxe JavaScript :

```
-- Lingo syntax
put(ilk(1)) -- #integer
put(ilk(1.05)) -- #float
```

```
// JavaScript syntax
trace(typeof(1)) // number
trace(typeof(1.05)) // number
```

Dans Lingo, vous pouvez convertir un nombre décimal en nombre entier avec la fonction `integer()`. Vous pouvez également convertir un nombre entier en nombre décimal en effectuant une opération mathématique sur le nombre entier, par exemple en multipliant un nombre entier par un nombre décimal. Dans la syntaxe JavaScript, la fonction `parseInt()` vous permet de convertir une chaîne ou un nombre décimal en nombre entier. Contrairement à la fonction `integer()` de Lingo, `parseInt()` arrondit au nombre inférieur. Par exemple, l'instruction suivante arrondit le nombre décimal 3,9 et le convertit au nombre entier 4 (Lingo) et 3 (syntaxe JavaScript) :

```
-- Lingo syntax
theNumber = integer(3.9) -- results in a value of 4

// JavaScript syntax
var theNumber = parseInt(3.9); // results in a value of 3
```

Dans Lingo, la fonction `value()` peut convertir une chaîne en une valeur numérique.

Vous pouvez également utiliser une notation exponentielle avec les nombres décimaux : par exemple, `-1,1234e-100` ou `123,4e+9`.

Dans Lingo, la fonction `float()` vous permet de convertir un nombre entier ou une chaîne en un nombre décimal. Dans la syntaxe JavaScript, la fonction `parseFloat()` vous permet de convertir une chaîne en un nombre décimal. Par exemple, l'instruction suivante enregistre la valeur 3,0000 (Lingo) et 3 (syntaxe JavaScript) dans la variable `theNumber`.

```
-- Lingo syntax
theNumber = float(3) -- results in a value of 3.0000

// JavaScript syntax
var theNumber = parseFloat(3) // results in a value of 3
```

Constantes

Une *constante* est une valeur déclarée dont le contenu ne change jamais.

Dans Lingo, les termes prédéfinis `TRUE`, `FALSE`, `VOID` et `EMPTY` sont des constantes car leur valeur ne change jamais. Les termes prédéfinis `BACKSPACE`, `ENTER`, `QUOTE`, `RETURN`, `SPACE` et `TAB` sont des constantes qui font référence aux touches du clavier. Par exemple, pour tester si la dernière touche enfoncée par l'utilisateur était la barre Espace, utilisez l'instruction suivante :

```
-- Lingo syntax
if _key.keyPressed() = SPACE then beep()
```

Dans la syntaxe JavaScript, vous pouvez accéder à des constantes prédéfinies à l'aide de types de données propres à la syntaxe JavaScript. Par exemple, l'objet Nombre contient des constantes telles que `Number.MAX_VALUE` et `Number.NaN`, l'objet Mathématique renferme des constantes telles que `Math.PI` et `Math.E`, etc.

Remarque : *cette référence ne fournit pas d'informations approfondies sur les constantes prédéfinies dans la syntaxe JavaScript. Pour plus d'informations sur ces constantes, consultez l'une des nombreuses documentations d'autres éditeurs sur le sujet.*

Dans la syntaxe JavaScript, vous pouvez également définir vos propres constantes à l'aide du mot-clé `const`. Par exemple, l'instruction suivante crée une constante appelée `items` et lui attribue la valeur 20. Cette valeur ne peut pas être modifiée après sa création.

```
// JavaScript syntax
const items = 20;
```

Pour plus d'informations sur les constantes, consultez « Constantes », page 146.

Symboles

Dans Lingo, un *symbole* est une chaîne ou toute autre valeur précédée du signe dièse (#).

Les symboles sont des constantes définies par l'utilisateur. Les comparaisons utilisant des symboles s'effectuent très rapidement, créant ainsi un code plus efficace.

Remarque : *dans Lingo, la majuscule pour le symbole correspond à la minuscule. Si vous essayez de convertir une chaîne (par exemple, Sonia) en symbole à l'aide de la fonction symbol(), vous obtiendrez comme résultat : sonia.*

Par exemple, la première instruction ci-dessous s'exécute plus rapidement que la seconde :

```
-- Lingo syntax
userLevel = #novice
userLevel = "novice"
```

Les symboles ne peuvent contenir ni espaces ni ponctuation.

Dans Lingo et la syntaxe JavaScript, convertissez une chaîne en symbole à l'aide de la méthode `symbol()`.

```
-- Lingo syntax
x = symbol("novice") -- results in #novice

// JavaScript syntax
var x = symbol("novice"); // results in #novice
```

Reconvertissez un symbole en chaîne à l'aide de la fonction `string()` (Lingo) ou la méthode `toString()` (syntaxe JavaScript).

```
-- Lingo syntax
x = string(#novice) -- results in "novice"

// JavaScript syntax
var x = symbol("novice").toString(); // results in "novice"
```

Dans la syntaxe JavaScript, vous ne pouvez pas comparer des symboles du même nom pour déterminer s'ils font référence au même symbole. Pour comparer des symboles du même nom, vous devez d'abord les convertir en chaîne en utilisant la méthode `toString()`.

Variables

Director utilise des *variables* pour conserver et actualiser les valeurs. Comme son nom l'indique, une variable contient une valeur qui peut être modifiée ou mise à jour pendant la lecture de l'animation. En modifiant la valeur d'une variable pendant la lecture de l'animation, Director peut par exemple stocker une URL, mémoriser le nombre de fois qu'un utilisateur prend part à une session de discussion en ligne, enregistrer si une opération réseau est terminée ou non, etc.

Il est conseillé de toujours attribuer une valeur connue à une variable la première fois que vous la déclarez. Cette opération est appelée *initialisation* d'une variable. L'initialisation d'une variable facilite le suivi de cette variable et permet de comparer ses différentes valeurs au fur et à mesure de la lecture de l'animation.

Les variables peuvent être globales ou locales. Une *variable locale* n'existe que tant que le gestionnaire dans lequel elle a été définie est en cours d'exécution. Une *variable globale* peut exister et conserver sa valeur tant que l'application Director est en cours d'exécution, notamment lorsqu'une animation passe à une autre animation. Une variable peut être globale au sein d'un gestionnaire individuel, un script spécifique ou une animation entière ; la portée dépend de la manière dont la variable globale est initialisée.

Si vous souhaitez qu'une variable soit disponible pendant toute l'animation, il est recommandé de la déclarer dans un gestionnaire `on prepareMovie` (Lingo) ou `function prepareMovie()` (syntaxe JavaScript). Elle est ainsi disponible dès le début de l'animation.

Pour plus d'informations sur l'utilisation des variables globales et locales, consultez « Utilisation de variables globales », page 17 et « Utilisation de variables locales », page 19.

Stockage et mise à jour de valeurs dans des variables

Les variables peuvent contenir des données pour tous les types de données de Director, qu'il s'agisse de nombres entiers, de chaînes, de valeurs `TRUE` ou `FALSE`, de symboles, de listes ou du résultat d'un calcul. Pour stocker les valeurs des propriétés et des variables, utilisez l'opérateur égal à (`=`).

Comme indiqué dans la section Types de données de cette référence, les variables de Lingo et de la syntaxe JavaScript sont dynamiques, ce qui signifie qu'elles contiennent des types de données différents à des moments différents. La possibilité de modifier le type d'une variable distingue Lingo d'autres langages tels que Java™ et C++, où cette possibilité n'existe pas.

Par exemple, l'instruction `set x = 1` crée la variable `x`, qui est une variable nombre entier, car vous lui avez attribué un entier. Si vous utilisez ensuite l'instruction `set x = "un"`, la variable `x` devient une variable à chaîne puisqu'elle contient maintenant une chaîne.

Vous pouvez convertir une chaîne en nombre à l'aide de la fonction `value()` (Lingo) ou de la méthode `parseInt()` (syntaxe JavaScript)ou un nombre en chaîne à l'aide de la fonction `string()` (Lingo) ou de la méthode `toString()` (syntaxe JavaScript).

Les valeurs de certaines propriétés peuvent être définies (la valeur est attribuée) et renvoyées (la valeur est récupérée) et les valeurs de certaines propriétés ne peuvent être que renvoyées. Les propriétés dont les valeurs peuvent être à la fois définies et renvoyées sont appelées *lecture/écriture*, celles qui peuvent seulement être renvoyées étant appelées *lecture seule*.

Il s'agit en général de propriétés décrivant une condition échappant au contrôle de Director. Par exemple, vous ne pouvez pas définir la propriété d'acteur `numChannels`, qui indique le nombre de pistes d'une animation possédant un contenu Adobe® Shockwave®. Par contre, vous pouvez récupérer le nombre de pistes en faisant référence à la propriété `numChannels` d'un acteur.

Affecter une valeur à une variable

❖ Utilisez l'opérateur égal à (`=`).

Par exemple, l'instruction suivante attribue une URL à la variable `placesToGo` :

```
// JavaScript syntax
var placesToGo = "http://www.adobe.com";
```

Les variables peuvent également contenir le résultat d'opérations mathématiques. Par exemple, l'instruction suivante ajoute le résultat d'une addition à la variable `mySum` :

```
-- Lingo syntax
mySum = 5 + 5 -- this sets mySum equal to 10
```

Comme exemple supplémentaire, l'instruction suivante renvoie l'acteur attribué à l'image-objet 2 en récupérant la valeur de la propriété `member` de l'image-objet et le place dans la variable `textMember`.

```
-- Lingo syntax
textMember = sprite(2).member
```

Il est recommandé d'utiliser des noms de variables qui indiquent le rôle de ces variables. Vos scripts n'en sont que plus faciles à lire. Par exemple, la variable `maSum` indique que cette variable contient le résultat d'une addition.

Tester la valeur de propriétés ou de variables

❖ Utilisez la fonction `put()` ou `trace()` dans la fenêtre Messages ou cochez les valeurs de la fenêtre Surveillance (`put()` et `trace()` proposent des fonctionnalités identiques et sont disponibles dans Lingo et la syntaxe JavaScript).

Par exemple, l'instruction suivante affiche la valeur attribuée à la variable `myNumber` dans la fenêtre Messages.

```
-- Lingo syntax
myNumber = 20 * 7
put(myNumber) -- displays 140 in the Message window

// JavaScript syntax
var myNumber = 20 * 7;
trace(myNumber) // displays 140 in the Message window
```

Utilisation de variables globales

Les variables globales peuvent être partagées par les gestionnaires, les scripts ou les animations. Une variable globale existe et garde sa valeur tant que Director est en cours d'exécution ou jusqu'à ce que vous appeliez la méthode `clearGlobals()`.

Dans Adobe Shockwave Player, les variables globales subsistent dans les animations affichées par la méthode `goToNetMovie()`, mais pas dans celles qui sont affichées par la méthode `goToNetPage()` .

Chaque gestionnaire déclarant une variable globale peut utiliser la valeur de cette variable. S'ils modifient cette valeur, la nouvelle valeur est accessible à tous les autres gestionnaires qui considèrent cette variable comme globale.

Il est recommandé de démarrer le nom de toutes les variables globales par un *g* minuscule. Cette convention permet d'identifier plus facilement les variables globales lors de l'examen de votre code.

Director propose une manière d'afficher toutes les variables globales actuelles et leurs valeurs actuelles, et d'effacer les valeurs de toutes les variables globales.

Afficher toutes les variables globales actuelles et leurs valeurs actuelles

❖ Utilisez la méthode `showGlobals()` de l'objet Global dans la fenêtre Messages.

Pour plus d'informations sur la fenêtre Messages, consultez « Débogage dans la fenêtre Messages », page 83.

Supprimer toutes les variables globales actuelles

❖ Utilisez la méthode `clearGlobals()` de l'objet Global de la fenêtre Messages pour attribuer à toutes les variables globales la valeur `VOID` (Lingo) ou `undefined` (syntaxe JavaScript).

Le contrôle des valeurs des variables globales, lors de la lecture d'une animation, s'effectue par le biais de l'Inspecteur d'objet. Pour plus d'informations sur l'Inspecteur d'objet, consultez « Débogage dans l'Inspecteur d'objet », page 87.

Variables globales dans Lingo

Dans Lingo, les variables sont considérées, par défaut, comme des variables locales, et vous n'avez donc pas besoin d'insérer un mot-clé avant le nom d'une variable. Pour déclarer une variable globale, vous devez faire précéder la variable du mot-clé `global`.

Si vous déclarez une variable globale au début d'un script et avant un gestionnaire, cette variable est disponible pour tous les gestionnaires de ce script. Si vous déclarez une variable globale au sein d'un gestionnaire, cette variable est disponible pour ce gestionnaire uniquement. Cependant, si vous déclarez une variable globale du même nom au sein de deux gestionnaires distincts, et que vous mettez à jour la valeur de la variable dans un gestionnaire, vous mettez également à jour la valeur de la variable dans l'autre gestionnaire.

Les exemples suivants illustrent ce qui se passe avec deux variables globales : `gScript`, qui est à la disposition de tous les gestionnaires dans le script, et `gHandler`, qui est disponible au sein de son propre gestionnaire et de tout autre gestionnaire qui la déclare à sa première ligne.

```
-- Lingo syntax
global gScript -- gScript is available to all handlers

on mouseDown
    global gHandler
    gScript = 25
    gHandler = 30
end

on mouseUp
    global gHandler
    trace(gHandler) -- displays 30
end
```

Dans Lingo, lorsque vous utilisez le terme `global` pour définir des variables globales, elles possèdent automatiquement une valeur initiale égale à `VOID`.

Variables globales dans la syntaxe JavaScript

Dans la syntaxe JavaScript, les variables sont considérées comme des variables globales par défaut. La portée d'une variable globale peut être déterminée par la manière dont elle est déclarée et la position à laquelle elle est déclarée.

- Si vous déclarez une variable au sein d'une fonction de la syntaxe JavaScript sans la faire précéder du mot-clé `var`, cette variable est disponible pour toutes les fonctions du script qui la contient.
- Si vous déclarez une variable en dehors d'une fonction de la syntaxe JavaScript, avec ou sans le mot-clé `var`, cette variable est disponible pour toutes les fonctions du script qui la contient.
- Si vous déclarez une variable à l'intérieur ou à l'extérieur d'une fonction de la syntaxe JavaScript en utilisant la syntaxe `_global.`*`varName`*, cette variable est disponible pour tous les scripts d'une animation.

L'exemple suivant utilise la syntaxe `_global.gMovie` dans un script pour déclarer la variable `gMovie` en tant que variable globale. Cette variable est disponible pour tous les scripts d'une animation :

```
// JavaScript syntax
_global.gMovie = 1; // Declare gMovie in one script

// Create a function in a separate script that operates on gMovie
function mouseDown() {
    _global.gMovie++;
    return(_global.gMovie);
}
```

L'exemple suivant déclare la variable globale `gScript` dans un script. Cette variable est uniquement disponible pour les fonctions de ce script.

```
// JavaScript syntax
var gScript = 1; // Declare gScript in a script

// gScript is available only to functions in the script that defines it
function mouseDown() {
    gScript++;
    return(gScript);
}
```

Dans la syntaxe JavaScript, lorsque vous définissez des variables avant les gestionnaires, elles reçoivent automatiquement la valeur initiale `undefined`.

Utilisation de variables locales

Une variable locale n'existe que tant que le gestionnaire dans lequel elle a été définie est en cours d'exécution. Cependant, après avoir créé une variable locale, vous pouvez l'utiliser dans d'autres expressions ou modifier sa valeur tant qu'un script se trouve dans le gestionnaire qui a défini la variable.

Il est préférable de définir une variable comme locale lorsque vous ne souhaitez l'utiliser que provisoirement dans un gestionnaire. Vous limitez ainsi les risques de modification accidentelle de sa valeur dans d'autres gestionnaires utilisant le même nom de variable.

Créer une variable locale

❖ Dans Lingo, attribuez une valeur à la variable à l'aide de l'opérateur égal à (=).

❖ Dans la syntaxe JavaScript, à l'intérieur d'une fonction, insérez le mot-clé `var` avant le nom de la variable puis attribuez-lui une valeur à l'aide de l'opérateur égal à.

> ***Remarque :*** *Etant donné que les variables de la syntaxe JavaScript sont, par défaut, globales, si vous essayez de déclarer une variable locale à l'intérieur d'une fonction sans utiliser le mot-clé* `var`*, votre script risque de créer un comportement inattendu. Ainsi, et bien que l'utilisation de* `var` *soit facultative, il est fortement recommandé de déclarer toutes les variables JavaScript locales à l'aide de* `var` *pour éviter tout comportement inattendu.*

Afficher toutes les variables locales actuelles d'un gestionnaire

❖ Dans Lingo, utilisez la fonction `showLocals()`.

Dans Lingo,vous pouvez utiliser cette méthode dans la fenêtre Messages ou dans des gestionnaires pour faciliter le débogage des scripts. Le résultat apparaît dans la fenêtre Messages. La méthode `showLocals()` ne s'applique pas à la syntaxe JavaScript.

Pour contrôler les valeurs des variables locales lors de la lecture d'une animation, utilisez l'Inspecteur d'objet. Pour plus d'informations sur l'Inspecteur d'objet, consultez « Débogage dans l'Inspecteur d'objet », page 87.

Opérateurs

Les *opérateurs* sont des éléments indiquant aux scripts de Lingo et de la syntaxe JavaScript comment combiner, comparer ou modifier les valeurs d'une expression. Plusieurs des opérateurs de Director sont partagés par Lingo et la syntaxe JavaScript, et certains sont propres à un langage.

Certains types d'opérateurs comprennent :

- les **opérateurs arithmétiques** (tels que `+`, `-`, / et `*`) ;
- les **opérateurs de comparaison** (tels que `<`,`>` et `>=`), qui comparent deux arguments ;
- les **opérateurs logiques** (`not`, `and`, `or`), qui combinent des conditions simples en conditions composées ;
- les **opérateurs de chaînes** (tels que `&`, `&&` et `+`) qui relient ou concatènent des chaînes de caractères.

Remarque : *il existe beaucoup plus de types d'opérateurs dans la syntaxe JavaScript que dans Lingo, et ils ne sont pas tous décrits dans cette référence. Pour plus d'informations sur les autres opérateurs dans JavaScript 1.5, reportez-vous à l'une des nombreuses documentations d'autres éditeurs sur le sujet.*

Les éléments sur lesquels les opérateurs agissent sont appelés des *opérandes*. Dans Lingo, il n'existe que des opérateurs binaires. Dans la syntaxe JavaScript, il existe des opérateurs binaires et unaires. Un *opérateur binaire* nécessite deux opérandes, l'un placé avant l'opérateur et l'autre après. Un *opérateur unaire* nécessite un seul opérande, placé soit avant soit après l'opérateur.

Dans l'exemple suivant, la première instruction illustre un opérateur binaire où les variables `x` et `y` sont des opérandes et le signe plus (`+`) l'opérateur. La seconde instruction illustre un opérateur unaire où la variable `i` est l'opérande et `++` l'opérateur.

```
// JavaScript syntax
x + y; // binary operator
i++; // unary operator
```

Pour plus d'informations sur les opérateurs, consultez « Opérateurs », page 626.

Ordre de priorité des opérateurs

Lorsqu'au moins deux opérateurs sont utilisés dans la même instruction, certains opérateurs sont prioritaires par rapport à d'autres selon une hiérarchie précise pour déterminer les opérateurs à exécuter en premier. Cette hiérarchie est appelée *ordre de priorité* des opérateurs. Par exemple, une multiplication est toujours effectuée avant une addition. Cependant, les éléments entre parenthèses sont prioritaires par rapport à la multiplication. Dans l'exemple suivant, en l'absence de parenthèses, la multiplication de cette instruction se produit en premier :

```
-- Lingo syntax
total = 2 + 4 * 3 -- results in a value of 14
```

Lorsque l'addition apparaît entre parenthèses, l'addition s'effectue en premier :

```
-- Lingo syntax
total = (2 + 4) * 3 -- results in a value of 18
```

Vous trouverez ci-dessous une description des types d'opérateurs et de leur ordre de priorité. Les opérateurs possédant une priorité élevée sont exécutés en premier. Par exemple, un opérateur dont l'ordre de priorité est 5 est exécuté avant un opérateur dont l'ordre de priorité est 4. Les opérations qui ont le même ordre de priorité sont exécutées de gauche à droite.

Opérateurs arithmétiques

Les *opérateurs arithmétiques* additionnent, soustraient, multiplient, divisent et effectuent d'autres opérations arithmétiques. Les parenthèses et le signe moins sont également des opérateurs arithmétiques.

Opérateur	Effet	Ordre de priorité
( )	Opérations permettant de contrôler la priorité.	5
-	Lorsque placé devant un chiffre, en inverse la valeur.	5
*	Effectue une multiplication.	4
mod	(Lingo seulement) Effectue des opérations de modulos.	4
/	Effectue une division.	4
%	(syntaxe JavaScript seulement) Renvoie le reste du nombre entier résultant d'une division de deux opérandes.	4
++	(syntaxe JavaScript seulement) Ajoute un à son opérande. S'il est utilisé comme opérateur préfixe (`++x`), renvoie la valeur de son opérande après avoir ajouté un. S'il est utilisé comme opérateur suffixe (`x++`), renvoie la valeur de son opérande avant d'ajouter un.	4
--	(syntaxe JavaScript seulement) Soustrait un de son opérande. La valeur renvoyée est analogue à celle de l'opérateur d'incréments.	4
+	Lorsque placé entre deux chiffres, effectue une addition.	3
-	Lorsque placé entre deux chiffres, effectue une soustraction.	3

Remarque : *dans Lingo, lorsque seuls des nombres entiers sont utilisés dans une opération, le résultat est toujours un nombre entier. Si vous utilisez des entiers et des nombres à virgule flottante dans la même opération, le résultat est toujours un nombre à virgule flottante. Dans la syntaxe JavaScript, tous les calculs renvoient comme résultat des nombres à virgule flottante.*

Si le résultat de la division d'un entier par un autre entier n'est pas un nombre entier, Lingo arrondit le résultat au nombre entier inférieur le plus proche. Par exemple, le résultat de `4/3` est `1`. Dans la syntaxe JavaScript, la valeur à virgule flottante, 1,333, est renvoyée.

Pour forcer Lingo à calculer une valeur sans arrondir le résultat, utilisez `float()` avec une ou plusieurs des valeurs dans l'expression. Par exemple, le résultat de `4/float(3)` est `1.333)`.

Opérateurs de comparaison

Les *opérateurs de comparaison* comparent deux valeurs et déterminent si la comparaison est vraie (true) ou fausse (false).

Opérateur	Signification	Ordre de priorité
==	(syntaxe JavaScript seulement) Deux opérandes sont égaux. Si les opérandes ne sont pas du même type de données, la syntaxe JavaScript essaie de les convertir en type de données approprié afin d'effectuer la comparaison.	1
===	(syntaxe JavaScript seulement) Deux opérandes sont égaux et du même type.	1
!=	(syntaxe JavaScript seulement) Deux opérandes ne sont pas égaux. Si les opérandes ne sont pas du même type de données, la syntaxe JavaScript essaie de les convertir en type de données approprié afin d'effectuer la comparaison.	1
!==	(syntaxe JavaScript seulement) Deux opérandes ne sont pas égaux et/ou du même type.	1
<>	(Lingo seulement) Deux opérandes ne sont pas égaux.	1

Opérateur (Suite)	Signification (Suite)	Ordre de priorité (Suite)
<	L'opérande de gauche est inférieur à l'opérande de droite.	1
<=	L'opérande de gauche est inférieur ou égal à l'opérande de droite.	1
>	L'opérande de gauche est supérieur à l'opérande de droite.	1
>=	L'opérande de gauche est supérieur ou égal à l'opérande de droite.	1
=	(Lingo seulement) Deux opérandes sont égaux.	1

Opérateurs d'affectation

Un *opérateur d'affectation* attribue une valeur à son opérande de gauche en fonction de la valeur de son opérande de droite. A l'exception de l'opérateur d'affectation de base égal à (=), tous les opérateurs d'affectation de raccourcis suivants ne s'appliquent qu'à la syntaxe JavaScript.

Opérateur	Signification	Ordre de priorité
=	Egal à	1
x += y	(syntaxe JavaScript seulement) x = x + y	1
x -= y	(syntaxe JavaScript seulement) x = x - y	1
x *= y	(syntaxe JavaScript seulement) x = x * y	1
x /= y	(syntaxe JavaScript seulement) x = x / y	1
x %= y	(syntaxe JavaScript seulement) x = x % y	1

Opérateurs logiques

Les *opérateurs logiques* testent si deux expressions logiques sont vraies (true) ou fausses (false).

Faites attention lorsque vous utilisez les opérateurs logiques et les opérateurs de chaînes dans Lingo et la syntaxe JavaScript. Par exemple, dans la syntaxe JavaScript, `&&` est un opérateur logique qui détermine si deux expressions sont vraies, mais dans Lingo, `&&` est un opérateur de chaînes qui concatène deux chaînes et insère un espace entre deux expressions.

Opérateur	Effet	Ordre de priorité
and	(Lingo seulement) Détermine si les deux expressions sont vraies.	4
&&	(syntaxe JavaScript seulement) Détermine si les deux expressions sont vraies.	4
or	(Lingo seulement) Détermine si une des expressions ou les deux, sont vraies.	4
||	(syntaxe JavaScript seulement) Détermine si une des expressions ou les deux, sont vraies.	4
not	(Lingo seulement) Inverse une expression.	5
!	(JavaScript seulement) Inverse une expression.	5

L'opérateur `not` (Lingo) ou ! (syntaxe JavaScript) est utile pour passer de la valeur `TRUE` ou `FALSE` à la valeur opposée. Par exemple, l'instruction suivante active le son s'il est désactivé ou le désactive s'il est activé :

```
-- Lingo syntax
_sound.soundEnabled = not (_sound.soundEnabled)

// JavaScript syntax
_sound.soundEnabled = !(_sound.soundEnabled);
```

Opérateurs de chaînes

Les *opérateurs de chaînes* combinent et définissent des chaînes.

Faites attention lorsque vous utilisez les opérateurs logiques et les opérateurs de chaînes dans Lingo et la syntaxe JavaScript. Par exemple, dans la syntaxe JavaScript, `&&` est un opérateur logique qui détermine si deux expressions sont vraies, mais dans Lingo, `&&` est un opérateur de chaînes qui concatène deux chaînes et insère un espace entre deux expressions.

Opérateur	Effet	Ordre de priorité
&	(Lingo seulement) Concatène deux chaînes.	2
+	(syntaxe JavaScript seulement) Concatène deux valeurs de chaîne et renvoie une troisième chaîne qui unit les deux opérandes.	2
+=	(syntaxe JavaScript seulement) Concatène une variable de chaîne et une valeur de chaîne et attribue la valeur renvoyée à la variable de chaîne.	2
&&	(Lingo seulement) Concatène deux chaînes et insère un espace entre elles.	2
"	Indique le début ou la fin d'une chaîne.	1

Constructions conditionnelles

Par défaut, Director exécute toujours les instructions d'un script en commençant par la première instruction et en continuant dans l'ordre dans lequel elles apparaissent, jusqu'à ce qu'il rencontre la dernière instruction ou une instruction l'envoyant à un autre endroit.

L'ordre d'exécution des instructions est tributaire de l'ordre dans lequel vous les placez. Par exemple, si vous rédigez une instruction nécessitant le calcul d'une valeur, vous devez d'abord placer une instruction calculant cette valeur.

Dans l'exemple suivant, la première instruction additionne deux nombres et la seconde affecte une représentation de la somme sous forme de chaîne à un acteur champ, appelé `Answer`, affiché sur la scène. La seconde instruction n'a pu être placée avant la première parce que la variable `x` n'a pas encore été définie.

```
-- Lingo syntax
x = 2 + 2
member("Answer").text = string(x)

// JavaScript syntax
var x = 2 + 2;
member("Answer").text = x.toString();
```

Lingo et la syntaxe JavaScript proposent des conventions servant à modifier l'ordre d'exécution ou les instructions de script par défaut et aussi à effectuer des actions en fonction de conditions spécifiques. Par exemple, vous pouvez choisir d'effectuer les actions suivantes dans vos scripts :

- exécuter une série d'instructions si une condition logique est vraie ou exécuter d'autres instructions si la condition logique est fausse ;

- évaluer une expression et essayer d'adapter sa valeur à une condition spécifique ;
- exécuter une série d'instructions plusieurs fois de suite jusqu'à ce qu'une condition spécifique soit remplie.

Test de conditions logiques

Pour exécuter une instruction ou une série d'instructions lorsqu'une condition donnée est vraie ou fausse, vous utilisez les structures `if...then...else` (Lingo) ou `if...else` (syntaxe JavaScript). Par exemple, vous pouvez créer une structure `if...then...else` ou `if...then` qui vérifie que le téléchargement du texte à partir d'Internet est terminé et qui, le cas échéant, le formate. Ces structures utilisent le modèle suivant pour tester les conditions logiques :

- Dans Lingo et la syntaxe JavaScript, les instructions qui vérifient si une condition est vraie ou fausse commencent par l'élément `if`.
- Dans Lingo, si la condition existe, les instructions suivant l'élément `then` sont exécutées. Dans la syntaxe JavaScript, les accolades ({ }) remplacent l'élément Lingo `then` et doivent encadrer chaque instruction `if`, `else` ou `else if`.
- Dans Lingo et la syntaxe JavaScript, si la condition n'existe pas, les scripts passent à l'instruction suivante du gestionnaire en utilisant l'élément `else` ou `else if`.
- Dans Lingo, l'élément `end if` indique la fin du test `if`. Dans la syntaxe JavaScript, le test `if` se termine automatiquement. Il n'y a donc pas d'élément qui termine explicitement le test.

Pour optimiser les performances de vos scripts, commencez par tester les conditions les plus vraisemblables.

Les instructions suivantes testent plusieurs conditions. Le terme `else if` spécifie l'exécution d'autres tests si les conditions précédentes se sont avérées fausses :

```
-- Lingo syntax
if _mouse.mouseMember = member(1) then
    _movie.go("Cairo")
else if _mouse.mouseMember = member(2) then
    _movie.go("Nairobi")
else
    _player.alert("You're lost.")
end if

// JavaScript syntax
if (_mouse.mouseMember = member(1)) {
    _movie.go("Cairo");
}
else if (_mouse.mouseMember = member(2)) {
    _movie.go("Nairobi");
}
else {
    _player.alert("You're lost.");
}
```

Lors de la rédaction de structures `if…then` dans Lingo, vous pouvez placer l'instruction ou les instructions après `then` sur la même ligne que celle de `then` ou les placer sur leur propre ligne en insérant un retour chariot après `then`. Si vous insérez un retour chariot, vous devez également placer une instruction `end if` à la fin de la structure `if...then`.

Lors de la rédaction de structures `if` dans la syntaxe JavaScript, vous pouvez placer l'instruction ou les instructions après `if` sur la même ligne que celle de `if` ou les placer sur leur propre ligne en insérant un retour chariot après `if`.

Par exemple, les instructions suivantes sont équivalentes :

```
-- Lingo syntax
if _mouse.mouseMember = member(1) then _movie.go("Cairo")

if _mouse.mouseMember = member(1) then
    _movie.go("Cairo")
end if

// JavaScript syntax
if (_mouse.mouseMember = member(1)) { _movie.go("Cairo"); }

if (_mouse.mouseMember = member(1)) {
    _movie.go("Cairo");
}
```

Pour plus d'informations sur l'utilisation des structures `if...then...else` et `if...else`, consultez « if », page 202.

Evaluation et correspondance d'expressions

Les structures `case` (Lingo) ou `switch...case` (syntaxe JavaScript) sont des raccourcis permettant d'éviter l'utilisation des structures `if...then...else` ou `if...then` dans les structures à branchements multiples. Les structures `case` et `switch...case` sont souvent plus efficaces et plus lisibles que beaucoup de structures `if...then...else` ou `if...then`.

Dans Lingo, la condition devant être testée suit le terme `case` dans la première ligne de la structure `case`. La comparaison commence par la première ligne, puis passe à la suivante, etc. jusqu'à ce que Lingo rencontre une expression correspondant à la condition testée. Lorsqu'une correspondance est trouvée, Director exécute les instructions Lingo qui suivent l'expression.

Dans la syntaxe JavaScript, la condition devant être testée suit le terme `switch` dans la première ligne de la structure. Chaque comparaison du test suit le terme `case` dans chaque ligne contenant un test. Vous pouvez mettre fin à une comparaison `case` en utilisant le terme facultatif `break`. Lorsque vous insérez le terme `break`, vous excluez le programme de la structure `switch` et exécutez toute instruction qui suit la structure. Si vous n'insérez pas `break`, la comparaison `case` suivante est exécutée.

Une structure `case` ou `switch...case` peut utiliser des comparaisons comme conditions de test.

Par exemple, les structures `case` et `switch...case` suivantes testent la dernière touche sur laquelle l'utilisateur a appuyé, et répond en conséquence :

- Si l'utilisateur a appuyé sur A, l'animation passe à l'image Pomme.
- Si l'utilisateur a appuyé sur B ou C, l'animation exécute la transition demandée et passe à l'image Oranges.
- Si l'utilisateur a appuyé sur n'importe quelle autre touche, l'ordinateur émet un bip sonore.

```
-- Lingo syntax
case (_key.key) of
    "a" : _movie.go("Apple")
    "b", "c":
         _movie.puppetTransition(99)
         _movie.go("Oranges")
    otherwise: _sound.beep()
end case

// JavaScript syntax
switch (_key.key) {
    case "a" :
        _movie.go("Apple");
```

```
            break;
        case "b":
        case "c":
             _movie.puppetTransition(99);
            _movie.go("Oranges");
            break;
            default: _sound.beep()
    }
```

Remarque : *dans la syntaxe JavaScript, vous ne pouvez effectuer qu'une seule comparaison par instruction* `case`.

Pour plus d'informations sur l'utilisation des structures `case`, consultez « case », page 196.

Répétition d'actions

Dans Lingo et la syntaxe JavaScript, vous pouvez répéter une action un certain nombre de fois ou tant qu'une condition spécifique existe.

Dans Lingo, pour répéter une action un certain nombre de fois, vous utilisez une structure `repeat with`. Spécifiez le nombre de répétitions sous forme de plage après l'instruction `repeat with`.

Pour répéter une action un certain nombre de fois dans la syntaxe JavaScript, vous utilisez la structure `for`. La structure `for` nécessite trois paramètres : le premier paramètre initialise généralement une variable compteur, le second précise une condition à évaluer à chaque fois dans la boucle et le troisième est généralement utilisé pour mettre à jour ou incrémenter la variable compteur.

Les structures `repeat with` et `for` servent à effectuer la même opération sur une série d'objets. Par exemple, la boucle suivante applique l'encre Fond transparent aux images-objets 2 à 10 :

```
-- Lingo syntax
repeat with n = 2 to 10
    sprite(n).ink = 36
end repeat

// JavaScript syntax
for (var n=2; n<=10; n++) {
    sprite(n).ink = 36;
}
```

Cet exemple exécute une action similaire, mais avec des nombres décroissants :

```
-- Lingo syntax
repeat with n = 10 down to 2
    sprite(n).ink = 36
end repeat

// JavaScript syntax
for (var n=10; n>=2; n--) {
    sprite(n).ink = 36;
}
```

Dans Lingo, pour répéter une série d'instructions tant qu'une condition spécifique existe, utilisez la structure `repeat while`.

Dans la syntaxe JavaScript, pour répéter une série d'instructions tant qu'une condition spécifique existe, utilisez la structure `while`.

Par exemple, les instructions suivantes font émettre un bip sonore continu à l'animation à chaque fois que l'utilisateur appuie sur le bouton de la souris :

```
-- Lingo syntax
repeat while _mouse.mouseDown
    _sound.beep()
end repeat

// JavaScript syntax
while (_mouse.mouseDown) {
    _sound.beep();
}
```

Les scripts Lingo et JavaScript continuent à effectuer une boucle sur les instructions de la boucle jusqu'à ce que la condition ne soit plus vraie ou jusqu'à ce que l'une des instructions envoie le script à l'extérieur de la boucle. Dans l'exemple précédent, le script quitte la boucle lorsque l'utilisateur relâche le bouton de la souris puisque la condition `mouseDown` cesse d'exister.

Pour quitter une boucle dans Lingo, utilisez l'instruction `exit repeat`.

Pour quitter une boucle dans la syntaxe JavaScript, vous pouvez utiliser le terme `break`. Le script quitte automatiquement la boucle lorsqu'une condition n'est plus vraie.

Par exemple, les instructions suivantes font émettre un bip sonore à l'animation à chaque fois que l'utilisateur appuie sur le bouton de la souris, sauf si le pointeur de la souris se trouve au-dessus de l'image-objet 1. Dans ce cas, le script quitte la boucle et le bip sonore s'arrête. La méthode `rollover()` indique si le pointeur se trouve au-dessus de l'image-objet spécifiée.

```
-- Lingo syntax
repeat while _mouse.stillDown
    _sound.beep()
    if _movie.rollOver(1) then exit repeat
end repeat

// JavaScript syntax
while (_mouse.stillDown) {
    _sound.beep();
    if (_movie.rollOver(1)) {
        break;
    }
}
```

Pour plus d'informations sur les structures `repeat while` et `while`, consultez « repeat while », page 213.

Evénements, messages et gestionnaires

Pour créer des scripts solides et utiles, il est essentiel de maîtriser les concepts et les fonctionnalités des événements, messages et gestionnaires. Si vous comprenez l'ordre dans lequel les événements et les messages sont envoyés et reçus, vous pouvez déterminer exactement quand des scripts ou des parties de scripts donnés doivent être exécutés. Ceci vous aide à déboguer vos scripts si certaines actions ne se produisent pas au moment prévu.

Les événements suivants se produisent durant la lecture d'une animation :

- des événements se produisent en réaction à une action du système ou à une action définie par l'utilisateur ;
- des messages correspondant à ces événements sont envoyés aux scripts d'une animation ;
- les gestionnaires compris dans les scripts contiennent les instructions qui sont exécutées lorsqu'un message est reçu.

Le nom d'un événement correspond au nom du message qu'il produit, et le gestionnaire de cet événement correspond à la fois à l'événement et au message. Par exemple, lorsque l'événement `mouseDown` se produit, Director crée et envoie aux scripts un message appelé `mouseDown` qui est ensuite pris en charge par un gestionnaire appelé `mouseDown`.

Evénements

Durant la lecture d'une animation, deux catégories d'événements se produisent :

- Les **événements système** se produisent sans aucune interaction entre un utilisateur et l'animation et sont prédéfinis et nommés dans Director, par exemple, lorsque la tête de lecture entre dans une image, lorsque vous cliquez sur une image-objet, etc.
- Les **événements définis par l'utilisateur** se produisent en réponse aux actions que vous définissez. Par exemple, vous pouvez créer un événement qui se produit lorsque la couleur de fond d'une image-objet passe du rouge au bleu, après qu'un son a été lu cinq fois, etc.

Plusieurs événements système tels que `prepareFrame`, `beginSprite`, etc., se produisent automatiquement et dans un ordre prédéfini durant la lecture d'une animation. D'autres événements système, en particulier des événements liés à la souris tels que `mouseDown`, `mouseUp` etc., ne se produisent pas nécessairement de manière automatique durant la lecture d'une animation, mais plutôt après qu'ils aient été déclenchés par un utilisateur.

Par exemple, au début d'une animation, l'événement `prepareMovie` est toujours le premier à se produire, l'événement `prepareFrame` toujours le second, etc. Cependant, les événements `mouseDown` et `mouseUp` peuvent ne jamais se produire dans une animation, à moins qu'un utilisateur ne les déclenche en cliquant sur l'animation.

Les listes suivantes répertorient les événements système qui se produisent toujours durant une animation et l'ordre dans lequel ils se produisent.

Les événements se produisent dans l'ordre suivant au démarrage de l'animation :

1 `prepareMovie`

2 `prepareFrame` Immédiatement après l'événement `prepareFrame`, Director lit les sons, dessine les images-objets et effectue les transitions ou les effets de palette. Cet événement se produit avant l'événement `enterFrame`. L'utilisation d'un gestionnaire `prepareFrame` est conseillée pour exécuter un script avant que l'image ne soit dessinée.

3 `beginSprite` Cet événement se produit lorsque la tête de lecture pénètre dans la zone d'une image-objet.

4 `startMovie` Cet événement se produit dans la première image lue.

Lorsqu'une animation rencontre une image, les événements se produisent dans l'ordre suivant :

1 `beginSprite` Cet événement ne se produit que si de nouvelles images-objets apparaissent dans l'image.

2 `stepFrame`

3 `prepareFrame`

4 `enterFrame` Après l'événement `enterFrame`, mais avant `exitFrame`, Director gère les délais exigés par les réglages de cadence, les événements d'inactivité et les événements clavier et souris.

5 `exitFrame`

6 `endSprite` Cet événement ne se produit que lorsque la tête de lecture sort d'une image-objet de l'image.

Les événements se produisent dans l'ordre suivant lorsque la lecture de l'animation s'arrête :

1 `endSprite` Cet événement ne se produit que si l'animation contient des images-objets.

2 `stopMovie`

Pour plus d'informations sur les événements système prédéfinis de Director, consultez « Evénements et messages », page 153.

Messages

Pour exécuter au bon moment l'ensemble d'instructions de script approprié, Director doit déterminer ce qui se passe dans l'animation et les instructions à exécuter pour répondre à certains événements.

Director utilise des *messages* pour indiquer que des événements spécifiques se produisent dans l'animation, tels qu'un clic sur une image-objet, l'enfoncement des touches du clavier, le démarrage de l'animation, l'entrée ou la sortie de la tête de lecture dans une image ou encore le renvoi d'une valeur spécifique par un script.

L'ordre dans lequel les messages sont envoyés aux objets de l'animation est le suivant :

1 Les messages sont d'abord envoyés aux comportements associés aux images-objets affectées par l'événement. Si plusieurs comportements sont associés à une image-objet, ceux-ci répondent au message dans l'ordre dans lequel ils ont été associés à cette image-objet.

2 Les messages sont ensuite envoyés au script de l'acteur de l'image-objet.

3 Les messages sont ensuite envoyés aux comportements associés à l'image actuelle.

4 Enfin, les messages sont envoyés aux scripts de l'animation.

Bien qu'il vous soit possible de définir le nom de vos messages, la plupart des événements communs survenant dans une animation possèdent des noms de message prédéfinis.

Pour plus d'informations sur les messages intégrés à Director, consultez « Evénements et messages », page 153.

Définition de messages personnalisés

En plus d'utiliser les noms de messages prédéfinis, vous pouvez définir vos propres messages et les noms des gestionnaires correspondants. Un message personnalisé peut appeler un autre script, un autre gestionnaire ou le gestionnaire de l'instruction même. Lorsque le gestionnaire appelé a terminé son exécution, l'exécution du gestionnaire qui l'a appelé reprend.

Les noms de messages et de gestionnaires personnalisés doivent répondre aux critères suivants :

- Ils doivent débuter par une lettre.
- Ils doivent inclure uniquement des caractères alphanumériques (pas de caractères spéciaux ou de ponctuation).
- Ils doivent être composés d'un ou de plusieurs mots reliés par un trait de soulignement (les espaces ne sont pas autorisés).
- Ils ne peuvent pas être identiques au nom d'un élément Lingo ou JavaScript prédéfini.

L'utilisation de mots-clés Lingo ou JavaScript prédéfinis pour les noms de messages et de gestionnaires peut prêter à confusion. Bien qu'il soit possible de remplacer ou d'augmenter explicitement la fonctionnalité d'un élément Lingo ou JavaScript en l'utilisant comme nom de message ou de gestionnaire, seuls les utilisateurs chevronnés devraient effectuer cette opération.

Lorsque vous utilisez plusieurs gestionnaires remplissant des fonctions similaires, donnez-leur des noms commençant de la même manière afin de les regrouper dans les listes alphabétiques, comme par exemple la liste qui apparaît lorsque vous sélectionnez l'option Edition > Rechercher > Gestionnaire.

Gestionnaires

Un *gestionnaire* est un ensemble d'instructions placées dans un script et exécutées en réponse à un événement déterminé et à un message subséquent. Bien que Director contienne des événements et des messages intégrés, vous devez créer vos propres gestionnaires pour chaque paire d'événements/messages que vous voulez gérer.

Stratégie de positionnement des gestionnaires

Vous pouvez placer des gestionnaires dans n'importe quel type de script, un script pouvant contenir plusieurs gestionnaires. Il est recommandé de regrouper les gestionnaires apparentés au même endroit afin d'en simplifier la gestion.

Les recommandations suivantes s'appliquent aux situations les plus actuelles :

- Pour associer un gestionnaire à une image-objet déterminée ou pour exécuter un gestionnaire en réponse à une action dans une image-objet précise, placez le gestionnaire dans un comportement affecté à l'image-objet.
- Pour définir un gestionnaire disponible à tout moment lorsque l'animation se trouve dans une image déterminée, placez-le dans un comportement affecté à l'image.

  Par exemple, pour qu'un gestionnaire réponde à un clic de la souris lorsque la tête de lecture est dans une image, quel que soit l'endroit où se produit le clic, placez un gestionnaire `mouseDown` ou `mouseUp` dans le comportement de l'image plutôt que dans un comportement d'image-objet.
- Pour définir un gestionnaire exécuté en réponse à des messages d'événements se produisant n'importe où dans l'animation, placez-le dans un script d'animation.
- Pour définir un gestionnaire exécuté en réponse à un événement affectant un acteur, quelles que soient les images-objets utilisant cet acteur, placez-le dans un script d'acteur.

Identification du moment auquel les gestionnaires reçoivent un message

Après avoir envoyé un message aux scripts, Director vérifie la présence de gestionnaires dans un ordre défini.

1 Director vérifie d'abord l'existence de gestionnaires dans l'objet à partir duquel le message a été envoyé. Si un gestionnaire est trouvé, le message est intercepté et le script du gestionnaire est exécuté.

2 S'il ne trouve pas de gestionnaire, Director vérifie, dans un ordre croissant, les scripts d'animation associés à un acteur et pouvant contenir un gestionnaire lié au message. Si un gestionnaire est trouvé, le message est intercepté et le script du gestionnaire est exécuté.

3 Si aucun gestionnaire n'a été trouvé, Director vérifie si un script d'image contient un gestionnaire pour le message. Si un gestionnaire est trouvé, le message est intercepté et le script du gestionnaire est exécuté.

4 S'il ne trouve pas de gestionnaire, Director vérifie, dans un ordre croissant, les scripts associés aux images-objets et pouvant contenir un gestionnaire lié au message. Si un gestionnaire est trouvé, le message est intercepté et le script du gestionnaire est exécuté.

Le message n'est pas automatiquement transmis aux emplacements restants après son interception par le gestionnaire. Cependant, dans Lingo, vous pouvez utiliser la méthode `pass()` pour ignorer cette règle par défaut et transmettre le message à d'autres objets.

Si le gestionnaire recherché n'est pas trouvé après l'envoi du message à tous les emplacements possibles, Director ignore le message.

L'ordre exact dans lequel Director envoie un message aux objets dépend du message même. Pour plus d'informations sur l'ordre des objets auxquels Director envoie des messages spécifiques, consultez l'entrée correspondant à chaque message dans « Evénements et messages », page 153.

Utilisation de paramètres pour transmettre des valeurs à un gestionnaire

L'utilisation de paramètres comme valeurs vous permet de transmettre à un gestionnaire les valeurs exactes nécessaires au moment voulu, sans tenir compte de l'endroit de l'animation ni du moment auquel vous appelez ce gestionnaire. Les paramètres peuvent être facultatifs ou obligatoires, selon le cas.

Créer les paramètres d'un gestionnaire

- Dans Lingo, placez les paramètres après le nom du gestionnaire.
- Dans la syntaxe JavaScript, mettez les paramètres entre parenthèses, puis placez-les après le nom du gestionnaire.

Les paramètres multiples doivent être séparés par des virgules.

Lorsque vous appelez un gestionnaire, vous devez fournir des valeurs spécifiques pour les paramètres qu'il utilise. Vous pouvez utiliser n'importe quel type de valeur, tel qu'un nombre, une variable à laquelle une valeur est affectée ou une chaîne. Les valeurs de l'instruction d'appel doivent suivre le même ordre que dans les paramètres du gestionnaire et être entourées de parenthèses.

Dans l'exemple suivant, la variable `mySum` appelle la méthode `addThem` qui reçoit les deux valeurs 2 et 4. Le gestionnaire `addThem` remplace les repères d'emplacement de paramètres `a` et `b` par les deux valeurs qu'il a reçues, stocke le résultat dans la variable locale `c`, puis utilise le mot-clé `return` pour renvoyer le résultat à la méthode originale qui est ensuite attribuée à `mySum`.

Etant donné que 2 figure en premier dans la liste des paramètres, il remplace `a` dans le gestionnaire. De même, étant donné que 4 est le second dans la liste des paramètres, il remplace `b` dans le gestionnaire.

```
-- Lingo syntax
mySum = addThem(2, 4) -- calling statement

on addThem a, b -- handler
    c = a + b
    return c -- returns the result to the calling statement
end

// JavaScript syntax
var mySum = addThem(2, 4); // calling statement

function addThem(a, b) { // handler
    c = a + b;
    return c; // returns the result to the calling statement
}
```

Dans Lingo, lorsque vous appelez une méthode personnalisée à partir d'un objet, une référence à l'objet Script de la mémoire est toujours transmise en tant que premier paramètre implicite au gestionnaire de la méthode. Cela signifie que vous devez prendre en charge la référence de l'objet Script dans votre gestionnaire.

Par exemple, imaginez que vous ayez rédigé une méthode d'image-objet personnalisée appelée `jump()` dont le paramètre est un nombre entier simple et que vous l'ayez placée dans un comportement. Lorsque vous appelez `jump()` d'une référence d'un objet Image-objet, le gestionnaire doit également inclure un paramètre représentant la référence de l'objet Script, et non pas uniquement le nombre entier. Dans ce cas, le paramètre désigné est représenté par le mot-clé `me`, mais tout autre terme fonctionne également.

```
-- Lingo syntax
myHeight = sprite(2).jump(5)

on jump(me,a)
    return a + 15 -- this handler works correctly, and returns 20
end
```

```
on jump(a)
    return a + 15 -- this handler does not work correctly, and returns 0
end
```

Vous pouvez également utiliser des expressions comme valeurs. Par exemple, l'instruction suivante utilise `3+6` pour remplacer `a` et `8>2` (ou `1`, représentant `TRUE`) pour remplacer `b` et renvoie `10` :

```
-- Lingo syntax
mySum = addThem(3+6, 8>2)
```

Dans Lingo, tous les gestionnaires commencent par le mot `on`, suivi du message auquel ils doivent répondre. La dernière ligne du gestionnaire est le mot `end`. Vous pouvez répéter le nom du gestionnaire après `end`, ce qui n'est pas obligatoire.

Dans la syntaxe JavaScript, tous les gestionnaires commencent par le mot `function`, suivi du message auquel ils doivent répondre. Les instructions comprenant le gestionnaire sont mises entre crochets et sont toutes des fonctions de la syntaxe JavaScript.

Renvoi de résultats avec les gestionnaires

Il est souvent utile qu'un gestionnaire vous indique la présence d'une condition ou le résultat d'une action spécifique.

Renvoyer des résultats avec un gestionnaire

❖ Utilisez le mot-clé `return` pour qu'un gestionnaire retourne la présence d'une condition ou le résultat d'une action. Par exemple, le gestionnaire `findColor` suivant retourne la couleur de l'image-objet 1 :

```
-- Lingo syntax
on findColor
    return sprite(1).foreColor
end

// JavaScript syntax
function findColor() {
    return(sprite(1).foreColor);
}
```

Vous pouvez également utiliser le mot-clé `return` tout seul pour quitter le gestionnaire actuel et ne renvoyer aucune valeur. Par exemple, le gestionnaire `jump` suivant ne renvoie rien si le paramètre `aVal` est égal à 5 ; autrement, il renvoie une valeur :

```
-- Lingo syntax
on jump(aVal)
    if aVal = 5 then return

    aVal = aVal + 10
    return aVal
end
// JavaScript syntax
function jump(aVal) {
    if(aVal == 5) {
        return;
    }
    else {
        aVal = aVal + 10;
        return(aVal);
    }
}
```

Lorsque vous définissez un gestionnaire renvoyant un résultat, vous devez le faire suivre de parenthèses quand vous l'appelez à partir d'un autre gestionnaire. Par exemple, l'instruction `put(findColor())` appelle le gestionnaire `on findColor`, puis affiche le résultat dans la fenêtre Messages.

Listes linéaires et listes de propriétés

Dans vos scripts, vous pouvez choisir de faire le suivi et la mise à jour de listes de données, telles qu'une série de noms ou les valeurs attribuées à un ensemble de variables. Lingo et la syntaxe JavaScript ont accès aux listes linéaires et aux listes de propriétés. Dans une liste linéaire, chaque élément est une valeur unique. Dans une liste de propriétés, chaque élément contient deux valeurs ; la première est un nom de propriété, la seconde est la valeur associée à cette propriété.

Etant donné que Lingo et la syntaxe JavaScript ont tous deux accès aux listes linéaires et aux listes de propriétés, il est recommandé d'utiliser les listes linéaires et les listes de propriétés si les valeurs de votre code sont partagées par les scripts Lingo et JavaScript.

Si certaines valeurs de votre code sont utilisées uniquement dans les scripts de la syntaxe JavaScript, nous vous recommandons d'utiliser des objets Tableau de JavaScript avec les listes de données. Pour plus d'informations sur l'utilisation des tableaux, consultez « Tableaux de la syntaxe JavaScript », page 40.

Création de listes linéaires

Pour créer une liste linéaire, effectuez l'une des opérations suivantes :

- Dans Lingo, utilisez la fonction de haut niveau `list()` ou l'opérateur de liste (`[ ]`) et séparez les éléments de la liste par des virgules.
- Dans la syntaxe JavaScript, utilisez la fonction de haut niveau `list()` et séparez les éléments de la liste par des virgules.

L'index d'une liste linéaire commence toujours par 1.

Lorsque vous utilisez la fonction de haut niveau `list()`, vous spécifiez que les éléments de la liste sont des paramètres de la fonction. Cette fonction peut s'avérer pratique si vous utilisez un clavier ne possédant pas de touches de crochets.

Les instructions suivantes créent toutes une liste linéaire de trois noms et l'affectent à une variable :

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- using the Lingo list operator
workerList = list("Bruno", "Heather", "Carlos") -- using list()

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos"); // using list()
```

Vous pouvez également créer des listes linéaires vides. Les instructions suivantes créent des listes linéaires vides :

```
-- Lingo syntax
workerList = [] -- using the Lingo list operator
workerList = list() -- using list() with no parameters

// JavaScript syntax
var workerList = list(); // using list() with no parameters
```

Création de listes de propriétés

Pour créer une liste de propriétés, effectuez l'une des opérations suivantes :

• Dans Lingo, utilisez soit la fonction de haut niveau `propList()`, soit l'opérateur de liste (`[:]`). Lorsque vous utilisez l'opérateur de liste pour créer une liste de propriétés, vous pouvez utiliser soit le signe deux-points (:) pour désigner des éléments de nom/valeur et des virgules pour séparer les éléments de la liste, soit des virgules pour désigner des éléments de nom/valeur et séparer les éléments dans la liste.

• Dans la syntaxe JavaScript, utilisez la fonction de haut niveau `propList()` et insérez des virgules pour désigner des éléments de nom/valeur et séparer les éléments dans la liste.

Lorsque vous utilisez la fonction de haut niveau `propList()`, vous spécifiez que les éléments de la liste de propriétés sont des paramètres de la fonction. Cette fonction peut s'avérer pratique si vous utilisez un clavier ne possédant pas de touches de crochets.

Les propriétés peuvent apparaître plusieurs fois dans une liste de propriétés donnée.

Les instructions suivantes créent toutes une liste de quatre propriétés (`left`, `top`, `right` et `bottom`) et leurs valeurs correspondantes :

```
-- Lingo syntax
sprite1Loc = [#left:100, #top:150, #right:300, #bottom:350]
sprite1Loc = ["left",400, "top",550, "right",500, "bottom",750]
sprite1Loc = propList("left",400, "top",550, "right",500, "bottom",750)

// JavaScript syntax
var sprite1Loc = propList("left",400, "top",550, "right",500, "bottom",750);
```

Vous pouvez également créer des listes de propriétés vides. Les instructions suivantes créent des listes de propriétés vides :

```
-- Lingo syntax
sprite1Loc = [:] -- using the Lingo property list operator
sprite1Loc = propList() -- using propList() with no parameters

// JavaScript syntax
var sprite1Loc = propList(); // using propList() with no parameters
```

Définition et récupération d'éléments de listes

Vous pouvez définir et récupérer des éléments individuels dans une liste. La syntaxe utilisée est différente en fonction du type de liste.

Définition d'une valeur dans une liste linéaire

❖ Effectuez l'une des opérations suivantes :

• Utilisez l'opérateur égal à (=).

• Utilisez la méthode `setAt()`.

Les instructions suivantes montrent comment définir la liste linéaire `workerList` contenant une valeur, `Heather`, puis ajoute `Carlos` en tant que seconde valeur dans la liste.

```
-- Lingo syntax
workerList = ["Heather"] -- define a linear list
workerList[2] = "Carlos" -- set the second value using the equal operator
workerList.setAt(2, "Carlos") -- set the second value using setAt()
```

```
// JavaScript syntax
var workerList = list("Heather"); // define a linear list
workerList[2] = "Carlos"; // set the second value using the equal operator
workerList.setAt(2, "Carlos"); // set the second value using setAt()
```

Récupération d'une valeur dans une liste linéaire

1 Utilisez la variable de la liste suivie du numéro indiquant la position de la valeur dans la liste. Encadrez ce nombre de crochets.

2 Utilisez la méthode `getAt()`.

Les instructions suivantes créent la liste linéaire `workerList`, puis attribue la seconde valeur de la liste à la variable `name2` :

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- define a linear list
name2 = workerList[2] -- use bracketed access to retrieve "Heather"
name2 = workerList.getAt(2) -- use getAt() to retrieve "Heather"

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos");
var name2 = workerList[2] // use bracketed access to retrieve "Heather"
var name2 = workerList.getAt(2) // use getAt() to retrieve "Heather"
```

Définition d'une valeur dans une liste de propriétés

❖ Effectuez l'une des opérations suivantes :

- Utilisez l'opérateur égal à (=).
- Dans Lingo, utilisez la méthode `setaProp()`.
- Utilisez la syntaxe à point.

Les instructions Lingo suivantes utilisent l'opérateur égal à pour faire de `sushi` la nouvelle valeur associée à la propriété `Bruno` :

```
-- Lingo syntax
foodList = [:] -- define an empty property list
foodList[#Bruno] = "sushi" -- associate sushi with Bruno
```

Les instructions Lingo suivantes utilisent `setaprop()` pour faire de `sushi` la nouvelle valeur associée à la propriété `Bruno` :

```
-- Lingo syntax
foodList = [:] -- define an empty property list
foodList.setaProp(#Bruno, "sushi") -- use setaProp()

// JavaScript syntax
foodList = propList() -- define an empty property list
foodList.setaProp("Bruno", "sushi") -- use setaProp()
```

Les instructions suivantes utilisent la syntaxe à points pour définir la valeur associée à `Bruno` de `sushi` à `teriyaki`.

```
-- Lingo syntax
foodList = [#Bruno:"sushi"] -- define a property list
trace(foodList) -- displays [#Bruno: "sushi"]
foodList.Bruno = "teriyaki" -- use dot syntax to set the value of Bruno
trace(foodList) -- displays [#Bruno: "teriyaki"]

// JavaScript syntax
var foodList = propList("Bruno", "sushi"); // define a property list
trace(foodList); // displays ["Bruno": "sushi"]
```

```
foodList.Bruno = "teriyaki" // use dot syntax to set the value of Bruno
trace(foodList) -- displays [#Bruno: "teriyaki"]
```

Récupération d'une valeur dans une liste de propriétés

❖ Effectuez l'une des opérations suivantes :

• Utilisez la variable de la liste, suivie du nom de la propriété associée à cette valeur. Encadrez cette propriété de crochets.

• Utilisez la méthode `getaProp()` ou `getPropAt()`.

• Utilisez la syntaxe à point.

Les instructions suivantes utilisent un accès par crochets pour récupérer les valeurs associées aux propriétés `breakfast` et `lunch`.

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
trace(foodList[#breakfast]) -- displays "Waffles"
trace(foodList[#lunch]) -- displays "Tofu Burger"

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
trace(foodList["breakfast"]); // displays Waffles
trace(foodList["lunch"]); // displays Tofu Burger
```

Les instructions suivantes utilisent `getaProp()` pour récupérer la valeur associée à la propriété `breakfast` et `getPropAt()` pour récupérer la propriété à la seconde position d'index de la liste.

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
trace(foodList.getaProp(#breakfast)) -- displays "Waffles"
trace(foodList.getPropAt(2)) -- displays #lunch

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
trace(foodList.getaProp("breakfast")) // displays Waffles
trace(foodList.getPropAt(2)) // displays lunch
```

Les instructions suivantes utilisent la syntaxe à points pour accéder aux valeurs associées aux propriétés dans une liste de propriétés :

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
trace(foodList.breakfast) -- displays "Waffles"

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
trace(foodList.lunch); // displays Tofu Burger
```

Vérification d'éléments dans les listes

Vous pouvez déterminer les caractéristiques d'une liste et le nombre d'éléments qu'elle contient en utilisant les méthodes suivantes :

• Pour afficher le contenu d'une liste, utilisez la fonction `put()` ou `trace()` en passant la variable contenant la liste en tant que paramètre.

• Pour déterminer le nombre d'éléments contenus dans une liste, utilisez la méthode `count()` (Lingo seulement) ou la propriété `count`.

• Pour déterminer le type d'une liste, utilisez la méthode `ilk()`.

• Pour déterminer la valeur maximale d'une liste, utilisez la méthode `max()`.

• Pour déterminer la valeur minimale d'une liste, utilisez la fonction `min()`.

• Pour déterminer la position d'une propriété spécifique, utilisez la commande `findPos`, `findPosNear` ou `getOne` .

Les instructions suivantes utilisent `count()` et `count` pour afficher le nombre d'éléments d'une liste :

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- define a linear list
trace(workerList.count()) -- displays 3
trace(workerList.count) -- displays 3

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos"); // define a linear list
trace(workerList.count); // displays 3
```

Les instructions suivantes utilisent `ilk()` pour déterminer le type d'une liste :

```
-- Lingo syntax
x = ["1", "2", "3"]
trace(x.ilk()) // returns #list

// JavaScript syntax
var x = list("1", "2", "3");
trace(x.ilk()) // returns #list
```

Les instructions suivantes utilisent `max()` et `min()` pour déterminer les valeurs maximales and minimales d'une liste :

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- define a linear list
trace(workerList.max()) -- displays "Heather"
trace(workerList.min()) -- displays "Bruno"

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos"); // define a linear list
trace(workerList.max()); // displays Heather
trace(workerList.min()); // displays Bruno
```

Les instructions suivantes utilisent `findPos` pour obtenir la position d'index d'une propriété spécifiée dans une liste de propriétés :

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
trace(foodList.findPos(#lunch)) -- displays 2

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
trace(foodList.findPos("breakfast")); // displays 1
```

Ajout et suppression d'éléments dans des listes

Vous pouvez ajouter des éléments à une liste ou en supprimer à l'aide des méthodes suivantes :

- Pour ajouter un élément à la fin d'une liste, utilisez la méthode `append()`.
- Pour ajouter un élément au bon endroit dans une liste triée, utilisez la méthode `add()` ou `addProp()`.
- Pour ajouter un élément à un emplacement spécifique d'une liste linéaire, utilisez la méthode `addAt()`.
- Pour ajouter un élément à un emplacement spécifique d'une liste de propriétés, utilisez la méthode `addProp()`.
- Pour supprimer un élément d'une liste, utilisez la méthode `deleteAt()`, `deleteOne()` ou `deleteProp()`.
- Pour remplacer un élément d'une liste, utilisez le méthode `setAt()` ou `setaProp()`.

Les instructions suivantes utilisent `append()` pour ajouter un élément à la fin d'une liste.

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- define a linear list
workerList.append("David")
trace(workerList) -- displays ["Bruno", "Heather", "Carlos", "David"]

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos"); // define a linear list
workerList.append("David");
trace(workerList); // displays ["Bruno", "Heather", "Carlos", "David"]
```

Les instructions suivantes utilisent `addProp()` pour ajouter une propriété et une valeur associée à une liste de propriétés :

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
foodList.addProp(#dinner, "Spaghetti") -- adds [#dinner: "Spaghetti"]

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
foodList.addProp("dinner", "Spaghetti"); // adds ["dinner": "Spaghetti"]
```

Il n'est pas nécessaire d'éliminer explicitement les listes. En effet, elles sont automatiquement supprimées dès qu'aucune variable n'y fait plus référence. Les autres types d'objets doivent être retirés de manière explicite, en donnant aux variables qui y font référence la valeur `VOID` (Lingo) ou `null` (syntaxe JavaScript).

Copie de listes

L'affectation d'une liste à une variable, puis l'affectation de cette variable à une autre variable ne créent pas automatiquement une copie de cette liste. Par exemple, la première instruction ci-dessous crée une liste contenant les noms de deux continents et affecte la liste à la variable `landList`. La seconde instruction affecte la même liste à une nouvelle variable `continentList`. Dans la troisième instruction, l'ajout de l'élément `Australie` à `landList` ajoute automatiquement l'élément `Australia` à la liste `continentList`. Ceci se produit car les deux noms de variables font référence au même objet Liste en mémoire. Le même comportement se produit lorsque vous utilisez un tableau dans la syntaxe JavaScript.

```
-- Lingo syntax
landList = ["Asia", "Africa"]
continentList = landList
landList.add("Australia") -- this also adds "Australia" to continentList
```

Création d'une copie d'une liste indépendante d'une autre liste

❖ Utilisez la méthode `duplicate()`.

Par exemple, les instructions suivantes créent une liste puis font une copie indépendante de la liste :

```
-- Lingo syntax
oldList = ["a", "b", "c"]
newList = oldList.duplicate() -- makes an independent copy of oldList

// JavaScript syntax
var oldList = list("a", "b", "c");
var newList = oldList.duplicate(); // makes an independent copy of oldList
```

Une fois `newList` créée, la modification de `oldList` ou de `newList` n'a aucun effet sur l'autre liste.

Tri de listes

Les listes sont triées dans l'ordre alphanumérique, les nombres étant triés avant les chaînes. Les chaînes sont triées en fonction de la première lettre, quel que soit le nombre de caractères qu'elles contiennent. Les listes triées sont traitées un peu plus rapidement que les listes non triées.

Une liste linéaire est triée en fonction des valeurs de la liste. Une liste de propriétés est triée en fonction des noms de propriétés de la liste ou du tableau.

Une fois les valeurs d'une liste linéaire ou d'une liste de propriétés triées, elles restent triées même si des valeurs sont ajoutées ou supprimées des listes.

Trier une liste

❖ Utilisez la méthode `sort()`.

Par exemple, les instructions suivantes trient une liste alphabétique non triée :

```
-- Lingo syntax
oldList = ["d", "a", "c", "b"]
oldList.sort() -- results in ["a", "b", "c", "d"]

// JavaScript syntax
var oldList = list("d", "a", "c", "b");
oldList.sort(); // results in ["a", "b", "c", "d"]
```

Création de listes multidimensionnelles

Vous pouvez également créer des listes multidimensionnelles qui vous permettent d'utiliser les valeurs de plusieurs listes à la fois.

Dans l'exemple suivant, les deux premières instructions créent les listes linéaires distinctes `list1` et `list2`. La troisième instruction crée une liste multidimensionnelle et l'affecte à `mdList`. Dans une liste multidimensionnelle, les instructions quatre et cinq utilisent des crochets pour accéder aux valeurs de la liste ; le premier crochet donne accès à une liste donnée, le second crochet donnant accès à la valeur se trouvant à la position d'index spécifiée dans la liste.

```
-- Lingo syntax
list1 = list(5,10)
list2 = list(15,20)
mdList = list(list1, list2)
trace(mdList[1][2]) -- displays 10
trace(mdList[2][1]) -- displays 15
```

```
// JavaScript syntax
var list1 = list(5,10);
var list2 = list(15,20);
var mdList = list(list1, list2);
trace(mdList[1][2]); // displays 10
trace(mdList[2][1]); // displays 15
```

Tableaux de la syntaxe JavaScript

Les tableaux de la syntaxe JavaScript sont similaires aux listes linéaires de Lingo, chaque élément d'un tableau étant une valeur unique. L'une des principales différences entre les tableaux de la syntaxe JavaScript et les listes linéaires de Lingo est que l'index d'un tableau commence toujours par 0.

Vous pouvez créer un tableau de la syntaxe JavaScript en utilisant l'objet Tableau. Vous pouvez utiliser soit des crochets (`[ ]`) soit le constructeur `Array` pour créer un tableau. Les deux instructions suivantes créent un tableau de deux valeurs :

```
// JavaScript syntax
var myArray = [10, 15]; // using square brackets
var myArray = new Array(10, 15); // using the Array constructor
```

Vous pouvez également créer des tableaux vides. Les deux instructions suivantes créent un tableau vide :

```
// JavaScript syntax
var myArray = [];
var myArray = new Array();
```

Remarque : *le manuel* Référence de scripting de Director *ne constitue pas une référence complète pour les objets Tableau de la syntaxe JavaScript. Pour plus d'informations sur l'utilisation des objets Tableau, reportez-vous à l'une des nombreuses documentations d'autres éditeurs sur le sujet.*

Vérification d'éléments dans les tableaux

Vous pouvez déterminer les caractéristiques d'un tableau et le nombre d'éléments qui y sont contenus en utilisant les méthodes suivantes :

- Pour afficher le contenu d'une liste, utilisez la fonction `put()` ou `trace()`, en passant la variable contenant la liste en tant que paramètre.
- Pour déterminer le nombre d'éléments contenus dans un tableau, utilisez la propriété `length` de l'objet Tableau.
- Pour déterminer le type d'un tableau, utilisez la propriété `constructor`.

  L'exemple suivant montre comment déterminer le nombre d'éléments contenus dans un tableau en utilisant la propriété `length` puis en renvoyant le type d'objet à l'aide de la propriété `constructor` :

  ```
  // JavaScript syntax
  var x = ["1", "2", "3"];
  trace(x.length) // displays 3
  trace(x.constructor == Array) // displays true
  ```

Ajout et suppression d'éléments dans les tableaux

Vous pouvez ajouter des éléments dans un tableau ou en supprimer à l'aide des méthodes suivantes :

- Pour ajouter un élément à la fin d'un tableau, utilisez la méthode `push()` de l'objet Tableau.
- Pour placer un élément au bon endroit dans un tableau trié, utilisez la méthode `splice()` de l'objet Tableau.

- Pour placer un élément à un endroit précis d'un tableau, utilisez la méthode `splice()` de l'objet Tableau.
- Pour supprimer un élément d'un tableau, utilisez la méthode `splice()` de l'objet Tableau.
- Pour remplacer un élément dans un tableau, utilisez la méthode `splice()` de l'objet Tableau.

L'exemple suivant montre comment utiliser la méthode `splice()` de l'objet Tableau pour ajouter des éléments à un tableau, les supprimer du tableau ou les remplacer :

```
// JavaScript syntax
var myArray = new Array("1", "2");
trace(myArray); displays 1,2

myArray.push("5"); // adds the value "5" to the end of myArray
trace(myArray); // displays 1,2,5

myArray.splice(3, 0, "4"); // adds the value "4" after the value "5"
trace(myArray); // displays 1,2,5,4

myArray.sort(); // sort myArray
trace(myArray); // displays 1,2,4,5

myArray.splice(2, 0, "3");
trace(myArray); // displays 1,2,3,4,5

myArray.splice(3, 2); // delete two values at index positions 3 and 4
trace(myArray); // displays 1,2,3

myArray.splice(2, 1, "7"); // replaces one value at index position 2 with "7"
trace(myArray); displays 1,2,7
```

Copie de tableaux

L'affectation d'un tableau à une variable, puis l'affectation de cette variable à une autre variable ne créent pas une copie distincte de ce tableau.

Par exemple, la première instruction ci-dessous crée un tableau contenant les noms de deux continents, puis affecte le tableau à la variable `landList`. La seconde instruction affecte la même liste à une nouvelle variable `continentList`. Dans la troisième instruction, l'ajout de l'élément `Australia` à `landList` ajoute automatiquement l'élément `Australia` au tableau `continentList`. Ceci se produit car les deux noms de variables font référence au même objet Tableau en mémoire.

```
// JavaScript syntax
var landArray = new Array("Asia", "Africa");
var continentArray = landArray;
landArray.push("Australia"); // this also adds "Australia" to continentList
```

Création d'une copie d'un tableau indépendante d'un autre tableau

❖ Utilisez la méthode `slice()` de l'objet Tableau.

Par exemple, les instructions suivantes créent un tableau puis utilisent `slice()` pour créer une copie indépendante du tableau :

```
// JavaScript syntax
var oldArray = ["a", "b", "c"];
var newArray = oldArray.slice(); // makes an independent copy of oldArray
```

Une fois `newArray` créé, la modification de `oldArray` ou de `newArray` n'a aucun effet sur l'autre tableau.

Tri des tableaux

Les tableaux sont triés dans l'ordre alphanumérique, les nombres étant triés avant les chaînes. Les chaînes sont triées en fonction de la première lettre, quel que soit le nombre de caractères qu'elles contiennent.

Trier un tableau

❖ Utilisez la méthode `sort()` de l'objet Tableau.

Les instructions suivantes trient un tableau alphabétique non trié :

```
// JavaScript syntax
var oldArray = ["d", "a", "c", "b"];
oldArray.sort(); // results in a, b, c, d
```

Les instructions suivantes trient un tableau alphanumérique non trié :

```
// JavaScript syntax
var oldArray = [6, "f", 3, "b"];
oldArray.sort(); // results in 3, 6, b, f
```

Le tri d'un tableau renvoie un nouveau tableau trié.

Création de tableaux multidimensionnels

Vous pouvez également créer des tableaux multidimensionnels qui vous permettent d'utiliser les valeurs de plusieurs tableaux à la fois.

Dans l'exemple suivant, les deux premières instructions créent les tableaux distincts `array1` et `array2`. La troisième instruction crée un tableau multidimensionnel et l'affecte à `mdArray`. Dans un tableau multidimensionnel, les instructions quatre et cinq utilisent des crochets pour accéder aux valeurs du tableau ; le premier crochet donne accès à un tableau donné, le second donnant accès à la valeur se trouvant à la position d'index spécifiée dans le tableau.

```
// JavaScript syntax
var array1 = new Array(5,10);
var array2 = [15,20];
var mdArray = new Array(array1, array2);
trace(mdArray[0][1]); // displays 10
trace(mdArray[1][0]); // displays 15
```

Chapitre 3 : Rédaction de scripts dans Director

Les scripts de Director® 11 prennent en charge toutes sortes de fonctionnalités pour des animations qui seraient impossibles à réaliser sans eux. Au fur et à mesure de la rédaction des scripts, vous vous rendrez probablement compte que des scripts de plus en plus compliqués sont nécessaires pour prendre en charge l'interactivité complexe de vos animations Director. Nous vous présentons ici les concepts et techniques de programmation intermédiaire et avancée, notamment des informations concernant la programmation orientée objet dans Director.

Si vous n'avez jamais créé de scripts dans Director, assurez-vous de lire la section « Principes de base de la programmation dans Director », page 4 en plus des rubriques suivantes.

Choix entre Lingo et la syntaxe JavaScript

Lingo et JavaScript donnent accès aux mêmes objets, événements et API de programmation. Par conséquent, le langage que vous choisissez d'utiliser pour rédiger vos scripts a peu d'importance. Il vous suffit de décider du langage que vous connaissez le mieux et avec lequel vous vous sentez le plus à l'aise.

Voici quelques idées générales concernant le fonctionnement des langages de programmation avec un modèle d'objet et d'événement donné dans Director :

- En général, un langage de programmation donné tel que Lingo ou JavaScript entoure un modèle d'objet et d'événement donné et donne accès à ces objets et événements.
- JavaScript est une implémentation d'ECMAScript qui entoure le modèle d'objet et d'événement d'un navigateur Web et donne accès aux objets et événements du navigateur.
- ActionScript est une implémentation d'ECMAScript qui entoure le modèle d'objet et d'événement d'Adobe® Flash® et donne accès aux objets et événements de Flash.
- L'implémentation de la syntaxe JavaScript de Director est une implémentation de ECMAScript qui entoure le modèle d'objet et d'événement de Director et donne accès aux objets et événements de Director.
- Lingo est une syntaxe personnalisée qui entoure le modèle d'objet et d'événement de Director et donne accès aux objets et événements de Director.

Lingo et JavaScript représentent les deux langages que vous pouvez utiliser pour accéder au même modèle d'objet et d'événement de Director. Les scripts rédigés dans l'un des langages ont les mêmes capacités que ceux rédigés dans l'autre.

Par conséquent, une fois que vous savez comment accéder aux API de programmation dans un langage, vous savez en gros comment y accéder dans l'autre langage. Par exemple, le code de la syntaxe JavaScript peut accéder à des types de données Lingo tels que les symboles, les listes linéaires, les listes de propriétés etc., créer et invoquer des scripts et comportements parents Lingo, créer et invoquer des Xtras et utiliser des expressions de sous-chaînes Lingo. De plus, les scripts de la syntaxe JavaScript et Lingo peuvent être utilisés au sein d'une seule animation. Cependant, un acteur script ne peut contenir qu'un type de syntaxe.

Il existe deux différences principales entre Lingo et la syntaxe JavaScript :

• Chaque langage comprend des conventions de syntaxe et de terminologie qui lui sont propres. Par exemple, la syntaxe d'un gestionnaire d'événement dans Lingo est différente de celle de JavaScript :

```
-- Lingo syntax
on mouseDown
    ...
end

// JavaScript syntax
function mouseDown() {
    ...
}
```

Pour plus d'informations sur les conventions de syntaxe et de terminologie utilisées pour chaque langage, consultez « Terminologie de programmation », page 5 et « Syntaxe de programmation », page 7.

• L'accès à certaines API de programmation varie légèrement d'un langage à l'autre. Par exemple, vous utiliseriez des constructions différentes pour accéder au deuxième mot du premier paragraphe d'un acteur texte :

```
-- Lingo syntax
member("News Items").paragraph[1].word[2]

// JavaScript syntax
member("News Items").getPropRef("paragraph", 1).getProp("word", 2);
```

Programmation avec la syntaxe à point

Que vous rédigiez des scripts avec Lingo ou la syntaxe JavaScript, vous le faites à l'aide de la syntaxe à point. Vous utilisez la syntaxe à point pour accéder aux propriétés ou méthodes liées à un objet. Une instruction utilisant la syntaxe à point doit commencer par une référence à un objet, suivi d'un point, puis par le nom de la propriété, de la méthode ou de la sous-chaîne que vous souhaitez spécifier. Chaque point d'une instruction représente un mouvement allant d'un niveau plus élevé et plus général de la hiérarchie des objets à un niveau inférieur et plus précis.

Par exemple, l'instruction suivante crée d'abord une référence à la bibliothèque de distribution appelée « Reportages » puis utilise la syntaxe à point pour accéder au nombre d'acteurs compris dans cette bibliothèque.

Pour identifier les sous-chaînes, les termes suivant le point servent à indiquer des éléments spécifiques du texte. Par exemple, la première instruction ci-dessous concerne le premier paragraphe de l'acteur texte appelé « Nouveautés ». La seconde instruction concerne le deuxième mot du premier paragraphe.

```
-- Lingo syntax
member("News Items").paragraph[1]
member("News Items").paragraph[1].word[2]

// JavaScript syntax
member("News Items").getPropRef("paragraph", 1);
member("News Items").getPropRef("paragraph", 1).getProp("word", 2);
```

Pour certains objets gérant l'accès des propriétés en cascade à des données ou à un type d'acteur précis, comme indiqué dans les deux instructions précédentes, l'accès aux propriétés n'est pas pris en charge par la syntaxe JavaScript normale. Vous devez donc utiliser les méthodes `getPropRef()` et `getProp()` pour accéder aux propriétés en cascade dans la syntaxe JavaScript.

Voici quelques remarques concernant cette exception dans la syntaxe JavaScript :

- Vous devez appliquer cette technique aux objets 3D, aux acteurs texte, aux acteurs champ et aux Xtras XML Parser auxquels vous accédez à l'aide de la syntaxe JavaScript.
- Vous devez utiliser la méthode `getPropRef()` pour stocker une référence à l'un des objets précédemment mentionnés ou à ses propriétés à l'aide de la syntaxe JavaScript.
- Vous devez utiliser la méthode `getProp()` pour récupérer une valeur de propriété de l'un des objets précédemment mentionnés ou de ses propriétés à l'aide la syntaxe JavaScript.
- Vous devez accéder aux objets 3D et aux propriétés en utilisant leurs noms complets dans la syntaxe JavaScript. Par exemple, dans Lingo, la propriété `shader` peut être utilisée comme un raccourci de la propriété `shaderList[1]`. Cependant, dans la syntaxe JavaScript, la propriété `shaderList[1]` doit être utilisée tout le temps.

Introduction aux objets de Director

Les *objets* sont généralement des groupes logiques de données nommées pouvant également contenir des méthodes qui agissent sur ces données. Dans cette version de Director, les API de programmation ont été groupées en objets et sont accessibles à travers ces objets. Chaque objet donne accès à un ensemble précis de données nommées et de types de fonctionnalités. Par exemple, l'objet Image-objet donne accès aux données et fonctionnalités d'une image-objet, l'objet Animation donne accès aux données et fonctionnalités d'une animation, etc.

Les objets utilisés dans Director appartiennent aux quatre catégories suivantes. Selon la fonctionnalité que vous voulez ajouter et la partie d'une animation à laquelle vous l'ajoutez, vous utiliserez les objets d'une ou plusieurs de ces catégories :

- Objets principaux
- Types de médias
- Objets de programmation
- Objets 3D

Objets principaux

Cette catégorie d'objets donne accès aux principales fonctionnalités et fonctions de Director, telles que le moteur du lecteur de Director, les fenêtres des animations, les images-objets, les sons, etc. Ces objets représentent la couche de base à travers laquelle on accède à toutes les API et autres catégories d'objets.

Il existe également un groupe de méthodes et propriétés de haut niveau qui vous permettent d'accéder directement à tous les objets principaux au lieu d'avoir à passer par la hiérarchie des objets.

Pour plus d'informations sur les objets principaux et leurs API, consultez « Objets principaux de Director », page 97.

Types de médias

Cette catégorie d'objets donne accès aux fonctionnalités des divers types de médias (RealMedia, DVD, GIF animé, etc.) ajoutés aux animations en tant qu'acteurs.

Les médias ne sont pas strictement des objets mais plutôt des acteurs qui se rapportent à un type de média précis. Lorsqu'un type de média est ajouté à une animation en tant qu'acteur, il hérite de la fonctionnalité de l'objet Acteur principal et étend l'objet Acteur en fournissant des fonctionnalités supplémentaires qui ne sont disponibles que pour le type de média spécifié. Par exemple, un acteur RealMedia a accès aux méthodes et propriétés de l'objet Acteur, et possède également d'autres méthodes et propriétés propres à RealMedia. Les autres types de médias affichent tous ce comportement.

Pour plus d'informations sur les types de médias disponibles et leurs API, consultez « Types de médias », page 114.

Objets de programmation

Cette catégorie d'objets, également connus sous le nom de *Xtras*, donne accès aux fonctionnalités des composants logiciels tels que XML Parser, Fileio, SpeechXtra, etc. qui sont installés dans Director et étendent les fonctionnalités principales de Director. Les Xtras existants fournissent certaines fonctions telles que l'importation de filtres et la connexion à Internet. De plus, si vous connaissez le langage de programmation C, vous pouvez créer vos propres Xtras.

Pour plus d'informations sur les objets de programmation disponibles et leurs API, consultez « Objets de programmation », page 132.

Objets 3D

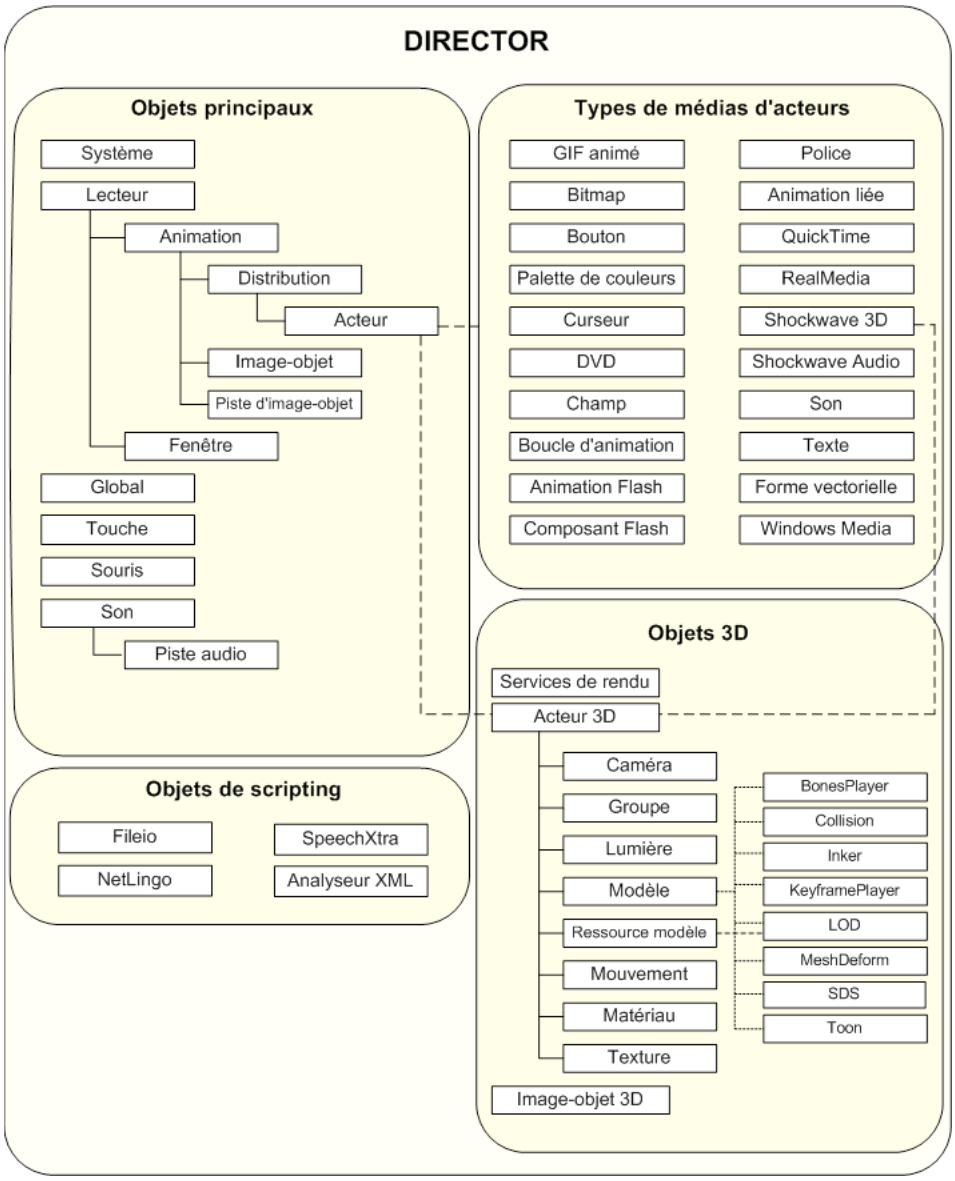
Cette catégorie d'objets donne accès aux fonctionnalités d'acteurs et de texte utilisées pour créer des animations 3D.

Pour plus d'informations sur les animations 3D, consultez les rubriques du document Utilisation de Director dans l'Aide de Director.

Pour plus d'informations sur les objets 3D disponibles et leurs API, consultez « Objets 3D », page 136.

Diagrammes de modèles d'objets

Les schémas suivants illustrent les principales relations de haut niveau entre les groupes d'objets et leurs hiérarchies dans Director. Pour plus d'informations sur la création d'objets, les propriétés et méthodes et les autres API, consultez les rubriques consacrées aux API correspondantes.



Fonctions et propriétés de haut niveau

Il existe un groupe de fonctions et propriétés de haut niveau qui vous permettent d'accéder directement aux objets principaux et aux fonctionnalités de Director. Vous utiliserez probablement souvent plusieurs de ces fonctions et propriétés lors de la création de références aux objets principaux, aux nouvelles images, aux listes, etc. Par exemple, la propriété de haut niveau `_movie` fait directement référence à l'objet principal Animation et la fonction de haut niveau `list()` crée une liste linéaire.

Les tableaux suivants dressent la liste des fonctions et propriétés de haut niveau.

Fonctions de haut niveau	
castLib()	randomVector()
channel() (niveau supérieur)	rect()
color()	script()
date() (formats), date() (système)	showLocals()
image()	sound()
isBusy()	sprite()
list()	symbol()
member()	timeout()
point()	trace()
propList()	vector()
put()	window()
random()	xtra()

Propriétés de haut niveau	
_global	_player
_key	_sound
_mouse	_system
_movie	

Introduction à la programmation orientée objet dans Director

En utilisant Lingo ou la syntaxe JavaScript, vous pouvez appliquer les principes de la programmation orientée objet à vos scripts. Cette opération facilite en général la programmation puisqu'elle vous permet de rédiger moins de codes et d'utiliser une logique plus simple pour effectuer des tâches, tout en améliorant la réutilisation et la modularité de votre code.

En fonction du langage de programmation utilisé, vous appliquez ces principes à l'aide de deux types de paradigmes :

- Dans Lingo, vous utilisez les scripts parents, les scripts ancêtres et les objets enfants pour simuler la programmation orientée objet.
- Dans la syntaxe JavaScript, vous utilisez les techniques de programmation orientée objet de style JavaScript standard pour créer des classes et des sous-classes.

Chaque paradigme vous permet d'appliquer les avantages apportés par la programmation orientée objet à vos scripts. Ainsi, le langage de programmation que vous choisissez d'utiliser a peu d'importance. Il vous suffit tout simplement d'appliquer les principes différemment.

Etant donné que chaque langage de programmation utilise un paradigme différent pour appliquer les principes orientés objet, les techniques valables pour un langage ne le sont pas pour l'autre. Il vous suffit donc de consulter le contenu qui s'applique au langage de programmation que vous utilisez :

- Pour plus d'informations sur la simulation de la programmation orientée objet dans Lingo, consultez « Programmation orientée objet avec la syntaxe Lingo », page 49.
- Pour plus d'informations sur la programmation orientée objet dans la syntaxe JavaScript, consultez « Programmation orientée objet avec la syntaxe JavaScript », page 58.

Programmation orientée objet avec la syntaxe Lingo

Dans Lingo, les scripts parents offrent les avantages de la programmation orientée objet. Vous pouvez utiliser des scripts parents pour générer des objets scripts qui ont une réponse et un comportement similaires tout en fonctionnant indépendamment les uns des autres.

Vous pouvez créer plusieurs copies (ou instances) d'un script parent à l'aide de Lingo. Chaque instance d'un script parent est un objet enfant. Vous pouvez créer des objets enfants sur demande au fur et à mesure de la lecture de l'animation. Director ne limite pas le nombre d'objets enfants que vous pouvez créer depuis un même script parent. Vous pouvez créer autant d'objets enfants que la mémoire de l'ordinateur peut en supporter.

Director peut créer plusieurs objets enfants depuis un même script parent de la même façon qu'il peut créer plusieurs instances d'un comportement pour différentes images-objets. Un script parent pourrait être assimilé à un modèle et un objet enfant à une implémentation du modèle parent.

Cette section sur les scripts parents et les objets enfants de Lingo présente les concepts de base concernant la rédaction des scripts parents, ainsi que la création et l'utilisation des objets enfants, et propose des exemples de script. Vous n'y trouverez pas les concepts de la programmation orientée objet, mais devrez cependant en comprendre les principes afin d'utiliser efficacement les scripts parents et les objets enfants. Vous trouverez dans le commerce de nombreux ouvrages présentant les notions de base de la programmation orientée objet.

Similarité avec les autres langages orientés objet

Si vous connaissez déjà un langage de programmation orienté objet tel que Java ou C++, vous comprenez probablement ces concepts, même s'il est possible que vous les connaissiez sous d'autres noms.

Les termes par lesquels Director décrit les scripts parents et les objets enfants correspondent aux termes couramment utilisés dans la programmation orientée objet :

Les **scripts parents** de Director correspondent aux classes dans la programmation orientée objet.

Les **objets enfants** de Director correspondent aux instances dans la programmation orientée objet.

Les **variables de propriétés** de Director correspondent aux variables d'instances ou d'acteurs dans la programmation orientée objet.

Les **gestionnaires** de Director correspondent aux méthodes dans la programmation orientée objet.

Les **scripts ancêtres** de Director correspondent à la superclasse ou classe de base dans la programmation orientée objet.

Scripts parents et objets enfants : notions de base

Dans Lingo, un *script parent* contient un ensemble de gestionnaires et de propriétés définissant un objet enfant ; il ne constitue pas un objet enfant en lui-même. Un *objet enfant* est une instance autonome et indépendante d'un script parent. Les objets enfants du même parent possèdent des gestionnaires et des propriétés identiques, les objets enfants d'un même groupe pouvant donc avoir des réponses similaires aux événements et aux messages.

Les scripts parents sont généralement utilisés pour construire des objets enfants facilitant l'organisation de l'animation. Ces objets enfants sont particulièrement utiles lorsqu'une animation nécessite plusieurs exécutions simultanées d'une même logique avec des paramètres différents. Vous pouvez également ajouter un objet enfant à la propriété `scriptInstanceList` d'un objet Image-objet ou la propriété `actorList` de l'objet Animation afin de contrôler l'animation.

Tous les objets enfants d'un même script parent possédant des gestionnaires identiques, ces objets enfants répondent de façon semblable aux événements. Toutefois, chaque objet enfant conserve ses propres valeurs pour les propriétés définies dans le script parent. Il s'ensuit que chaque objet enfant peut avoir un comportement différent de celui des objets enfants apparentés tout en provenant du même script parent.

Par exemple, vous pouvez créer un script parent définissant des objets enfants de champs texte modifiables, possédant chacun leurs propres paramètres de propriété, texte et couleur, quels que soient les réglages des autres champs texte. En modifiant les valeurs des propriétés d'objets enfants précis, vous pouvez modifier ces caractéristiques pendant la lecture de l'animation, sans influer sur les autres objets enfants basés sur le même script parent.

De même, un objet enfant peut également contenir une propriété réglée sur `TRUE` ou `FALSE`, quel qu'en soit le réglage pour les objets enfants apparentés.

Un script parent se rapporte au nom d'un acteur script contenant les variables de propriétés et les gestionnaires. Un objet enfant créé à partir d'un script parent est essentiellement une nouvelle instance de l'acteur script.

Différences entre les objets enfants et les comportements

Bien que les objets enfants et les comportements soient similaires en ce sens qu'ils peuvent tous deux posséder plusieurs instances, ils présentent également plusieurs différences importantes. La principale différence entre les objets enfants et les comportements réside dans le fait que les comportements sont associés à des emplacements du scénario puisqu'ils sont affectés à des images-objets. Les objets comportement sont automatiquement créés à partir d'initialisateurs dans le scénario lorsque la tête de lecture passe d'une image à une autre et rencontre des images-objets associées à des comportements. Les scripts enfants, eux, doivent être explicitement créés par un gestionnaire.

Les comportements et les scripts enfants diffèrent dans la façon dont ils deviennent associés aux images-objets. Director associe automatiquement un comportement à l'image-objet à laquelle il est associé, alors que vous devez explicitement associer un script enfant à une image-objet. Les objets enfants ne requièrent pas de références d'images-objets et n'existent que dans la mémoire.

Ancêtres : principes de base

Les scripts parents peuvent déclarer des *ancêtres*, c'est-à-dire des scripts supplémentaires dont un script enfant peut appeler et utiliser les gestionnaires et les propriétés.

La création de scripts ancêtres permet de créer un ensemble de gestionnaires et de propriétés que vous pouvez utiliser et réutiliser pour plusieurs scripts parents.

Un script parent fait d'un autre script parent son ancêtre en affectant le script à sa propriété `ancestor`. Par exemple, l'instruction suivante transforme le script Objet_Principal en ancêtre du script parent dans lequel l'instruction survient :

```
-- Lingo syntax
ancestor = new(script "What_Everyone_Does")
```

Lorsque les gestionnaires et les propriétés ne sont pas définis dans un objet enfant, Director recherche la propriété ou le gestionnaire requis dans les ancêtres de l'enfant, en commençant par son script parent. Si un gestionnaire est appelé ou qu'une propriété est testée et que le script parent ne contient aucune définition correspondante, Director recherche une définition dans le script ancêtre. Si ce script ancêtre contient une définition, celle-ci est utilisée.

Un objet enfant ne peut avoir qu'un ancêtre à la fois, mais ce script ancêtre peut posséder à son tour un ancêtre, qui peut également en avoir un, et ainsi de suite. Cela vous permet de créer des générations de scripts parents dont les gestionnaires sont accessibles à un objet enfant.

Rédaction d'un script parent

Un script parent contient le code nécessaire à la création d'objets enfants et en définit les actions et propriétés possibles. Vous devez d'abord décider du comportement envisagé des objets enfants. Rédigez ensuite un script parent qui effectue les opérations suivantes :

- Déclare les éventuelles variables de propriétés requises ; ces variables représentant des propriétés pour lesquelles chaque objet enfant peut contenir une valeur indépendamment des autres objets enfants.
- Définit les valeurs initiales des propriétés et des variables de l'objet enfant dans le gestionnaire `on new`.
- Contient des gestionnaires supplémentaires contrôlant les actions de l'objet enfant.

Déclaration des variables de propriétés

Chaque objet enfant créé à partir du même script parent contient, dans un premier temps, les mêmes valeurs pour ses variables de propriétés. La valeur d'une variable de propriété n'appartient qu'à l'objet enfant auquel elle est associée. Chaque variable de propriété et sa valeur persistent tant que l'objet enfant existe. La valeur initiale d'une variable de propriété est généralement définie dans le gestionnaire `on new`. Si la propriété n'est pas initialement définie, sa valeur initiale est `VOID`.

Déclarer une variable de propriété

❖ Utilisez le mot-clé `property` au début du script parent.

Définir et tester des variables de propriétés en dehors de l'objet enfant

❖ Définissez et testez les variables de propriétés de la même façon que toute autre propriété dans vos scripts à l'aide de la syntaxe *`objectRef.propertyName`*.

Par exemple, l'instruction suivante définit la propriété `speed` d'un objet `car1` :

```
car1.speed = 55
```

Création du nouveau gestionnaire

Chaque script parent utilise généralement un gestionnaire `on new`. Ce gestionnaire crée le nouvel objet enfant lorsqu'un autre script émet une commande `new(script`*`parentScriptName`*`)`, qui ordonne au script parent défini de créer un objet enfant basé sur lui-même. Si nécessaire, le gestionnaire `on new` du script parent peut aussi définir les valeurs initiales des propriétés de l'objet enfant.

Le gestionnaire `on new` commence toujours par l'expression `on new`, suivie de la variable `me` et de tout paramètre communiqué au nouvel objet enfant.

Le gestionnaire `on new` suivant crée un nouvel objet enfant à partir du script parent et initialise la propriété `spriteNum` de l'enfant à l'aide de la valeur qui lui est associée dans le paramètre `aSpriteNum`. L'instruction `return me` renvoie l'objet enfant vers le gestionnaire qui a appelé initialement le gestionnaire `on new`.

```
-- Lingo syntax
property spriteNum

on new me, aSpriteNum
    spriteNum = aSpriteNum
    return me
end
```

Pour plus d'informations sur l'appel des gestionnaires `on new`, consultez « Création d'un objet enfant », page 53.

Ajout de gestionnaires supplémentaires

Le comportement de l'objet enfant est déterminé par l'inclusion des gestionnaires produisant le comportement escompté dans le script parent. Par exemple, vous pouvez ajouter un gestionnaire pour changer la couleur de l'image-objet.

Le script parent suivant définit une valeur pour la propriété `spriteNum` et contient un second gestionnaire qui modifie la propriété `foreColor` de l'image-objet :

```
-- Lingo syntax
property spriteNum

on new me, aSpriteNum
    spriteNum = aSpriteNum
    return me
end

on changeColor me
    spriteNum.foreColor = random(255)
end
```

Référence à l'objet actuel

D'une manière générale, les objets enfants multiples sont créés à partir du même script parent et chacun d'eux utilise plus d'un gestionnaire. La variable de paramètre spéciale `me` indique aux gestionnaires de l'objet enfant qu'ils doivent agir sur les propriétés de cet objet, et non sur celles d'autres objets enfants. De la sorte, lorsqu'un gestionnaire dans un objet enfant fait référence à des propriétés, il emploie les valeurs de son propre objet enfant pour ces propriétés.

Le terme `me` doit toujours être la première variable de paramètre indiquée dans chaque définition de gestionnaire d'un script parent. Il est toujours important de définir `me` comme premier paramètre pour les scripts parents et de passer le même paramètre si vous devez appeler d'autres gestionnaires dans le même script parent. Ceux-ci sont en effet les gestionnaires de chacun des objets enfants du script.

Lorsque vous faites référence à des propriétés définies dans des scripts ancêtres, vous devez utiliser le paramètre `me` comme source de la référence. En effet, la propriété, bien qu'elle soit définie dans le script ancêtre, n'en reste pas moins une propriété de l'objet enfant. Par exemple, l'instruction suivante utilise `me` pour désigner un objet et accéder aux propriétés définies dans un de ses ancêtres :

```
-- Lingo syntax
x = me.y -- access ancestor property y
```

La variable `me` étant présente dans tous les gestionnaires de l'objet enfant, elle indique que tous les gestionnaires contrôlent ce même objet enfant.

Création d'un objet enfant

Toute l'existence des objets enfants se déroule dans la mémoire ; ils ne sont pas enregistrés avec une animation. Seuls les scripts parents et ancêtres sont enregistrés sur disque.

Pour créer un nouvel objet enfant, utilisez la méthode `new()` et affectez à cet objet enfant un nom de variable ou une position dans une liste afin de pouvoir l'identifier et l'utiliser plus tard.

Pour créer un objet enfant et l'affecter à une variable, utilisez la syntaxe suivante :

```
-- Lingo syntax
variableName = new(script "scriptName", parameter1, parameter2, ...)
```

Le paramètre `scriptName` est le nom du script parent, et *`parameter1`*, *`parameter2`*, `...` représentent tout paramètre que vous passez au gestionnaire `on new` de l'objet enfant. La méthode `new()` crée un objet enfant dont l'ancêtre est `scriptName`. Elle appelle ensuite le gestionnaire `on new` de l'objet enfant, avec les paramètres indiqués.

L'instruction `new()` peut provenir de n'importe quel endroit de l'animation. Vous pouvez personnaliser les paramètres initiaux de l'objet enfant en modifiant les valeurs des paramètres transmis avec l'instruction `new()`.

Chaque objet enfant n'utilise que la mémoire nécessaire à l'enregistrement des valeurs actuelles de ses propriétés et variables ainsi qu'une référence au script parent. Par conséquent, vous pouvez généralement créer et gérer autant d'objets enfants que vous le souhaitez.

Des instructions `new()` supplémentaires permettent de produire d'autres objets enfants à partir du même script parent.

Vous pouvez créer des objets enfants sans initialiser immédiatement leurs variables de propriétés en utilisant la méthode `rawNew()`. La méthode `rawNew()` effectue cette opération en créant l'objet enfant sans appeler le gestionnaire `on new` du script parent. Au cas où de grandes quantités d'objets enfants seraient requises, `rawNew()` permet de créer les objets à l'avance et de reporter l'affectation des valeurs de propriété jusqu'à ce que chaque objet soit requis.

L'instruction suivante crée un objet enfant à partir du script parent Véhicule sans initialiser ses variables de propriétés, puis l'affecte à la variable `car1`.

```
-- Lingo syntax
car1 = script("Car").rawNew()
```

Pour initialiser les propriétés de l'un de ces objets enfants, appelez son gestionnaire `on new` :

```
car1.new
```

Vérification des propriétés d'un objet enfant

Vous pouvez vérifier les valeurs de variables de propriétés spécifiques dans différents objets enfants à l'aide de la syntaxe *`objectName.propertyName`*. Par exemple, l'instruction suivante affecte à la variable `x` la valeur de la propriété `carSpeed` de l'objet enfant dans la variable `car1`.

```
-- Lingo syntax
x = car1.carSpeed
```

La consultation des propriétés d'un objet depuis l'extérieur de cet objet peut s'avérer utile pour obtenir des informations sur des groupes d'objets, comme la vitesse moyenne de tous les véhicules d'un jeu de course automobile. Vous pouvez également utiliser les propriétés d'un objet afin de déterminer le comportement d'autres objets qui en dépendent.

En plus de vérifier les propriétés que vous affectez, vous pouvez déterminer si un objet enfant contient un gestionnaire spécifique ou rechercher le script parent d'où provient un objet. Cette fonction est très pratique si des objets proviennent de scripts parents similaires mais présentant des différences très subtiles.

Par exemple, vous pouvez créer un scénario dans lequel un script parent parmi d'autres peut servir à créer un objet enfant. Vous pouvez ensuite déterminer de quel script parent un objet enfant déterminé provient à l'aide de la fonction `script()`, qui renvoie le nom du script parent d'un objet.

Les instructions suivantes permettent de vérifier si l'objet `car1` a été créé à partir du script parent `car` :

```
-- Lingo syntax
if car1.script = script("Car") then
    _sound.beep()
end if
```

Vous pouvez aussi obtenir la liste des gestionnaires d'un objet enfant à l'aide de la méthode `handlers()` ou vérifier si un gestionnaire particulier existe dans un objet enfant en utilisant la méthode `handler()`.

L'instruction suivante place une liste des gestionnaires de l'objet enfant `car1` dans la variable `myHandlerList` :

```
-- Lingo syntax
myHandlerList = car1.handlers()
```

Cette liste pourrait se présenter comme suit :

```
[#start, #accelerate, #stop]
```

Les instructions suivantes utilisent la méthode `handler()` pour vérifier si le gestionnaire `on accelerate` existe dans l'objet enfant `car1` :

```
-- Lingo syntax
if car1.handler(#accelerate) then
    put("The child object car1 contains the handler named on accelerate.")
end if
```

Suppression d'un objet enfant

Vous pouvez supprimer un objet enfant d'une animation en modifiant la valeur de toutes les variables qui contiennent une référence à cet objet enfant. Si l'objet enfant est affecté à une liste, comme `actorList`, vous devez également supprimer l'objet de la liste.

Supprimer un objet enfant et les variables qui y font référence

❖ Affectez `VOID` à chaque variable.

Director supprime l'objet enfant lorsque rien ne lui fait plus référence. Dans l'exemple suivant, `ball1` contient la seule référence à un objet enfant particulier, et la définit comme étant `VOID` pour supprimer l'objet de la mémoire.

```
-- Lingo syntax
ball1 = VOID
```

Supprimer un objet de la liste actorList

❖ Utilisez la méthode `delete()` pour supprimer l'élément de la liste.

Utilisation de scriptInstanceList

Vous pouvez utiliser la propriété `scriptInstanceList` pour ajouter dynamiquement de nouveaux comportements à une image-objet. Normalement, `scriptInstanceList` est la liste des instances de comportements créées à partir des initialisateurs de comportements définis dans le scénario. Si vous ajoutez des objets enfants créés à partir de scripts parents à la liste, les objets enfants reçoivent les messages envoyés à d'autres comportements.

Par exemple, l'instruction suivante ajoute un objet enfant à la propriété `scriptInstanceList` de l'image-objet 10 :

```
-- Lingo syntax
add(sprite(10).scriptInstanceList, new(script "rotation", 10))
```

Le script suivant est le script parent possible auquel l'instruction précédente fait référence :

```
-- Lingo syntax parent script "rotation"
property spriteNum

on new me, aSpriteNum
    spriteNum = aSpriteNum
    return me
end

on prepareFrame me
    sprite(spriteNum).rotation = sprite(spriteNum).rotation + 1
end
```

Lorsqu'un objet enfant est ajouté à `scriptInstanceList`, vous devez initialiser la propriété `spriteNum` de l'objet enfant. Cette opération est généralement effectuée à partir d'un paramètre passé au gestionnaire `on new`.

Remarque : *le message `beginSprite` n'est pas envoyé aux objets enfants ajoutés dynamiquement.*

Pour plus d'informations sur la propriété `scriptInstanceList`, consultez « scriptInstanceList », page 1017.

Utilisation de actorList

Vous pouvez établir une liste spéciale d'objets enfants (ou de tout autre objet) qui reçoivent un message personnel à chaque fois que la tête de lecture entre dans une image ou que la méthode `updateStage()` met la scène à jour.

La liste spéciale est `actorList`, qui ne contient que les objets ayant été explicitement ajoutés à la liste.

Le message est le message `stepFrame`, qui n'est émis que lorsque la tête de lecture entre dans une image ou que la commande `updateStage()` est utilisée.

Les objets de `actorList` reçoivent un message `stepFrame` au lieu d'un message `enterFrame` à chaque image. Si les objets disposent d'un gestionnaire `on stepFrame`, le script du gestionnaire est exécuté à chaque fois que la tête de lecture entre dans une nouvelle image ou que la méthode `updateStage()` met la scène à jour.

Parmi les applications possibles de `actorList` et `stepFrame` figurent l'animation d'objets enfants utilisés en tant qu'images-objets ou la mise à jour d'un compteur qui suit le nombre de fois que la tête de lecture entre dans une image.

Un gestionnaire `on enterFrame` pourrait produire les mêmes résultats, mais la propriété `actorList` et le gestionnaire `stepFrame` sont conçus pour des performances optimales dans Director. Les objets de `actorList` répondent mieux aux messages `stepFrame` qu'aux messages `enterFrame` ou aux messages personnalisés émis après une méthode `updateStage()`.

Ajouter un objet dans actorList

❖ Utilisez la propriété `actorList` comme suit, *`childObject`* faisant référence à l'objet enfant à ajouter :

```
-- Lingo syntax
_movie.actorList.add(childObject)
```

Le gestionnaire `stepFrame` de l'objet, défini dans son script parent ou ancêtre, est alors exécuté automatiquement à chaque fois que la tête de lecture avance. L'objet est transmis en tant que premier paramètre, `me`, au gestionnaire `on stepFrame`.

Director n'efface pas le contenu de la liste `actorList` lorsqu'il passe à une autre animation, ce qui peut provoquer un comportement imprévisible dans cette dernière. Pour éviter le transfert des objets enfants de l'animation actuelle dans la nouvelle animation, insérez une instruction qui efface `actorList` dans le gestionnaire `on prepareMovie` de la nouvelle animation.

Supprimer des objets enfants de actorList

❖ Donnez à `actorList` la valeur `[ ]`, qui représente une liste vide.

Pour plus d'informations sur `actorList`, consultez « actorList », page 655.

Création d'objets de temporisation

Un objet de temporisation est un objet qui agit comme un minuteur et qui envoie un message à la fin du délai. Ce type d'objet est utile pour les scénarios qui nécessitent l'exécution de certains événements à intervalles réguliers ou après un certain délai.

Les objets de temporisation peuvent envoyer des messages appelant des gestionnaires dans des objets enfants ou des scripts d'animation. Pour créer un objet de temporisation, utilisez le mot-clé `new()`. Vous devez spécifier le nom de l'objet, le gestionnaire à appeler et la fréquence avec laquelle le gestionnaire doit être appelé. Lorsqu'un objet de temporisation est créé, Director maintient une liste des objets de temporisation actuellement actifs, nommée `timeOutList`.

La syntaxe décrite ci-dessous est nécessaire pour toutes les animations créées dans Adobe Director 11ou pour les anciennes animations lues dans Adobe Director 11 dont la propriété `scriptExecutionStyle` est définie sur 10. Les animations créées dans Director MX et les versions antérieures ont une propriété `scriptExecutionStyle` définie sur 9, ce qui vous permet d'utiliser la syntaxe trouvée dans Director MX et les versions antérieures.

Créer des objets de temporisation

```
-- Lingo syntax when scriptExecutionStyle is set to 9
variableName = timeout().new(timeoutName, timeoutPeriod, #timeoutHandler, {,
targetObject})

-- Lingo syntax when scriptExecutionStyle is set to 10
variableName = timeout().new(timeoutName, timeoutPeriod, timeoutHandler, targetObject)
variableName = new timeout(timeoutName, timeoutPeriod, timeoutHandler, targetObject)

// JavaScript syntax
variableName = new timeout(timeoutName, timeoutPeriod, timeoutFunction, targetObject)
```

Cette instruction utilise les éléments suivants :

- *`variableName`* est la variable dans laquelle vous placez l'objet de temporisation.
- `timeout` indique le type d'objet Lingo que vous créez.
- *`timeoutName`* est le nom que vous donnez à l'objet de temporisation. Ce nom figure dans la liste `timeOutList`. C'est la propriété `#name` de l'objet.
- `new` crée un nouvel objet.
- *`intMilliseconds`* indique la fréquence avec laquelle l'objet de temporisation doit appeler le gestionnaire indiqué. C'est la propriété `#period` de l'objet. Par exemple, une valeur de 2 000 appelle le gestionnaire indiqué toutes les 2 secondes.
- `#`*`handlerName`* est le nom du gestionnaire que l'objet doit appeler. C'est la propriété `#timeOutHandler` de l'objet. Vous la représentez sous la forme d'un symbole en précédant le nom du signe `#`. Par exemple, un gestionnaire nommé `accelerate` serait noté `#accelerate`.

• `targetObject` indique le gestionnaire de l'objet enfant à appeler. C'est la propriété `#target` de l'objet. Ce paramètre permet d'être spécifique si plusieurs objets enfants contiennent les mêmes gestionnaires. Si vous omettez ce paramètre, Director recherche le gestionnaire indiqué dans le script d'animation.

L'instruction suivante crée un objet de temporisation nommé `timer1` qui appelle le gestionnaire `on accelerate` dans l'objet enfant `car1` toutes les 2 secondes :

```
-- Lingo syntax
myTimer = timeOut("timer1").new(2000, #accelerate, car1)
```

Pour déterminer le moment auquel le message de temporisation suivant est envoyé par un objet de temporisation déterminé, consultez sa propriété `#time`. La valeur renvoyée est le moment, en millièmes de secondes, auquel le message de temporisation suivant est envoyé. Par exemple, l'instruction suivante détermine le moment auquel le prochain message de temporisation est envoyé à partir de l'objet de temporisation `timer1` et l'affiche dans la fenêtre Messages :

```
-- Lingo syntax
put(timeout("timer1").time)
```

Utilisation de timeOutList

Lorsque vous commencez à créer des objets de temporisation, vous pouvez utiliser `timeOutList` pour déterminer le nombre d'objets de temporisation actifs à un moment particulier.

L'instruction suivante attribue la variable `x` au nombre d'objets contenus dans `timeOutList` en utilisant la propriété `count` :

```
-- Lingo syntax
x = _movie.timeoutList.count
```

Vous pouvez également faire référence à un objet de temporisation en employant son numéro dans la liste.

L'instruction suivante supprime le second objet de temporisation dans `timeOutList` en utilisant la méthode `forget()` :

```
-- Lingo syntax
timeout(2).forget()
```

Relais d'événements système au moyen d'objets de temporisation

Lorsque vous créez des objets de temporisation qui font référence à des objets enfants précis, vous permettez à ces derniers de recevoir des événements système. Les objets de temporisation relaient ces événements vers leurs objets enfants cibles. Les événements système qui peuvent être reçus par des objets enfants sont, par exemple, `prepareMovie`, `startMovie`, `stopMovie`, `prepareFrame` et `exitFrame`. En incluant des gestionnaires pour ces événements dans les objets enfants, vous ordonnez aux objets enfants de leur répondre en fonction de vos besoins. Les événements système reçus par les objets enfants sont également reçus par les scripts d'animation, les scripts d'image et les autres scripts définis comme devant leur répondre.

Le script parent suivant contient un gestionnaire pour l'événement système `exitFrame` ainsi qu'un gestionnaire personnalisé `slowDown` :

```
-- Lingo syntax
property velocity

on new me
    velocity = random(55)
end

on exitFrame
    velocity = velocity + 5
end
```

```
on slowDown mph
    velocity = velocity - mph
end
```

Association de propriétés personnalisées avec des objets de temporisation

Si vous souhaitez associer des propriétés personnalisées avec un objet de temporisation, vous devez créer un objet de temporisation dont la cible n'est pas une référence à un objet instance de script. Lorsque vous utilisez cette technique, les données cibles deviennent des données associées à l'objet de temporisation pouvant être utilisées dans votre gestionnaire de temporisation.

L'exemple suivant vous montre comment utiliser cette technique :

```
-- Lingo syntax
-- initialize a timeout object and pass it a data property list (tData)
-- instead of a reference to a script instance object
tData = [#beta: 0]
tTO = timeout("betaData").new(50,#targetHandler,tData)

-- within a movie script, create the targetHandler handler
on targetHandler (aData)
    -- increment and display the beta property
    tData.beta = tData.beta + 1
    put(tData.beta)
end targetHandler
```

Dans l'exemple précédent, la propriété `beta` continue d'être incrémentée. Cela signifie que vous pouvez initialiser plusieurs objets de temporisation qui appellent le même gestionnaire de script d'animation. Chaque objet de temporisation peut posséder sa propre liste de données.

En général, sachez que :

- Lors de l'utilisation d'une instance de script en tant que cible, le gestionnaire cible de cette instance de script en particulier est appelé. Vous ne pouvez pas utiliser des propriétés personnalisées avec cette technique.
- Lors de l'utilisation d'une référence autre qu'une instance de script (une liste de propriété, par exemple) en tant que cible, le gestionnaire cible dans un script d'animation est appelé. Vous pouvez utiliser des propriétés personnalisées avec cette technique.

Programmation orientée objet avec la syntaxe JavaScript

La programmation orientée objet dans la syntaxe JavaScript est quelque peu différente de celle d'autres langages orientés objet tels que Java et C++. Alors que certains langages orientés objet sont basés sur les classes, la syntaxe JavaScript est basée sur les prototypes.

Les deux-points qui suivent comparent à un niveau élevé les langages basés sur les classes aux langages basés sur les prototypes tels que la syntaxe JavaScript :

- Dans les langages basés sur les classes, vous créez des définitions de classes se rapportant aux propriétés et méthodes initiales qui caractérisent toutes les instances créées à partir de ces classes. Une définition de classe contient des méthodes spéciales, appelées *méthodes de construction*, qui sont utilisées pour créer des instances de cette classe. Lorsqu'une instance est créée à l'aide de l'opérateur `new` en association avec une méthode de construction

particulière, cette instance hérite de toutes les propriétés de sa classe parent. Cette instance peut également effectuer toute autre tâche de traitement qui lui est propre en fonction de la construction qui a été appelée.

Dans une définition de classe, vous procédez à l'opération d'héritage en créant une sous-classe qui hérite de toutes les propriétés de sa classe parent, en plus de définir les nouvelles propriétés et de modifier éventuellement les propriétés héritées. La classe parent à partir de laquelle une sous-classe a été créée est aussi appelée *superclasse*.

- Dans les langages basés sur les prototypes tels que la syntaxe JavaScript, il n'existe pas de distinction entre les classes, les instances, les sous-classes, etc. Tous ces éléments sont considérés comme des objets. Dans la syntaxe JavaScript, au lieu d'utiliser des définitions de classes, vous utilisez des *objets prototypes* en tant que modèle à partir duquel de nouveaux objets sont créés. Tout comme dans les langages basés sur les classes, dans la syntaxe JavaScript, vous créez un nouvel objet en utilisant l'opérateur `new` en association avec une fonction de construction.

  Dans la syntaxe JavaScript, au lieu d'utiliser les superclasses et les sous-classes, vous associez les objets prototypes à des fonctions de construction pour effectuer l'héritage. Ce procédé est très similaire à celui qui consiste à utiliser les superclasses et les sous-classes, à l'exception de la terminologie qui est différente.

  De plus, contrairement aux langages basés sur les classes, la syntaxe JavaScript vous permet d'ajouter et de supprimer des propriétés à un objet ou une série d'objets à l'exécution. Par exemple, si vous ajoutez une propriété à un objet prototype à l'exécution, les objets d'instances apparentés héritent également de cette propriété.

Terminologie orientée objet

Dans la syntaxe JavaScript, étant donné que tous les types correspondent à des objets, les termes basés sur les classes telles que *superclasse*, *sous-classe*, *classe*, *instance*, etc. n'ont aucune signification technique littérale. Cependant, ces termes correspondent essentiellement à des objets dans la syntaxe JavaScript et peuvent être utilisés pour illustrer les différents types d'objets de la syntaxe JavaScript. Ainsi, ces termes basés sur les classes sont utilisés de manière interchangeable avec le terme *objet* tout au long de la discussion sur la programmation orientée objet dans la syntaxe JavaScript et signifient ce qui suit :

- **superclasse** Toute classe à partir de laquelle on peut créer des sous-classes (objets) ; une classe parent.
- **sous-classe** Toute classe ayant été créée à partir d'une superclasse (objet) ; une classe enfant.
- **classe** Terme générique signifiant superclasse ou sous-classe ; une classe parent ou enfant.
- **instance** ou **instance d'objet** Objet unique ayant été créé à partir d'une superclasse.

Classes personnalisées

L'un des principaux avantages de la programmation orientée objet est la possibilité de créer des classes personnalisées qui vous permettent d'ajouter des fonctionnalités personnalisées à vos scripts. Les classes prédéfinies, telles que Objet, Chaîne, Mathématique, etc. fournies par la syntaxe JavaScript sont utiles dans certains cas, mais les fonctionnalités qu'elles offrent peuvent ne pas s'appliquer à la tâche que vous souhaitez accomplir. Supposons, par exemple, que vous souhaitez que certains objets de votre animation représentent des types de transports, tels que des véhicules, des bateaux, des avions, etc. et que vous souhaitez que chaque catégorie affiche des caractéristiques et fonctionnalités uniques. Ni les classes prédéfinies de la syntaxe JavaScript, ni les objets Director prédéfinis ne peuvent directement vous fournir la fonctionnalité dont vous avez besoin. C'est pourquoi vous devez créer une nouvelle classe pour chaque type de transport afin de pouvoir définir des caractéristiques uniques pour chacun d'entre eux.

N'oubliez pas que lorsque vous créez des classes personnalisées de la syntaxe JavaScript, vous avez toujours accès à toutes les fonctions et fonctionnalités des objets Director prédéfinis. Cela signifie que même si les objets Director prédéfinis ne vous fournissent pas directement les fonctionnalités dont vous avez besoin, vous pouvez toujours les utiliser dans vos classes personnalisées pour accéder à leurs valeurs et à leurs fonctionnalités prédéfinies.

Fonctions de construction

Dans la syntaxe JavaScript, une fonction de construction représente la classe qui contient le modèle à partir duquel les nouvelles instances d'objets sont créées. Les fonctions de construction créent et initialisent (définissent l'état par défaut) les propriétés des nouveaux objets.

Le format des fonctions de construction est fondamentalement identique à celui des fonctions de méthode standard de la syntaxe JavaScript. Cependant, la différence entre la fonction de construction et la fonction de méthode réside dans le fait que la fonction de construction utilise un mot-clé spécial `this` pour représenter une référence à un nouvel objet initialisé. En règle générale, une fonction de méthode effectue des actions sur un ensemble spécifique de données d'un objet.

L'exemple suivant vous montre une manière de créer une fonction de construction Rectangle qui pourrait être utilisée pour initialiser la hauteur et la largeur de nouveaux objets Rectangle :

```
function Rectangle(w, h) {
    this.width = w;
    this.height = h;
}
```

Vous pouvez également créer une fonction de construction en utilisant la syntaxe *fonction littérale*. La syntaxe fonction littérale fournit les mêmes fonctionnalités que la syntaxe précédente et correspond à une autre manière de rédiger une construction. L'exemple suivant vous montre comment utiliser la syntaxe fonction littérale pour créer une fonction de construction Rectangle identique à la précédente :

```
Rectangle = function(w, h) {
    this.width = w;
    this.height = h;
}
```

Remarque : *lorsque vous définissez des fonctions de construction à appliquer à une animation, assurez-vous de les placer dans un script d'animation afin qu'elles soient disponibles globalement.*

Il est recommandé de donner des noms aux fonctions de construction qui correspondent à leurs fonctionnalités et d'utiliser une lettre majuscule au début du nom, comme `Rectangle` ou `Circle`.

En général, les fonctions de construction sont utilisées pour initialiser de nouveaux objets, mais peuvent également renvoyer l'objet si vous le souhaitez. Si vous retournez l'objet initialisé, l'objet renvoyé devient la valeur de l'expression `new`.

Instances d'objets

La manière la plus commune de créer une nouvelle instance d'objet est d'utiliser l'opérateur `new` suivi du nom de la fonction de construction. Les exemples suivants créent de nouvelles instances d'objets :

```
var objRandom = new Object(); // assigns a reference to an Object object
var objString = new String(); // assigns a reference to a String object
```

Une fonction de construction peut parfois définir des paramètres transmis par une nouvelle instance d'objet afin d'initialiser l'état de l'instance d'objet. Si une fonction de construction définit les paramètres utilisés lors de l'initialisation de nouvelles instances d'objets, les valeurs de propriétés sont initialisées ainsi :

- Si vous transmettez des valeurs à la fonction de construction lors de l'initialisation, les propriétés qui ont reçu les valeurs sont définies selon ces valeurs.
- Si vous ne transmettez pas ces valeurs à la fonction de construction lors de l'initialisation, les propriétés qui n'ont pas reçu les valeurs sont définies sur `undefined`.

Lorsque vous créez de nouvelles instances d'objets, le mot-clé `this` est utilisé dans le corps de la fonction de construction associée afin de faire référence à la nouvelle instance d'objet. Par conséquent, une nouvelle instance d'objet est initialisée avec toutes les propriétés définies à l'aide de la syntaxe `this.propertyName`.

Dans l'exemple suivant, une fonction de construction `Circle` utilise le mot-clé `this` pour spécifier les noms des trois propriétés qui sont associées aux nouvelles instances d'objets. L'instruction suivant la construction initialise une nouvelle instance d'objet en transmettant des valeurs à la construction. Ces valeurs sont utilisées en tant que valeurs initiales des propriétés spécifiées par le mot-clé `this`.

```
// Circle constructor function
function Circle(x, y, r) {
    this.xCoord = x;
    this.yCoord = y;
    this.radius = r;
}

// xCoord = 10, yCoord = 15, radius = 5
var objCircle = new Circle(10, 15, 5);
```

Une fois l'instance `objCircle` initialisée, vous pouvez accéder à ses propriétés. A l'aide de l'instance `objCircle` créée précédemment, vous pouvez donner à certaines variables les valeurs de leurs propriétés.

```
var theXCoord = objCircle.xCoord; // assigns the value 10 to theXCoord
var theYCoord = objCircle.yCoord; // assigns the value 15 to theYCoord
var theRadius = objCircle.radius; // assigns the value 5 to theRadius
```

Remarque : *pour plus d'informations sur l'utilisation de la syntaxe à point pour accéder aux propriétés et méthodes d'un objet, consultez* *« Programmation avec la syntaxe à point », page 44*.

Il est recommandé de donner aux nouveaux objets des noms qui correspondent à leur fonctionnalité et d'utiliser des minuscules, comme `objRectangle` ou `objCircle`.

Vous pouvez également créer une instance d'objet en utilisant la syntaxe *objet littéral*. Ainsi, vous n'avez pas besoin d'utiliser l'opérateur `new` et la fonction de construction. Utilisez cette technique uniquement lorsque vous n'avez besoin que d'une instance d'un objet n'ayant pas été défini dans une fonction de construction. Dans l'exemple suivant, une instance d'objet est créée avec `x = 1`, `y = 2` et `rayon = 2` :

```
var objSmallCircle = { x:1, y:2, radius:2 };
```

Héritage d'objets

Non seulement la programmation orientée objet est capable de créer vos propres classes personnalisées, mais grâce à elle, les sous-classes peuvent également hériter des propriétés et méthodes des superclasses à partir desquelles elles ont été créées. L'héritage vous facilite la création d'objets possédant des propriétés et fonctionnalités intégrées.

Dans la syntaxe JavaScript, une superclasse correspond à la classe de base à partir de laquelle toutes les autres sous-classes sont créées : la superclasse Objet. La superclasse Objet comprend quelques propriétés et méthodes de base. Les sous-classes qui sont créées à l'aide du modèle Objet héritent toujours de ces propriétés et méthodes de base et définissent leurs propres propriétés et méthodes. Les sous-classes provenant de ces classes héritent de l'objet Objet, de leurs superclasses, et ainsi de suite. Tous les autres objets que vous créez prolongent cette chaîne d'héritage.

Par exemple, l'objet Objet contient la propriété `constructor` et la méthode `toString()`. Si vous créez une nouvelle classe nommée `SubObj1`, elle correspond à une sous-classe de l'objet Objet et hérite ainsi de la propriété `constructor` et de la méthode `toString()` de l'objet Objet. Ensuite, si vous créez une autre classe nommée `SubObj2` en utilisant `SousObj1` en tant que superclasse, `SubObj2` hérite également de la propriété `constructor` et de la méthode `toString()` de l'objet Objet, en plus de toutes les autres propriétés et méthodes personnalisées que vous avez définies dans `SubObj1`.

Deux des propriétés importantes dont vos fonctions de construction personnalisées héritent de la superclasse Objet sont `prototype` et `constructor`. La propriété `prototype` représente l'objet prototype de la classe, qui vous permet d'ajouter des variables (propriétés) et des méthodes aux instances d'objets. C'est le moyen par lequel l'héritage est en général implémenté dans la syntaxe JavaScript. La propriété `constructor` représente la fonction de construction en elle-même. Dans les sections suivantes, l'utilisation de ces propriétés est expliquée.

Objets prototypes

Comme indiqué précédemment, lorsque vous créez une sous-classe, cette dernière hérite automatiquement des propriétés et méthodes de la superclasse à laquelle elle se rapporte. Dans la syntaxe JavaScript, l'héritage est en général implémenté en utilisant des objets prototypes. En réalité, une sous-classe hérite des propriétés et méthodes de l'objet prototype de sa superclasse, et non de la superclasse en elle-même. Ce point important apporte un avantage sérieux : toutes les propriétés et méthodes ne doivent pas être obligatoirement copiées d'une classe à une instance d'objet de cette classe, ce qui peut considérablement réduire la quantité de mémoire nécessaire à la création de nouvelles instances d'objets.

Dans la syntaxe JavaScript, chaque classe, y compris la classe prédéfinie Objet, contient uniquement un objet prototype. Chaque instance d'objet créée à partir d'une classe a accès aux propriétés et méthodes dans l'objet prototype de cette classe. Par conséquent, l'objet prototype d'une classe est en général le seul objet qui stocke les propriétés et méthodes pour cette classe ; une instance d'objet contient uniquement les propriétés nécessaires à l'initialisation de cette instance.

Dans votre code, il semble que chaque instance d'objet comprenne réellement ces propriétés et méthodes parce que vous pouvez y accéder directement à partir de l'instance d'objet. Mais en réalité, l'instance utilise l'objet prototype pour y accéder. L'objet prototype d'une classe est automatiquement créé lorsque vous créez la classe. Vous pouvez accéder à l'objet prototype à l'aide de la propriété `prototype` de la classe.

Etant donné que l'objet prototype d'une classe stocke les propriétés partagées par toutes les instances d'objets, il peut définir correctement les propriétés et méthodes dont les valeurs sont partagées parmi toutes les instances d'objets. Le fait de partager les propriétés et les méthodes parmi les instances d'objets vous facilite la création d'instances qui affichent un comportement défini par défaut et la personnalisation de toutes les instances associées au comportement par défaut.

Les objets prototypes ne sont en général pas adaptés pour définir les propriétés et méthodes dont les valeurs peuvent varier selon les instances d'objets. Dans ces cas-là, vous pouvez définir ces propriétés et méthodes au sein de la classe elle-même.

Pour spécifier l'étendue d'une propriété ou méthode personnalisée, vous pouvez la définir selon l'un des quatre types suivants :

- Variables d'instances
- Méthodes d'instances
- Variables de classes
- Méthodes de classes

Variables d'instances

Les *variables d'instances* correspondent à toutes les variables (propriétés) définies dans une fonction de construction et sont copiées dans chaque instance d'objet de cette construction. Toutes les instances d'objets possèdent leurs propres copies de variables d'instances. Cela signifie que s'il existe cinq instances d'objets pour une classe `Circle`, cinq copies de chaque variable d'instance sont définies dans la classe. Etant donné que chaque instance d'objet possède sa propre copie de variable d'instance, chaque instance d'objet peut affecter une valeur unique à une variable

d'instance sans pour autant modifier les valeurs des autres copies de la variable d'instance. Vous pouvez accéder directement aux variables d'instances à partir des instances d'objets qui leur sont propres.

Dans l'exemple suivant, quatre variables d'instances (`make`, `model`, `color` et `speed`) sont définies dans une fonction de construction. Ces quatre variables d'instances sont directement disponibles à partir de toutes les instances d'objets de la construction `Car`.

```
function Car(make, model, color) { // define a Car class
    this.make = make;
    this.model = model;
    this.color = color;
    this.speed = 0;
}
```

L'instance d'objet suivante `objCar` contient les quatre variables d'instances. Bien que la valeur de la variable d'instance `speed` ne soit pas passée à la construction `Car`, `objCar` possède toujours une propriété `speed` dont la valeur initiale est 0 parce que la variable `speed` est définie dans la construction `Car`.

```
// objCar.make="Subaru", objCar.model="Forester",
// objCar.color="silver", objCar.speed = 0
var objCar = new Car("Subaru", "Forester", "silver");
```

Méthodes d'instances

Les *méthodes d'instances* correspondent à toute méthode accessible via une instance d'objet. Les instances d'objets ne possèdent pas leurs propres copies de méthodes d'instances. Les méthodes d'instances sont en fait définies en premier en tant que fonctions, et ensuite les propriétés de l'objet prototype de la fonction de construction sont définies sur les valeurs de la fonction. Les méthodes d'instances utilisent le mot-clé `this` dans le corps de la fonction de construction définie afin de faire référence à l'instance d'objet sur laquelle elles opèrent. Bien qu'une instance d'objet donnée ne possède pas une copie d'une méthode d'instance, vous pouvez toujours accéder directement aux méthodes d'instances à partir des instances d'objets qui y sont associées.

L'exemple suivant définit une fonction nommée `Car_increaseSpeed()`. Le nom de la fonction est ensuite affecté à la propriété `increaseSpeed` de l'objet prototype de la classe `Car` :

```
// increase the speed of a Car
function Car_increaseSpeed(x) {
    this.speed += x;
    return this.speed;
}
Car.prototype.increaseSpeed = Car_increaseSpeed;
```

Une instance d'objet `Car` peut ensuite accéder à la méthode `increaseSpeed()` et affecter sa valeur à la variable à l'aide de la syntaxe suivante :

```
var objCar = new Car("Subaru", "Forester", "silver");
var newSpeed = objCar.increaseSpeed(30);
```

Vous pouvez également créer une méthode d'instance à l'aide de la syntaxe de fonction littérale. En utilisant la syntaxe de fonction littérale, il n'est plus nécessaire ni de définir une fonction, ni d'affecter un nom de propriété au nom de la fonction.

Dans l'exemple suivant, la syntaxe de fonction littérale est utilisée pour définir une méthode `increaseSpeed()` qui contient les mêmes fonctionnalités que la fonction `increaseSpeed()` définie précédemment :

```
// increase the speed of a Car
Car.prototype.increaseSpeed = function(x) {
    this.speed += x;
    return this.speed;
}
```

Variables de classes

Egalement appelées variables *statiques*, elles correspondent à toutes les variables (propriétés) associées à une classe et non à une instance d'objet. Il n'existe qu'une copie de variable de classe, quel que soit le nombre d'instances d'objets créées à partir de cette classe. Les variables de classes n'utilisent pas l'objet prototype pour implémenter l'héritage. Vous pouvez accéder directement à une variable de classe via la classe et non via une instance d'objet ; vous devez définir une classe dans une fonction de construction avant de pouvoir définir des variables de classes.

Dans l'exemple suivant, deux variables de classes sont définies (`MAX_SPEED` et `MIN_SPEED`) :

```
function Car() { // define a Car class
    ...
}

Car.MAX_SPEED = 165;
Car.MIN_SPEED = 45;
```

Vous avez directement accès aux variables de classes `MAX_SPEED` et `MIN_SPEED` à partir de la classe `Car`.

```
var carMaxSpeed = Car.MAX_SPEED; // carMaxSpeed = 165
var carMinSpeed = Car.MIN_SPEED; // carMinSpeed = 45
```

Méthodes de classes

Egalement appelées méthodes *statiques*, elles correspondent à toutes les méthodes associées à une classe et non à une instance d'objet. Il n'existe qu'une copie de méthode de classe, quel que soit le nombre d'instances d'objets créées à partir de cette classe. Les méthodes de classes n'utilisent pas l'objet prototype pour implémenter l'héritage. Vous avez directement accès à une méthode de classe via la classe et non via une instance d'objet ; vous devez définir une classe dans une fonction de construction avant de pouvoir définir des méthodes de classes.

Dans l'exemple suivant, on définit une fonction nommée `setInitialSpeed()` qui peut modifier la vitesse par défaut des nouvelles instances de véhicules. Le nom de la fonction est affecté à la propriété `setInitialSpeed` de la classe `Car` :

```
function Car(make, model, color) { // define a Car class
    this.make = make;
    this.model = model;
    this.color = color;
    this.speed = Car.defaultSpeed;
}
Car.defaultSpeed = 10; // initial speed for new Car instances
// increase the speed of a Car
function Car_setInitialSpeed(x) {
    Car.defaultSpeed = x;
}
Car.setInitialSpeed = Car_setInitialSpeed;
```

Vous pouvez accéder directement à la méthode de classe `setInitialSpeed()` à partir de la classe `Car`.

```
var newSpeed = Car.setInitialSpeed(30);
```

Vous pouvez également créer une méthode de classe à l'aide de la syntaxe de fonction littérale. Dans l'exemple suivant, la syntaxe de fonction littérale est utilisée pour définir une méthode `setInitialSpeed()` qui contient les mêmes fonctionnalités que la fonction `setInitialSpeed()` définie précédemment :

```
// increase the speed of a Car
Car.setInitialSpeed = function(x) {
    Car.defaultSpeed = x;
}
```

Etapes recommandées pour définir une classe

La liste suivante décrit les étapes que nous vous recommandons de suivre lorsque vous définissez une classe :

1 Définissez une fonction de construction utilisée en tant que modèle à partir duquel les instances d'objets sont initialisées. Vous pouvez également définir toute variable d'instance de la fonction de construction en utilisant le mot-clé `this` pour faire référence à une instance d'objet.

2 Définissez toute méthode d'instance, ainsi que toute variable d'instance, stockée dans l'objet prototype d'une classe. Ces méthodes et variables d'instances sont disponibles pour toutes les instances d'objets et vous pouvez y accéder via l'objet prototype de la classe.

3 Définissez toute méthode de classe, variable de classe et constante stockées dans la classe elle-même. Vous ne pouvez accéder à ces méthodes et variables de classes que via la classe elle-même.

Dans votre code, lorsque vous accédez à une propriété d'une instance d'objet, la syntaxe JavaScript recherche l'instance d'objet de cette propriété. Si l'instance ne contient pas de propriété, la syntaxe JavaScript recherche alors l'objet prototype de la superclasse à partir de laquelle l'instance a été créée. Etant donné qu'une instance d'objet est recherchée avant l'objet prototype de la classe à partir de laquelle il a été créé, les propriétés de l'instance d'objet masquent les propriétés à l'objet prototype de leurs superclasses. Cela signifie qu'une instance d'objet et sa superclasse peuvent réellement définir une propriété avec le même nom mais avec des valeurs différentes.

Suppression de variables

Vous pouvez supprimer une variable de classe ou une variable d'instance à l'aide de l'opérateur `delete`. L'exemple suivant illustre ce processus.

```
function Car() { // define a Car constructor function
    ...
}
Car.color = "blue"; // define a color property for the Car class
Car.prototype.engine = "V8"; // define an engine property for the prototype

var objCar = new Car();

trace(Car.color); // displays "blue"
trace(objCar.engine); // displays "V8"

delete Car.color;
delete Car.prototype.engine;

trace(Car.color); // displays undefined
trace(objCar.engine); // displays undefined
```

Accès à la propriété de construction d'un objet prototype

Lorsque vous définissez une classe en créant une fonction de construction, la syntaxe JavaScript crée un objet prototype pour cette classe. Lorsque l'objet prototype est créé, il comprend au départ une propriété `constructor` qui se rapporte à la fonction de construction elle-même. Vous pouvez utiliser la propriété `constructor` d'un objet prototype pour déterminer le type de tout objet donné.

Dans l'exemple suivant, la propriété `constructor` contient une référence à la fonction de construction utilisée pour créer l'instance d'objet. La valeur de la propriété `constructor` est en réalité une référence à la construction elle-même et n'est pas une chaîne qui contient le nom de la construction :

```
function Car() { // define a Car class
    // initialization code here
```

```
}
var myCar = new Car(); // myCar.constructor == function Car() {}
```

Création dynamique de propriétés

Un des autres avantages dont vous pouvez bénéficier en utilisant un objet prototype pour implémenter l'héritage est que les propriétés et méthodes ajoutées à un objet prototype sont automatiquement disponibles pour les instances d'objets. Cette remarque est vraie même si une instance d'objet a été créée avant que les propriétés ou les méthodes n'aient été ajoutées.

Dans l'exemple suivant, la propriété `color` est ajoutée à l'objet prototype d'une classe `Car` après la création d'une instance d'objet `Car`.

```
function Car(make, model) { // define a Car class
    this.make = make;
    this.model = model;
}
var myCar = new Car("Subaru", "Forester"); // create an object instance

trace(myCar.color); // returns undefined

// add the color property to the Car class after myCar was initialized
Car.prototype.color = "blue";

trace(myCar.color); // returns "blue"
```

Vous pouvez également ajouter des propriétés aux instances d'objets après que les instances ont été créées. Lorsque vous ajoutez une propriété à une instance d'objet spécifique, cette propriété est disponible uniquement pour cette instance d'objet spécifique. A l'aide de l'instance d'objet `myCar` précédemment créée, les instructions suivantes ajoutent la propriété `colour` à `myCar` après sa création.

```
trace(myCar.color); // returns undefined

myCar.color = "blue"; // add the color property to the myCar instance

trace(myCar.color); // returns "blue"

var secondCar = new Car("Honda", "Accord"); // create a second object instance

trace(secondCar.color); // returns undefined
```

Rédaction de scripts dans la fenêtre Script

Lorsque vous rédigez des scripts pour une animation, ces scripts peuvent considérablement augmenter en quantité et en complexité. La décision des méthodes ou des propriétés à utiliser, de la structuration efficace des scripts et de leur position judicieuse exige un plan d'action rigoureux et de nombreux tests au fur et à mesure que la complexité de l'animation augmente.

L'aspect le plus important des scripts réside dans la formulation d'un objectif et dans une compréhension totale de ce que vous souhaitez accomplir et ce, avant de commencer à rédiger ces scripts. C'est en fait aussi important, et en général aussi laborieux, que la création des storyboards de l'animation.

Une fois le plan d'ensemble de votre animation conçu, vous pouvez commencer à rédiger puis à tester les scripts. Attendez-vous à y passer du temps. Il est très rare qu'un script produise le résultat attendu dès la première rédaction, les premiers tests ou les premières opérations de débogage.

Il est donc conseillé de commencer de manière progressive et de tester vos scripts fréquemment. Dès qu'une partie d'un script produit l'effet escompté, rédigez la partie suivante, et ainsi de suite. Cette méthode vous permet d'identifier rapidement les bogues et garantit la précision de vos scripts au fur et à mesure que leur complexité augmente.

La fenêtre Script propose un certain nombre de fonctionnalités qui vous permettent de créer et modifier vos scripts.

Dans Director, la fenêtre Script vous permet d'ajouter aux animations une interactivité avancée, basée sur les scripts. Dans la fenêtre Script, vous pouvez programmer à l'aide de Lingo ou de la syntaxe JavaScript. Lingo est le langage de programmation traditionnel de Director. La syntaxe JavaScript, quant à elle, a été récemment introduite afin de permettre aux développeurs multimédia qui préfèrent utiliser JavaScript d'utiliser cette syntaxe au sein de Director.

En créant des scripts dans la fenêtre Script, vous pouvez réaliser la plupart des tâches qui peuvent être accomplies dans l'interface graphique de Director, telles que le déplacement des images-objets sur la scène ou la lecture de sons. Cependant, le plus grand avantage de la création de scripts est surtout la souplesse que vous apportez à une animation. Au lieu de lire une série d'images exactement comme le scénario l'exige, les scripts peuvent contrôler la lecture de l'animation en fonction de conditions et d'événements précis.

Remarque : *en plus de la fenêtre Script, dans laquelle vous pouvez créer vos propres scripts, Director contient un jeu d'instructions toutes prêtes (appelées* comportements*), que vous pouvez simplement faire glisser sur les images-objets et les images. Les comportements permettent d'ajouter une dimension interactive basée sur les scripts, sans avoir à rédiger de scripts. Pour plus d'informations sur les comportements, consultez la rubrique Comportements dans l'Aide de Director.*

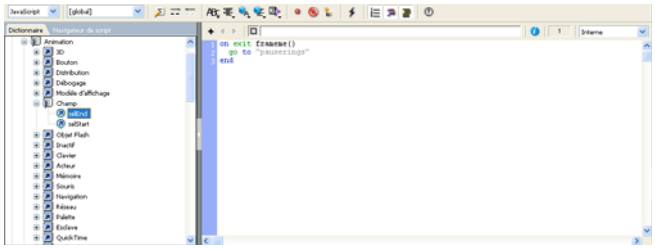
La fenêtre Script possède un panneau de l'explorateur et un Editeur de script. Par défaut, le panneau de l'explorateur apparaît à gauche de l'Editeur de script. Vous pouvez voir le panneau de l'explorateur dans la vue du dictionnaire ou dans celle du navigateur de scripts. Pour définir la position par défaut du panneau de l'explorateur, utilisez la boîte de dialogue Préférences de la fenêtre Script (Edition > Préférences > Script).

Mode dictionnaire

Le mode dictionnaire affiche une liste de fonctions de script Lingo/JavaScript intégrées organisées sous la forme d'une arborescence. Les fonctions sont classées selon leur catégorie et par ordre alphabétique comme un index.

Utilisez le mode dictionnaire pour effectuer les opérations suivantes :

- parcourir les fonctions intégrées des scripts Lingo et JavaScript ;
- utiliser les fonctions intégrées pour créer des scripts.



Rechercher des fonctions et créer des scripts à l'aide du mode dictionnaire

1 Choisissez Fenêtre > Script. La fenêtre Script apparaît.

2 Cliquez sur l'onglet Dictionnaire.

3 Sélectionnez Lingo ou JavaScript dans le menu local. Les fonctions intégrées correspondantes s'affichent dans le panneau ci-dessous.

Lingo : affiche des fonctions pour les scripts Lingo, les scripts Lingo 3D et les Xtras de programmation utilisés dans l'animation en cours. Les fonctions sont organisées en catégories (Globales, Animation, Lecteur, etc.).

JavaScript : affiche des fonctions pour les scripts JavaScript, les scripts JavaScript 3D et les Xtras de programmation utilisés dans l'animation en cours. Les fonctions sont organisées en catégories (Globales, Animation, Lecteur, etc.).

4 Agrandissez chaque catégorie pour afficher les fonctions qui lui sont associées en cliquant sur le signe plus (+) en regard de chaque catégorie. Pour afficher les fonctions par ordre alphabétique, agrandissez la catégorie de l'index.

5 Pour ajouter une fonction à l'Editeur de script pour la création de scripts, double-cliquez sur la fonction.

6 Enregistrez le script.

7 Cliquez sur l'icône Recompiler tous les scripts modifiés.

Mode navigateur de scripts

Le mode navigateur de scripts affiche les scripts et les gestionnaires associés qui ont été utilisés dans l'animation. Dans ce mode, vous pouvez créer de nouveaux scripts et gestionnaires.

Utilisez le mode navigateur de scripts pour effectuer les opérations suivantes :

- parcourir les scripts et les gestionnaires de l'animation actuelle comme une arborescence ou une liste ;
- trier les scripts en fonction du nom du script, du nom de l'acteur, du numéro de l'acteur ou du type de script dans l'affichage sous forme de liste ;
- localiser un gestionnaire dans l'Editeur de script ;
- créer des scripts pour chaque type de script ou acteur script.

Rechercher et créer des scripts à l'aide du mode navigateur de scripts

1 Choisissez Fenêtre > Script pour ouvrir la fenêtre Script.

2 Cliquez sur l'onglet Navigateur de scripts. Les scripts s'affichent sous la forme d'une arborescence dans le panneau ci-dessous. Pour afficher les scripts sous la forme d'une liste, cliquez sur le bouton de la vue du navigateur de scripts.

Affichage sous forme d'arborescence : les scripts sont classés selon le type de script (comme les scripts de comportement, les scripts d'animation, les scripts parents et la bibliothèque de distribution) dans lequel ils sont créés. Les scripts d'acteurs sont également répertoriés dans cette liste. Pour ouvrir un script dans l'Editeur de script, double-cliquez sur le script. Les gestionnaires appartenant à un script apparaissent sous la forme d'une arborescence sous le nœud de script correspondant.

3 Pour afficher une liste de gestionnaires compilés dans le script, agrandissez le nœud <nom> du script. Les gestionnaires non compilés ne s'affichent pas.

4 Pour localiser un gestionnaire dans l'Editeur de script, double-cliquez sur le gestionnaire. Le gestionnaire est mis en surbrillance dans l'Editeur de script. Vous pouvez également cliquer sur l'icône Passer au gestionnaire dans la barre de programmation. Pour plus d'informations, consultez « Recherche de gestionnaires et de texte dans les scripts », page 74.

5 Pour ajouter ou supprimer un commentaire dans le script, cliquez sur l'icône Insérer une marque de commentaire ou Supprimer la marque de commentaire dans la barre de programmation.

6 Pour activer ou désactiver un point d'arrêt, cliquez sur l'icône Activer/désactiver le point d'arrêt dans la barre de programmation. Vous pouvez également appuyer sur F9 ou cliquer sur la barre bleue en regard de la partie de code.

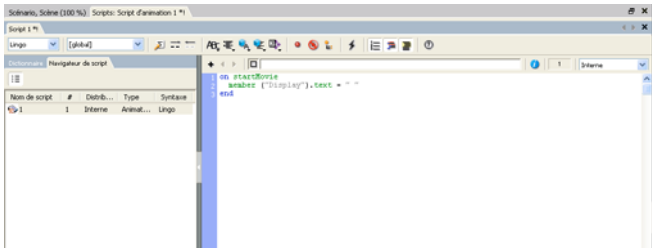
7 **Affichage sous forme de liste :** les scripts sont organisés dans une colonne sous la forme d'une liste en fonction du nom du script, du nom de l'acteur, du numéro de l'acteur et du type de script. Pour trier une liste dans une colonne, cliquez sur l'en-tête de colonne souhaité.

8 Pour créer un nouveau script dans l'affichage sous forme d'arborescence, cliquez avec le bouton droit sur un type de script et sélectionnez Ajouter un nouveau <type de script> dans le menu local. Lorsque vous saisissez un nom pour le script dans l'Editeur de script, le nom du script s'affiche dans la liste.

9 Enregistrez le script.

10 Cliquez sur l'icône Recompiler tous les scripts modifiés.

Les scripts d'acteurs sont répertoriés dans le dossier des scripts d'acteurs dans l'affichage sous forme d'arborescence et en tant que noms de scripts dans l'affichage sous forme de liste. Etant donné que ces scripts sont associés à un acteur spécifique et qu'ils ne sont pas réellement des acteurs, ils sont supprimés du navigateur de scripts uniquement lorsque vous supprimez l'acteur associé.



Pour plus d'informations sur les autres façons de créer et d'ouvrir des scripts, consultez « Exécution d'opérations élémentaires », page 75.

Remarque : *pour fermer un onglet Script, cliquez sur le bouton X dans l'onglet ou cliquez avec le bouton droit sur la zone de l'onglet puis sélectionnez l'option Fermer <type de script:nom du script>.*

Ouverture et fermeture de plusieurs scripts

Vous pouvez ouvrir plusieurs scripts sous forme d'onglets dans la fenêtre Script. Etant donné que la fenêtre Script ne peut afficher qu'un nombre limité d'onglets, certains onglets peuvent être masqués. Pour parcourir les onglets masqués, utilisez les icônes « > » et « < » permettant de parcourir les onglets de la fenêtre Script.

Ouvrir plusieurs fenêtres Script

❖ Effectuez l'une des opérations suivantes :

- Depuis la fenêtre Script, sélectionnez Fenêtre > Nouvelle fenêtre Script.
- Appuyez sur Alt+w+n.

Vous pouvez utiliser le raccourci Alt+w+n pour ouvrir des instances supplémentaires de n'importe quelle fenêtre active. Par exemple, si vous avez ouvert la fenêtre Vecteur, utilisez le raccourci Alt+w+n pour ouvrir des fenêtres Vecteur supplémentaires.

Fermer un onglet Script

1 Cliquez sur l'onglet de la fenêtre Script que vous souhaitez fermer.

2 Effectuez l'une des opérations suivantes :

- Cliquez sur l'onglet que vous souhaitez fermer (s'il n'est pas l'onglet actif), puis cliquez sur le bouton X.

• Cliquez avec le bouton droit sur l'onglet ou sur la zone en regard de l'onglet, puis sélectionner Fermer <type de script:nom du script>.

Définition des préférences de la fenêtre Script

Vous pouvez modifier la police du texte affiché dans la fenêtre Script et attribuer des couleurs différentes aux différents composants du code Lingo. Pour modifier la police par défaut du texte de la fenêtre Script et les couleurs des différents éléments du code, vous utilisez la fenêtre Script. Director affecte automatiquement une couleur distincte aux différents éléments du code, sauf si vous désactivez Coloration automatique.

Pour afficher le panneau de l'explorateur dans d'autres fenêtres, cliquez sur l'icône Flèche de la barre de fractionnement située entre l'éditeur de script et le panneau de l'explorateur.

Définir les préférences de la fenêtre Script

1 Choisissez Edition > Préférences > Script.

2 Pour choisir la police par défaut, cliquez sur le bouton Police et sélectionnez les attributs de la police dans la boîte de dialogue Police.

3 Pour choisir la couleur par défaut du texte affiché dans la fenêtre Script, choisissez une couleur dans la puce Couleur.

4 Pour choisir la couleur d'arrière-plan de la fenêtre Script, choisissez une couleur dans le menu Arrière-plan.

5 Pour que la fenêtre Script colorie automatiquement certains éléments du code, cochez la case Coloration automatique. Cette option est activée par défaut. Lorsque l'option Coloration automatique est désactivée, le texte a la couleur par défaut.

6 Pour que la nouvelle fenêtre Script formate automatiquement vos scripts avec la mise en retrait, activez l'option Format automatique. Cette option est activée par défaut.

Remarque : *les fonctions de coloration automatique et de formatage automatique ne s'appliquent pas aux codes JavaScript. Par conséquent, si vous créez des scripts à l'aide de la syntaxe JavaScript, les boutons Coloration automatique et Format automatique sont désactivés dans la fenêtre Script et les termes tels que* `function`*,* `var` *et* `this` *apparaissent avec la couleur par défaut.*

7 Pour que la nouvelle fenêtre Script affiche les numéros de ligne associés à vos scripts, activez l'option Numérotation des lignes. Cette option est activée par défaut.

8 Si l'option Coloration automatique est activée, choisissez les couleurs des éléments de code suivants :

- Mots-clés
- Commentaires
- Constantes
- Personnalisé (termes que vous définissez dans votre propre code)

9 Pour changer la couleur de fond de la colonne des numéros de ligne, cliquez sur Numérotation des lignes et choisissez une nouvelle couleur.

10 Pour changer l'emplacement des volets Pile d'appels, Variables et Surveillance dans la fenêtre Débogueur, sélectionnez Gauche, Droit, Haut ou Bas dans le menu Volets de débogage.

11 Sélectionnez Lingo ou JavaScript dans le menu local Type de script par défaut. Director utilise l'option sélectionnée par défaut lors de l'ouverture du panneau de l'explorateur

Remarque : *cette modification agit au niveau de l'application et reste en mémoire lorsque vous fermez puis ouvrez à nouveau Director.*

12 Pour placer le panneau de l'explorateur à un autre emplacement adjacent à l'éditeur de script dans la fenêtre Script, sélectionnez Gauche, Haut, Droit ou Bas dans le menu local du panneau de l'explorateur. Par défaut, le panneau de l'explorateur apparaît à gauche de l'Editeur de script.

Remarque : *lors du débogage d'un script, une instance distincte de l'éditeur de script s'ouvre avec le panneau Débogueur. Cette fenêtre de type Document peut être réorganisée dans l'application en la déplaçant avec les autres fenêtres ouvertes.*

Suppression de scripts

Vous pouvez supprimer des scripts à partir du panneau de l'explorateur.

Supprimer un script

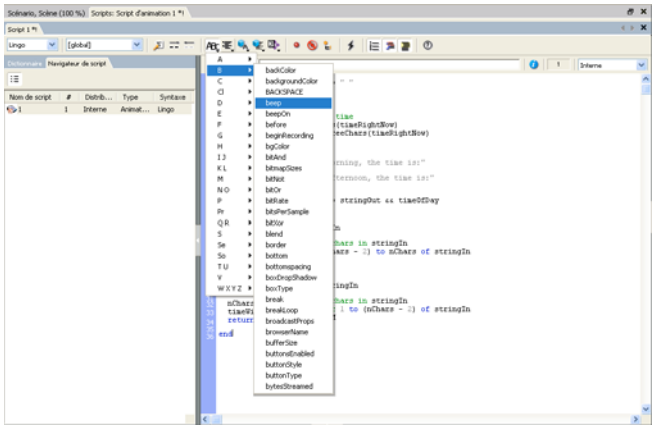
❖ Effectuez l'une des opérations suivantes :

- Dans le navigateur de script, cliquez avec le bouton droit sur une entrée de script, puis sur Supprimer.
- Dans l'affichage sous forme de liste de l'explorateur de script, sélectionnez un script, puis cliquez sur Supprimer.

Remarque : *lorsque vous supprimez un script déjà ouvert, l'onglet correspondant est également supprimé de la fenêtre Script. Si le script ouvert que vous avez supprimé était le seul script de votre animation, la fenêtre Script est désactivée, et l'onglet dans lequel s'affichait votre script indique la mention* Script. *Pour activer l'éditeur de texte, cliquez sur le signe plus (+) situé dans le panneau de l'éditeur de script.*

Insertion de termes de programmation communs

La fenêtre Script offre des menus locaux contenant les termes de programmation communs que vous pouvez utiliser pour insérer des instructions dans un script. Les mêmes menus sont également disponibles dans la fenêtre Messages.



Dans les fenêtres Script et Messages, vous pouvez sélectionner la syntaxe de programmation à utiliser pour un script donné.

Sélectionner la syntaxe de programmation

❖ Dans le menu local Syntaxe de script, sélectionnez Lingo ou JavaScript.

Après avoir sélectionné une syntaxe de programmation, entrez le code dans la syntaxe que vous avez choisie. Si vous essayez de compiler un script dans une syntaxe autre que celle que vous avez choisie, vous obtenez une erreur de script.

Lorsque vous entrez des scripts dans la fenêtre Script, vous pouvez insérer ou supprimer des marques de commentaire sur une seule ou plusieurs lignes de code à l'aide des boutons Insérer une marque de commentaire et Supprimer la marque de commentaire. Selon la syntaxe choisie, les boutons Insérer une marque de commentaire et Supprimer la marque de commentaire affichent les bons repères de commentaire pour cette syntaxe ; Lingo utilisant des tirets doubles (`--`) et la syntaxe JavaScript deux barres obliques (`//`).

Insérer une marque de commentaire dans le code

❖ Mettez en surbrillance la ou les lignes de code auxquelles vous voulez ajouter un commentaire et cliquez sur Insérer une marque de commentaire.

Remarque : *vous utilisez le bouton Insérer une marque de commentaire pour ajouter des commentaires sur plusieurs lignes de code JavaScript, Director place deux barres obliques avant chaque ligne. Vous pouvez également insérer des commentaires sur plusieurs lignes de code en tapant le signe /* avant la première ligne de code et le signe */ après la dernière ligne ; mais vous devez le faire manuellement.*

Supprimer une marque de commentaire dans le code

❖ Mettez en surbrillance la ou les lignes de code dont vous voulez supprimer les commentaires et cliquez sur Supprimer la marque de commentaire.

Les fenêtres Script et Messages contiennent toutes deux les menus suivants :

- Le menu alphabétique est une liste alphabétique de tous les éléments Lingo, à l'exception du Lingo 3D.
- Le menu par catégorie est une liste des éléments Lingo répertoriés selon leurs fonctions. Il n'inclut pas les éléments Lingo 3D.
- Le menu alphabétique 3D de Lingo est une liste alphabétique de tous les éléments Lingo 3D, présentés par ordre alphabétique.
- Le menu par catégorie 3D de Lingo est une liste de tous les éléments Lingo 3D répertoriés selon leurs fonctions.
- Le menu local des Xtras de programmation inclut les méthodes et propriétés de tous les Xtras de programmation trouvés, qu'il s'agisse d'Xtras Adobe ou autres.

Remarque : *les Xtras de programmation figurant dans le menu local sont limités à ceux qui prennent en charge la méthode* `Interface()` *et dont les noms apparaissent effectivement dans le menu local. Bien que certains types de médias d'acteurs tels 3D et DVD prennent également en charge la méthode* `Interface()`*, ils ne figurent pas dans le menu local Xtras de programmation parce qu'ils ne sont pas implémentés en tant qu'Xtras de programmation dans Director.*

L'élément sélectionné dans les menus locaux est inséré par Director à l'emplacement du curseur dans la fenêtre Script.

Si un élément nécessite des paramètres supplémentaires, des repères de noms indiquant les informations supplémentaires requises sont insérés. Lorsque plusieurs arguments ou paramètres sont nécessaires, le premier est mis en surbrillance pour vous inviter à le saisir et le remplacer. Vous devez sélectionner et remplacer les autres paramètres vous-même.

Certains types d'acteurs et d'Xtras de programmation offrent des termes de programmation qui ne figurent pas dans les menus locaux. Ces types d'acteurs et d'Xtras possèdent généralement leur propre documentation ; vous pouvez également trouver des informations à partir de Director.

Afficher la liste des Xtras disponibles

❖ Emettez soit `put(_player.xtraList)`, soit `trace(_player.xtraList)` dans la fenêtre Messages.

Afficher la liste des Xtras de programmation disponibles

❖ Emettez soit `put(_player.scriptingXtraList)`, soit `trace(_player.scriptingXtraList)` dans la fenêtre Messages.

Afficher la liste des méthodes et propriétés d'un Xtra

❖ Dans le menu local Xtras de programmation, placez-vous sur un Xtra et, dans le menu secondaire, cliquez sur Put Interface. Les méthodes et propriétés de cet Xtra apparaissent dans la fenêtre Messages.

Saisie et modification de texte

La saisie et la modification de texte dans une fenêtre Script se font de la même manière que dans n'importe quel champ.

Les opérations d'édition les plus communes effectuées dans une fenêtre Script sont les suivantes :

- Pour sélectionner un mot, double-cliquez dessus.
- Pour sélectionner un script entier, choisissez Edition > Tout sélectionner.
- Pour commencer une nouvelle ligne, entrez un retour chariot.
- Dans Lingo, pour renvoyer une longue ligne de code à la ligne en insérant un symbole de continuation, appuyez sur Alt+Entrée (Windows®) ou sur Option+Retour (Macintosh®) à l'endroit où vous voulez insérer un retour à la ligne. Le symbole de continuation (\) qui apparaît indique que l'instruction continue sur la ligne suivante.

  Pour renvoyer une longue ligne de code à la ligne dans la syntaxe JavaScript, insérez un saut de ligne en appuyant sur Entrée (Windows) ou Retour (Macintosh). Le symbole de continuation Lingo entraîne une erreur de script dans les scripts de la syntaxe JavaScript.

- Pour trouver un gestionnaire dans le script actuel, choisissez son nom dans le menu local Passer au gestionnaire de la fenêtre Script.
- Pour compiler les scripts que vous venez de modifier, cliquez sur le bouton Recompiler tous les scripts modifiés de la fenêtre Script ou fermez cette dernière. Lorsque vous modifiez un script, un astérisque apparaît dans la barre de titre de la fenêtre Script, indiquant que le script doit être recompilé.
- Pour compiler tous les scripts d'une animation, sélectionnez Recompiler tous les scripts dans le menu Contrôle.
- Pour reformater un script avec la mise en retrait, appuyez sur la touche de tabulation dans la fenêtre Script.

  Director place automatiquement les instructions en retrait lorsque leur syntaxe est correcte. Si une ligne du script n'est pas correctement mise en retrait, la syntaxe de cette ligne est incorrecte.

- Pour ouvrir une seconde fenêtre Script, appuyez sur la touche Alt (Windows) ou Option (Macintosh), tout en cliquant sur le bouton Nouvel acteur dans la fenêtre Script. Cette opération peut notamment se révéler utile lorsque vous modifiez simultanément deux sections différentes d'un long script.
- Pour activer ou désactiver la numérotation des lignes, cliquez sur le bouton Numérotation des lignes.
- Pour activer ou désactiver l'option Coloration automatique, cliquez sur le bouton Coloration automatique. L'option Coloration automatique affiche chaque type d'élément Lingo (propriétés, commandes, etc.) dans une couleur différente.
- Pour activer ou désactiver le formatage automatique, cliquez sur le bouton Format automatique. L'option Format automatique applique la mise en retrait correcte à vos scripts chaque fois que vous ajoutez un retour chariot ou que vous appuyez sur la touche de tabulation.

Remarque : *les fonctions Coloration automatique et Format automatique ne s'appliquent pas aux codes JavaScript. Par conséquent, si vous créez des scripts à l'aide de la syntaxe JavaScript, les boutons Coloration automatique et Format automatique sont désactivés dans la fenêtre Script et les termes tels que* `function`, `var` *et* `this` *apparaissent avec la couleur par défaut.*

Recherche de gestionnaires et de texte dans les scripts

La commande Rechercher du menu Edition permet de rechercher des gestionnaires et de rechercher et modifier du texte et des gestionnaires dans les scripts.

Rechercher des gestionnaires dans les scripts

1. Choisissez Edition > Rechercher > Gestionnaire.

   La boîte de dialogue Rechercher un gestionnaire apparaît.

   La colonne la plus à gauche de la boîte de dialogue Rechercher un gestionnaire affiche les noms de tous les gestionnaires de l'animation. La colonne du milieu affiche le numéro de l'acteur associé au script du gestionnaire, ainsi que son nom. La colonne la plus à droite affiche la distribution dans laquelle se trouve l'acteur.

2. Sélectionnez le gestionnaire à rechercher.
3. Cliquez sur Rechercher.

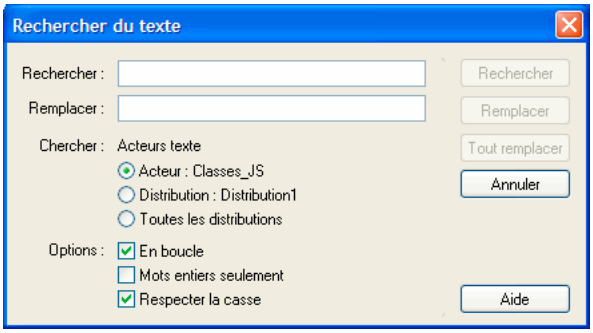
   Le gestionnaire apparaît dans la fenêtre Script.

   La barre de titre en haut de la fenêtre Script indique le type du script.

Rechercher du texte dans les scripts

1. Activez la fenêtre Script.
2. Choisissez Edition > Rechercher > Texte.

   La boîte de dialogue Rechercher du texte apparaît.



3. Saisissez le texte à rechercher dans le champ Rechercher, puis cliquez sur Rechercher.

   Par défaut, la recherche ne fait pas la distinction entre les majuscules et les minuscules : `ThisHandler`, `thisHandler` et `THISHANDLER` sont considérés comme identiques lors de la recherche. Cliquez sur la case Respecter la casse pour que la recherche prenne en compte les majuscules et les minuscules.

Spécifier les acteurs dans lesquels effectuer la recherche

- Sélectionnez l'option appropriée dans Chercher : Scripts.

Reprendre la recherche au début une fois qu'elle atteint la fin

❖ Sélectionnez l'option En boucle.

Ne rechercher que des mots entiers et non des fragments de mots correspondant au mot recherché

❖ Sélectionnez l'option Mots entiers seulement.

Rechercher l'occurrence suivante du texte spécifié dans le champ Rechercher

❖ Choisissez Edition > Poursuivre la recherche.

Trouver toutes les occurrences du texte sélectionné

1 Sélectionnez le texte.

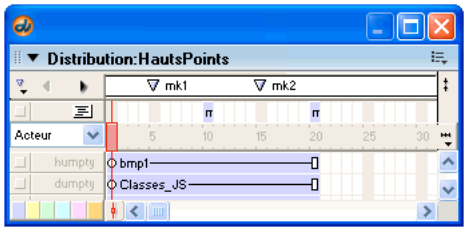
2 Choisissez Edition > Rechercher > Sélection.

Exécution d'opérations élémentaires

Vous trouverez ci-dessous les opérations élémentaires permettant de créer, associer et ouvrir des scripts.

Créer un comportement d'image (script associé à une image)

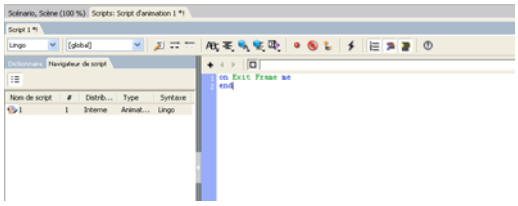
❖ Double-cliquez sur la piste des comportements dans l'image à laquelle vous souhaitez affecter le comportement.



***A.** Pistes des comportements*

Lorsque vous créez un nouveau comportement, celui-ci reçoit automatiquement le premier numéro de distribution disponible dans la fenêtre Distribution active.

Lorsque vous créez un nouveau comportement d'image, la fenêtre Script apparaît et contient automatiquement le gestionnaire Lingo `on exitFrame`. La première ligne contient l'expression `on exitFrame`, suivie d'une ligne dans laquelle le curseur clignote puis d'une autre comprenant le mot `end`. Cela vous permet d'associer facilement et rapidement un comportement Lingo commun à l'image. Pour que ce gestionnaire puisse fonctionner avec la syntaxe JavaScript, remplacez `on exitFrame` par `function exitFrame() {` puis `end` par `}`.



Un des comportements d'image les plus fréquents est celui qui maintient la tête de lecture en boucle dans une même image. Il peut s'avérer utile lorsque vous souhaitez que votre animation maintienne la lecture sur une même image en attendant que l'utilisateur clique sur un bouton ou dans l'attente de la fin de la lecture d'une vidéo numérique ou d'un fichier audio.

Maintenir la tête de lecture sur une seule image

❖ Dans un comportement d'image, tapez l'instruction suivante sur la ligne qui suit directement l'instruction `on exitFrame` (Lingo) ou `function exitFrame()` (syntaxe JavaScript) :

```
-- Lingo syntax
_movie.go(_movie.frame)

// JavaScript syntax
_movie.go(_movie.frame);
```

La propriété `frame` de l'objet Animation se rapporte à l'image actuellement occupée par la tête de lecture. Cette instruction demande essentiellement à la tête de lecture de « revenir au niveau de l'image active ».

Créer un comportement d'image-objet (script associé à une image-objet)

❖ Dans le scénario ou sur la scène, sélectionnez l'image-objet à laquelle vous souhaitez associer le comportement. Choisissez ensuite Fenêtre > Inspecteur de comportement et choisissez Nouveau comportement dans le menu local Comportements.

Lorsque vous créez un nouveau comportement d'image-objet, la fenêtre Script apparaît et contient automatiquement le gestionnaire Lingo `on mouseUp`. La première ligne contient l'expression `on mouseUp`, suivie d'une ligne dans laquelle le curseur clignote puis d'une autre comprenant le mot `end`. Ceci vous permet d'associer rapidement et aisément un comportement commun à l'image-objet. Pour que ce gestionnaire puisse fonctionner avec la syntaxe JavaScript, remplacez `on mouseUp` par `function mouseUp() {` puis `end` par `}`.

Ouvrir un comportement afin de le modifier

1 Double-cliquez sur le comportement dans la fenêtre Distribution.

L'Inspecteur de comportement apparaît.

2 Cliquez sur l'icône Script de l'acteur dans l'Inspecteur de comportement.

La fenêtre Script de l'acteur affiche le comportement.

Vous pouvez également ouvrir la fenêtre Script d'animation et faire défiler les scripts jusqu'à ce que le comportement recherché apparaisse.

Supprimer un comportement d'un emplacement dans le scénario

❖ Sélectionnez l'emplacement, puis supprimez le script de la liste affichée dans l'Inspecteur des propriétés (onglet Comportement).

Associer des comportements existants à des images-objets ou à des images

❖ Effectuez l'une des opérations suivantes :

• Faites glisser un comportement d'une distribution vers une image-objet ou une image du scénario ou (pour les images-objets) vers une image-objet de la scène.

• Dans le scénario, sélectionnez les images-objets ou les images auxquelles vous souhaitez associer le comportement. Choisissez ensuite Fenêtre > Inspecteur de comportement et choisissez le comportement existant dans le menu-local Comportements.

Créer un script d'animation (script associé à une animation)

Effectuez l'une des opérations suivantes :

• Pour créer un script d'animation à l'aide de l'explorateur Script dans l'affichage sous forme d'arborescence :

1 Sélectionnez un nœud d'animation sous la distribution dans laquelle vous voulez ajouter le script d'animation.

2 Cliquez avec le bouton droit et sélectionnez Ajouter > Nouveau script d'animation.

• Si le script actuel de la fenêtre Script est un script d'animation, cliquez sur le bouton Nouveau script de cette fenêtre. Le bouton Nouveau script crée toujours un script du même type que le script actuel.

• Si le script actuel de la fenêtre Script n'est pas un script d'animation, cliquez sur le bouton Nouveau script puis remplacez le type du nouveau script à l'aide du menu local Type de l'onglet Script de l'Inspecteur des propriétés.

• Si aucune image-objet ni aucun script n'est sélectionné dans la distribution, dans le scénario ou sur la scène, ouvrez une nouvelle fenêtre Script. Par défaut, un nouveau script d'animation est créé.

Ouvrir un script d'animation ou un script parent afin de le modifier

❖ Une fois le script ouvert à l'aide de l'explorateur Script, double-cliquez sur le script dans la fenêtre Distribution.

Modifier le type d'un script

1 Sélectionnez le script dans la fenêtre Distribution ou ouvrez-le dans la fenêtre Script.

2 Cliquez sur l'onglet Script de l'Inspecteur des propriétés et choisissez un type de script dans le menu-local Type.

Faire défiler les scripts dans la fenêtre Script

❖ Utilisez les flèches Acteur précédent et Acteur suivant situées en haut de la fenêtre Script pour vous déplacer vers l'avant ou l'arrière dans les scripts.

Copier un script

❖ Sélectionnez le script dans la fenêtre Distribution, puis choisissez Dupliquer dans le menu Edition.

Pour créer un script automatiquement associé à chaque image-objet créée à partir d'un acteur spécifique, associez le script à l'acteur proprement dit.

Créer un script associé à un acteur ou ouvrir un script associé à un acteur existant

❖ Effectuez l'une des opérations suivantes :

• Cliquez avec le bouton droit de la souris (Windows) ou cliquez en maintenant la touche Ctrl enfoncée (Macintosh) sur un acteur dans la fenêtre Distribution, puis choisissez Script dans le menu contextuel.

• Sélectionnez un acteur dans la fenêtre Distribution et cliquez sur le bouton Script de l'acteur dans la fenêtre Distribution.

Utilisation de scripts liés

En plus des scripts stockés sous la forme d'acteurs internes, vous pouvez placer des scripts dans des fichiers texte externes et les lier à votre animation Director. Ces scripts liés sont similaires aux fichiers d'images ou de vidéo numérique que vous pouvez importer dans une animation Director.

Parmi les avantages découlant de l'utilisation de scripts liés, citons les suivants :

• Une personne peut travailler sur le fichier Director alors qu'une autre travaille sur le script.

• Il est facile d'échanger des scripts avec d'autres personnes.

• Vous pouvez contrôler les scripts séparément du fichier Director, dans une application de contrôle de code source telle que Microsoft® Visual SourceSafe® ou Perforce® de Perforce Software. Ce type d'applications évite que les différents programmeurs qui travaillent ensemble sur un projet Director écrasent le travail des autres.

Les scripts liés ne sont utilisés par Director qu'en cours de création. A l'exécution, les projections Director et Adobe® Shockwave® Player utilisent une copie interne spéciale des données du script stockée dans l'animation. De la sorte, il n'est pas nécessaire de distribuer vos scripts liés avec vos animations, et il est impossible à l'utilisateur final de les copier.

Importer un script sous la forme d'un fichier texte lié

1. Choisissez Fichier > Importer.
2. Choisissez Script comme type de fichier à importer.
3. Sélectionnez les fichiers de script que vous souhaitez importer.

   Vous pouvez importer des fichiers possédant les extensions .txt, .ls ou .js. L'extension .ls est l'extension désignant les scripts liés de Director.

   Pour créer une liste des fichiers à importer, vous pouvez utiliser les boutons Ajouter et Tout ajouter. Une telle liste est notamment utile si vous souhaitez importer des scripts de plusieurs endroits différents.
4. Choisissez Lier au fichier externe.
5. Cliquez sur Importer.

Vous pouvez modifier les scripts liés de manière normale dans la fenêtre Script de Director. Les modifications que vous apportez sont écrites dans les fichiers externes à chaque fois que vous enregistrez l'animation Director. Si vous avez importé le script lié depuis un serveur UNIX®, les fins de ligne UNIX sont préservées. Si vous importez un script dont le fichier texte est verrouillé, il vous est impossible de le modifier dans Director.

Il est impossible d'appliquer des couleurs de texte personnalisées aux scripts liés dans la fenêtre Script. Par contre, la fonction Coloration automatique des scripts est activée pour les scripts liés.

Transformer un acteur script interne en acteur script lié externe

1. Sélectionnez l'acteur interne, puis cliquez sur l'onglet Script de l'Inspecteur des propriétés.
2. Cliquez sur Lier le script sous.
3. Entrez le nom du fichier dans la boîte de dialogue Enregistrer le script sous.
4. Cliquez sur Enregistrer.

Recharger un script lié après sa modification

❖ Utilisez la méthode `unload()` de l'objet Acteur.

Si un script lié est modifié en dehors de Director, vous pouvez le recharger avec la méthode `unload()` dans la fenêtre Messages. L'instruction suivante permet de purger l'acteur script `monScript` puis de le recharger :

```
-- Lingo syntax
member("myScript").unload()

// JavaScript syntax
member("myScript").unload();
```

Chapitre 4 : Débogage de scripts dans Director

Les scripts ne répondent pas toujours immédiatement aux instructions. Chaque script présente souvent une erreur de syntaxe : il s'agit généralement d'un mot mal écrit ou d'une partie du script absente. Il arrive aussi que le script fonctionne mais ne produise pas le résultat escompté. Des erreurs ou des bogues survenant presque toujours lors de la rédaction de scripts, il est recommandé de prévoir le temps nécessaire au débogage lors du développement des projets multimédia.

Au fur et à mesure de votre apprentissage, vous rencontrerez probablement d'autres types de problèmes, car lorsque vous maîtriserez un sujet, vous commencerez seulement à découvrir les autres. Toutefois, les principales techniques de dépannage présentées dans ce chapitre sont destinées à la fois aux utilisateurs débutants et expérimentés.

Le meilleur moyen de corriger un bogue dans vos scripts dépend d'une situation à l'autre. Il n'existe pas de procédure standard permettant de résoudre un problème. Vous devez utiliser plusieurs des outils et techniques présentés ci-après :

- Présentation générale et détaillée de l'interaction des scripts dans une animation
- Expérimentation et pratique des principales méthodes de débogage

Les outils suivants sont destinés à vous aider à identifier les problèmes dans les scripts :

- Lorsque la fonction de suivi est activée, la **fenêtre Messages** affiche un enregistrement des images lues et des gestionnaires en cours d'exécution dans l'animation.
- La **fenêtre Débogueur** affiche les valeurs des variables globales, les propriétés du script actuellement en cours d'exécution, la séquence de gestionnaires exécutée pour parvenir au niveau actuel, ainsi que la valeur des variables et des expressions que vous avez sélectionnées.
- La **fenêtre Script** vous permet de saisir des commentaires, d'insérer des points d'arrêt dans le script et de sélectionner des variables dont la valeur apparaît dans l'Inspecteur d'objet.
- L'**Inspecteur d'objet** vous permet d'afficher et de définir les valeurs des objets et des propriétés que vous avez sélectionnés.

Bonnes habitudes de rédaction de scripts

De bonnes habitudes de rédaction de scripts vous permettent d'éviter dès le départ bon nombre de problèmes de programmation.

- Veillez à rédiger les scripts par petits lots et testez directement chacun de vos scripts, au fur et à mesure que vous les créez. Cette procédure permettra d'isoler les éventuels problèmes afin de les identifier plus facilement.
- Insérez des commentaires expliquant l'objectif des instructions et des valeurs du script. Le script est alors plus facile à comprendre lorsque vous y reviendrez ultérieurement ou qu'un autre utilisateur devra l'utiliser. Par exemple, les commentaires des instructions suivantes indiquent l'objectif de la structure `if…then` et clarifient la répétition de la boucle :

```
-- Lingo syntax
-- Loop until the "s" key is pressed
```

```
repeat while not(_key.keyPressed("s"))
    _sound.beep()
end repeat

// JavaScript syntax
// Loop until the "s" key is pressed
while(!_key.keyPressed("s")) {
    _sound.beep();
}
```

• Vérifiez si la syntaxe du script est correcte. Utilisez les menus locaux de la fenêtre Script pour insérer des versions préformatées des éléments de programmation. Consultez les rubriques des API de cette référence pour vérifier si les instructions sont rédigées correctement.

• Utilisez les noms de variable, qui indiquent l'objectif d'une variable. Par exemple, une variable contenant un nombre devrait porter un nom tel que `newNumber` plutôt que `ABC`.

Opérations de débogage de base

Le processus de débogage implique des étapes de stratégie et d'analyse, et non une procédure standard rigoureuse. Cette section décrit les approches de débogage fondamentales utiles aux programmeurs pour déboguer tous les types de code, et pas uniquement le code Lingo ou la syntaxe JavaScript.

Avant d'apporter une modification majeure à une animation, veillez à toujours en faire une copie de sauvegarde. Il est recommandé de nommer les copies par incréments (par exemple, nomDeFichier_01.dir, nomDeFichier_02.dir, nomDeFichier_03.dir, etc.) afin de pouvoir suivre les diverses étapes d'une animation.

Identification du problème

Cela peut paraître évident, mais rappelons que la première chose à faire, lors d'une procédure de débogage, est d'identifier le problème. La fonction d'un bouton est-elle faussée ? L'animation accède-t-elle à une autre image que celle prévue ? La modification d'un champ s'avère-t-elle impossible ?

Essayez aussi de déterminer ce que vous attendez d'un script donné et comparez ensuite la fonction escomptée de ce script avec sa fonction réelle. Cette opération vous aide à déterminer clairement votre objectif et les parties de cet objectif qui n'ont pas été réalisées.

Si vous avez copié un script ou une partie d'un script à partir d'une autre animation ou d'un exemple écrit, vérifiez si ce script a été conçu pour des conditions spécifiques. Il est peut-être nécessaire qu'une piste d'image-objet soit déjà programmée. Il se peut également que les noms d'acteur doivent suivre une convention stylistique spécifique.

Localisation du problème

Pour localiser un problème, procédez comme suit :

• Revenez en arrière pour localiser l'emplacement de l'apparition du problème.

• Utilisez la fenêtre Messages pour suivre les images parcourues par l'animation et identifier les gestionnaires exécutés par vos scripts.

• Déterminez le comportement présumé des scripts et ce qui, dans ces instructions, est associé au problème. Par exemple, si un acteur texte ne peut pas être édité alors qu'il devrait l'être, localisez l'emplacement de la propriété `editable` de l'acteur dans vos scripts.

- Si vous ne parvenez pas à modifier comme prévu une image-objet sur la scène, vérifiez si la méthode `updateStage()` n'est pas requise à un emplacement précis.

- Vérifiez si le problème survient sur tous les ordinateurs ou sur un seul. Tâchez de définir si le problème survient uniquement lorsque l'affichage est réglé sur millions de couleurs. Il se peut qu'un élément de l'ordinateur interfère avec l'application.

Concentrez-vous sur des lignes de script précises en insérant un point d'arrêt, c'est-à-dire un point où le script interrompt son exécution et appelle la fenêtre Débogueur, dans une ligne. Ceci vous permettra d'analyser les conditions à ce point précis, avant de poursuivre l'opération du script. Pour plus d'informations sur l'insertion de points d'arrêt dans un script, consultez « Débogage dans la fenêtre Débogueur », page 90.

Résolutions de problèmes simples

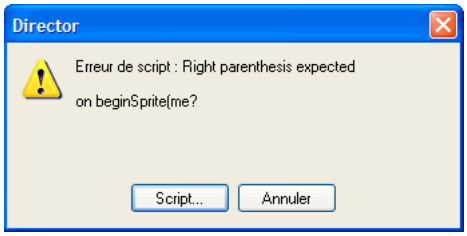
Lorsque vous découvrez un bogue, consultez tout d'abord les résolutions de problèmes simples.

Le premier test de débogage s'effectue lorsque vous compilez votre script. Pour compiler votre script, utilisez l'une des méthodes suivantes :

- Dans la fenêtre Script, cliquez sur Recompiler tous les scripts modifiés.
- Dans le menu Contrôle, cliquez sur Recompiler tous les scripts.
- Appuyez sur Maj+F8.
- Fermez la fenêtre Script.

Il est généralement recommandé de compiler des scripts en utilisant l'une des trois premières méthodes. Pour pouvoir utiliser la quatrième option, vous devez fermer la fenêtre Script à chaque fois que vous voulez compiler un script.

Lorsque vous compilez votre script, Director® présente un message d'erreur si le script contient une syntaxe incorrecte. Le message affiche généralement la ligne dans laquelle le problème a été détecté initialement. Un point d'interrogation apparaît au point précis où Director a initialement détecté le problème.



Par exemple, la première ligne du message précédent vous indique que l'erreur en question est une erreur de syntaxe et vous donne une explication. La deuxième ligne du message d'erreur affiche la ligne de code contenant l'erreur de syntaxe.

Recherche d'erreurs de syntaxe

Les erreurs de syntaxe sont certainement à l'origine des bogues les plus actuels dans la programmation. Lorsqu'un script échoue, il est recommandé de vérifier immédiatement les points suivants :

- Les termes sont écrits correctement, les espaces sont placés aux endroits appropriés et la ponctuation correcte est utilisée. Director ne peut pas interpréter une syntaxe incorrecte.
- Des guillemets sont placés de part et d'autre des noms d'acteurs, des libellés et des chaînes dans l'instruction.

• Tous les paramètres requis sont présents. A chaque élément doivent être associés des paramètres spécifiques. Consultez les rubriques consacrées aux API dans cette référence pour déterminer si un élément nécessite d'autres paramètres.

Recherche de bogues simples

Si votre script se compile sans afficher de message d'erreur, il risque de contenir un bogue. Si votre script ne produit pas les résultats escomptés, vérifiez les points suivants :

• Les valeurs des paramètres sont-elles correctes ? Par exemple, l'utilisation d'une valeur incorrecte pour le nombre de bips sonores que doit générer la méthode `beep()` produit un autre nombre de bips.

• Les valeurs sujettes à modifications, telles que les variables et le contenu d'acteurs texte, ont-elles les valeurs escomptées ? Vous pouvez afficher ces valeurs dans l'Inspecteur d'objet en sélectionnant le nom de l'objet, puis en cliquant sur Inspecteur d'objet dans la fenêtre Script ou, dans la fenêtre Messages, en utilisant les fonctions `put()` ou `trace()`.

• Les éléments de programmation produisent-ils les résultats escomptés ? Vous pouvez examiner leur comportement en vous reportant aux rubriques consacrées aux API dans cette référence.

• La casse est-elle correcte (si le script est rédigé dans la syntaxe JavaScript) ? La syntaxe JavaScript est sensible à la casse, ce qui signifie que les méthodes, fonctions, propriétés et variables doivent être saisies en respectant l'emploi des majuscules et minuscules.

Si vous appelez une méthode ou fonction en ne respectant pas la casse, vous obtenez une erreur de script.

Si vous essayez d'accéder à une variable ou une propriété en utilisant une casse incorrecte, il est possible que vous ne receviez pas d'erreur, mais votre script risque de ne pas avoir le comportement voulu. Par exemple, le gestionnaire `mouseUp` suivant contient une instruction qui essaie d'accéder à la propriété `itemLabel` en utilisant une casse incorrecte. Ce script ne renverra pas d'erreur, mais créera automatiquement une nouvelle variable contenant une casse incorrecte. La valeur de la nouvelle variable est `undefined`.

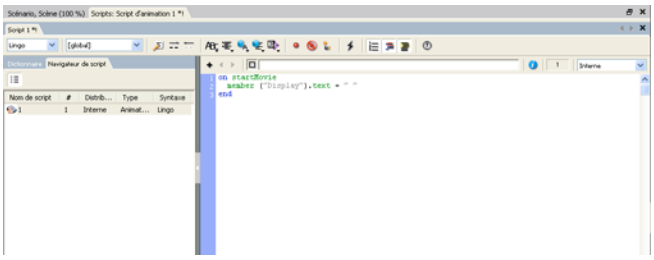
```
// JavaScript syntax
function beginSprite() {
    this.itemLabel = "Blue prints";
}

function mouseUp() {
    trace(this.itemlabel) // creates the itemlabel property
}
```

Débogage dans la fenêtre Script

La fenêtre Script propose un certain nombre de fonctions qui vous permettent de déboguer vos scripts.

Ouverture de la fenêtre Script :

❖ Choisissez Fenêtre > Script.



Rédaction d'un commentaire associé à la ligne de code actuelle :

❖ Cliquez sur Insérer une marque de commentaire.

Supprimer le commentaire de la ligne de code actuelle :

❖ Cliquez sur Supprimer la marque de commentaire.

Activer ou désactiver des points d'arrêt dans la ligne de code actuelle :

❖ Cliquez sur Activer/désactiver le point d'arrêt.

Désactiver tous les points d'arrêt :

❖ Cliquez sur Ignorer les points d'arrêt.

Ajouter l'expression ou la variable sélectionnée à l'Inspecteur d'objet :

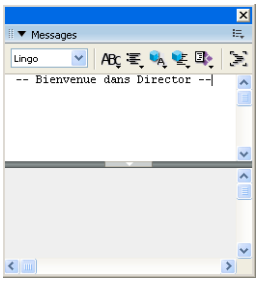
❖ Cliquez sur Inspecteur d'objet.

Débogage dans la fenêtre Messages

La fenêtre Messages vous permet de tester les commandes de programmation et d'en contrôler le processus lors de la lecture d'une animation.

Ouvrir la fenêtre Messages :

❖ Choisissez Fenêtre > Messages.



Gestion de la fenêtre Messages

La fenêtre Messages contient un volet de saisie et un volet de résultat. Le contenu du volet de saisie est modifiable. Le contenu du volet de résultat est en lecture seule. Le seul moyen d'afficher le texte dans le volet de résultat est d'appeler la fonction `put()` ou `trace()`.

Vous pouvez ajuster la taille des volets de saisie et de résultat en faisant glisser le séparateur horizontal situé entre les deux volets.

Redimensionner le volet de résultat

❖ Faites glisser le séparateur horizontal vers un nouvel emplacement.

Masquer complètement le volet de résultat

❖ Cliquez sur le bouton Réduire/Agrandir, au centre du séparateur horizontal.

Lorsque le volet de résultat est masqué, les sorties des scripts en cours d'exécution sont affichées dans le volet de saisie.

Afficher le volet de résultat lorsqu'il est masqué

❖ Cliquez de nouveau sur le bouton Réduire/Agrandir.

Supprimer le contenu de la fenêtre Messages

❖ Cliquez sur le bouton Effacer.

Si le volet de résultat est visible, son contenu est effacé.

Si le volet de résultat n'est pas visible, le contenu du volet de saisie est effacé.

Effacer une partie du contenu du volet de résultat

1. Sélectionnez le texte à effacer.
2. Appuyez sur la touche Retour arrière ou Effacement.

Copier un texte dans le volet de saisie ou de résultat

1. Sélectionnez le texte.
2. Choisissez Edition > Copier.

Test de scripts dans la fenêtre Messages

Vous pouvez tester les instructions Lingo et JavaScript pour vérifier leur fonctionnement en les saisissant dans la fenêtre Messages et en observant les résultats. Lorsque vous saisissez une instruction dans la fenêtre Messages, Director exécute la commande immédiatement, qu'une animation soit ou non en cours d'exécution.

Avant de saisir les instructions que vous voulez tester, vous devez d'abord sélectionner la syntaxe de programmation (Lingo ou JavaScript) à tester.

Sélectionner la syntaxe de programmation

❖ Dans le menu local Syntaxe de script, sélectionnez Lingo ou JavaScript.

Tester une instruction d'une ligne

1. Saisissez directement l'instruction dans la fenêtre Messages.
2. Appuyez sur Entrée (Windows®) ou sur Retour (Mac®). Director exécute l'instruction.

Si l'instruction est valide, la fenêtre Messages en affiche le résultat dans le volet de résultat, en bas de l'écran. Si le script n'est pas valide, un message d'erreur apparaît.

Par exemple, si vous saisissez l'instruction ci-après dans la fenêtre Messages :

```
-- Lingo syntax
put(50+50)

// JavaScript syntax
trace(50+50);
```

puis que vous appuyez sur la touche Entrée (Windows) ou Retour (Macintosh), le résultat apparaît dans le volet de résultat :

```
-- Lingo syntax
-- 100

// JavaScript syntax
// 100
```

Si vous saisissez l'instruction suivante dans la fenêtre Messages :

```
-- Lingo syntax
_movie.stage.bgColor = 255

// JavaScript syntax
_movie.stage.bgColor = 255;
```

puis que vous appuyez sur la touche Entrée (Windows®) ou Retour (Mac®), la scène apparaît en noir.

Vous pouvez tester plusieurs lignes de code en une seule opération en copiant et collant des instructions dans la fenêtre Messages ou en appuyant simultanément sur les touches Maj et Retour (Entrée) après chaque ligne de code.

Exécuter plusieurs lignes de code par copier/coller

1 Copiez les lignes de code dans le Presse-papiers.

2 Entrez une ligne vierge dans la fenêtre Messages.

3 Collez le code dans le volet de saisie de la fenêtre Messages.

4 Placez le point d'insertion à la fin de la dernière ligne de code.

5 Appuyez sur Ctrl+Entrée (Windows) ou Ctrl+Retour (Mac). Director trouve la première ligne vierge au-dessus du point d'insertion et exécute successivement chaque ligne de code après la ligne vierge.

Saisir manuellement plusieurs lignes de code

1 Entrez une ligne vierge dans la fenêtre Messages.

2 Entrez la première ligne de code.

3 Appuyez sur Maj+Retour (Entrée) à la fin de la ligne.

4 Répétez les étapes 2 et 3 jusqu'à la dernière ligne de code.

5 Appuyez sur Ctrl+Entrée (Windows) ou **Ctrl+Retour (Mac)**. Director trouve la première ligne vierge au-dessus du point d'insertion et exécute successivement chaque ligne de code après la ligne vierge.

Vous pouvez tester un gestionnaire sans exécuter l'animation, en écrivant le gestionnaire dans une fenêtre de script d'animation ou de script de comportement, puis en l'appelant depuis la fenêtre Messages.

Tester un gestionnaire

1 Copiez et collez ou saisissez manuellement un gestionnaire à plusieurs lignes dans la fenêtre Messages, comme indiqué dans les deux procédures précédentes.

2 Placez le point d'insertion à la fin de la dernière ligne de code.

3 Appuyez sur Entrée (Windows) ou sur Retour (Mac). Le gestionnaire est exécuté.

Toutes les sorties provenant des instructions `put()` ou `trace()` dans le gestionnaire sont affichées dans la fenêtre Messages.

Tout comme la fenêtre Script, la fenêtre Messages contient des menus locaux des commandes de programmation. Lorsque vous sélectionnez une commande dans l'un de ces menus locaux, la commande apparaît automatiquement dans la fenêtre Messages en présentant le premier argument fourni. Plusieurs menus sont disponibles et permettent un accès rapide au catalogue complet des termes de programmation.

Les menus locaux comprennent les éléments suivants :

- Lingo par ordre alphabétique : toutes les commandes, à l'exception de Lingo 3D, présentées par ordre alphabétique.
- Lingo par catégorie : toutes les commandes, à l'exception de Lingo 3D, présentées par catégorie.
- Lingo 3D par ordre alphabétique : tous les termes Lingo 3D, présentés par ordre alphabétique.
- Lingo 3D par catégorie : tous les termes Lingo, présentés par catégorie.
- Les Xtras de programmation incluent les méthodes et propriétés de tous les Xtras de programmation trouvés, qu'il s'agisse d'Xtras Adobe® ou autres.

Remarque : *Les Xtras de programmation figurant dans le menu local sont limités à ceux qui prennent en charge la méthode* `Interface()` *et dont les noms apparaissent effectivement dans le menu local. Bien que certains types de médias d'acteurs tels 3D et DVD prennent également en charge la méthode* `Interface()`*, ils ne figurent pas dans le menu local Xtras de programmation parce qu'ils ne sont pas implémentés en tant qu'Xtras de programmation dans Director.*

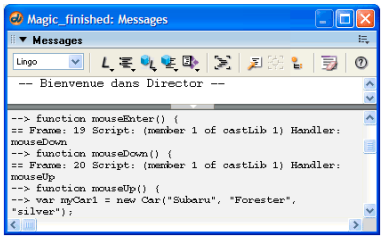
Gestion de scripts dans la fenêtre Messages

Vous pouvez régler le volet de résultat de la fenêtre Messages de manière à afficher un enregistrement des instructions qu'une animation exécute lors de sa lecture. Ceci s'avère utile pour retracer le suivi du flux de votre code et examiner le résultat d'instructions spécifiques. Il existe deux manières d'effectuer cette opération.

Afficher des instructions dans le volet de résultat

❖ Effectuez l'une des opérations suivantes :

- Dans la fenêtre Messages, cliquez sur la fonction de suivi.
- Donnez à la propriété `traceScript` de l'objet Animation la valeur `TRUE`.



Les entrées placées après un double signe égal (==) indiquent ce qui s'est produit dans l'animation, par exemple la dernière image ouverte, le script en cours d'exécution ou le résultat d'une méthode ou de la définition d'une valeur.

Par exemple, la ligne suivante contient plusieurs renseignements :

```
== Frame: 39 Script: 1 Handler: mouseUp
```

- L'animation a accédé à l'image 39.
- L'animation a exécuté le script 1, le premier script associé à l'image.

- L'animation a exécuté le gestionnaire `mouseUp` dans le script 1 après que l'animation a accédé à l'image.

Les entrées situées après une flèche constituée d'un double tiret et d'un signe supérieur à (`-->`) indiquent les lignes de votre code qui ont été exécutées. Par exemple, les lignes Lingo suivantes :

```
--> _sound.fadeOut(1, 5*60)
--> if leftSide < 10 then
--> if leftSide < 200 then
--> _movie.go("Game Start")
```

indiquent que ces instructions Lingo ont été exécutées. Supposons que vous souhaitiez déterminer la raison pour laquelle la tête de lecture n'a pas accédé à l'image appelée « Début du jeu ». Si la ligne `--> _movie.go ("Game Start")` ne s'est pas affichée dans la fenêtre Messages, il se peut que la condition de l'instruction précédente ne soit pas celle escomptée.

Le volet de résultat de la fenêtre Messages peut contenir une grande quantité de texte lorsque la fonction de suivi est activée. Pour supprimer le contenu du volet de résultat, cliquez sur le bouton Effacer. Si le volet de résultat n'est pas visible, le contenu du volet de saisie est effacé.

Vous pouvez assurer le suivi des valeurs de variables et d'autres objets en sélectionnant le nom de l'objet dans la fenêtre Messages et en cliquant sur le bouton Inspecteur d'objet. L'objet est ajouté à l'Inspecteur d'objet, où sa valeur est affichée et actualisée lors de la lecture de l'animation. Pour plus d'informations sur l'Inspecteur d'objet, consultez « Débogage dans l'Inspecteur d'objet », page 87.

Lorsque vous êtes en mode de débogage, vous pouvez suivre les modifications d'une variable en sélectionnant cette dernière dans la fenêtre Messages et en cliquant sur le bouton Surveiller l'expression. Director ajoute ensuite la variable au volet Surveillance dans la fenêtre Débogueur, dans laquelle sa valeur est affichée et actualisée pendant que vous travaillez dans la fenêtre Débogueur. Pour plus d'informations sur le volet Surveillance, consultez « Débogage dans la fenêtre Débogueur », page 90.

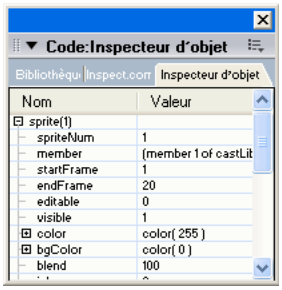
Débogage dans l'Inspecteur d'objet

L'Inspecteur d'objet permet d'afficher et de définir les propriétés d'un grand nombre d'objets qui ne sont pas affichées dans l'Inspecteur des propriétés. Il s'agit notamment des objets de programmation tels que les variables globales, les listes, les objets enfants de scripts parents, de toutes les propriétés d'acteur 3D, des propriétés d'images-objets, des expressions de scripts, etc. En outre, l'Inspecteur d'objet affiche les modifications apportées aux propriétés d'objet lors de la lecture de l'animation, par exemple les modifications dues aux scripts ou apportées aux propriétés de scénario de l'image-objet. Ces types de modifications à l'exécution ne sont pas affichés dans l'Inspecteur des propriétés lors de la lecture de l'animation.

Pour afficher les valeurs des variables JavaScript dans l'Inspecteur d'objet, vous devez les déclarer sans qu'elles soient précédées de l'instruction var.

Ouvrir l'Inspecteur d'objet

❖ Choisissez Fenêtre > Inspecteur d'objet.



Présentation détaillée des structures d'objets

L'Inspecteur d'objet est très utile pour comprendre la structure d'objets complexes. Par exemple, les acteurs 3D contiennent un grand nombre de couches de propriétés. L'Inspecteur d'objet affichant une représentation visuelle de la structure imbriquée de ces propriétés, il vous aide à comprendre l'organisation de ces propriétés, ainsi que leurs interactions. Il est important de comprendre la structure des propriétés des objets dans Director lors de la rédaction des scripts Lingo.

La possibilité d'examiner le changement de valeur des propriétés lors de la lecture d'une animation est pratique pour comprendre le fonctionnement de l'animation. Cela s'avère particulièrement utile lors des procédures de test et de débogage des scripts, car vous pouvez constater les changements de valeurs en fonction des scripts que vous avez rédigés.

La fenêtre Débogueur de Director affiche également ces informations, mais uniquement en mode de débogage. Pour plus d'informations sur le débogage, consultez « Débogage avancé », page 95.

Objets visibles

Voici quelques exemples d'objets que vous pouvez entrer dans l'Inspecteur d'objet :

- Images-objets, telles que `sprite(3)`
- Acteurs, tels que `member("3d")`
- Variables globales, telles que `gMyList`
- Objets enfants, tels que `gMyChild`
- Objets Adobe® Flash®, tels que `gMyFlashObject` ; pour plus d'informations sur l'utilisation d'objets Flash dans Director, consultez les rubriques du manuel Utilisation de Director dans l'Aide de Director.
- Expressions de script, telles que `sprite(7).blend`

Affichage d'objets

Il existe trois façons de visualiser un objet dans l'Inspecteur d'objet. Vous pouvez faire glisser les éléments directement dans l'Inspecteur d'objet, saisir manuellement le nom d'un de ses éléments ou utiliser le bouton Inspecteur d'objet dans les fenêtres Messages et Script.

Glisser un élément dans l'Inspecteur d'objet

❖ Effectuez l'une des opérations suivantes :

- Sélectionnez une image-objet dans la fenêtre Scénario et faites-la glisser dans l'Inspecteur d'objet.
- Sélectionnez un acteur dans la fenêtre Acteur et faites-le glisser dans l'Inspecteur d'objet.

- Sélectionnez le nom d'un objet dans les fenêtres Script, Messages ou Texte et faites-le glisser dans l'Inspecteur d'objet.

Saisir manuellement un objet dans l'Inspecteur d'objet

1 Double-cliquez dans la première cellule vide de la colonne Nom de l'Inspecteur d'objet.

2 Tapez le nom de l'objet dans la cellule. Utilisez le même nom que celui utilisé pour cet objet dans vos scripts.

3 Appuyez sur Entrée (Windows) ou sur Retour (Mac). Si l'objet possède des sous-propriétés, un signe plus (+) est affiché sur sa gauche.

4 Cliquez sur le signe plus. Les propriétés de l'objet s'affichent en dessous de celui-ci. Les propriétés contenant des sous-propriétés sont affichées avec un signe plus sur leur gauche. Cliquez sur chaque signe plus pour afficher les sous-propriétés.

Afficher un objet à l'aide du bouton Inspecteur d'objet

1 Dans la fenêtre Script, mettez en surbrillance la partie d'une instruction se rapportant à un objet.

2 Dans la fenêtre Script, cliquez sur Inspecteur d'objet. Si l'objet possède des sous-propriétés, un signe plus (+) est affiché sur sa gauche.

3 Cliquez sur le signe plus. Les propriétés de l'objet s'affichent en dessous de celui-ci. Les propriétés contenant des sous-propriétés sont affichées avec un signe plus sur leur gauche. Cliquez sur chaque signe plus pour afficher les sous-propriétés.

Remarque : *L'inspection de nombreux objets ou de gros objets individuels dans l'Inspecteur d'objet risque de poser des problèmes importants de performance durant la programmation, particulièrement lorsque l'option Interrogation automatique est activée. Par exemple, l'inspection d'une liste contenant 10 000 entrées peut entraîner un fort ralentissement de Director pendant que l'affichage est mis à jour.*

Recherche d'objets

Vous pouvez également accéder au contenu de l'Inspecteur d'objet à l'aide des touches fléchées de votre clavier.

Monter ou descendre dans la liste des éléments

❖ Utilisez les touches fléchées Haut et Bas.

Afficher les sous-propriétés d'un élément

❖ Sélectionnez l'élément et appuyez sur la touche fléchée Droite.

Masquer les sous-propriétés d'un élément

❖ Sélectionnez l'élément et appuyez sur la touche fléchée Gauche.

Utilisation de l'option Interrogation automatique

Les propriétés système, telles que `milliseconds` et `colorDepth`, ne sont actualisées dans l'Inspecteur d'objet que lorsque l'option Interrogation automatique est activée. L'utilisation de l'interrogation automatique augmente la charge de travail du processeur, ce qui risque de ralentir les performances de votre animation lorsque vous ajoutez un certain nombre de propriétés système à l'Inspecteur d'objet.

Activer l'option Interrogation automatique

1 Cliquez avec le bouton droit de la souris (Windows) ou cliquez en maintenant la touche Ctrl enfoncée (Mac) dans l'Inspecteur d'objet. Le menu contextuel de l'Inspecteur d'objet apparaît.

2 Sélectionnez Interrogation automatique dans le menu contextuel. Lorsque l'option Interrogation automatique est activée, une coche apparaît en regard de l'option correspondante dans le menu.

Désactiver l'option Interrogation automatique

❖ Sélectionnez à nouveau Interrogation automatique dans le menu contextuel.

Modification des valeurs d'un objet ou d'une propriété

Vous pouvez définir la valeur d'un objet ou d'une propriété dans l'Inspecteur d'objet en saisissant une nouvelle valeur dans le champ situé à droite du nom de l'objet ou de la propriété.

Définir la valeur d'un objet ou d'une propriété

1 Double-cliquez sur la valeur à droite du nom de l'élément.

2 Saisissez la nouvelle valeur de l'élément.

3 Appuyez sur Entrée (Windows) ou sur Retour (Mac). La nouvelle valeur est définie et est immédiatement reflétée dans l'animation.

Vous pouvez saisir une expression de script comme valeur pour un élément. Par exemple, vous pouvez définir la valeur `sprite(3).locH` pour l'expression `sprite(8).locH + 20`.

Suppression d'objets

Vous pouvez également retirer des éléments de l'Inspecteur d'objet.

Suppression d'un élément de l'Inspecteur d'objet :

❖ Sélectionnez l'élément et appuyez sur la touche Retour arrière (Windows) ou Effacement (Macintosh).

Suppression de tout le contenu de l'Inspecteur d'objet :

❖ Cliquez avec le bouton droit de la souris (Windows) ou cliquez en maintenant la touche Ctrl enfoncée (Macintosh) dans l'Inspecteur d'objet et choisissez Effacer tout dans le menu local.

Lorsque vous ouvrez une animation différente de celle sur laquelle vous travaillez actuellement, les objets précédemment entrés dans l'Inspecteur d'objet y sont conservés. Ceci facilite la comparaison de différentes versions d'une même animation. Lorsque vous quittez Director, les éléments de l'Inspecteur d'objet ne sont pas conservés.

Débogage dans la fenêtre Débogueur

La fenêtre Débogueur constitue un mode spécial de la fenêtre Script. Elle fournit plusieurs outils permettant de localiser la cause de problèmes dans les scripts. Le Débogueur vous permet de localiser rapidement les éléments de votre code qui sont à l'origine du problème. La fenêtre Débogueur permet de rédiger des scripts ligne par ligne, d'ignorer les gestionnaires imbriqués, de modifier le texte des scripts et de visualiser les valeurs des variables et d'autres objets au fur et à mesure de leur modification. L'apprentissage des outils de la fenêtre Débogueur permet d'accroître l'efficacité de votre programmation.

La fenêtre Débogueur permet également de localiser et de corriger les erreurs dans vos scripts. Elle comprend plusieurs outils qui vous permettent d'effectuer les opérations suivantes :

- Afficher la partie du script contenant la ligne de code actuelle.

- Retracer la séquence des gestionnaires appelés avant le gestionnaire actuel.
- Exécuter certaines parties du gestionnaire actuel.
- Exécuter certaines parties des gestionnaires appelés depuis le gestionnaire actuel.
- Afficher la valeur d'une variable locale, d'une variable globale ou d'une propriété associée au code qui fait l'objet de la recherche.

Activation du mode de débogage

La fenêtre Débogueur ne s'affiche que lorsqu'un script est interrompu. Cette interruption survient lorsque Director détecte une erreur ou un point d'arrêt dans un script.

La boîte de dialogue Erreur de script apparaît lorsqu'une erreur de script survient. Cette boîte de dialogue affiche les informations associées à l'erreur détectée, vous demande si vous souhaitez corriger le bogue dans le script, modifier le script dans la fenêtre Script ou annuler.

Activation du mode de débogage :

❖ Effectuez l'une des opérations suivantes :

- Cliquez sur Déboguer dans la boîte de dialogue Erreur de script.
- Placez un point d'arrêt dans un script.

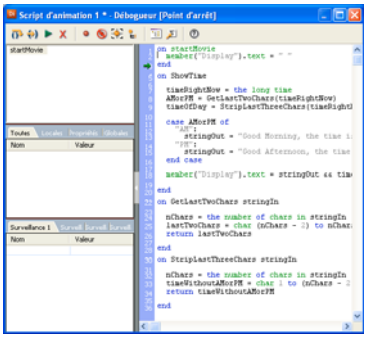
Lorsque Director détecte un point d'arrêt en cours d'exécution, l'exécution du script est interrompue et la fenêtre Script passe en mode de débogage. La lecture de l'animation se poursuit, mais l'exécution de vos scripts est interrompue jusqu'à ce que vous utilisiez la fenêtre Débogueur pour indiquer la procédure que Director doit suivre. Si plusieurs fenêtres Script sont ouvertes, Director recherche celle contenant le script dans lequel le point d'arrêt a été détecté et fait passer cette fenêtre en mode de débogage.

Ajout d'un point d'arrêt afin de provoquer l'ouverture de la fenêtre Débogueur :

1 Dans la fenêtre Script, ouvrez le script qui doit contenir le point d'arrêt.

2 Cliquez sur la marge gauche de la fenêtre Script, à côté de la ligne de code dans laquelle le point d'arrêt doit apparaître ou placez un point d'insertion sur la ligne de code et cliquez sur Activer/désactiver le point d'arrêt. L'exécution de votre code est interrompue au début de cette ligne et la fenêtre Script passe en mode de débogage.

Si la fenêtre Script est ouverte lorsque Director détecte une erreur de script ou un point d'arrêt, la fenêtre Débogueur remplace automatiquement la fenêtre Script.



Arrêt de la procédure de débogage :

❖ Effectuez l'une des opérations suivantes :

• Cliquez sur le bouton Relancer le script dans la fenêtre Débogueur. Cette opération rétablit l'exécution normale du script.

• Cliquez sur le bouton Arrêter le débogage dans la fenêtre Débogueur. Cette opération met fin à la session de débogage et à l'animation.

La fenêtre Script apparaît à nouveau, à la place de la fenêtre Débogueur.

Lorsque la fenêtre Débogueur apparaît, elle présente la ligne de code actuelle et vous propose plusieurs choix pour la suite de l'exécution.

Recherche de la ligne de code actuelle :

❖ Dans le volet Script, recherchez la flèche verte affichée en regard d'une ligne de code.

La flèche verte pointe vers la ligne actuelle. Vous ne pouvez pas sélectionner une autre ligne de code en cliquant dessus dans le volet Script.

Affichage de la pile d'appels dans la fenêtre Débogueur

Le volet Pile d'appels affiche la séquence des gestionnaires imbriqués exécutés avant la ligne de code actuelle. Cette séquence est appelée « pile d'appels ». Utilisez la pile d'appels pour retracer la structure de votre code lors de la procédure de débogage. Vous pouvez visualiser les variables associées à un gestionnaire spécifique en cliquant sur le nom du gestionnaire dans le volet Pile d'appels. Les variables sont affichées dans le volet des variables.

Affichage des variables dans la fenêtre Débogueur

Le volet des variables de la fenêtre Débogueur affiche les variables associées au gestionnaire actuel. Le gestionnaire actuel est celui qui est affiché dans le volet Script et le dernier gestionnaire affiché dans le volet Pile d'appels. Vous pouvez également afficher les variables associées aux gestionnaires précédents dans la pile d'appels. Les modifications apportées aux valeurs des variables d'un script sont affichées en rouge. Pour plus d'informations sur la progression dans les scripts, consultez « Progression dans les scripts dans la fenêtre Débogueur », page 93.

Affichage de variables associées à un gestionnaire dans la pile d'appels :

❖ Cliquez sur le nom du gestionnaire dans le volet Pile d'appels. Les variables sont affichées dans le volet des variables.

Le volet des variables contient quatre onglets vous permettant de visualiser les variables :

Le volet **Toutes** affiche les variables globales et locales associées au gestionnaire actuel.

Le volet **Locales** affiche uniquement les variables locales associées au gestionnaire actuel.

Le volet **Propriétés** affiche les propriétés déclarées dans le script actuel.

Le volet **Globales** affiche uniquement les variables globales associées au gestionnaire actuel.

Tri des variables dans le volet des variables :

• Pour trier les variables par nom, cliquez sur le mot *Nom* qui apparaît au-dessus des noms de variable.

• Pour trier les variables en ordre alphabétique inversé, cliquez une seconde fois sur le mot *Nom*.

Vous pouvez modifier les valeurs des variables locales du gestionnaire actuel et des variables globales dans le volet des variables. Vous ne pouvez pas modifier les valeurs des variables locales qui ne sont pas situées dans le gestionnaire actuel.

Modification de la valeur d'une variable dans le volet des variables :

1. Double-cliquez sur la valeur de la variable dans la colonne Valeur.
2. Saisissez la nouvelle valeur de la variable.
3. Appuyez sur Entrée (Windows) ou sur Retour (Mac).

Affichage des objets dans la fenêtre Débogueur

Le volet Surveillance de la fenêtre Débogueur permet de visualiser les variables et autres objets associés au gestionnaire actuel, ainsi que les objets associés aux autres gestionnaires. L'ajout d'objets dans le volet Surveillance vous permet de suivre leurs valeurs au fur et à mesure de leur modification grâce aux scripts. Lorsque la valeur d'un objet change en raison de l'exécution d'une ligne de code, Director affiche la couleur du nom de l'objet en rouge dans le volet Surveillance.

Le volet Surveillance affiche uniquement les objets que vous avez ajoutés. Vous pouvez utiliser chacun des quatre onglets du volet Surveillance pour organiser les objets en groupes.

Ajout au volet Surveillance d'un objet dont le nom est affiché dans le volet Script :

1. Cliquez sur le nom de l'objet dans le volet Script.
2. Cliquez sur le bouton Surveiller l'expression.

Ajout au volet Surveillance d'un objet dont le nom n'est pas affiché dans le volet Script :

1. Double-cliquez sur la première cellule vide de la colonne Nom du volet Surveillance.
2. Saisissez le nom de l'objet dans la cellule et appuyez sur Entrée (Windows) ou sur Retour (Macintosh).

   Si l'objet possède des propriétés, un signe plus (+) est affiché en regard du nom de l'objet.

Affichage des propriétés d'un objet :

❖ Cliquez sur le signe plus (+) en regard du nom de l'objet.

Le volet Surveillance permet d'organiser les objets de plusieurs façons.

Organisation des objets dans le volet Surveillance :

❖ Effectuez l'une des opérations suivantes :

• Pour trier les objets dans le volet Surveillance, cliquez sur l'en-tête de colonne Nom affiché en haut de la colonne de gauche. Les noms d'objets de la colonne sont présentés par ordre alphabétique.

• Pour trier les objets en ordre alphabétique inversé, cliquez une seconde fois sur l'en-tête de colonne Nom.

• Pour organiser les objets en groupes, utilisez les onglets du volet Surveillance. Pour ajouter un objet à un volet spécifique, cliquez sur l'onglet de votre choix avant d'ajouter l'objet.

• Pour effacer le contenu d'un onglet dans le volet Surveillance, sélectionnez le volet, puis cliquez avec le bouton droit (Windows) ou cliquez en maintenant la touche Ctrl enfoncée (Macintosh) dans le volet Surveillance et sélectionnez Effacer tout.

Progression dans les scripts dans la fenêtre Débogueur

La fenêtre Débogueur fournit un ensemble d'outils permettant une exécution lente des scripts, ce qui vous permet de visualiser l'effet de chaque ligne de code dans votre animation. Vous pouvez exécuter une ligne de code à la fois et décider si vous souhaitez exécuter les gestionnaires ligne par ligne ou pour l'ensemble des lignes.

Exécution de la ligne de code actuelle indiquée par la flèche verte uniquement :

❖ Cliquez sur le bouton Exécuter le script pas à pas.

La plupart des gestionnaires comprennent des instructions d'appel des autres gestionnaires. Vous pouvez centrer votre attention sur ces gestionnaires imbriqués ou les ignorer et vous limiter au code du gestionnaire actuel.

Lorsque vous savez que les gestionnaires sont exécutés comme prévu et que vous souhaitez vous concentrer sur le code dans le gestionnaire actuel, la fenêtre Débogueur peut ignorer les gestionnaires imbriqués et accéder directement à la prochaine ligne de code dans le gestionnaire actuel. Lorsque le débogueur ignore un gestionnaire imbriqué, il exécute le gestionnaire, mais n'affiche pas le code du gestionnaire ni ne marque de pause dans le gestionnaire imbriqué.

Processus pour ignorer des gestionnaires imbriqués :

❖ Cliquez sur le bouton Exécuter le script pas à pas dans la fenêtre Débogueur.

Ce bouton exécute la ligne de code actuelle, ainsi que les gestionnaires imbriqués appelés par la ligne, puis s'arrête sur la ligne suivante du gestionnaire.

Si vous suspectez un dysfonctionnement des gestionnaires imbriqués et souhaitez examiner leur comportement, la fenêtre Débogueur vous permet également d'exécuter les gestionnaires imbriqués ligne par ligne.

Exécution des gestionnaires imbriqués ligne par ligne :

❖ Cliquez sur le bouton Exécuter le script en détail dans la fenêtre Débogueur.

Un clic sur le bouton Exécuter le script en détail lance l'exécution de la ligne de code actuelle et poursuit le flux normal dans les gestionnaires imbriqués appelés par l'intermédiaire de cette ligne. Une fois le traitement d'un gestionnaire imbriqué terminé, la fenêtre Débogueur s'arrête sur la prochaine ligne de code dans le gestionnaire de niveau supérieur.

Lorsque la procédure de débogage est terminée, vous pouvez quitter le Débogueur à tout moment.

Reprise de l'exécution normale du code et sortie de la fenêtre Débogueur :

❖ Cliquez sur le bouton Relancer le script.

Sortie de la fenêtre Débogueur et arrêt de la lecture de l'animation :

❖ Cliquez sur le bouton Arrêter le débogage.

Modification de scripts en mode de débogage

Lorsque vous êtes en mode de débogage, vous pouvez modifier vos scripts directement dans la fenêtre Débogueur. Cette méthode vous permet de corriger les erreurs dès que vous les rencontrez, puis de poursuivre la procédure de débogage.

Modification d'un script dans la fenêtre Débogueur :

1 Cliquez dans le volet Script et placez le point d'insertion à l'endroit où vous souhaitez commencer à taper.

2 Apportez les modifications au script.

Vous pouvez passer directement à un gestionnaire spécifique en sélectionnant son nom et en cliquant sur le bouton Passer au gestionnaire.

3 Lorsque la procédure de débogage et de modification des scripts est terminée, cliquez sur le bouton Arrêter le débogage. La fenêtre Script repasse en mode Script.

4 Cliquez sur le bouton Recompiler tous les scripts modifiés.

Débogage de projections et d'animations Shockwave

Cette section traite du débogage durant l'exécution dans les projections et les animations Director qui comprennent un contenu Adobe® Shockwave®. Vous pouvez utiliser la fenêtre Messages ou activer les dialogues d'erreurs de script pour déboguer les projections et les animations Shockwave.

Débogage à l'aide de la fenêtre Messages :

❖ Donnez à la propriété `debugPlaybackEnabled` de l'objet Lecteur la valeur `TRUE`.

Lorsque cette propriété possède la valeur `TRUE`, la lecture d'une projection ou d'une animation Shockwave ouvre une fenêtre Messages (Windows) ou un fichier texte Messages (Macintosh) et les résultats des appels de fonction `put()` ou `trace()` sont insérés dans ces formats.

Si, à tout moment durant l'animation, vous donnez à la propriété `debugPlaybackEnabled` la valeur `FALSE`, la fenêtre ou le fichier texte Messages se ferme et ne peut être rouverte durant cette session de lecture, même si vous redonnez à la propriété `debugPlaybackEnabled` la valeur `TRUE` plus tard dans cette même section de lecture.

Débogage en activant les dialogues d'erreurs de script :

❖ Dans le fichier .ini d'une projection ou d'une animation Shockwave, donnez à la propriété `DisplayFullLingoErrorText` la valeur 1.

Vous créez ainsi, dans la boîte de dialogue, un texte d'erreurs plus descriptif que le texte d'erreurs générique. Par exemple, un message d'erreur générique peut avoir l'aspect suivant :

```
Script error: Continue?
```

L'attribution de la valeur 1 à la propriété `DisplayFullLingoErrorText pourrait générer le message d'erreur suivant :`

```
Script error: list expected
```

Pour obtenir des informations sur la création et la modification d'un fichier .ini pour une projection ou une animation Shockwave, consultez le fichier modèle .ini de Director se trouvant dans le dossier d'installation racine de Director.

Débogage avancé

Si le problème n'est pas facile à identifier, tentez les approches suivantes :

• Déterminez la section dans laquelle se situe le problème. Par exemple, si un clic sur un bouton ne produit pas le résultat escompté, vérifiez le script affecté à ce bouton.

Si une image-objet exécute une action erronée, vérifiez les valeurs des propriétés attachées à cette image-objet. Sont-elles définies sur les valeurs souhaitées ?

• Recherchez la séquence d'exécution du script. Lorsqu'une section de l'animation ne réagit pas comme vous l'espériez, tâchez tout d'abord de retracer la séquence des événements de l'animation. Consultez les autres scripts dans la hiérarchie des messages pour vous assurer que Director exécute le gestionnaire correct.

• Consultez les informations de suivi dans la fenêtre Messages, qui présentent les images parcourues par l'animation, ainsi que les gestionnaires appelés au cours de la lecture de l'animation.

• Essayez d'utiliser les fonctions Exécuter le script pas à pas et Exécuter le script en détail dans la fenêtre Débogueur et voyez si les résultats diffèrent de ce que vous attendiez.

• Vérifiez les variables et les expressions. Analysez le changement des valeurs lors de la lecture de l'animation. Observez si elles changent au mauvais moment ou si elles ne changent pas du tout. Si la même variable est utilisée dans plusieurs gestionnaires, vérifiez si chaque gestionnaire qui utilise la variable a défini cette variable comme globale.

Vous pouvez suivre les variables et les expressions en affichant leurs valeurs dans le volet Surveillance de la fenêtre Débogueur ou dans l'Inspecteur d'objet.

• N'apportez qu'une modification à la fois. N'hésitez pas à apporter des modifications dans un gestionnaire pour vérifier si les changements peuvent résoudre le problème ou produire des résultats qui aident à le localiser.

Veillez toutefois à ne pas résoudre un problème en en créant un autre. Apportez une modification à la fois et annulez-la si le problème n'est pas résolu. Si vous apportez trop de modifications avant de résoudre effectivement un problème, vous risquez de ne plus pouvoir déterminer quel était le problème initial, voire même d'en créer de nouveaux.

• Recréez la section. Si vous ne trouvez pas de solution, tâchez de recréer la section depuis le début. Par exemple, si une image-objet ne réagit pas correctement lorsque le pointeur la survole, créez une simple animation contenant uniquement cette image-objet et le gestionnaire, avec la méthode `rollOver()`.

Si vous copiez/collez simplement les scripts, cela risque de copier le problème. En revanche, si vous recréez la section, vous serez amené à reconstruire la logique depuis son premier niveau, et vous pouvez alors vérifier si Director réagit comme vous le souhaitez. Si la section que vous avez recréée ne fonctionne toujours pas comme prévu, il se peut qu'une erreur provienne de la logique de la section.

Si la section que vous avez recréée fonctionne correctement, comparez-la avec l'animation d'origine pour noter leurs différences. Vous pouvez également copier la section dans l'originale et vérifier si le problème est résolu.

Chapitre 5 : Objets principaux de Director

Les objets principaux de Director® donnent accès aux fonctionnalités et options disponibles dans Director, aux projections et à Adobe® Shockwave® Player. Les objets principaux incluent le moteur du lecteur de Director, les fenêtres des animations, les images-objets, les sons, etc. Ils représentent la couche de base à travers laquelle on accède à presque toutes les API et autres catégories d'objets, à l'exception des objets de programmation qui étendent les fonctionnalités principales de Director.

Pour voir comment les objets principaux sont liés les uns aux autres, ainsi qu'aux autres objets de Director, consultez « Diagrammes de modèles d'objets », page 46.

Bibliothèque de distribution

Représente une seule bibliothèque de distribution dans une animation.

Une animation peut contenir une ou plusieurs bibliothèques. Une bibliothèque de distribution peut être constituée d'un ou plusieurs acteurs qui représentent des médias dans une animation, tels que les sons, le texte, les graphiques et les autres animations.

Vous pouvez créer une référence à une bibliothèque de distribution en utilisant la fonction de haut niveau `castLib()` ou la propriété `castLib` de l'objet Animation. Par exemple, si une animation contient une bibliothèque de distribution appelée `scripts`, vous pouvez créer une référence à cette bibliothèque en procédant comme suit :

- Utilisez la méthode de haut niveau castLib().

```
-- Lingo syntax
libScript = castLib("scripts")

// JavaScript syntax
var libScript = castLib("scripts");
```

- Utilisez la propriété castLib de l'objet Animation.

```
-- Lingo syntax
libScript = _movie.castLib["scripts"]

// JavaScript syntax
var libScript = _movie.castLib["scripts"];
```

Récapitulatif des méthodes pour l'objet Bibliothèque de distribution

Méthode
findEmpty()

Récapitulatif des propriétés pour l'objet Bibliothèque de distribution

Propriété
fileName (distribution)
member (distribution)
name
number (distribution)
preLoadMode
selection

Voir aussi

castLib, castLib(), Acteur, Animation, Lecteur, Image-objet, Fenêtre

Global

Fournit un emplacement de stockage de variables globales. Ces variables sont disponibles dans Lingo et la syntaxe JavaScript.

Vous pouvez accéder à l'objet Global en utilisant la propriété de haut niveau `_global`. Vous pouvez soit affecter `_global` à une variable, soit utiliser directement la propriété `_global` pour accéder aux méthodes de l'objet Global et à toute variable globale définie.

- Affectez `_global` à une variable.

```
-- Lingo syntax
objGlobal = _global

// JavaScript syntax
var objGlobal = _global;
```

- Utilisez directement la propriété _global.

```
-- Lingo syntax
_global.showGlobals()

// JavaScript syntax
_global.showGlobals();
```

- Accédez à une variable globale.

```
-- Lingo syntax
global gSuccess

on mouseDown
    gSuccess = "Congratulations!"
    put(gSuccess) -- displays "Congratulations!"
end

// JavaScript syntax
_global.gSuccess = "Congratulations!";

function mouseDown() {
    trace(_global.gSuccess); // displays "Congratulations!"
}
```

Récapitulatif des méthodes pour l'objet Global

Méthode
clearGlobals()
showGlobals()

Voir aussi

_global

Touche

Utilisé pour contrôler les opérations effectuées au clavier par un utilisateur.

Vous pouvez accéder à l'objet Touche en utilisant la propriété de haut niveau _key. Vous pouvez soit affecter _key à une variable, soit utiliser directement la propriété _key pour accéder aux méthodes et propriétés de l'objet Touche.

- Affectez _key à une variable.

```
-- Lingo syntax
objKey = _key

// JavaScript syntax
var objKey = _key;
```

- Utilisez directement la propriété _key.

```
-- Lingo syntax
isCtrlDown = _key.controlDown

// JavaScript syntax
var isCtrlDown = _key.controlDown;
```

Récapitulatif des méthodes pour l'objet Touche

Méthode
keyPressed()

Récapitulatif des propriétés pour l'objet Touche

Propriété
commandDown
controlDown
key
keyCode
optionDown
shiftDown

Voir aussi

_key

Acteur

Représente un acteur au sein d'une bibliothèque de distribution. Les acteurs sont les médias et les éléments de script d'une animation. Les acteurs média peuvent être du texte, des bitmaps, des formes, etc. Les acteurs script incluent les comportements, les scripts d'animation, etc.

Un acteur peut être référencé soit par son numéro, soit par son nom.

- Lorsque vous faites référence à un acteur en utilisant son numéro, Director le recherche dans une bibliothèque de distribution précise et en extrait les données. Cette méthode est plus rapide que celle qui consiste à faire référence à l'acteur par son nom. Toutefois, étant donné que Director ne met pas automatiquement à jour les références aux numéros des acteurs dans les scripts, toute référence par numéro à un acteur qui a changé de position dans sa bibliothèque de distribution est rompue.

- Lorsque vous faites référence à un acteur en utilisant son nom, Director effectue des recherches dans toutes les bibliothèques de distribution d'une animation, de la première à la dernière, et extrait les données de l'acteur lorsqu'il trouve son nom. Cette méthode est plus lente que celle qui consiste à faire référence à l'acteur par son numéro, surtout lorsqu'il s'agit d'animations de grande taille contenant plusieurs bibliothèques de distribution et acteurs. Cependant, une référence à un nom d'acteur permet à cette dernière de rester intacte, même si l'acteur change de position dans sa bibliothèque de distribution.

Vous pouvez créer une référence à un acteur en utilisant la fonction de haut niveau `member()` ou la propriété `member` de l'objet Distribution, Animation ou Image-objet.

Les exemples suivants illustrent la création d'une référence à un acteur.

- Utilisez la fonction de haut niveau `member()`.

```
-- Lingo syntax
objTree = member("bmpTree")

// JavaScript syntax
var objTree = member("bmpTree");
```

- Utilisez la propriété `member` de l'objet Image-objet.

```
-- Lingo syntax
objTree = sprite(1).member;

// JavaScript syntax
var objTree = sprite(1).member;
```

Récapitulatif des méthodes pour l'objet Acteur

Méthode
copyToClipBoard()
duplicate() (acteur)
erase()
importFileInto()
move()

Méthode (Suite)
pasteClipBoardInto()
preLoad() (acteur)
unLoad() (acteur)

Récapitulatif des propriétés pour l'objet Acteur

Propriété	
castLibNum	modifiedDate
comments	name
creationDate	number (acteur)
fileName (acteur)	purgePriority
height	rect (acteur)
hilite	regPoint
linked	scriptText
loaded	size
media	thumbNail
mediaReady	type (acteur)
modified	width
modifiedBy	

Voir aussi

Types de médias, member(), member (distribution), member (animation), member (image-objet), Animation, Lecteur, Objets de programmation, Image-objet, Fenêtre

Souris

Permet d'accéder aux opérations effectuées par un utilisateur avec la souris, telles que les déplacements de la souris et les clics de souris.

Vous pouvez accéder à l'objet Souris en utilisant la propriété de haut niveau _mouse. Vous pouvez soit affecter _mouse à une variable, soit utiliser directement la propriété _mouse pour accéder aux propriétés de l'objet Souris.

- Affectez _mouse à une variable.

```
-- Lingo syntax
objMouse = _mouse

// JavaScript syntax
var objMouse = _mouse;
```

- Utilisez directement la propriété _mouse.

```
-- Lingo syntax
isDblClick = _mouse.doubleClick
```

```
// JavaScript syntax
var isDblClick = _mouse.doubleClick;
```

Récapitulatif des propriétés pour l'objet Souris

Propriété	
clickLoc	mouseLoc
clickOn	mouseMember
doubleClick	mouseUp
mouseChar	mouseV
mouseDown	mouseWord
mouseH	rightMouseDown
mouseItem	rightMouseUp
mouseLine	stillDown

Voir aussi

_mouse

Animation

Représente une animation en cours d'exécution dans le lecteur de Director.

Le lecteur de Director peut contenir une ou plusieurs animations. Une animation peut contenir une ou plusieurs bibliothèques. Une bibliothèque de distribution peut être constituée d'un ou de plusieurs acteurs qui représentent les médias et les éléments de script d'une animation. Les acteurs média peuvent être du texte, des bitmaps, des formes, etc. Les acteurs script incluent les comportements, les scripts d'animation, etc. Les images-objets sont créées à partir d'acteurs et utilisées sur la scène d'une animation.

Vous pouvez faire référence à l'animation en cours d'exécution en utilisant la propriété de haut niveau `_movie`. Vous pouvez faire référence à une animation quelconque du lecteur en utilisant la propriété `movie` de l'objet Fenêtre.

- Faites référence à l'animation en cours d'exécution.

```
-- Lingo syntax
objMovie = _movie

// JavaScript syntax
var objMovie = _movie;
```

- Utilisez la propriété movie de l'objet Fenêtre pour accéder à l'animation d'une fenêtre donnée.

```
-- Lingo syntax
objMovie = _player.window[2].movie

// JavaScript syntax
var objMovie = _player.window[2].movie;
```

Vous pouvez non seulement utiliser une référence à une animation pour accéder aux méthodes et propriétés de l'animation en question, mais aussi appeler des gestionnaires Lingo et JavaScript et accéder aux acteurs et aux images-

objets de l'animation, ainsi qu'à leurs méthodes et propriétés. Cette procédure est différente des versions précédentes de Director dans lesquelles vous deviez utiliser la commande `tell` pour travailler avec les animations. Avec l'objet Animation, le travail avec les animations est simplifié.

Récapitulatif des méthodes pour l'objet Animation

Méthode	
beginRecording()	newMember()
cancelIdleLoad()	preLoad() (animation)
clearFrame()	preLoadMember()
constrainH()	preLoadMovie()
constrainV()	printFrom()
delay()	puppetPalette()
deleteFrame()	puppetSprite()
duplicateFrame()	puppetTempo()
endRecording()	puppetTransition()
finishIdleLoad()	ramNeeded()
frameReady() (animation)	rollOver()
go()	saveMovie()
goLoop()	sendAllSprites()
goNext()	sendSprite()
goPrevious()	stopEvent()
idleLoadDone()	unLoad() (animation)
insertFrame()	unLoadMember()
label()	unLoadMovie()
marker()	updateFrame()
mergeDisplayTemplate()	updateStage()

Récapitulatif des propriétés pour l'objet Animation

Propriété	
aboutInfo	frameTransition
active3dRenderer	idleHandlerPeriod
actorList	idleLoadMode
allowCustomCaching	idleLoadPeriod
allowGraphicMenu	idleLoadTag
allowSaveLocal	idleReadChunkSize

Propriété (Suite)	
allowTransportControl	imageCompression
allowVolumeControl	imageQuality
allowZooming	keyboardFocusSprite
beepOn	lastChannel
buttonStyle	lastFrame
castLib	markerList
centerStage	member (animation)
copyrightInfo (animation)	name
displayTemplate	paletteMapping
dockingEnabled	path (animation)
editShortCutsEnabled	preferred3dRenderer
enableFlashLingo	preLoadEventAbort
exitLock	score
fileFreeSize	scoreSelection
fileSize	script
fileVersion	sprite (animation)
fixStageSize	stage
frame	timeoutList
frameLabel	traceLoad
framePalette	traceLogFile
frameScript	traceScript
frameSound1	updateLock
frameSound2	useFastQuads
frameTempo	xtraList (animation)

Voir aussi

`_movie`, `Bibliothèque de distribution`, `Acteur`, `movie`, `Lecteur`, `Image-objet`, `Fenêtre`

Lecteur

Représente le moteur de lecture principal utilisé pour gérer et exécuter l'environnement auteur, les animations dans une fenêtre (MIAW), les projections et Shockwave Player.

L'objet Lecteur donne accès à toutes les animations et fenêtres qu'il gère, en plus des Xtras disponibles.

Vous pouvez créer une référence à l'objet Lecteur en utilisant la propriété de haut niveau `_player`.

- Affectez `_player` à une variable.

```
-- Lingo syntax
objPlayer = _player

// JavaScript syntax
var objPlayer = _player;
```

- Utilisez directement la propriété _player.

```
-- Lingo syntax
_player.alert("The movie has ended.")

// JavaScript syntax
_player.alert("The movie has ended.");
```

Récapitulatif des méthodes pour l'objet Lecteur

Méthode	
alert()	getPref()
appMinimize()	halt()
cursor()	open() (lecteur)
externalParamName()	quit()
externalParamValue()	setPref()
flushInputEvents()	windowPresent()

Récapitulatif des propriétés pour l'objet Lecteur

Propriété	
activeCastLib	netPresent
activeWindow	netThrottleTicks
alertHook	organizationName
applicationName	productName
applicationPath	productVersion
currentSpriteNum	safePlayer
debugPlaybackEnabled	scriptingXtraList
digitalVideoTimeScale	searchCurrentFolder
disableImagingTransformation	searchPathList
emulateMultibuttonMouse	serialNumber
externalParamCount	sound (lecteur)
frontWindow	switchColorDepth
inlineImeEnabled	toolXtraList
itemDelimiter	transitionXtraList
lastClick	userName

Propriété (Suite)	
lastEvent	window
lastKey	xtra
lastRoll	xtraList (lecteur)
mediaXtraList	

Voir aussi

_player, Bibliothèque de distribution, Acteur, Animation, Image-objet, Fenêtre

Son

Contrôle la lecture audio des huit pistes audio disponibles.

L'objet Son est composé d'objets Piste audio représentant des pistes audio individuelles.

Vous pouvez créer une référence à l'objet Son en utilisant la propriété de haut niveau _sound.

- Affectez _sound à une variable.

```
-- Lingo syntax
objSound = _sound

// JavaScript syntax
var objSound = _sound;
```

- Utilisez la propriété _sound pour accéder à la propriété soundDevice de l'objet Son.

```
-- Lingo syntax
objDevice = _sound.soundDevice

// JavaScript syntax
var objDevice = _sound.soundDevice;
```

Récapitulatif des méthodes pour l'objet Son

Méthode
beep()
channel() (son)

Récapitulatif des propriétés pour l'objet Son

Propriété
soundDevice
soundDeviceList
soundEnabled

Propriété (Suite)
soundKeepDevice
soundLevel
soundMixMedia

Voir aussi

_sound, Piste audio

Piste audio

Représente une piste audio individuelle au sein de l'objet Son.

Les pistes audio disponibles sont au nombre de huit. Vous pouvez utiliser un objet Piste audio dans un script pour accéder à l'une des huit pistes audio et la modifier.

Remarque : *vous ne pouvez modifier que les deux premières pistes audio dans le scénario de l'interface utilisateur de Director.*

Vous pouvez créer une référence à un objet Piste audio en utilisant la méthode de haut niveau `sound()`, la propriété `sound` de l'objet Lecteur ou la méthode `channel()` de l'objet Son. Par exemple, vous pouvez faire référence à la piste audio 2 en utilisant l'une des méthodes suivantes :

- Utilisez la méthode de haut niveau `sound()`.

```
-- Lingo syntax
objSoundChannel = sound(2)

// JavaScript syntax
var objSoundChannel = sound(2);
```

- Utilisez la propriété `sound` de l'objet Lecteur.

```
-- Lingo syntax
objSoundChannel = _player.sound[2]

// JavaScript syntax
var objSoundChannel = _player.sound[2];
```

- Utilisez la méthode `channel()` de l'objet Son.

```
-- Lingo syntax
objSoundChannel = _sound.channel(2)

// JavaScript syntax
var objSoundChannel = _sound.channel(2);
```

Récapitulatif des méthodes pour l'objet Piste audio

Méthode	
breakLoop()	play() (piste audio)
fadeIn()	playFile()
fadeOut()	playNext() (piste audio)

Méthode (Suite)	
fadeTo()	queue()
getPlayList()	rewind() (piste audio)
isBusy()	setPlayList()
pause() (piste audio)	stop() (piste audio)

Récapitulatif des propriétés pour l'objet Piste audio

Propriété	
channelCount	member (piste audio)
elapsedTime	pan
endTime	sampleCount
loopCount	sampleRate
loopEndTime	startTime
loopsRemaining	status
loopStartTime	volume (piste audio)

Voir aussi

`channel() (son)`, `sound (lecteur)`, `sound()`, `Son`

Image-objet

Représente une occurrence d'un acteur dans une piste d'image-objet du scénario.

Un objet Image-objet couvre une plage d'image-objet, c'est-à-dire la gamme d'images d'une piste d'image-objet donnée. Un objet Piste d'image-objet représente une piste d'image-objet entière, quel que soit le nombre d'images-objets qu'elle contient.

Remarque : *composants d'acteurs Flash® : vous ne pouvez accéder à ces composants placés sur la scène (sous la forme d'images-objets Flash) lorsqu'ils sont invisibles qu'en utilisant l'objet Acteur. L'utilisation de l'objet Image-objet pour accéder à une image-objet Flash avec une propriété d'invisibilité renverra un message d'erreur.*

Une image-objet peut être référencée soit par son numéro, soit par son nom.

• Lorsque vous faites référence à une image-objet par son numéro, Director effectue une recherche dans toutes les images-objets qui existent dans l'image en cours du scénario, en commençant par la piste portant le numéro le plus bas, et extrait les données de l'image-objet lorsqu'il la trouve. Cette méthode est plus rapide que celle qui consiste à faire référence à une image-objet par son nom. Toutefois, étant donné que Director ne met pas automatiquement à jour les références aux numéros d'images-objets dans les scripts, toute référence par numéro à une image-objet qui a changé de position dans la scène est rompue.

• Lorsque vous faites référence à une image-objet par son nom, Director fait une recherche dans toutes les images-objets qui existent dans l'image en cours du scénario, en commençant par la piste portant le numéro le plus bas, et extrait les données de l'image-objet lorsqu'il la trouve. Cette méthode est plus lente que celle qui consiste à faire référence à l'image-objet par son numéro, surtout lorsqu'il s'agit d'animations de grande taille contenant plusieurs bibliothèques de distribution, acteurs et images-objets. Cependant, une référence à un nom d'image-objet permet à cette référence de rester intacte, même si l'image-objet change de position sur la scène.

Vous pouvez créer une référence à un objet Image-objet en utilisant la fonction de haut niveau `sprite()`, la propriété `sprite` de l'objet Animation ou la propriété `sprite` de l'objet Piste d'image-objet.

• Utilisez la fonction de haut niveau `sprite()`.

```
-- Lingo syntax
objSprite = sprite(1)

// JavaScript syntax
var objSprite = sprite(1);
```

• Utilisez la propriété `sprite` de l'objet Animation.

```
-- Lingo syntax
objSprite = _movie.sprite["willowTree"]

// JavaScript syntax
var objSprite = _movie.sprite["willowTree"];
```

• Utilisez la propriété `sprite` de l'objet Piste d'image-objet.

```
-- Lingo syntax
objSprite = channel(3).sprite

// JavaScript syntax
var objSprite = channel(3).sprite;
```

Vous pouvez utiliser une référence à un objet Image-objet pour accéder à l'acteur à partir duquel l'image-objet a été créée. Toute modification effectuée sur l'acteur à partir duquel une image-objet a été créée est également répercutée dans l'image-objet. L'exemple suivant illustre la modification du texte d'un acteur texte à partir duquel l'image-objet 5 a été créée. Ce changement apporté à l'acteur est également répercuté dans l'image-objet 5.

```
-- Lingo syntax
labelText = sprite(5)
labelText.member.text = "Weeping Willow"

// JavaScript syntax
var labelText = sprite(5);
labelText.member.text = "Weeping Willow";
```

Récapitulatif des propriétés pour l'objet Image-objet

Propriété	
backColor	locV
blend (image-objet)	locZ
bottom	member (image-objet)
constraint	name (image-objet)
cursor	quad

Propriété (Suite)	
editable	rect (image-objet)
endFrame	right
filterlist	rotation
flipH	skew
flipV	spriteNum
foreColor	startFrame
height	top
ink	width
left	
locH	

Voir aussi

`Bibliothèque de distribution`, `Acteur`, `Animation`, `Lecteur`, `sprite (animation)`, `sprite (piste d'image-objet)`, `sprite()`, `Piste d'image-objet`, `Fenêtre`

Piste d'image-objet

Représente une piste d'image-objet individuelle dans le scénario.

Un objet Image-objet couvre une plage d'image-objet, c'est-à-dire la gamme d'images d'une piste d'image-objet donnée. Un objet Piste d'image-objet représente une piste d'image-objet entière, quel que soit le nombre d'images-objets qu'elle contient.

Les pistes d'images-objets sont contrôlées par le scénario par défaut. Utilisez l'objet Piste d'image-objet pour faire passer le contrôle d'une piste d'image-objet au script lors d'une session d'enregistrement de scénario.

Une piste d'image-objet peut être référencée soit par son numéro, soit par son nom.

- Lorsque vous faites référence à une piste d'image-objet par son numéro, vous accédez directement à la piste. Cette méthode est plus rapide que celle qui consiste à faire référence à une piste d'image-objet par son nom. Toutefois, étant donné que Director ne met pas automatiquement à jour les références aux numéros des pistes d'images-objets dans les scripts, toute référence par numéro à une piste d'image-objet qui a changé de position dans le scénario est rompue.
- Lorsque vous faites référence à une piste d'image-objet par son nom, Director fait une recherche dans toutes les pistes, en commençant par la piste portant le numéro le plus petit, et extrait les données de la piste d'image-objet lorsqu'il la trouve. Cette méthode est plus lente que celle qui consiste à faire référence à la piste d'image-objet par son numéro, surtout lorsqu'il s'agit d'animations de grande taille contenant plusieurs bibliothèques de distribution, acteurs et images-objets. Cependant, une référence à un nom de piste d'image-objet permet à cette référence de rester intacte, même si la piste d'image-objet change de position dans le scénario.

Vous pouvez créer une référence à un objet Piste d'image-objet en utilisant la méthode de haut niveau `channel()` et en faisant référence au numéro ou au nom de la piste.

```
-- Lingo syntax
objSpriteChannel = channel(2) -- numbered reference
objSpriteChannel = channel("background") -- named reference

// JavaScript syntax
var objSpriteChannel = channel(2); // numbered reference
var objSpriteChannel = channel("background"); // named reference
```

Vous pouvez utiliser une référence à un objet Piste d'image-objet pour accéder à l'image-objet en cours d'utilisation dans une piste d'image-objet particulière. L'exemple suivant illustre l'accès à la couleur d'arrière-plan de l'image-objet en cours d'utilisation dans la piste d'image-objet 2.

```
-- Lingo syntax
labelSprite = channel(2).sprite.backColor

// JavaScript syntax
var labelSprite = channel(2).sprite.backColor;
```

Récapitulatif des méthodes pour l'objet Piste d'image-objet

Méthode
makeScriptedSprite()
removeScriptedSprite()

Récapitulatif des propriétés pour l'objet Piste d'image-objet

Propriété
name (piste d'image-objet)
number (piste d'image-objet)
scripted
sprite (piste d'image-objet)

Voir aussi

Bibliothèque de distribution, channel() (niveau supérieur), Acteur, Animation, Lecteur, Image-objet, Fenêtre

Système

Donne accès aux informations sur le système et l'environnement, telles que les méthodes système.

Vous pouvez créer une référence à l'objet Système en utilisant la propriété de haut niveau `_system`.

- Affectez `_system` à une variable.

```
-- Lingo syntax
objSystem = _system

// JavaScript syntax
var objSystem = _system;
```

- Utilisez directement la propriété _system.

```
-- Lingo syntax
sysDate = _system.date()

// JavaScript syntax
var sysDate = _system.date();
```

Récapitulatif des méthodes pour l'objet Système

Méthode
date() (système)
restart()
shutDown()
time() (système)

Récapitulatif des propriétés pour l'objet Système

Propriété
colorDepth
deskTopRectList
environmentPropList
milliseconds

Voir aussi

_system

Fenêtre

Représente une fenêtre dans laquelle une animation est en cours d'exécution, telle que la fenêtre Scène et toute autre animation dans une fenêtre (MIAW) en cours d'utilisation.

Vous pouvez créer une référence à un objet Fenêtre en utilisant la fonction de haut niveau `window()`, la propriété `window` de l'objet Lecteur ou la propriété `windowList` de l'objet Lecteur.

- Utilisez la méthode de haut niveau `window()`.

```
-- Lingo syntax
objWindow = window("Sun")

// JavaScript syntax
var objWindow = window("Sun");
```

- Utilisez la propriété `window` de l'objet Lecteur.

```
-- Lingo syntax
objWindow = _player.window["Sun"]

// JavaScript syntax
var objWindow = _player.window["Sun"];
```

- Utilisez la propriété `windowList` de l'objet Lecteur.

```
-- Lingo syntax
objWindow = _player.windowList[1]

// JavaScript syntax
var objWindow = _player.windowList[1];
```

Remarque : *lorsque vous créez une référence au nom d'une fenêtre en utilisant soit la fonction de haut niveau* `window()`*, soit la propriété* `window` *de l'objet Lecteur, cette référence n'est créée que si une fenêtre portant ce nom existe. Si aucune fenêtre portant ce nom n'existe, la référence contient* `VOID` *(Lingo) ou* `null` *(syntaxe JavaScript).*

La propriété `scriptExecutionStyle` de l'objet Animation est définie par défaut sur 10 et la propriété `windowType` est évitée par défaut en faveur des listes de propriétés `appearanceOptions` et `titlebarOptions`. Si la propriété `scriptExecutionStyle` est définie sur 9, la propriété `windowType` est complètement opérationnelle.

Récapitulatif des méthodes pour l'objet Fenêtre

Méthode	
close()	moveToBack()
forget() (fenêtre)	moveToFront()
maximize()	open() (fenêtre)
mergeProps()	restore()
minimize()	

Récapitulatif des propriétés pour l'objet Fenêtre

Propriété	
appearanceOptions	resizable
bgColor (fenêtre)	sizeState
dockingEnabled	sourceRect
drawRect	title (fenêtre)
fileName (fenêtre)	titlebarOptions
image (fenêtre)	type (fenêtre)
movie	visible
name	windowBehind
picture (fenêtre)	windowInFront
rect (fenêtre)	

Voir aussi

`Bibliothèque de distribution`, `Acteur`, `Animation`, `Lecteur`, `Image-objet`, `window()`, `window`, `windowList`

Chapitre 6 : Types de médias

Les types de médias de Director® donnent accès aux fonctionnalités des divers types de médias (RealMedia®, DVD, GIF animé, etc.), qui sont ajoutés aux animations en tant qu'acteurs.

Les médias ne sont pas strictement des objets mais plutôt des acteurs qui se rapportent à un type de média précis. Lorsqu'un type de média est ajouté à une animation en tant qu'acteur, il hérite de la fonctionnalité de l'objet Acteur principal et étend l'objet Acteur en fournissant des fonctionnalités supplémentaires qui ne sont disponibles que pour le type de média spécifié. Par exemple, un acteur RealMedia a accès aux méthodes et propriétés de l'objet Acteur, et possède également d'autres méthodes et propriétés propres à RealMedia. Les autres types de médias affichent tous ce comportement.

Pour voir comment les types de médias d'acteurs sont liés les uns aux autres, ainsi qu'aux autres objets de Director, consultez « Diagrammes de modèles d'objets », page 46.

GIF animé

Représente un acteur GIF animé.

Vous pouvez ajouter un acteur GIF animé à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#animgif)

// JavaScript syntax
_movie.newMember("animgif");
```

Certaines des méthodes ou propriétés suivantes peuvent ne s'appliquer qu'aux images-objets créées à partir d'un acteur GIF animé.

Récapitulatif des méthodes pour le type de média GIF animé

Méthode
resume()
rewind() (GIF animé, Flash)

Récapitulatif des propriétés pour le type de média GIF animé

Propriété
directToStage
frameRate
linked
path (animation)
playBackMode

Voir aussi

Acteur

Bitmap

Représente un acteur bitmap.

Vous pouvez utiliser les objets images bitmap pour effectuer des opérations simples affectant le contenu d'un acteur bitmap entier, telles que le changement des couleurs d'arrière-plan et de premier plan de l'acteur ou pour effectuer une délicate manipulation des pixels d'une image, telle que le recadrage, le dessin et la copie de pixels.

Vous pouvez ajouter un acteur bitmap à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#bitmap)

// JavaScript syntax
_movie.newMember("bitmap");
```

Certaines des méthodes ou propriétés suivantes peuvent ne s'appliquer qu'aux images-objets créées à partir d'un acteur bitmap.

Récapitulatif des méthodes pour le type de média Bitmap

Méthode
crop() (image)
pictureP()

Récapitulatif des propriétés pour le type de média Bitmap

Propriété	
alphaThreshold	imageCompression
backColor	imageQuality
blend (image-objet)	palette
depth (bitmap)	picture (acteur)
dither	rect (image)
foreColor	trimWhiteSpace
image (image)	useAlpha

Voir aussi

Acteur

Bouton

Représente un acteur bouton ou case à cocher.

Vous pouvez ajouter un acteur bouton à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#button)

// JavaScript syntax
_movie.newMember("button");
```

Récapitulatif des propriétés pour le type de média Bouton

Propriété
hilite

Voir aussi

Acteur

Palette de couleurs

Représente la palette de couleurs associée à un acteur bitmap.

Un acteur palette de couleurs n'a aucune méthode ou propriété à laquelle il est possible d'accéder directement. Les méthodes et propriétés suivantes sont simplement associées aux palettes de couleurs.

Vous pouvez ajouter un acteur palette de couleurs à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#palette)

// JavaScript syntax
_movie.newMember("palette");
```

Vous pouvez associer un acteur bitmap à un acteur palette de couleurs à l'aide de la propriété `palette` de l'acteur bitmap. L'exemple suivant attribue la propriété `palette` de l'acteur bitmap `bmpMember` à l'acteur palette de couleurs `colorPaletteMember`. La valeur de la propriété `palette` reflète le numéro de l'acteur palette de couleurs.

```
-- Lingo syntax
member("bmpMember").palette = member("colorPaletteMember")

// JavaScript syntax
member("bmpMember").palette = member("colorPaletteMember");
```

Après avoir associé un acteur bitmap à un acteur palette de couleurs, vous ne pouvez pas supprimer l'acteur palette de couleurs avant d'avoir supprimé son association à l'acteur bitmap.

Récapitulatif des méthodes pour le type de média Palette de couleurs

Méthode
color()

Récapitulatif des propriétés pour le type de média Palette de couleurs

Propriété
depth (bitmap)
palette
paletteMapping

Voir aussi

Bitmap, Acteur, palette

Curseur

Représente un acteur curseur.

Vous pouvez ajouter un acteur curseur à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#cursor)

// JavaScript syntax
_movie.newMember("cursor");
```

Récapitulatif des propriétés pour le type de média Curseur

Propriété
castMemberList
cursorSize
hotSpot
interval

Voir aussi

Acteur

DVD

Représente un acteur DVD.

Vous pouvez ajouter un acteur DVD à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#dvd)
```

```
// JavaScript syntax
_movie.newMember("dvd");
```

Certaines des méthodes ou propriétés suivantes peuvent ne s'appliquer qu'aux images-objets créées à partir d'un acteur DVD.

Récapitulatif des événements pour le type de média DVD

Les événements DVD suivants sont toujours pris en charge par un gestionnaire d'événement `DVDeventNotification`. Lorsque l'un de ces événements se produit, le gestionnaire d'événement `DVDeventNotification` le reçoit sous forme de paramètre. Certains de ces événements contiennent également d'autres informations qui sont transmises sous forme de deuxième ou troisième paramètre à `DVDeventNotification`. Pour plus d'informations sur l'utilisation des événements suivants avec le gestionnaire `DVDeventNotification`, consultez « on DVDeventNotification », page 158.

Evénement	
angleChange	noFirstPlayChain
audioStreamChange	parentalLevelChange
buttonChange	playbackStopped
chapterAutoStop	playPeriodAutoStop
chapterStart	rateChange
diskEjected	stillOff
diskInserted	stillOn
domainChange	titleChange
error	UOPchange
karaokeMode	warning

Récapitulatif des méthodes pour le type de média DVD

Méthode	
activateAtLoc()	rootMenu()
activateButton()	selectAtLoc()
frameStep()	selectButton()
chapterCount()	selectButtonRelative()
pause() (DVD)	stop() (DVD)
play() (DVD)	subPictureType()
returnToTitle()	titleMenu()

Récapitulatif des propriétés pour le type de média DVD

Propriété	
angle (DVD)	duration (DVD)
angleCount	folder
aspectRatio	frameRate (DVD)
audio (DVD)	fullScreen
audioChannelCount	mediaStatus (DVD)
audioExtension	playRate (DVD)
audioFormat	resolution (DVD)
audioSampleRate	selectedButton
audioStream	startTimeList
audioStreamCount	stopTimeList
buttonCount	subPicture
chapter	subPictureCount
chapterCount	title (DVD)
closedCaptions	titleCount
currentTime (DVD)	videoFormat
domain	volume (DVD)

Voir aussi

Acteur

Champ

Représente un acteur champ.

Vous pouvez ajouter un acteur champ à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#field)

// JavaScript syntax
_movie.newMember("field");
```

Récapitulatif des méthodes pour le type de média Champ

Méthode	
charPosToLoc()	pointToItem()
lineHeight()	pointToLine()
linePosToLocV()	pointToParagraph()

Méthode (Suite)	
locToCharPos()	pointToWord()
locVToLinePos()	scrollByLine()
pointToChar()	scrollByPage()

Récapitulatif des propriétés pour le type de média Champ

Propriété	
alignment	fontStyle
autoTab	lineCount
border	margin
boxDropShadow	pageHeight
boxType	scrollTop
dropShadow	selEnd
editable	selStart
font	text
fontSize	wordWrap

Voir aussi

Acteur

Boucle d'animation

Représente un acteur boucle d'animation.

Vous pouvez ajouter un acteur boucle d'animation à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#filmloop)

// JavaScript syntax
_movie.newMember("filmloop");
```

Récapitulatif des propriétés pour le type de média Boucle d'animation

Propriété
media
regPoint

Voir aussi

Acteur

Composant Flash

Représente un composant Adobe® Flash® inséré dans un acteur ou une image-objet comprenant un contenu Flash.

Un composant Flash offre des fonctionnalités toutes prêtes qui étendent les fonctionnalités existantes des acteurs ou des images-objets comprenant un contenu Flash. Ils sont entièrement créés et supportés par la communauté de développement de Director.

Director prend en charge les composants Flash suivants :

Composant Flash	Description
Button	Bouton d'interface utilisateur rectangulaire redimensionnable.
CheckBox	Partie fondamentale de tout formulaire ou application Web. Vous pouvez l'utiliser à chaque fois que vous avez besoin de regrouper un jeu de valeurs `true` ou `false` qui ne s'excluent pas mutuellement.
DateChooser	Calendrier permettant à un utilisateur de choisir une date.
Label	Une ligne de texte unique.
List	Zone de liste déroulante à un ou plusieurs choix.
NumericStepper	Permet à un utilisateur de passer en revue un jeu de nombres ordonné.
RadioButton	Partie fondamentale de tout formulaire ou application Web. Vous pouvez l'utiliser à chaque fois que vous voulez qu'un utilisateur effectue une sélection dans un groupe d'options.
ScrollPane	Affiche des clips d'animation, ainsi que des fichiers JPEG et SWF, dans une zone déroulante.
TextArea	Champ de texte à plusieurs lignes.
TextInput	Composant d'une seule ligne qui enrobe l'objet ActionScript TextField.
Tree	Permet à un utilisateur de visualiser des données hiérarchiques.

Un composant Flash a accès aux mêmes API que celles auxquelles accède un acteur ou une image-objet Flash normale, en plus des fonctionnalités se rapportant à ce composant. Pour plus d'informations sur de ces composants Flash, consultez les rubriques du document Utilisation de Director dans l'Aide de Director.

Vous pouvez ajouter un acteur de composant Flash à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#flashcomponent)

// JavaScript syntax
_movie.newMember("flashcomponent");
```

Voir aussi

Animation Flash, Acteur

Animation Flash

Représente un acteur ou une image-objet comprenant un contenu Flash.

Vous pouvez ajouter un acteur animation Flash à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#flash)

// JavaScript syntax
_movie.newMember("flash");
```

Un acteur ou une image-objet animation Flash peut également contenir des composants Flash. Les composants Flash fournissent des fonctionnalités toutes prêtes qui étendent les fonctionnalités existantes des acteurs ou des images-objets animation Flash. Pour plus d'informations sur les composants Flash pris en charge par Director, consultez « Composant Flash », page 121.

Certaines des méthodes ou propriétés ci-après peuvent ne s'appliquer qu'aux images-objets créées à partir d'un acteur animation Flash.

Récapitulatif des méthodes pour le type de média Animation Flash

Méthode	
callFrame()	printAsBitmap()
clearAsObjects()	rewind() (GIF animé, Flash)
clearError()	setCallback()
findLabel()	setFlashProperty()
flashToStage()	settingsPanel()
getFlashProperty()	setVariable()
getVariable()	showProps()
goToFrame()	stageToFlash()
hitTest()	stop() (Flash)
hold()	stream()
newObject()	tellTarget()
print()	

Récapitulatif des propriétés pour le type de média Animation Flash

Propriété	
actionsEnabled	originPoint
broadcastProps	originV
bufferSize	playBackMode
buttonsEnabled	playing
bytesStreamed	posterFrame
centerRegPoint	quality
clickMode	rotation

Propriété (Suite)	
defaultRect	scale (acteur)
defaultRectMode	scaleMode
eventPassMode	sound (acteur)
fixedRate	static
flashRect	streamMode
frameCount	streamSize
imageEnabled	viewH
linked	viewPoint
mouseOverButton	viewScale
originH	viewV
originMode	

Voir aussi

`Composant Flash`, `Acteur`

Police

Représente un acteur police.

Vous pouvez ajouter un acteur police à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#font)

// JavaScript syntax
_movie.newMember("font");
```

Récapitulatif des propriétés pour le type de média Police

Propriété
bitmapSizes
characterSet
fontStyle
originalFont
recordFont

Voir aussi

`Acteur`

Animation liée

Représente un acteur animation liée.

Vous pouvez ajouter un acteur animation liée à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#movie)

// JavaScript syntax
_movie.newMember("movie");
```

Récapitulatif des propriétés pour le type de média Animation liée

Propriété
scriptsEnabled

Voir aussi

Acteur

QuickTime

Représente un acteur QuickTime®.

Vous pouvez ajouter un acteur QuickTime à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#quicktimemedia)

// JavaScript syntax
_movie.newMember("quicktimemedia");
```

Certaines des méthodes ou propriétés suivantes peuvent ne s'appliquer qu'aux images-objets créées à partir d'un acteur QuickTime.

Récapitulatif des méthodes pour le type de média QuickTime

Méthode	
enableHotSpot()	qtRegisterAccessKey()
getHotSpotRect()	qtUnRegisterAccessKey()
nudge()	setTrackEnabled()
ptToHotSpotID()	swing()
QuickTimeVersion()	

Récapitulatif des propriétés pour le type de média QuickTime

Propriété	
audio (RealMedia)	scale (acteur)
currentTime (QuickTime, AVI)	staticQuality
fieldOfView	tilt
hotSpotEnterCallback	trackCount (acteur)
hotSpotExitCallback	trackCount (image-objet)
invertMask	trackEnabled
isVRMovie	trackNextKeyTime
loopBounds	trackNextSampleTime
mask	trackPreviousKeyTime
motionQuality	trackPreviousSampleTime
mouseLevel	trackStartTime (acteur)
node	trackStartTime (image-objet)
nodeEnterCallback	trackStopTime (acteur)
nodeExitCallback	trackStopTime (image-objet)
nodeType	trackText
pan (propriété QTVR)	trackType (acteur)
percentStreamed (acteur)	trackType (image-objet)
playRate	translation
preLoad (acteur)	triggerCallback
rotation	warpMode

Voir aussi

`Acteur`

RealMedia

Représente un acteur RealMedia.

Vous pouvez ajouter un acteur RealMedia à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#realmedia)

// JavaScript syntax
_movie.newMember("realmedia");
```

Certaines des méthodes ou propriétés suivantes peuvent ne s'appliquer qu'aux images-objets créées à partir d'un acteur RealMedia.

Récapitulatif des méthodes pour le type de média RealMedia

Méthode
pause() (RealMedia, SWA, Windows Media)
play() (RealMedia, SWA, Windows Media)
realPlayerNativeAudio()
realPlayerPromptToInstall()
realPlayerVersion()
seek()
stop() (RealMedia, SWA, Windows Media)

Récapitulatif des propriétés pour le type de média RealMedia

Propriété	
audio (RealMedia)	password
currentTime (RealMedia)	pausedAtStart (RealMedia, Windows Media)
displayRealLogo	percentBuffered
duration (RealMedia, SWA)	soundChannel (RealMedia)
image (RealMedia)	state (RealMedia)
lastError	userName (RealMedia)
mediaStatus (RealMedia, Windows Media)	video (RealMedia, Windows Media)

Voir aussi

Acteur

Shockwave 3D

Représente un acteur Adobe® Shockwave® 3D.

Un acteur Shockwave 3D (ou, tout simplement, 3D) est différent des autres acteurs car il contient un univers 3D complet. Un univers 3D contient un ensemble d'objets propres aux acteurs 3D vous permettant d'ajouter des fonctionnalités 3D à une animation.

Vous pouvez ajouter un acteur 3D à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#shockwave3d)

// JavaScript syntax
_movie.newMember("shockwave3d");
```

Pour plus d'informations sur les objets et API dont disposent les acteurs 3D, consultez « Objets 3D », page 136.

Voir aussi

Acteur

Shockwave Audio

Représente un acteur Shockwave Audio.

Vous pouvez ajouter un acteur Shockwave Audio à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#swa)

// JavaScript syntax
_movie.newMember("swa");
```

Récapitulatif des événements pour le type de média Shockwave Audio

Evénement
on cuePassed

Récapitulatif des méthodes pour le type de média Shockwave Audio

Méthode
getError() (Flash, SWA)
getErrorString()
isPastCuePoint()
pause() (RealMedia, SWA, Windows Media)
play() (RealMedia, SWA, Windows Media)
preLoadBuffer()
stop() (RealMedia, SWA, Windows Media)

Récapitulatif des propriétés pour le type de média Shockwave Audio

Propriété	
bitRate	percentStreamed (acteur)
bitsPerSample	preLoadTime
channelCount	sampleRate
copyrightInfo (SWA)	sampleSize
cuePointNames	soundChannel (SWA)
cuePointTimes	state (Flash, SWA)
duration (RealMedia, SWA)	streamName
loop (acteur)	URL
mostRecentCuePoint	volume (acteur)
numChannels	

Voir aussi

Acteur

Son

Représente un acteur utilisé pour stocker et faire référence à des échantillons de sons.

Les échantillons de sons sont contrôlés par les objets principaux Son et Piste audio. Un acteur son ne possède pas d'API et utilise les API des objets Son et Piste audio pour contrôler son comportement.

Vous pouvez ajouter un acteur son à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#sound)

// JavaScript syntax
_movie.newMember("sound");
```

Pour plus d'informations sur les objets et API que vous pouvez utiliser pour contrôler des échantillons de sons, consultez « Son », page 106 et « Piste audio », page 107.

Voir aussi

Acteur

Texte

Représente un acteur texte.

Vous pouvez ajouter un acteur texte à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#text)

// JavaScript syntax
_movie.newMember("text");
```

Récapitulatif des événements pour le type de média Texte

Evénement
on hyperlinkClicked

Récapitulatif des méthodes pour le type de média Texte

Méthode
count()
pointInHyperlink()
pointToChar()
pointToItem()

Méthode (Suite)
pointToLine()
pointToParagraph()
pointToWord()

Récapitulatif des propriétés pour le type de média Texte

Propriété	
antiAlias	hyperlink
antiAliasThreshold	hyperlinkRange
bottomSpacing	hyperlinks
charSpacing	hyperlinkState
firstIndent	kerning
fixedLineSpace	kerningThreshold
font	RTF
fontStyle	selectedText
HTML	useHypertextStyles

Voir aussi

Acteur

Forme vectorielle

Représente un acteur forme vectorielle.

Vous pouvez ajouter un acteur forme vectorielle à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#vectorshape)

// JavaScript syntax
_movie.newMember("vectorshape");
```

Certaines des méthodes ou propriétés suivantes peuvent ne s'appliquer qu'aux images-objets créées à partir d'un acteur forme vectorielle.

Récapitulatif des méthodes pour le type de média Forme vectorielle

Méthode
addVertex()
deleteVertex()
moveVertex()

Méthode (Suite)
moveVertexHandle()
newCurve()
showProps()

Récapitulatif des propriétés pour le type de média Forme vectorielle

Propriété	
antiAlias	imageEnabled
backgroundColor	originH
broadcastProps	originMode
centerRegPoint	originPoint
closed	originV
curve	regPointVertex
defaultRect	scale (acteur)
defaultRectMode	scaleMode
endColor	strokeColor
fillColor	strokeWidth
fillCycles	vertex
fillDirection	vertexList
fillMode	viewH
fillOffset	viewPoint
fillScale	viewScale
flashRect	viewV
gradientType	

Voir aussi

Acteur

Windows Media

Représente un acteur Windows Media®.

Vous pouvez ajouter un acteur Windows Media à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Lingo syntax
_movie.newMember(#windowsmedia)

// JavaScript syntax
_movie.newMember("windowsmedia");
```

Certaines des méthodes ou propriétés suivantes peuvent ne s'appliquer qu'aux images-objets créées à partir d'un acteur Windows Media.

Récapitulatif des méthodes pour le type de média Windows Media

Méthode
pause() (RealMedia, SWA, Windows Media)
play() (RealMedia, SWA, Windows Media)
playFromToTime()
rewind() (Windows Media)
stop() (RealMedia, SWA, Windows Media)

Récapitulatif des propriétés pour le type de média Windows Media

Propriété	
audio (Windows Media)	pausedAtStart (RealMedia, Windows Media)
directToStage	playRate (Windows Media)
duration (acteur)	video (RealMedia, Windows Media)
height	volume (Windows Media)
loop (Windows Media)	width
mediaStatus (RealMedia, Windows Media)	

Voir aussi

Acteur

Chapitre 7 : Objets de programmation

Les objets de programmation, aussi appelés Xtras, dans Director donnent accès aux fonctionnalités des composants logiciels qui sont installés dans Director® et étendent les fonctionnalités principales de ce dernier. Les Xtras existants fournissent certaines fonctions telles que l'importation de filtres et la connexion à Internet. De plus, si vous connaissez le langage de programmation C, vous pouvez créer vos propres Xtras.

Pour voir comment les objets de programmation sont liés les uns aux autres, ainsi qu'aux autres objets de Director, consultez « Diagrammes de modèles d'objets », page 46.

Fileio

Permet d'effectuer des opérations d'entrée et de sortie de fichier.

Vous pouvez créer une référence à un objet Fileio à l'aide de l'opérateur `new`.

```
-- Lingo syntax
objFileio = new xtra("fileio")

// JavaScript syntax
var objFileio = new xtra("fileio");
```

Récapitulatif des méthodes pour l'objet Fileio

Méthode	
closeFile()	readFile()
createFile()	readLine()
delete()	readToken()
deleteFile()	readWord()
displayOpen()	setFilterMask()
displaySave()	setFinderInfo()
error()	setNewLineConversion()
fileName()	setPosition()
getFinderInfo()	status()
getLength()	
getOSDirectory()	version()
getPosition()	writeChar()
openFile()	writeReturn()
readChar()	writeString()

Xtra MUI

L'Xtra MUI offre des boîtes de dialogue complètement opérationnelles et configurées selon vos souhaits. Celles-ci ne nécessitent pas la mémoire ou l'espace disque d'une animation MIAW simulant une boîte de dialogue.

Vous pouvez créer une référence à un objet Xtra MUI à l'aide de l'opérateur `new`.

```
-- Lingo syntax
objMui = new xtra("Mui")

// JavaScript syntax
var objMui = new xtra("Mui");
```

Récapitulatif des méthodes pour l'objet XML Parser

Méthode
Alert()
fileOpen()
fileSave()
GetItemPropList
getURL()
GetWidgetList()
GetWindowPropList
Initialize
ItemUpdate()
run
stop
WindowOperation

NetLingo

Permet d'effectuer des opérations de réseau comme par exemple, obtenir ou lire en continu un support à partir d'un réseau, vérifier la disponibilité du réseau, vérifier la progression d'une opération de réseau, etc.

Vous pouvez créer une référence à un objet NetLingo à l'aide de l'opérateur `new`.

```
-- Lingo syntax
objNetLingo = new xtra("netlingo")

// JavaScript syntax
var objNetLingo = new xtra("netlingo");
```

Récapitulatif des méthodes pour l'objet NetLingo

Méthode	
browserName()	netDone()
cacheDocVerify()	netError()
cacheSize()	netLastModDate()
clearCache	netMIME()
downloadNetThing	netStatus
externalEvent()	netTextResult()
getLatestNetID	postNetText
getNetText()	preloadNetThing()
getStreamStatus()	proxyServer
gotoNetMovie	tellStreamStatus()
gotoNetPage	URLEncode
netAbort	

SpeechXtra

Permet d'ajouter à une animation la fonctionnalité de conversion de texte en voix.

Vous pouvez créer une référence à un objet SpeechXtra à l'aide de l'opérateur `new`.

```
-- Lingo syntax
objSpeech = new xtra("speechxtra")

// JavaScript syntax
var objSpeech = new xtra("speechxtra");
```

Récapitulatif des méthodes pour l'objet SpeechXtra

Méthode	
voiceCount()	voiceSet()
voiceGet()	voiceSetPitch()
voiceGetAll()	voiceSetRate()
voiceGetPitch()	voiceSetVolume()
voiceGetRate()	voiceSpeak()
voiceGetVolume()	voiceState()
voiceInitialize()	voiceStop()
voicePause()	voiceWordPos()
voiceResume()	

XML Parser

Permet d'effectuer des analyses XML.

Vous pouvez créer une référence à un objet XML Parser à l'aide de l'opérateur `new`.

```
-- Lingo syntax
objXml = new xtra("xmlparser")

// JavaScript syntax
var objXml = new xtra("xmlparser");
```

Récapitulatif des méthodes pour l'objet XML Parser

Méthode
count()
doneParsing()
getError() (XML)
ignoreWhiteSpace()
makeList()
makeSubList()
parseString()
parseURL()

Récapitulatif des propriétés pour l'objet XML Parser

Propriété
attributeName
attributeValue
child (XML)
name (XML)

Chapitre 8 : Objets 3D

Les objets 3D permettent d'ajouter des fonctionnalités 3D à une animation. Ces objets sont à la fois exposés à Lingo et à la syntaxe JavaScript au sein de Director, des projections et d'Adobe® Shockwave® Player.

Vous avez accès à ces objets 3D par le biais des acteurs Shockwave 3D (ou, tout simplement, 3D). Vous pouvez également créer des images-objets 3D à partir des acteurs 3D. Les acteurs 3D ainsi que les images-objets 3D comportent des fonctionnalités qui leur sont propres. Ils ont également accès aux fonctionnalités des acteurs et images-objets autres que 3D dont les API sont respectivement indiqués par les objets principaux `Member` et `Sprite`.

Un acteur 3D est différent des autres acteurs car il contient un univers 3D complet. Un univers 3D contient les objets permettant d'accéder aux fonctionnalités 3D. Tous les objets d'un univers 3D sont basés sur un objet principal appelé nœud. La forme la plus simple d'un nœud dans un univers 3D est un objet Groupe ; cet objet Groupe est fondamentalement le nœud de base. Tous les autres objets d'un univers 3D sont basés sur un objet Groupe, ce qui signifie que les autres objets héritent des fonctionnalités d'un objet Groupe en plus de posséder des fonctionnalités propres à ces objets.

Pour voir comment les objets 3D sont liés les uns aux autres, ainsi qu'aux autres objets de Director, consultez « Diagrammes de modèles d'objets », page 46.

Director® est livré avec deux Xtras vous donnant accès aux objets 3D :

- Xtra 3D Asset (3DAuth.x32 sous Windows®, 3D Auth Xtra sous Macintosh®) prend en charge la fenêtre média 3D dans Director.
- Xtra 3D Media (Shockwave 3D Asset.x32 sous Windows, 3D Asset Xtra sous Macintosh) prend en charge le média 3D en lui-même.

Pour accéder aux objets 3D lors de la création ou de l'exécution, votre animation doit comprendre un Xtra 3D Asset.

Caméra

Représente un objet Caméra.

Une caméra contrôle la manière dont une image-objet 3D visualise l'univers 3D. Une image-objet 3D affiche une vue de caméra particulière dans l'univers.

Vous pouvez créer une référence à une caméra en utilisant la propriété `camera` de l'objet 3D `Member`. La propriété `camera` choisit la caméra située à un endroit précis de l'index dans la liste des caméras. Dans Lingo, vous pouvez directement utiliser la propriété `camera` de l'objet 3D `Member` pour créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence à la deuxième caméra de la « chambre familiale » de l'acteur 3D est créée et affectée à la variable `myCamera`.

```
-- Lingo syntax
myCamera = member("family room").camera[2]

// JavaScript syntax
var myCamera = member("family room").getPropRef("camera", 2);
```

Récapitulatif des méthodes pour l'objet Caméra

Méthode
addBackdrop
addOverlay
insertBackdrop
insertOverlay
removeBackdrop
removeOverlay

Récapitulatif des propriétés pour l'objet Caméra

Propriété	
backdrop	fog.far (brouillard)
backdrop[].blend (3D)	fog.near (brouillard)
backdrop[].loc (fond et recouvrement)	hither
backdrop[].regPoint (3D)	orthoHeight
backdrop[].rotation (fond et recouvrement)	overlay
backdrop[].scale (3D)	overlay[].blend (3D)
backdrop[].source	overlay[].loc (fond et recouvrement)
backdrop.count (3D)	overlay[].regPoint (3D)
child (3D)	overlay[].rotation (fond et recouvrement)
colorBuffer.clearAtRender	overlay[].scale (3D)
colorBuffer.clearValue	overlay[].source
fieldOfView (3D)	overlay.count (3D)
fog.color()	projection
fog.decayMode	rootNode
fog.enabled (brouillard)	yon

Voir aussi

`Group`, `Lumière`, `Modèle`, `Ressource de modèle`, `Mouvement`, `Matériau`, `Texture`

Group

Représente un modèle n'ayant ni ressource ni matériaux.

Un groupe est un nœud de base et correspond tout simplement à un point dans l'espace qui est représenté par une transformation. Vous pouvez affecter des enfants et des parents à ce nœud afin de regrouper des modèles, des lumières, des caméras ou d'autres groupes.

Le groupe de base est appelé un univers, qui est essentiellement synonyme de l'univers 3D de l'acteur proprement dit.

Vous pouvez créer une référence à un groupe en utilisant la propriété `group` de l'objet 3D `Member`. La propriété `group` choisit le groupe situé à un endroit précis de l'index dans la liste des groupes. Dans Lingo, vous pouvez directement utiliser la propriété `group` de l'objet 3D `Member` pour créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence au premier groupe de l'acteur 3D `space` est créée et affectée à la variable `myGroupe`.

```
-- Lingo syntax
myGroup = member("space").group[1]

// JavaScript syntax
var myGroup = member("space").getPropRef("group", 1);
```

Récapitulatif des méthodes pour l'objet Groupe

Méthode	
addChild	pointAt
addToWorld	registerScript()
clone	removeFromWorld
cloneDeep	rotate
getWorldTransform()	scale (commande)
isInWorld()	translate

Récapitulatif des propriétés pour l'objet Groupe

Propriété
name (3D)
parent
pointAtOrientation
transform (propriété)
userData
worldPosition

Voir aussi

`Caméra`, `Lumière`, `Modèle`, `Ressource de modèle`, `Mouvement`, `Matériau`, `Texture`

Lumière

Représente une lumière dans un univers 3D.

Les lumières sont utilisées pour éclairer un univers 3D. Sans lumière, les objets de l'univers ne seraient pas visibles.

Vous pouvez créer une référence à une lumière en utilisant la propriété `light` de l'objet 3D `Member`. La propriété `light` choisit la lumière située à un endroit précis de l'index dans la liste des lumières. Dans Lingo, vous pouvez directement utiliser la propriété `light` de l'objet 3D `Member` pour créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence à la troisième lumière de la « salle d'animation » de l'acteur 3D est créée et affectée à la variable `myLight`.

```
-- Lingo syntax
myLight = member("film room").light[3]

// JavaScript syntax
var myLight = member("film room").getPropRef("light", 3);
```

Récapitulatif des propriétés pour l'objet Lumière

Propriété
attenuation
color (lumière)
specular (lumière)
spotAngle
spotDecay
type (lumière)

Voir aussi

`Caméra`, `Group`, `Modèle`, `Ressource de modèle`, `Mouvement`, `Matériau`, `Texture`

Acteur

Représente un acteur Shockwave 3D.

Un acteur Shockwave 3D (ou, tout simplement, 3D) contient un univers 3D complet. Un univers 3D contient l'ensemble des objets que vous utilisez pour ajouter des fonctionnalités 3D à une animation.

Vous pouvez créer une référence à un acteur 3D en utilisant la fonction de haut niveau `member()` ou la propriété `member` de l'objet Animation ou Image-objet. Ces techniques sont les mêmes que celles utilisées pour créer une référence à un acteur qui n'est pas un acteur 3D.

- Utilisez la fonction de haut niveau `member()`.

```
-- Lingo syntax
3dMember = member("magic")

// JavaScript syntax
var 3dMember = member("magic");
```

- Utilisez la propriété `member` de l'objet Image-objet.

```
-- Lingo syntax
3dMember = sprite(1).member;

// JavaScript syntax
```

```
var 3dMember = sprite(1).member;
```

Récapitulatif des méthodes pour l'objet Acteur

Méthode	
camera()	model
cloneModelFromCastmember	modelResource
cloneMotionFromCastmember	motion()
deleteCamera	newCamera
deleteGroup	newGroup
deleteLight	newLight
deleteModel	newMesh
deleteModelResource	newModel
deleteMotion	newModelResource
deleteShader	newShader
deleteTexture	newTexture
extrude3D	resetWorld
group()	revertToWorldDefaults
light()	shader()
loadFile()	texture()

Récapitulatif des propriétés pour l'objet Acteur

Propriété	
ambientColor	loop (3D)
animationEnabled	model
bevelDepth	modelResource
bevelType	motion
bytesStreamed (3D)	percentStreamed (3D)
camera	preLoad (3D)
cameraPosition	reflectivity
cameraRotation	shader
diffuseColor	smoothness
directionalColor	specularColor
directionalPreset	state (3D)
directToStage	streamSize (3D)
displayFace	texture

Propriété (Suite)	
displayMode	textureMember
group	textureType
light	tunnelDepth

Voir aussi

Caméra, Group, Lumière, Modèle, Ressource de modèle, Mouvement, Matériau, Image-objet, Texture

Modèle

Représente un objet visible qu'un utilisateur peut voir dans un univers 3D.

Un modèle utilise une ressource de modèle et occupe une position et une orientation particulières dans un univers 3D. Une ressource de modèle est un élément de géométrie 3D qui peut être utilisé pour dessiner des modèles 3D. Un modèle définit également l'apparence de la ressource de modèle, telle que les textures et les matériaux utilisés. Pour plus d'informations sur la relation entre les modèles et les ressources de modèle, consultez les rubriques du document Utilisation de Director dans l'Aide de Director.

Vous pouvez créer une référence à un modèle en utilisant la propriété `model` de l'objet 3D `Member`. La propriété `model` choisit le modèle situé à un endroit précis de l'index dans la liste des modèles. Dans Lingo, vous pouvez directement utiliser la propriété `model` de l'objet 3D `Member` pour créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence au deuxième modèle de l'acteur 3D `Transportation` est créée et affectée à la variable `myModel`.

```
-- Lingo syntax
myModel = member("Transportation").model[2]

// JavaScript syntax
var myModel = member("Transportation").getPropRef("model", 2);
```

Un modèle comprend également des modificateurs qui contrôlent le rendu du modèle et le comportement de son animation. Les modificateurs sont associés à un modèle en utilisant la méthode `addModifier()`. Lorsqu'un modificateur est associé à un modèle, ses propriétés peuvent être manipulées à l'aide d'un script.

Les modificateurs suivants sont disponibles avec un modèle :

Modificateur	Description
Lecteur de segments	Modifie la géométrie d'un modèle dans le temps.
Collision	Permet à un modèle d'être informé d'une collision et d'y répondre.
Inker	Ajoute une silhouette, des plis et des bords de délimitation à un modèle existant.
Lecteur d'images-clés	Modifie les propriétés `transform` d'un modèle dans le temps.
Niveau de détail (LOD)	Fournit un contrôle par modèle sur le nombre de polygones utilisés pour rendre un modèle, en fonction de sa distance par rapport à la caméra. Le modificateur LOD est également disponible pour les ressources de modèle.

Modificateur (Suite)	Description (Suite)
Déformation de maille	Modifie la géométrie d'une ressource de modèle existante lors de l'exécution.
Subdivision surfaces (SDS)	Entraîne le rendu du modèle avec davantage de détails géométriques dans la zone du modèle que la caméra est en train de fixer.
Dessin animé	Modifie le rendu d'un modèle pour imiter un style de dessin animé.

Pour plus d'informations sur les méthodes, propriétés et événements disponibles pour les modificateurs, consultez les rubriques du document Utilisation de Director dans l'Aide de Director.

Ressource de modèle

Représente un élément de la géométrie 3D utilisé pour dessiner des modèles 3D.

Un modèle utilise une ressource de modèle et occupe une position et une orientation particulières dans un univers 3D. Un modèle définit également l'apparence de la ressource de modèle, telle que les textures et les matériaux utilisés.

Pour plus d'informations sur la relation entre les modèles et les ressources de modèle et leur utilisation, consultez les rubriques du document Utilisation de Director dans l'Aide de Director.

Vous pouvez créer une référence à une ressource de modèle en utilisant la propriété `modelResource` de l'objet 3D `Member`. La propriété `modelResource` choisit la ressource de modèle située à un endroit précis de l'index dans la liste des ressources de modèle. Dans Lingo, vous pouvez directement utiliser la propriété `modelResource` de l'objet 3D `Member` pour créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence à la deuxième ressource de modèle de l'acteur 3D `wheels` est créée et affectée à la variable `myModelResource`.

```
-- Lingo syntax
myModelResource = member("wheels").modelResource[2]

// JavaScript syntax
var myModelResource = member("wheels").getPropRef("modelResource", 2);
```

Mouvement

Représente une séquence d'animation prédéfinie qui implique le mouvement d'un modèle ou d'un composant de modèle.

Les mouvements individuels peuvent être définis pour être exécutés seuls ou avec d'autres mouvements. Par exemple, le mouvement d'un personnage qui court peut être combiné à un mouvement de saut pour simuler le saut au-dessus d'une flaque d'eau.

Vous pouvez créer une référence à un mouvement en utilisant la propriété `motion` de l'objet 3D `Member`. La propriété `motion` choisit le mouvement situé à un endroit précis de l'index dans la liste des mouvements. Dans Lingo, vous pouvez directement utiliser la propriété `motion` de l'objet 3D `Member` pour créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence au quatrième mouvement de l'acteur 3D `athlete` est créée et affectée à la variable `myMotion`.

```
-- Lingo syntax
myMotion = member("athlete").motion[4]

// JavaScript syntax
var myMotion = member("athlete").getPropRef("motion", 4);
```

Services de rendu

Représente l'objet global qui comprend une liste de propriétés dont les valeurs influencent les propriétés de rendu communes de tous les acteurs et images-objets 3D.

Vous pouvez accéder à l'objet de services de rendu global en utilisant la fonction de haut niveau `myRenderer`.

Dans l'exemple suivant, on accède à la propriété de rendu de l'objet de services de rendu global et la valeur est affectée à la variable `monRendu`.

```
-- Lingo syntax
myRenderer = getRendererServices().renderer

// JavaScript syntax
var myRenderer = getRendererServices().renderer;
```

Récapitulatif des méthodes pour l'objet Services de rendu

Méthode
getHardwareInfo()

Récapitulatif des propriétés pour l'objet Services de rendu

Propriété
modifiers
primitives
renderer
rendererDeviceList
textureRenderFormat

Voir aussi

Acteur, Image-objet

Matériau

Représente la couleur de la surface d'un modèle.

Des images peuvent être dessinées à la surface d'un modèle en appliquant une ou plusieurs textures à chaque matériau.

Vous pouvez créer une référence à un matériau en utilisant la propriété `shader` de l'objet 3D `Member`. La propriété `shader` choisit la nuance située à un endroit précis de l'index dans la liste des matériaux. Dans Lingo, vous pouvez directement utiliser la propriété `shader` de l'objet 3D `Member` pour créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence au deuxième matériau de l'acteur 3D `triangle` est créée et affectée à la variable `myShader`.

```
-- Lingo syntax
myShader = member("triangle").shader[2]

// JavaScript syntax
var myShader = member("triangle").getPropRef("shader", 2);
```

Image-objet

Représente une image-objet 3D créée à partir d'un acteur Shockwave 3D.

Vous pouvez créer une référence à une Image-objet 3D en utilisant la fonction de haut niveau `sprite()`, la propriété `sprite` de l'objet Animation ou la propriété `sprite` de l'objet Piste d'image-objet. Ces techniques sont les mêmes que celles utilisées pour créer une référence à une image-objet qui n'est pas une image-objet 3D.

- Utilisez la fonction de haut niveau `sprite()`.

  ```
  -- Lingo syntax
  3dSprite = sprite(1)

  // JavaScript syntax
  var 3dSprite = sprite(1);
  ```

- Utilisez la propriété `sprite` de l'objet Animation.

  ```
  -- Lingo syntax
  3dSprite = _movie.sprite["willowTree"]

  // JavaScript syntax
  var 3dSprite = _movie.sprite["willowTree"];
  ```

- Utilisez la propriété `sprite` de l'objet Piste d'image-objet.

  ```
  -- Lingo syntax
  3dSprite = channel(3).sprite

  // JavaScript syntax
  var 3dSprite = channel(3).sprite;
  ```

Récapitulatif des méthodes pour l'objet Image-objet

Méthode
addCamera
cameraCount()
deleteCamera

Récapitulatif des propriétés pour l'objet Image-objet

Propriété
antiAliasingEnabled
backColor
camera
directToStage

Voir aussi

`Caméra`, `Acteur`

Texture

Représente la texture appliquée à un matériau.

Vous pouvez créer une référence à une texture en utilisant la propriété `texture` de l'objet 3D `Member`. La propriété `texture` choisit la texture située à un endroit précis de l'index dans la liste des textures. Dans Lingo, vous pouvez directement utiliser la propriété `texture` de l'objet 3D `Member` pour créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence à la première texture de l'acteur 3D `triangle` est créée et affectée à la variable `myTexture`.

```
-- Lingo syntax
myTexture = member("triangle").texture[1]

// JavaScript syntax
var myTexture = member("triangle").getPropRef("texture", 1);
```

Chapitre 9 : Constantes

Ce chapitre répertorie dans l'ordre alphabétique toutes les constantes disponibles dans Director®.

La plupart de ces constantes ne s'appliquent qu'à Lingo. Toutefois, la syntaxe JavaScript comporte certaines constantes semblables aux constantes Lingo répertoriées ici ; lorsque tel est le cas, ce chapitre indique la syntaxe JavaScript et fournit des exemples d'utilisation de cette dernière pour vous aider à mettre en correspondance les constantes Lingo avec leurs équivalents JavaScript les plus proches. Pour plus d'informations sur les constantes de la syntaxe JavaScript, reportez-vous à l'une des nombreuses documentations d'autres éditeurs sur le sujet.

" (chaîne)

Syntaxe

```
--Lingo syntax
"

// JavaScript syntax
"
```

Description

Constante de chaîne ; lorsqu'ils sont utilisés avant et après une chaîne, les guillemets droits signifient que la chaîne qu'ils contiennent est une chaîne littérale, et non une variable, une valeur numérique ou un élément de script. Les guillemets droits doivent toujours encadrer les noms littéraux des acteurs, des distributions, des fenêtres et des fichiers externes.

Exemple

L'instruction suivante utilise les guillemets droits pour indiquer que la chaîne « San Francisco » est une chaîne littérale, le nom d'un acteur :

```
--Lingo syntax
put member("San Francisco").loaded

// JavaScript syntax
put(member("San Francisco").loaded);
```

Voir aussi

QUOTE

BACKSPACE

Syntaxe

```
-- Lingo syntax
BACKSPACE

// JavaScript syntax
51 // value of _key.keyCode
```

Description

Constante ; représente la touche Arrière/Retour arrière. Cette touche est appelée Retour arrière sur un clavier Windows® et Arrière sur un clavier Mac®.

Exemple

Le gestionnaire `on keyDown` suivant vérifie si l'utilisateur a appuyé sur la touche Retour arrière et, le cas échéant, appelle le gestionnaire `clearEntry` :

```
--Lingo syntax
on keyDown
    if (_key.key = BACKSPACE) then clearEntry
    _movie.stopEvent()
end keyDown

// JavaScript syntax
function keyDown() {
    if (_key.keyCode == 51) {
        clearEntry();
        _movie.stopEvent();
    }
}
```

EMPTY

Syntaxe

```
--Lingo syntax
EMPTY

// JavaScript syntax
""
```

Description

Constante de caractères ; représente la chaîne vide, `""`, c'est-à-dire une chaîne sans caractères.

Exemple

L'instruction suivante efface tous les caractères de l'acteur champ Notice en donnant au champ la valeur `EMPTY` :

```
--Lingo syntax
member("Notice").text = EMPTY

// JavaScript syntax
member("Notice").text = "";
```

ENTER

Syntaxe

```
--Lingo syntax
ENTER

// JavaScript syntax
3 // value of _key.keyCode
```

Description

Constante de caractères ; représente la touche Entrée sur un clavier Windows ou la touche Retour sur un clavier Mac correspondant à un retour chariot.

Sur les claviers PC, l'élément `ENTER` ne se rapporte qu'à la touche Entrée du pavé numérique.

Pour une animation lue en tant qu'applet, utilisez `RETURN` pour spécifier la touche Retour sous Windows et la touche Entrée sur le Mac.

Exemple

L'instruction suivante vérifie si l'utilisateur a appuyé sur la touche Entrée et, le cas échéant, envoie la tête de lecture sur l'image `addSum` :

```
-- Lingo syntax
on keyDown
    if (_key.key = ENTER) then _movie.go("addSum")
end

// JavaScript syntax
function keyDown() {
    if (_key.keyCode == 3) {
        _movie.go("addSum");
    }
}
```

Voir aussi

RETURN (constante)

FALSE

Syntaxe

```
-- Lingo syntax
FALSE

// JavaScript syntax
false
```

Description

Constante ; s'applique à une expression qui est logiquement fausse (`FALSE`), telle que 2 > 3. Lorsqu'elle est traitée comme un nombre, la constante `FALSE` correspond à la valeur numérique 0. Inversement, 0 est traité comme `FALSE`.

Exemple

L'instruction suivante désactive la propriété `soundEnabled` en lui attribuant la valeur `FALSE` :

```
-- Lingo syntax
_sound.soundEnabled = FALSE

// JavaScript syntax
_sound.soundEnabled = false;
```

Voir aussi

if, not, TRUE

PI

Syntaxe

```
-- Lingo syntax
PI

// JavaScript syntax
Math.PI
```

Description

Constante ; renvoie la valeur de pi (π), c'est-à-dire le rapport de la circonférence d'un cercle à son diamètre, sous la forme d'un nombre à virgule flottante. La valeur est arrondie au nombre de décimales défini par la propriété `floatPrecision`.

Exemple

L'instruction suivante utilise la constante `PI` dans une équation destinée à calculer l'aire d'un cercle :

```
-- Lingo syntax
vRadius = 3
vArea = PI*power(vRadius, 2)
trace(vArea) -- results in 28.2743

// JavaScript syntax
var vRadius = 3;
vArea = Math.PI*Math.pow(vRadius, 2);
trace(vArea); // results in 28.274333882308138
```

QUOTE

Syntaxe

```
--Lingo syntax
QUOTE

// JavaScript syntax
\"
```

Description

Constante ; représente le caractère guillemet dans une chaîne puisque ce caractère est lui-même utilisé dans les scripts Lingo pour délimiter les chaînes.

Exemple

L'instruction suivante insère des guillemets dans une chaîne :

```
-- Lingo syntax
put("Can you spell" && QUOTE & "Adobe" & QUOTE & "?")

// JavaScript syntax
put("Can you spell \"Adobe\"?");
```

Le résultat correspond à un jeu de guillemets placés autour du mot Adobe® :

```
Can you spell "Adobe"?
```

RETURN (constante)

Syntaxe

```
-- Lingo syntax
RETURN

// JavaScript syntax
36 // value of _key.keyCode
\n // when used in a string
```

Description

Constante ; représente un retour de chariot.

Exemple

L'instruction suivante reprend la lecture d'une animation en pause lorsque l'utilisateur appuie sur le retour de chariot :

```
-- Lingo syntax
if (_key.key = RETURN) then _movie.go(_movie.frame + 1)

// JavaScript syntax
if (_key.keyCode == 36) {
    _movie.go(_movie.frame + 1);
}
```

L'instruction suivante utilise la constante de caractères `RETURN` pour insérer un retour chariot entre deux lignes d'un message d'alerte :

```
-- Lingo syntax
_player.alert("Last line in the file." & RETURN & "Click OK to exit.")

// JavaScript syntax
_player.alert("Last line in the file." + "\n" + " Click OK to exit");
```

Sous Windows, vous devez normalement insérer un caractère de saut de ligne supplémentaire à la fin de chaque ligne. L'instruction suivante crée une chaîne de deux caractères nommée CRLF fournissant le saut de ligne additionnel :

```
CRLF = RETURN & numToChar(10)
```

SPACE

Syntaxe

```
-- Lingo syntax
SPACE

// JavaScript syntax
49 // value of _key.keyCode
```

Description

Constante ; valeur en lecture seule représentant un espace.

Exemple

L'instruction suivante affiche « Age de bronze » dans la fenêtre Messages :

```
-- Lingo syntax
put("Age"&SPACE&"Of"&SPACE&"Aquarius")
```

TAB

Syntaxe

```
-- Lingo syntax
TAB

// JavaScript syntax
48 // value of _key.keyCode
```

Description

Constante ; représente la touche de tabulation.

Exemple

L'instruction suivante vérifie si le caractère saisi est le caractère de tabulation et, le cas échéant, appelle le gestionnaire `doNextField` :

```
-- Lingo syntax
if (_key.key = TAB) then doNextField

// JavaScript syntax
if (_key.keyCode == 48) {
    doNextField();
}
```

Les instructions suivantes font avancer ou reculer la tête de lecture selon que l'utilisateur appuie sur les touches Tabulation ou Maj+Tabulation :

```
-- Lingo syntax
if (_key.key = TAB) then
    if (_key.shiftDown) then
        _movie.go(_movie.frame - 1)
    else
        _movie.go(_movie.frame + 1)
    end if
end if

// JavaScript syntax
if (_key.keyCode == 48) {
    if (_key.shiftDown) {
        _movie.go(_movie.frame - 1);
    } else {
        _movie.go(_movie.frame + 1);
    }
}
```

Voir aussi

BACKSPACE, EMPTY, RETURN (constante)

TRUE

Syntaxe

```
-- Lingo syntax
TRUE

// JavaScript syntax
true
```

Description

Constante ; représente la valeur d'une expression logique, telle que `2 < 3`. Elle correspond généralement à une valeur numérique de 1, mais tout entier non nul donne `TRUE` dans une comparaison.

Exemple

L'instruction suivante active la propriété `soundEnabled` en lui attribuant la valeur `TRUE` :

```
-- Lingo syntax
_sound.soundEnabled = TRUE

// JavaScript syntax
_sound.soundEnabled = true;
```

Voir aussi

`FALSE`, `if`

VOID

Syntaxe

```
-- Lingo syntax
VOID

// JavaScript syntax
null
```

Description

Constante ; indique la valeur `VOID`.

Exemple

L'instruction suivante vérifie si la variable `currentVariable` présente la valeur `VOID` :

```
-- Lingo syntax
if currentVariable = VOID then
    put("This variable has no value")
end if

// JavaScript syntax
if (currentVariable == undefined) {
    put("This variable has no value");
}
```

Voir aussi

`voidP()`

Chapitre 10 : Evénements et messages

Ce chapitre répertorie dans l'ordre alphabétique tous les événements et messages disponibles dans Director®.

on activateApplication

Syntaxe

```
-- Lingo syntax
on activateApplication
    statement(s)
end

// JavaScript syntax
function activateApplication() {
    statement(s);
}
```

Description

Gestionnaire intégré ; exécuté lorsque la projection passe au premier plan. Ce gestionnaire est pratique lorsqu'une projection est exécutée dans une fenêtre et que l'utilisateur l'envoie à l'arrière-plan pour travailler dans une autre application. Le gestionnaire est exécuté lorsque la projection repasse au premier plan. Toutes les MIAW exécutées dans la projection peuvent également utiliser ce gestionnaire.

En cours de programmation, ce gestionnaire n'est appelé que lorsque l'option Animer en arrière-plan est activée dans les préférences générales.

Sous Windows®, ce gestionnaire n'est pas appelé lorsque la projection n'est que réduite et qu'aucune autre application n'est passée au premier plan.

Exemple

Le gestionnaire suivant lit un son à chaque fois que l'utilisateur ramène la projection au premier plan :

```
-- Lingo syntax
on activateApplication
    sound(1).queue(member("openSound"))
    sound(1).play()
end

// JavaScript syntax
function activateApplication() {
    sound(1).queue(member("openSound"));
    sound(1).play();
}
```

Voir aussi

on deactivateApplication, activeCastLib, on deactivateWindow

on activateWindow

Syntaxe

```
-- Lingo syntax
on activateWindow
        statement(s)
end

// JavaScript syntax
function activateWindow()
    statement(s);
}
```

Description

Message système et gestionnaire d'événements ; contient des instructions exécutées dans une animation lorsque l'utilisateur clique sur la fenêtre inactive et que la fenêtre passe au premier plan.

Vous pouvez utiliser un gestionnaire `on activateWindow` dans un script que vous souhaitez exécuter chaque fois que l'animation devient active.

Le fait de cliquer sur l'animation principale (scène principale) ne génère pas de gestionnaire `on activateWindow`.

Exemple

Le gestionnaire suivant lit le son Hourra lorsque la fenêtre dans laquelle l'animation est exécutée devient active :

```
-- Lingo syntax
on activateWindow
    sound(2).play(member("Hurray"))
end

// JavaScript syntax
function activateWindow() {
    sound(2).play(member("Hurray"));
}
```

Voir aussi

activeWindow, close(), on deactivateWindow, frontWindow, on moveWindow, open() (fenêtre)

on beginSprite

Syntaxe

```
-- Lingo syntax
on beginSprite
    statement(s)
end

// JavaScript syntax
function beginSprite() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées lorsque la tête de lecture passe à une image contenant une image-objet jusqu'alors inconnue. A l'instar de `endSprite`, cet événement est généré une seule fois, même si la tête de lecture effectue une boucle sur une image, étant donné que le déclencheur est une image-objet que la tête de lecture n'avait pas encore rencontrée. L'événement est généré avant `prepareFrame`.

Director crée des instances de tout script de comportement lié à l'image-objet lorsque le message `beginSprite` est envoyé.

La référence d'objet `me` est transmise à cet événement s'il est utilisé dans un comportement. Le message est envoyé aux comportements et aux scripts d'images.

Si une image-objet commence dans la première image lue dans le cadre de l'animation, le message `beginSprite` est envoyé après le message `prepareMovie`, mais avant les messages `prepareFrame` et `startMovie`.

Remarque : *N'oubliez pas que certaines propriétés d'image-objet, telles que `rect`, ne sont peut-être pas accessibles dans un gestionnaire beginSprite. En effet, le système doit calculer la propriété, ce qui ne peut se faire tant que l'image-objet n'a pas été dessinée.*

Les commandes `go`, `play` et `updateStage` sont désactivées dans un gestionnaire `on beginSprite`.

Exemple

Le gestionnaire suivant lit l'acteur son Georges Brassens lorsque l'image-objet commence :

```
-- Lingo syntax
on beginSprite me
    sound(1).play(member("Stevie Wonder"))
end

// JavaScript syntax
function beginSprite() {
    sound(1).play(member("Stevie Wonder"));
}
```

Voir aussi

`on endSprite`, `on prepareFrame`, `scriptInstanceList`

on closeWindow

Syntaxe

```
-- Lingo syntax
on closeWindow
    statement(s)
end

// JavaScript syntax
function closeWindow() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient les instructions exécutées lorsque l'utilisateur ferme la fenêtre d'une animation en cliquant sur sa case de fermeture.

Le gestionnaire `on closeWindow` constitue un emplacement adéquat pour insérer les commandes Lingo que vous souhaitez exécuter à chaque fermeture de la fenêtre de l'animation.

Exemple

Le gestionnaire suivant transmet à Director la commande `forget` pour qu'il libère la fenêtre en cours de la mémoire lorsque l'utilisateur ferme la fenêtre dans laquelle l'animation est en cours de lecture :

```
-- Lingo syntax
on closeWindow
    -- perform general housekeeping here
    window(1).forget()
end

// JavaScript syntax
function closeWindow() {
    // perform general housekeeping here
    window(1).forget();
}
```

on cuePassed

Syntaxe

```
-- Lingo syntax
on cuePassed({me,} channelID, cuePointNumber,cuePointName)
    statement(s)
end

// JavaScript syntax
function cuePassed(channelID, cuePointNumber,cuePointName) {
    statement(s);
}
```

Description

Message système et gestionnaire d'événements ; contient des instructions exécutées à chaque fois qu'un son ou qu'une image-objet passe par un point de repère sur son média.

- `me` Le paramètre facultatif `me` représente la valeur `scriptInstanceRef` du script appelé. Vous devez inclure ce paramètre si vous utilisez le message dans un comportement. Si vous omettez ce paramètre, les autres arguments ne sont pas traités correctement.
- *`channelID`* Numéro de la piste audio ou d'image-objet spécifique du fichier où le point de repère a été franchi.
- *`cuePointNumber`* Nombre ordinal du point de repère qui déclenche l'événement dans la liste des points de repère de l'acteur.
- *`cuePointName`* Nom du point de repère rencontré.

Le message est transmis, dans l'ordre, aux scripts d'image-objet, d'acteur, d'image et d'animation. Pour que l'image-objet reçoive l'événement, elle doit être la source du son, telle qu'une animation QuickTime® ou un acteur SWA. Utilisez la propriété `isPastCuePoint` pour vérifier les points de repère dans les comportements relatifs aux images-objets qui ne génèrent aucun son.

Exemple

Le gestionnaire suivant placé dans un script d'animation ou d'image affiche tous les points de repère pour la piste audio 1 dans la fenêtre Messages :

```
-- Lingo syntax
on cuePassed channel, number, name
    if (channel = #Sound1) then
        put("CuePoint" && number && "named" && name && "occurred in sound 1")
    end if
end

// JavaScript syntax
function cuePassed(channel, number, name) {
    if (channel == symbol("Sound1")) {
        put("CuePoint " + number + " named " + name + "occurred in sound 1");
    }
}
```

Voir aussi

scriptInstanceList, cuePointNames, cuePointTimes, isPastCuePoint()

on deactivateApplication

Syntaxe

```
-- Lingo syntax
on deactivateApplication
    statement(s)
end

// JavaScript syntax
function deactivateApplication() {
    statement(s);
}
```

Description

Gestionnaire intégré ; exécuté lorsque la projection passe à l'arrière-plan. Ce gestionnaire est utile lorsqu'une projection est exécutée dans une fenêtre et que l'utilisateur l'envoie à l'arrière-plan pour travailler dans une autre application. Toutes les MIAW exécutées dans la projection peuvent également utiliser ce gestionnaire.

En cours de programmation, ce gestionnaire n'est appelé que lorsque l'option Animer en arrière-plan est activée dans les préférences générales.

Sous Windows, ce gestionnaire n'est pas appelé lorsque la projection n'est que réduite et qu'aucune application n'est passée au premier plan.

Exemple

Le gestionnaire suivant lit un son à chaque fois que l'utilisateur envoie la projection à l'arrière-plan :

```
-- Lingo syntax
on deactivateApplication
    sound(1).queue(member("closeSound"))
    sound(1).play()
end

// JavaScript syntax
function deactivateApplication() {
    sound(1).queue(member("closeSound"));
    sound(1).play();
}
```

Voir aussi

add (texture 3D), activeCastLib, on deactivateWindow

on deactivateWindow

Syntaxe

```
-- Lingo syntax
on deactivateWindow
    statement(s)
end

// JavaScript syntax
function deactivateWindow() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événements ; contient des instructions exécutées lorsque la fenêtre dans laquelle l'animation est lue est désactivée. Le gestionnaire d'événement `on deactivate` constitue un emplacement adéquat pour insérer le code Lingo que vous souhaitez exécuter chaque fois qu'une fenêtre est désactivée.

Exemple

Le gestionnaire suivant lit le son Ronflement lorsque la fenêtre dans laquelle l'animation est lue est désactivée :

```
-- Lingo syntax
on deactivateWindow
    sound(2).play(member("Snore"))
end

// JavaScript syntax
function deactivateWindow() {
    sound(2).play(member("Snore"));
}
```

on DVDeventNotification

Syntaxe

```
-- Lingo syntax
on DVDeventNotification objectRef, event {, eventArg1} {, eventArg2} {, eventArg3}
    statement(s)
end DVDeventNotification

// JavaScript syntax
function DVDeventNotification (objectRef, event {, eventArg1} {, eventArg2} {, eventArg3}) {
    statement(s);
}
```

Description

Gestionnaire d'événement de DVD spécifié par l'auteur. Contient des instructions qui s'exécutent en réponse aux événements survenant pendant la lecture d'un DVD.

Ce gestionnaire peut être utilisé pour assurer le suivi de tous les événements de DVD. Dans les exemples de script ci-dessus, *objectRef*, le premier paramètre transmis au gestionnaire DVDeventNotification, est une référence à l'objet DVDeventNotification proprement dit. L'événement réel qui se produit est toujours transmis en tant que deuxième paramètre, *event*. Certains événements contiennent des informations supplémentaires les concernant sous la forme d'un troisième paramètre, *eventArg1*. Dans certains cas, un quatrième et un cinquième paramètre, *eventArg2* et *eventArg3*, peuvent comporter des informations d'événement supplémentaires.

Le tableau suivant répertorie les événements pouvant survenir pendant la lecture d'un DVD.

Evénement	**Description**
angleChange	Survient en cas de modification du nombre d'angles disponibles ou du numéro d'angle de l'utilisateur en cours. Lorsque cet événement survient, les informations supplémentaires suivantes sont transmises à `DVDeventNotification` : • *eventArg2* : nombre entier indiquant le nombre d'angles disponibles. Lorsque le nombre d'angles disponibles est de 1, la vidéo en cours n'est pas multiangle. • *eventArg3* : nombre entier indiquant le numéro d'angle de l'utilisateur en cours.
audioStreamChange	Survient en cas de modification du numéro de flux audio de l'utilisateur en cours pour le titre principal. Lorsque cet événement survient, les informations supplémentaires suivantes sont transmises à `DVDeventNotification` : • *eventArg2* : nombre entier indiquant le nouveau numéro de flux audio de l'utilisateur en cours. Le flux `0xFFFFFFFF` indique qu'aucun flux n'est sélectionné.
buttonChange	Survient en cas de modification du nombre de boutons disponibles ou du numéro de bouton actuellement sélectionné. Lorsque cet événement survient, les informations supplémentaires suivantes sont transmises à `DVDeventNotification` : • *aeventArg2* : nombre entier indiquant le nombre de boutons disponibles. • *eventArg3* : nombre entier indiquant le numéro de bouton actuellement sélectionné. Le numéro de bouton 0 signifie qu'aucun bouton n'est sélectionné.
chapterAutoStop	Survient en cas d'arrêt de la lecture par suite d'un arrêt automatique.
chapterStart	Survient au démarrage de la lecture d'un nouveau programme dans le domaine `title`. Lorsque cet événement survient, les informations supplémentaires suivantes sont transmises à `DVDeventNotification` : • *eventArg2* : nombre entier indiquant le nouveau numéro de chapitre.
diskEjected	Survient en cas d'éjection d'un DVD.
diskInserted	Survient en cas d'insertion d'un DVD.

Evénement(Suite)	**Description(Suite)**
domainChange	Survient en cas de modification du domaine du lecteur de DVD. Lorsque cet événement survient, les informations supplémentaires suivantes sont transmises à `DVDeventNotification` : • *`eventArg1`* : valeur indiquant le nouveau domaine. Le nouveau domaine peut correspondre à l'une des valeurs suivantes : • `firstPlay` : le navigateur de DVD effectue une initialisation par défaut d'un DVD. • `videoManagerMenu` : le navigateur de DVD affiche les menus de la totalité du disque. • `videoTitleSetMenu` : le navigateur de DVD affiche les menus de la série de titres en cours. • `title` : le navigateur de DVD affiche le titre en cours. • `stop` : le navigateur de DVD se trouve dans le domaine `stop`.
error	Survient en cas de détection d'une condition d'erreur de DVD. Lorsque cet événement survient, les informations supplémentaires suivantes sont transmises à `DVDeventNotification` : • *`eventArg2`* : Valeur indiquant une condition d'erreur. La condition d'erreur peut correspondre à l'une des valeurs suivantes : • `copyProtectFail` : L'échange de la clé de protection d'un DVD contre la copie a échoué. La lecture s'est arrêtée. • `invalidDVD_0Disc` : le disque DVD-Video a été conçu de façon incorrecte pour la version de spécification 1.*x*. La lecture s'est arrêtée. • `invalidDiscRegion` : le disque DVD-Video ne peut pas être lu car il n'a pas été conçu pour être lu dans la zone du système. • `lowParentalLevel` : Le niveau de contrôle parental du lecteur est inférieur au niveau de contrôle parental minimal disponible dans le contenu du DVD. La lecture s'est arrêtée. • `macrovisionFail` : La distribution Macrovision a échoué. La lecture s'est arrêtée. • `incompatibleSystemAndDecoderRegions` : aucun disque ne peut être lu car la zone du système ne correspond pas à celle du décodeur. • `incompatibleDiscAndDecoderRegions` : le disque ne peut pas être lu car il n'a pas été conçu pour être lu dans la zone du décodeur. • `unexpected` : un événement imprévu s'est produit ; le contenu a peut-être été conçu de façon incorrecte. La lecture s'est arrêtée.
karaokeMode	Survient lorsque le mode audio est défini sur `karaoke`.
noFirstPlayChain	Survient lorsque le disque DVD ne comporte pas de programme FP_PGC (First Play Program Chain) et que le navigateur de DVD ne procède ni au chargement automatique d'un PGC, ni au démarrage de la lecture.
parentalLevelChange	Survient lorsque le niveau de contrôle parental du contenu créé est sur le point de changer. Lorsque cet événement survient, les informations supplémentaires suivantes sont transmises à `DVDeventNotification` : • `eventArg2` : nombre entier indiquant le nouveau niveau de contrôle parental dans le lecteur.
playbackStopped	Survient en cas d'arrêt de la lecture. Le navigateur de DVD a achevé la lecture du PGC et n'a détecté aucune autre instruction de passage à une autre lecture.

Evénement(Suite)	Description(Suite)
playPeriodAutoStop	Survient en cas d'arrêt de la lecture par suite d'un arrêt automatique.
rateChange	Survient en cas de modification de la cadence de lecture. Lorsque cet événement survient, les informations supplémentaires suivantes sont transmises à `DVDeventNotification` : • `eventArg2` : nombre entier indiquant la nouvelle cadence de lecture. Une valeur inférieure à (<) 0 indique un mode de lecture vers l'arrière. Une valeur supérieure à (>) 0 indique un mode de lecture vers l'avant. Cette valeur constitue la cadence de lecture réelle multipliée par 10 000.
stillOff	Survient à la fin d'une temporisation (PGC, cellule ou VOBU).
stillOn	Survient au début d'une temporisation (PGC, cellule ou VOBU). Lorsque cet événement survient, les informations supplémentaires suivantes sont transmises à `DVDeventNotification` : • `eventArg2` : valeur booléenne indiquant si des boutons sont disponibles. La valeur zéro (0) indique que des boutons sont disponibles. La valeur un (1) indique qu'aucun bouton n'est disponible. • `eventArg3` : nombre entier ou adresse indiquant le nombre de secondes de la temporisation. `0xFFFFFFFF` indique une temporisation infinie.
titleChange	Survient en cas de modification du numéro de titre en cours. Lorsque cet événement survient, les informations supplémentaires suivantes sont transmises à `DVDeventNotification` : • `eventArg2` : nombre entier ou adresse indiquant le nouveau numéro de titre.
UOPchange	Survient en cas de modification de l'un des mécanismes de lecture ou de recherche disponibles. Lorsque cet événement survient, les informations supplémentaires suivantes sont transmises à `DVDeventNotification` : • `eventArg2` : nombre entier ou adresse indiquant les mécanismes de lecture ou de recherche explicitement désactivés par le disque DVD.
warning	Survient en cas de détection d'une condition d'avertissement de DVD. Lorsque cet événement survient, les informations supplémentaires suivantes sont transmises à `DVDeventNotification` : • `eventArg2` : nombre entier ou adresse indiquant la condition d'avertissement. La condition d'avertissement peut correspondre à l'une des valeurs suivantes : • `invalidDVD1_0Disc` : le disque DVD-Video a été conçu de façon incorrecte. La lecture peut se poursuivre, mais un comportement imprévu risque de se produire. • `formatNotSupported` : un décodeur ne prend pas en charge le format en cours. La lecture d'un flux risque de ne pas fonctionner. • `illegalNavCommand` : le processeur de commandes de navigation de DVD interne a tenté de traiter une commande incorrecte. • `open`. • `seek`. • `read`.

Voir aussi

DVD

on endSprite

Syntaxe

```
-- Lingo syntax
on endSprite
    statement(s)
end

// JavaScript syntax
function endSprite() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient du code Lingo exécuté lorsque la tête de lecture sort d'une image-objet et passe à une image dans laquelle l'image-objet n'existe pas. Il est généré après `exitFrame`.

Placez les gestionnaires `on endSprite` dans un script de comportement.

Director détruit les instances des scripts de comportement liés à l'image-objet immédiatement après l'événement `endSprite`.

Le gestionnaire d'événement reçoit la référence de comportement ou de script d'image `me` lorsqu'il est utilisé dans un comportement. Ce message `endSprite` est envoyé après le message `exitFrame` si la tête de lecture se poursuit jusqu'à la fin de l'image.

Les méthodes `go()`, `play()` et `updateStage()` sont désactivées dans un gestionnaire `on endSprite`.

Exemple

Le gestionnaire suivant est exécuté lorsque la tête de lecture quitte une image-objet :

```
-- Lingo syntax
on endSprite me
    -- clean up
    gNumberOfSharks = gNumberOfSharks - 1
    sound(5).stop()
end

// JavaScript syntax
function endSprite() {
    // clean up
    gNumberOfSharks--;
    sound(5).stop();
}
```

Voir aussi

on beginSprite, on exitFrame

on enterFrame

Syntaxe

```
-- Lingo syntax
on enterFrame
    statement(s)
end
```

```
// JavaScript syntax
function enterFrame() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées à chaque fois que la tête de lecture entre dans l'image.

Placez les gestionnaires `on enterFrame` dans un script de comportement, d'image ou d'animation, comme suit :

- Pour affecter le gestionnaire à une seule image-objet, placez-le dans un comportement lié à l'image-objet.
- Pour affecter le gestionnaire à une seule image, placez-le dans le script de l'image.
- Pour affecter le gestionnaire à chaque image (à moins que vous n'indiquiez explicitement le contraire à l'animation), placez le gestionnaire `on enterFrame` dans un script d'animation. Le gestionnaire est exécuté à chaque fois que la tête de lecture entre dans une image à moins que le script d'image n'ait son propre gestionnaire. Si le script d'image comporte son propre gestionnaire, le gestionnaire `on enterFrame` du script d'image prévaut sur le gestionnaire `on enterFrame` du script d'animation.

L'ordre des événements d'image est `stepFrame`, `prepareFrame`, `enterFrame` et `exitFrame`.

Cet événement reçoit la référence d'objet `me` lorsqu'il est utilisé dans un comportement.

Exemple

Le gestionnaire suivant désactive la condition d'asservissement pour les images 1 à 5 à chaque fois que la tête de lecture entre dans l'image :

```
-- Lingo syntax
on enterFrame
    repeat with i = 1 to 5
        _movie.puppetSprite(i, FALSE)
    end repeat
end

// JavaScript syntax
function enterFrame() {
    for (i=1;i<=5;i++) {
        _movie.puppetSprite(i, false);
    }
}
```

on EvalScript

Syntaxe

```
-- Lingo syntax
on EvalScript aParam
    statement(s)
end

// JavaScript syntax
function EvalScript(aParam) {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; dans une animation comportant du contenu Adobe® Shockwave®, contient des instructions exécutées lorsque le gestionnaire reçoit un message `EvalScript` de la part d'un navigateur. Le paramètre est une chaîne reçue du navigateur web.

- Le message `EvalScript` peut inclure une chaîne que Director peut interpréter comme une instruction Lingo. Lingo n'accepte pas de chaînes imbriquées. Si le gestionnaire que vous appelez attend une chaîne comme paramètre, passez le paramètre comme symbole.

- Le gestionnaire `on EvalScript` est appelé par la méthode de programmation JavaScript or VBScript `EvalScript()` dans un navigateur.

Seuls les comportements devant être contrôlés par les utilisateurs doivent être inclus dans `on EvalScript` ; pour des raisons de sécurité, ne donnez pas un accès complet à tous les comportements.

Remarque : *Si vous placez un mot-clé return à la fin de votre gestionnaire EvalScript, la valeur renvoyée peut être utilisée par JavaScript dans le navigateur web.*

Exemple

L'exemple suivant indique comment faire passer la tête de lecture à une image spécifique en fonction de l'image passée comme paramètre :

```
-- Lingo syntax
on EvalScript aParam
    _movie.go(aParam)
end

// JavaScript syntax
function EvalScript(aParam) {
    _movie.go(aParam);
}
```

Le gestionnaire suivant exécute l'instruction `_movie.go(unParamètre)` s'il reçoit un message `EvalScript` contenant l'argument chien, chat ou arbre :

```
-- Lingo syntax
on EvalScript aParam
    case aParam of
        "dog", "cat", "tree": _movie.go(aParam)
    end case
end

// JavaScript syntax
function EvalScript(aParam) {
    switch(aParam) {
        case "dog", "cat", "tree": _movie.go(aParam);
    }
}
```

Une instruction d'appel possible dans JavaScript serait `EvalScript ("dog")`.

Le gestionnaire suivant prend un argument qui pourrait être un nombre ou un symbole :

```
-- Lingo syntax
on EvalScript aParam
    if word 1 of aParam = "myHandler" then
        _movie.go(aParam)
    end if
end
```

```
// JavaScript syntax
function EvalScript(aParam) {
    if (aParam.indexOf("myHandler",0)) {
        _movie.go(aParam);
    }
}
```

Le gestionnaire suivant nécessite normalement une chaîne comme argument. L'argument est reçu sous la forme d'un symbole, puis converti en chaîne dans le gestionnaire par la fonction `string` :

```
-- Lingo syntax
on myHandler aParam
    _movie.go(string(aParam))
end

// JavaScript syntax
function myHandler(aParam) {
    _movie.go(aParam.toString());
}
```

Voir aussi

externalEvent(), return (mot-clé)

on exitFrame

Syntaxe

```
-- Lingo syntax
on exitFrame
    statement(s)
end

// JavaScript syntax
function exitFrame() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées chaque fois que la tête de lecture quitte l'image à laquelle le gestionnaire `on exitFrame` est lié. Le gestionnaire `on exitFrame` est utile pour insérer des instructions Lingo réinitialisant des conditions devenues inutiles après la sortie de l'image.

Placez les gestionnaires `on exitFrame` dans des scripts de comportement, d'image ou d'animation de la façon suivante :

- Pour affecter le gestionnaire à une seule image-objet, placez-le dans un comportement lié à l'image-objet.
- Pour affecter le gestionnaire à une seule image, placez-le dans le script de l'image.
- Pour affecter le gestionnaire à chaque image (à moins que vous n'indiquiez explicitement le contraire), placez le gestionnaire dans un script d'animation. Le gestionnaire `on exitFrame` est exécuté chaque fois que la tête de lecture quitte l'image, à moins que le script d'image ne comporte son propre gestionnaire `on exitFrame`. Si le script d'image comporte son propre gestionnaire `on exitFrame`, ce gestionnaire prévaut sur celui du script d'animation.

Lorsqu'il est utilisé dans un comportement, cet événement reçoit la référence `me` du script d'image-objet ou d'image. L'ordre des événements d'image est `prepareFrame`, `enterFrame` et `exitFrame`.

Exemple

Le gestionnaire suivant désactive toutes les conditions d'asservissement lorsque la tête de lecture quitte l'image :

```
-- Lingo syntax
on exitFrame me
    repeat with i = 48 down to 1
        sprite(i).scripted = FALSE
    end repeat
end

// JavaScript syntax
function exitFrame() {
    for (i=48; i>=1; i--);
        sprite(i).scripted = false;
    }
}
```

Le gestionnaire suivant déplace la tête de lecture vers une image spécifiée si la valeur de la variable globale `vTotal` dépasse 1 000 lorsque la tête de lecture quitte l'image :

```
// JavaScript syntax
function exitFrame() {
    if (_global.vTotal > 1000) {
        _movie.go("Finished");
    }
}
```

Voir aussi

`on enterFrame`

on getBehaviorDescription

Syntaxe

```
-- Lingo syntax
on getBehaviorDescription
    statement(s)
end

// JavaScript syntax
function getBehaviorDescription() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient du code Lingo renvoyant la chaîne qui apparaît dans le volet de description de l'Inspecteur de comportement lorsque le comportement est sélectionné.

La chaîne de description est facultative.

Director envoie le message `getBehaviorDescription` aux comportements liés à une image-objet à l'ouverture de l'Inspecteur de comportement. Placez le gestionnaire `on getBehaviorDescription` dans un comportement.

Le gestionnaire peut contenir des caractères de retour de chariot pour formater les descriptions multilignes.

Exemple

L'instruction suivante affiche Barre de défilement vertical de champ de texte multiligne dans le volet de description :

```
-- Lingo syntax
on getBehaviorDescription
    return "Vertical Multiline textField Scrollbar"
end

// JavaScript syntax
function getBehaviorDescription() {
    return "Vertical Multiline textField Scrollbar";
}
```

Voir aussi

on getPropertyDescriptionList, on getBehaviorTooltip, on runPropertyDialog

on getBehaviorTooltip

Syntaxe

```
-- Lingo syntax
on getBehaviorTooltip
    statement(s)
end

// JavaScript syntax
function getBehaviorTooltip() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions Lingo renvoyant la chaîne apparaissant dans une info-bulle pour un script de la palette des bibliothèques.

Director envoie le message `getBehaviorTooltip` au script lorsque le curseur s'arrête au-dessus de ce dernier dans la Palette des bibliothèques. Placez le gestionnaire `on getBehaviorTooltip` dans le comportement.

L'utilisation du gestionnaire est facultative. Si aucun gestionnaire n'est fourni, le nom de l'acteur apparaît dans l'info-bulle.

Le gestionnaire peut contenir des caractères de retour de chariot pour formater les descriptions multilignes.

Exemple

L'instruction suivante affiche « Pièce de puzzle » dans le volet de description :

```
-- Lingo syntax
on getBehaviorTooltip
    return "Jigsaw puzzle piece"
end

// JavaScript syntax
function getBehaviorTooltip() {
    return "Jigsaw puzzle piece";
}
```

Voir aussi

on getPropertyDescriptionList, on getBehaviorDescription, on runPropertyDialog

on getPropertyDescriptionList

Syntaxe

```
-- Lingo syntax
on getPropertyDescriptionList
    statement(s)
end

// JavaScript syntax
function getPropertyDescriptionList() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient du code Lingo générant une liste de définitions et de libellés pour les paramètres qui apparaissent dans la boîte de dialogue Paramètres d'un comportement.

Placez le gestionnaire `on getPropertyDescriptionList` dans un script de comportement. Les comportements ne contenant pas de gestionnaire `on getPropertyDescriptionList` n'apparaissent pas dans la boîte de dialogue Paramètres et ne sont pas modifiables à partir de l'interface de Director.

Le message `on getPropertyDescriptionList` est envoyé lorsqu'une action entraînant l'ouverture de l'Inspecteur de comportement se produit : soit lorsque l'utilisateur fait glisser un comportement sur le scénario, soit lorsqu'il double-clique sur un comportement dans l'Inspecteur de comportement.

Les paramètres `#default`, `#format` et `#comment` sont obligatoires pour chaque paramètre. Les valeurs possibles de ces paramètres sont :

`#default`	Valeur initiale du paramètre.
`#format`	`#integer #float #string #symbol #member #bitmap #filmloop #field #palette #picture #sound #button #shape #movie #digitalvideo #script #richtext #ole #transition #xtra #frame #marker #ink #boolean`
`#comment`	Chaîne descriptive apparaissant à gauche du champ modifiable du paramètre dans la boîte de dialogue Paramètres.
#range	Fourchette de valeurs possibles pouvant être affectées à une propriété. Cette fourchette est spécifiée sous la forme d'une liste linéaire avec plusieurs valeurs ou comme un minimum et maximum sous la forme d'une liste de propriétés : `[#min : valeurMin, #max : valeurMax]`.

Exemple

Le gestionnaire suivant définit les paramètres d'un comportement apparaissant dans la boîte de dialogue Paramètres. Chaque instruction commençant par `addProp` ajoute un paramètre à la liste appelée Description. Chaque élément ajouté à la liste définit une propriété et ses valeurs `#default`, `#format` et `#comment` :

```
on getPropertyDescriptionList
    description = [:]
    description.addProp(#dynamic, [#default:1, #format:#boolean, #comment:"Dynamic"])
    description.addProp(#fieldNum, [#default:1, #format:#integer,#comment:"Scroll which
sprite:"])
    description.addProp(#extentSprite, [#default:1,#format:#integer,#comment: "Extend
Sprite:"])
    description.addProp(#proportional, [#default:1,#format:#boolean, #comment:
"Proportional:"])
    return description
end
```

Voir aussi

addProp, on getBehaviorDescription, on runPropertyDialog

on hyperlinkClicked

Syntaxe

```
-- Lingo syntax
on hyperlinkClicked me, data, range
    statement(s)
end

// JavaScript syntax
function hyperlinkClicked(data, range) {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; utilisé pour déterminer si l'utilisateur a cliqué sur un lien hypertexte.

Ce gestionnaire d'événement comprend les paramètres suivants :

- `me` Utilisé dans un comportement pour identifier l'instance d'image-objet.
- `data` Données proprement dites du lien hypertexte ; chaîne saisie dans l'Inspecteur de texte lors de la modification de l'acteur texte.
- `range` Plage de caractères du lien hypertexte dans le texte (il est possible d'obtenir le texte de la plage proprement dite en utilisant la syntaxe réfActeur.char[plage[1]..plage[2]).

Ce gestionnaire doit être attaché à une image-objet en tant que script de comportement. Evitez de placer ce gestionnaire dans un script d'acteur.

Exemple

Le comportement suivant affiche un lien utilisé pour examiner le lien hypertexte sélectionné, passer à une URL si nécessaire, puis afficher le texte du lien dans la fenêtre Messages :

```
property spriteNum
on hyperlinkClicked(me, data, range)
    if data starts "http://" then
        gotoNetPage(data)
    end if
    currentMember = sprite(spriteNum).member
    anchorString = currentMember.char[range[1]..range[2]]
    put("The hyperlink on"&&anchorString&&"was just clicked.")
end
// JavaScript syntax
function hyperlinkClicked(data, range) {
    var st = data.slice(0,7);
    var ht = "http://";
    if (st = ht) {
        gotoNetPage(data);
    }
    var currentMember = sprite(this.spriteNum).member;
    var r1 = currentMember.getPropRef("char", range[1]).hyperlinkRange;
    var a = r1[1] - 1;
    var b = r1[2];
    var st = new String(currentMember.text);
```

```
    var anchorString = st.slice(a, b);
    put("The hyperlink on " + anchorString + " was just clicked.");
}
```

on idle

Syntaxe

```
-- Lingo syntax
on idle
    statement(s)
end

// JavaScript syntax
function idle() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées lorsque l'animation n'a pas d'autres événements à traiter et constitue un endroit pratique pour insérer les instructions Lingo à exécuter aussi fréquemment que possible, comme celles qui mettent à jour les valeurs des variables globales et affichent les conditions de l'animation actuelle.

Les instructions des gestionnaires `on idle` étant fréquemment exécutées, il est recommandé de ne pas insérer de code Lingo dont le traitement exige trop de temps dans un gestionnaire `on idle`.

Il est souvent préférable d'insérer les gestionnaires `on idle` dans des scripts d'image plutôt que dans des scripts d'animation pour tirer parti du gestionnaire `on idle` uniquement lorsqu'il y a lieu.

Director peut charger des acteurs à partir d'une distribution interne ou externe pendant un événement `idle`. Toutefois, il ne peut pas charger des acteurs liés lors d'un événement `idle`.

Le message `idle` est uniquement envoyé aux scripts d'image et d'animation.

Exemple

Le gestionnaire suivant met à jour l'heure affichée dans l'animation lorsqu'il n'y a aucun autre événement à traiter :

```
-- Lingo syntax
on idle
    member("Time").text = _system.time()
end idle

// JavaScript syntax
function idle() {
    member("Time").text = _system.time();
}
```

Voir aussi

`idleHandlerPeriod`

on isOKToAttach

Syntaxe

```
-- Lingo syntax
on isOKToAttach me, aSpriteType, aSpriteNum
    statement(s)
end

// JavaScript syntax
function isOKToAttach(aSpriteType, aSpriteNum) {
    statement(s)
}
```

Description

Gestionnaire intégré ; vous pouvez ajouter ce gestionnaire à un comportement, en vue de vérifier le type d'image-objet auquel le comportement est associé et empêcher que ce comportement soit attaché à des types d'images-objets non appropriés.

Lorsque le comportement est associé à une image-objet, le gestionnaire est exécuté et Director lui transmet le type et le numéro de l'image-objet. L'argument `me` contient une référence au comportement associé à l'image-objet.

Ce gestionnaire est exécuté avant le gestionnaire `on getPropertyDescriptionList`.

L'auteur Lingo peut vérifier deux types d'images-objets. `#graphic` inclut tous les acteurs graphiques tels que les formes, les bitmaps, les vidéos numériques, le texte, etc. `#script` indique le comportement associé à la piste des scripts. Dans ce cas, `spriteNum` est 1.

Pour chacun de ces types d'images-objets, le gestionnaire doit renvoyer la valeur `TRUE` ou `FALSE`. La valeur `TRUE` indique que le comportement peut être associé à l'image-objet. La valeur `FALSE` empêche l'association du comportement à l'image-objet.

Si le comportement ne contient pas de gestionnaire `on isOKToAttach`, le comportement peut être associé à n'importe quelle image ou image-objet.

Ce gestionnaire est appelé lors de l'association initiale du comportement à l'image-objet ou à la piste des scripts et lors de l'association d'un nouveau comportement à une image-objet à l'aide de l'Inspecteur de comportement.

Exemple

L'instruction suivante vérifie le type d'image-objet à laquelle le comportement est associé et renvoie la valeur `TRUE` pour toutes les images-objets graphiques à l'exception des formes et la valeur `FALSE` pour la piste des scripts :

```
-- Lingo syntax
on isOKToAttach me, aSpriteType, aSpriteNum
    case aSpriteType of
        #graphic: -- any graphic sprite type
            return sprite(aSpriteNum).member.type <> #shape
            -- works for everything but shape cast members
        #script: --the frame script channel
            return FALSE -- doesn't work as a frame script
    end case
end

// JavaScript syntax
function isOKToAttach(aSpriteType, aSpriteNum) {
    switch (aSpriteType) {
        case symbol("graphic"): // any graphic sprite type
            return sprite(aSpriteNum).member.type != symbol("shape");
```

```
            // works for everything but shape cast members
        case symbol("script"): // the frame script channel
            return false; // doesn't work as a frame script
    }
}
```

on keyDown

Syntaxe

```
-- Lingo syntax
on keyDown
    statement(s)
end

// JavaScript syntax
function keyDown() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient les instructions exécutées lorsqu'une touche est enfoncée.

Lorsqu'une touche est enfoncée, Director recherche un gestionnaire `on keyDown` aux emplacements suivants dans cet ordre : gestionnaire d'événement principal, script d'image-objet champ modifiable, script d'acteur champ, script d'image et script d'animation. Pour les images-objets et les acteurs, les gestionnaires `on keyDown` ne fonctionnent qu'avec les acteurs texte et champ modifiables. Un événement `keyDown` n'a aucun effet sur les autres types d'acteurs, tels que les bitmaps. Si le fait d'appuyer sur une touche doit produire un résultat identique dans toute l'animation, définissez `keyDownScript`.

Director arrête la recherche dès qu'il rencontre le premier gestionnaire `on keyDown`, sauf si celui-ci contient une commande `pass` exigeant la transmission du message `keyDown` à l'emplacement suivant.

Le gestionnaire d'événement `on keyDown` est idéal pour placer des instructions Lingo créant des raccourcis clavier ou d'autres fonctions d'interface, en fonction des actions que vous souhaitez déclencher lorsque l'utilisateur appuie sur une touche du clavier.

Lorsque l'animation est lue sous forme d'applet, un gestionnaire `on keyDown` intercepte toujours les touches enfoncées, même s'il est vide. Lorsque l'utilisateur saisit du texte dans un champ modifiable, le gestionnaire `on keyDown` associé à ce champ doit inclure la commande `pass` pour que les caractères saisis apparaissent dans le champ.

L'endroit où vous placez un gestionnaire `on keyDown` peut affecter le moment de son exécution.

- Pour appliquer le gestionnaire à une image-objet champ modifiable particulière, placez-le dans un script d'image-objet.
- Pour appliquer le gestionnaire à un acteur champ modifiable quelconque, placez-le dans un script d'acteur.
- Pour appliquer le gestionnaire à une image entière, placez-le dans un script d'image.
- Pour appliquer le gestionnaire à l'animation entière, placez-le dans un script d'animation.

Vous pouvez annuler l'effet d'un gestionnaire `on keyDown` en plaçant un autre gestionnaire `on keyDown` dans un emplacement qui est vérifié par Lingo avant celui du gestionnaire à remplacer. Par exemple, vous pouvez outrepasser un gestionnaire `on keyDown` affecté à un acteur en plaçant un gestionnaire `on keyDown` dans un script d'image-objet.

Exemple

Le gestionnaire suivant vérifie si la touche Retour a été enfoncée et, le cas échéant, fait passer la tête de lecture à une autre image :

```
-- Lingo syntax
on keyDown
    if (_key.key = RETURN) then _movie.go("AddSum")
end keyDown

// JavaScript syntax
function keyDown() {
    if (_key.keyCode == 36) {
        _movie.go("AddSum");
    }
}
```

Voir aussi

charToNum(), keyDownScript, keyUpScript, key, keyCode, keyPressed()

on keyUp

Syntaxe

```
-- Lingo syntax
on keyUp
    statement(s)
end

// JavaScript syntax
function keyUp() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées lorsque l'utilisateur relâche une touche. Le gestionnaire `on keyUp` est semblable au gestionnaire `on keyDown`, à ceci près que l'événement se produit après l'apparition d'un caractère si une image-objet champ ou texte est modifiable à l'écran.

Lorsque l'utilisateur relâche une touche, Lingo recherche un gestionnaire `on keyUp` aux emplacements suivants dans cet ordre : gestionnaire d'événement principal, script d'image-objet champ modifiable, script d'acteur champ, script d'image et script d'animation. Pour les images-objets et les acteurs, les gestionnaires `on keyUp` ne fonctionnent qu'avec les chaînes modifiables. Un événement `keyUp` n'a aucun effet sur les autres types d'acteurs, tels que les bitmaps. Si le fait de relâcher une touche doit produire un résultat identique dans toute l'animation, définissez `keyUpScript`.

Lingo arrête la recherche dès qu'il rencontre le premier gestionnaire `on keyUp`, sauf si celui-ci contient une commande `pass` exigeant la transmission du message `keyUp` à l'emplacement suivant.

Le gestionnaire d'événement `on keyUp` constitue un emplacement adéquat pour insérer du code Lingo créant des raccourcis clavier ou d'autres fonctions d'interface en fonction des actions que vous souhaitez déclencher lorsque l'utilisateur relâche une touche du clavier.

Lorsque l'animation est lue sous forme d'applet, un gestionnaire `on keyUp` intercepte toujours les touches enfoncées, même s'il est vide. Lorsque l'utilisateur saisit du texte dans un champ modifiable, le gestionnaire `on keyUp` associé à ce champ doit inclure la commande `pass` pour que les caractères saisis apparaissent dans le champ.

L'endroit où vous placez un gestionnaire `on keyUp` peut affecter le moment de son exécution de la façon suivante :

- Pour appliquer le gestionnaire à une image-objet champ modifiable spécifique, placez-le dans un comportement.
- Pour appliquer le gestionnaire à un acteur champ modifiable quelconque, placez-le dans un script d'acteur.
- Pour appliquer le gestionnaire à une image entière, placez-le dans un script d'image.
- Pour appliquer le gestionnaire à l'animation entière, placez-le dans un script d'animation.

Vous pouvez annuler l'effet d'un gestionnaire `on keyUp` en plaçant un autre gestionnaire `on keyUp` à un emplacement qui est vérifié par Lingo avant celui du gestionnaire à remplacer. Par exemple, vous pouvez outrepasser le gestionnaire `on keyUp` affecté à un acteur en plaçant un gestionnaire `on keyUp` dans un script d'image-objet.

Exemple

Le gestionnaire suivant vérifie si la touche Retour a été relâchée et, le cas échéant, fait passer la tête de lecture à une autre image :

```
-- Lingo syntax
on keyUp
    if (_key.key = RETURN) then _movie.go("AddSum")
end keyUp

// JavaScript syntax
function keyUp() {
    if (_key.keyCode == 36) {
        _movie.go("AddSum");
    }
}
```

Voir aussi

on keyDown, keyDownScript, keyUpScript

on mouseDown (gestionnaire d'événement)

Syntaxe

```
-- Lingo syntax
on mouseDown
    statement(s)
end

// JavaScript syntax
function mouseDown() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées lorsque l'utilisateur appuie sur le bouton de la souris.

Lorsque le bouton de la souris est enfoncé, Lingo recherche un gestionnaire `on mouseDown` aux emplacements suivants dans cet ordre : gestionnaire d'événement principal, script d'image-objet, script d'acteur, script d'image et script d'animation. Lingo arrête la recherche dès qu'il rencontre le premier gestionnaire `on mouseDown`, sauf si celui-ci contient une commande `pass` exigeant la transmission du message `mouseDown` à l'emplacement suivant.

Pour que l'animation réponde toujours de la même manière lorsque le bouton de la souris est enfoncé, définissez `mouseDownScript` ou placez un gestionnaire mouseDown dans un script d'animation.

Le gestionnaire d'événement `on mouseDown` constitue un emplacement adéquat pour insérer du code Lingo faisant clignoter des images, déclenchant des effets sonores ou entraînant le déplacement d'images-objets lorsque l'utilisateur appuie sur le bouton de la souris.

L'endroit où vous placez un gestionnaire `on mouseDown` peut affecter le moment de son exécution.

- Pour appliquer le gestionnaire à une image-objet spécifique, placez-le dans un script d'image-objet.
- Pour appliquer le gestionnaire à un acteur quelconque, placez-le dans un script d'acteur.
- Pour appliquer le gestionnaire à une image entière, placez-le dans un script d'image.
- Pour appliquer le gestionnaire à l'animation entière, placez-le dans un script d'animation.

Vous pouvez annuler l'effet d'un gestionnaire `on mouseDown` en plaçant un autre gestionnaire `on mouseDown` dans un emplacement qui est vérifié par Lingo avant celui du gestionnaire à remplacer. Par exemple, vous pouvez outrepasser un gestionnaire `on mouseDown` affecté à un acteur en plaçant un gestionnaire `on mouseDown` dans un script d'image-objet.

Lorsqu'il est utilisé dans un comportement, cet événement reçoit la référence `me` du script d'image-objet ou d'image.

Exemple

Le gestionnaire suivant vérifie si l'utilisateur clique à un endroit quelconque de la scène et, le cas échéant, fait passer la tête de lecture à une autre image :

```
-- Lingo syntax
on mouseDown
    if (_mouse.clickOn = 0) then _movie.go("AddSum")
end

// JavaScript syntax
function mouseDown() {
    if (_mouse.clickOn == 0) {
        _movie.go("AddSum");
    }
}
```

Le gestionnaire suivant, affecté à un script d'image-objet, lit un son lorsque l'utilisateur clique sur l'image-objet :

```
-- Lingo syntax
on mouseDown
    sound(1).play(member("Crickets"))
end

// JavaScript syntax
function mouseDown() {
    sound(1).play(member("Crickets"));
}
```

Voir aussi

clickOn, mouseDownScript, mouseUpScript

on mouseEnter

Syntaxe

```
-- Lingo syntax
on mouseEnter
    statement(s)
end

// JavaScript syntax
function mouseEnter() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient les instructions exécutées lorsque le pointeur de la souris entre en contact avec la zone référencée de l'image-objet. Il n'est pas nécessaire que le bouton de la souris soit enfoncé.

Si l'image-objet est un acteur bitmap dont l'encre est Dessin seul, la zone active correspond à la partie de l'image affichée. Dans tous les autres cas, elle correspond au rectangle de délimitation de l'image-objet.

Lorsqu'il est utilisé dans un comportement, cet événement reçoit la référence `me` du script d'image-objet ou d'image.

Exemple

L'instruction suivante est un comportement de bouton simple qui change l'image bitmap du bouton lorsque la souris survole, puis sort du bouton :

```
-- Lingo syntax
property spriteNum

on mouseEnter me
    -- Determine current cast member and switch to next in cast
    currentMember = sprite(spriteNum).member.number
    sprite(spriteNum).member = currentMember + 1
end

on mouseLeave me
    -- Determine current cast member and switch to previous in cast
    currentMember = sprite(spriteNum).member.number
    sprite(spriteNum).member = currentMember - 1
end

// JavaScript syntax
var spriteNum;

function mouseEnter() {
    // Determine current cast member and switch to next in cast
    currentMember = sprite(spriteNum).member.number;
    sprite(spriteNum).member = currentMember + 1;
}

function mouseLeave() {
    // Determine current cast member and switch to previous in cast
    currentMember = sprite(spriteNum).member.number;
    sprite(spriteNum).member = currentMember - 1;
}
```

Voir aussi

on mouseLeave, on mouseWithin

on mouseLeave

Syntaxe

```
-- Lingo syntax
on mouseLeave
    statement(s)
end

// JavaScript syntax
function mouseLeave() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient les instructions exécutées lorsque la souris sort de la zone référencée de l'image-objet. Il n'est pas nécessaire que le bouton de la souris soit enfoncé.

Si l'image-objet est un acteur bitmap dont l'encre est Dessin seul, la zone active correspond à la partie de l'image affichée. Dans tous les autres cas, elle correspond au rectangle de délimitation de l'image-objet.

Lorsqu'il est utilisé dans un comportement, cet événement reçoit la référence `me` du script d'image-objet ou d'image.

Exemple

L'instruction suivante est un comportement de bouton simple qui change l'image bitmap du bouton lorsque la souris survole le bouton, puis en sort :

```
-- Lingo syntax
property spriteNum

on mouseEnter me
    -- Determine current cast member and switch to next in cast
    currentMember = sprite(spriteNum).member.number
    sprite(spriteNum).member = currentMember + 1
end

on mouseLeave me
    -- Determine current cast member and switch to previous in cast
    currentMember = sprite(spriteNum).member.number
    sprite(spriteNum).member = currentMember - 1
end

// JavaScript syntax
var spriteNum;

function mouseEnter() {
    // Determine current cast member and switch to next in cast
    currentMember = sprite(spriteNum).member.number;
    sprite(spriteNum).member = currentMember + 1;
}

function mouseLeave() {
    // Determine current cast member and switch to previous in cast
    currentMember = sprite(spriteNum).member.number;
    sprite(spriteNum).member = currentMember - 1;
}
```

Voir aussi

on mouseEnter, on mouseWithin

on mouseUp (gestionnaire d'événement)

Syntaxe

```
-- Lingo syntax
on mouseUp
    statement(s)
end

// JavaScript syntax
function mouseUp() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient les instructions exécutées lorsque le bouton de la souris est relâché.

Lorsque l'utilisateur relâche le bouton de la souris, Lingo recherche un gestionnaire `on mouseUp` aux emplacements suivants dans cet ordre : gestionnaire d'événement principal, script d'image-objet, script d'acteur, script d'image et script d'animation. Lingo arrête la recherche dès qu'il rencontre le premier gestionnaire `on mouseUp`, sauf si celui-ci contient une commande `pass` exigeant la transmission du message `mouseUp` à l'emplacement suivant.

Pour obtenir la même réponse dans toute l'animation lorsque l'utilisateur relâche le bouton de la souris, définissez `mouseUpScript`.

Le gestionnaire d'événement `on mouseUp` constitue un emplacement adéquat pour insérer du code Lingo modifiant l'aspect des objets, tels que des boutons, une fois que l'utilisateur a cliqué dessus. Pour ce faire, il suffit de changer l'acteur de l'image-objet sur laquelle l'utilisateur a cliqué, dès qu'il relâche le bouton de la souris.

L'endroit où vous placez un gestionnaire `on mouseUp` peut affecter le moment de son exécution de la manière suivante :

- Pour appliquer le gestionnaire à une image-objet spécifique, placez-le dans un script d'image-objet.
- Pour appliquer le gestionnaire à un acteur quelconque, placez-le dans un script d'acteur.
- Pour appliquer le gestionnaire à une image entière, placez-le dans un script d'image.
- Pour appliquer le gestionnaire à l'animation entière, placez-le dans un script d'animation.

Vous pouvez annuler l'effet d'un gestionnaire `on mouseUp` en plaçant un autre gestionnaire `on mouseUp` dans un emplacement qui est vérifié par Lingo avant celui du gestionnaire à annuler. Par exemple, vous pouvez outrepasser un gestionnaire `on mouseUp` affecté à un acteur en plaçant un gestionnaire `on mouseUp` dans un script d'image-objet.

Lorsqu'il est utilisé dans un comportement, cet événement reçoit la référence `me` du script d'image-objet ou d'image.

Exemple

Le gestionnaire suivant, affecté à l'image-objet 10, change l'acteur associé à cette image-objet à chaque fois que l'utilisateur relâche le bouton de la souris après avoir cliqué sur l'image-objet :

```
-- Lingo syntax
on mouseUp
    sprite(10).member = member("Dimmed")
end

// JavaScript syntax
function mouseUp() {
```

```
    sprite(10).member = member("Dimmed");
}
```

Voir aussi

on mouseDown (gestionnaire d'événement)

on mouseUpOutside

Syntaxe

```
-- Lingo syntax
on mouseUpOutside me
    statement(s)
end

// JavaScript syntax
function mouseUpOutside() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; envoyé lorsque l'utilisateur clique sur une image-objet, puis relâche le bouton de la souris en dehors de cette dernière.

Exemple

L'instruction suivante lit un son lorsque l'utilisateur clique sur une image-objet, puis relâche le bouton de la souris en dehors du rectangle de délimitation de cette dernière :

```
-- Lingo syntax
on mouseUpOutside me
    sound(1).play(member("Professor Long Hair"))
end

// JavaScript syntax
function mouseUpOutside() {
    sound(1).play(member("Professor Long Hair"));
}
```

Voir aussi

on mouseEnter, on mouseLeave, on mouseWithin

on mouseWithin

Syntaxe

```
-- Lingo syntax
on mouseWithin
    statement(s)
end

// JavaScript syntax
function mouseWithin() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient les instructions exécutées lorsque la souris est dans la zone référencée de l'image-objet. Il n'est pas nécessaire que le bouton de la souris soit enfoncé.

Si l'image-objet est un acteur bitmap dont l'encre est Dessin seul, la zone active correspond à la partie de l'image affichée. Dans tous les autres cas, elle correspond au rectangle de délimitation de l'image-objet.

Lorsqu'il est utilisé dans un comportement, cet événement reçoit la référence `me` du script d'image-objet ou d'image.

Exemple

L'instruction suivante affiche la position de la souris lorsqu'elle se trouve au-dessus d'une image-objet :

```
-- Lingo syntax
on mouseWithin
    member("Display").text = string(_mouse.mouseH)
end

// JavaScript syntax
function mouseWithin() {
    member("Display").text = _mouse.mouseH.toString();
}
```

Voir aussi

on mouseEnter, on mouseLeave

on moveWindow

Syntaxe

```
-- Lingo syntax
on moveWindow
    statement(s)
end

// JavaScript syntax
function moveWindow() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient les instructions exécutées lorsqu'une fenêtre est déplacée, par exemple lorsque l'utilisateur fait glisser une animation vers un nouvel emplacement de la scène. Ce gestionnaire constitue un endroit adéquat pour insérer le code Lingo qui doit être exécuté chaque fois que la fenêtre d'une animation est déplacée.

Exemple

Le gestionnaire suivant affiche un message dans la fenêtre Messages lorsque la fenêtre d'une animation en cours de lecture est déplacée :

```
-- Lingo syntax
on moveWindow
    put("Just moved window containing" && _movie.name)
end
```

```
// JavaScript syntax
function moveWindow() {
    put("Just moved window containing " + _movie.name);
}
```

Voir aussi

activeWindow, name (3D), windowList

on openWindow

Syntaxe

```
-- Lingo syntax
on openWindow
    statement(s)
end

// JavaScript syntax
function openWindow() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événements ; contient des instructions exécutées lorsque Director ouvre l'animation en tant qu'animation dans une fenêtre. C'est aussi un endroit idéal pour insérer des instructions Lingo à exécuter à chaque fois que l'animation s'ouvre dans une fenêtre.

Exemple

Le gestionnaire suivant exécute le fichier audio Hourra lorsque la fenêtre de l'animation s'ouvre :

```
-- Lingo syntax
on openWindow
    sound(2).play(member("Hurray"))
end

// JavaScript syntax
function openWindow() {
    sound(2).play(member("Hurray"));
}
```

on prepareFrame

Syntaxe

```
-- Lingo syntax
on prepareFrame
    statement(s)
end

// JavaScript syntax
function prepareFrame {
    statement(s);
}
```

Description

Message système et gestionnaire d'événements ; contient des instructions exécutées immédiatement avant le dessin de l'image actuelle.

Contrairement aux événements `beginSprite` et `endSprite`, un événement `prepareFrame` est généré chaque fois que la tête de lecture arrive sur une image.

Le gestionnaire `on prepareFrame` constitue un emplacement adéquat pour changer les propriétés de l'image-objet avant que celle-ci ne soit dessinée.

S'il est utilisé dans un comportement, le gestionnaire `on prepareFrame` reçoit la référence `me`.

Les commandes `go`, `play` et `updateStage` sont désactivées dans un gestionnaire `on prepareFrame`.

Exemple

Le gestionnaire suivant définit la propriété `locH` de l'image-objet à laquelle le comportement est associé :

```
-- Lingo syntax
on prepareFrame me
    sprite(me.spriteNum).locH= _mouse.mouseH
end

// JavaScript syntax
function prepareFrame() {
    sprite(spriteNum).locH= _mouse.mouseH;
}
```

Voir aussi

`on enterFrame`

on prepareMovie

Syntaxe

```
-- Lingo syntax
on prepareMovie
    statement(s)
end

// JavaScript syntax
function prepareMovie() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événements ; contient des instructions exécutées après que l'animation précharge les acteurs, mais avant qu'elle ne :

- crée des instances de comportements liées aux images-objets de la première image lue ;
- prépare la première image lue, y compris son dessin, la lecture des sons et l'exécution des transitions et des effets de palette.

Les nouvelles variables globales utilisées pour le comportement des images-objets de la première image doivent être initialisées dans le gestionnaire `on prepareMovie`. Il est inutile de redéfinir les variables globales déjà définies par l'animation précédente.

Un gestionnaire `on prepareMovie` constitue un emplacement adéquat pour insérer du code Lingo permettant de créer des variables globales, d'initialiser des variables, de lire un son pendant que le reste de l'animation se charge en mémoire ou de vérifier et d'ajuster les paramètres de l'ordinateur tels que le codage des couleurs.

Les commandes `go`, `play` et `updateStage` sont désactivées dans un gestionnaire `on prepareMovie`.

Exemple

Le gestionnaire suivant crée une variable globale lorsque l'animation débute :

```
-- Lingo syntax
on prepareMovie
    global currentScore
    currentScore = 0
end

// JavaScript syntax
function prepareMovie() {
    _global.currentScore = 0;
}
```

Voir aussi

`on enterFrame`, `on startMovie`

on resizeWindow

Syntaxe

```
-- Lingo syntax
on resizeWindow
    statement(s)
end

// JavaScript syntax
function resizeWindow() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient les instructions exécutées lorsqu'une animation est lue dans une fenêtre et que l'utilisateur redimensionne cette fenêtre en faisant glisser sa case de redimensionnement ou l'un de ses bords.

Un gestionnaire d'événement `on resizeWindow` constitue un emplacement adéquat pour insérer du code Lingo relatif aux dimensions de la fenêtre, entraînant par exemple le positionnement des images-objets ou le recadrement d'une vidéo numérique.

Exemple

Le gestionnaire suivant déplace l'image-objet 3 vers les coordonnées stockées dans la variable `emplacementCentral` lorsque la fenêtre lue par l'animation est redimensionnée :

```
-- Lingo syntax
on resizeWindow centerPlace
    sprite(3).loc = centerPlace
end
```

```
// JavaScript syntax
function resizeWindow(centerPlace) {
    sprite(3).loc = centerPlace;
}
```

Voir aussi

drawRect, sourceRect

on rightMouseDown (gestionnaire d'événement)

Syntaxe

```
-- Lingo syntax
on rightMousedown
    statement(s)
end

// JavaScript syntax
function rightMouseDown() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événements ; sous Windows, spécifie les instructions exécutées lorsque l'utilisateur appuie sur le bouton droit de la souris. Sur les ordinateurs Mac, les instructions sont exécutées lorsque l'utilisateur appuie simultanément sur le bouton de la souris et sur la touche Ctrl, et que la propriété `emulateMultiButtonMouse` présente la valeur `TRUE` ; si cette propriété reçoit la valeur `FALSE`, ce gestionnaire d'événement n'a aucun effet sur le Mac.

Exemple

Le gestionnaire suivant ouvre la fenêtre Aide lorsque l'utilisateur appuie sur le bouton droit de la souris sous Windows :

```
-- Lingo syntax
on rightMousedown
    window("Help").open()
end

// JavaScript syntax
function rightMouseDown() {
    window("Help").open();
}
```

on rightMouseUp (gestionnaire d'événement)

Syntaxe

```
-- Lingo syntax
on rightMouseUp
    statement(s)
end

// JavaScript syntax
function rightMouseUp() {
```

```
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; sous Windows, spécifie les instructions exécutées lorsque l'utilisateur relâche le bouton droit de la souris. Sur les ordinateurs Mac, les instructions sont exécutées lorsque l'utilisateur relâche le bouton de la souris tout en appuyant sur la touche Ctrl, et que la propriété `emulateMultiButtonMouse` présente la valeur `TRUE` ; si cette propriété reçoit la valeur `FALSE`, ce gestionnaire d'événement n'a aucun effet sur le Mac.

Exemple

Le gestionnaire suivant ouvre la fenêtre Aide lorsque l'utilisateur relâche le bouton droit de la souris sous Windows :

```
-- Lingo syntax
on rightMouseUp
    window("Help").open()
end

// JavaScript syntax
function rightMouseUp() {
    window("Help").open();
}
```

on runPropertyDialog

Syntaxe

```
-- Lingo syntax
on runPropertyDialog me, currentInitializerList
    statement(s)
end

// JavaScript syntax
function runPropertyDialog(currentInitializerList) {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient un Lingo définissant des valeurs spécifiques pour les paramètres d'un comportement dans la boîte de dialogue Paramètres. Le message `runPropertyDialog` est envoyé chaque fois que le comportement est associé à une image-objet ou que l'utilisateur modifie les valeurs initiales de la propriété du comportement d'une image-objet.

Les paramètres actuels des propriétés initiales d'un comportement sont passés au gestionnaire sous forme de liste de propriétés. Si le gestionnaire `on runPropertyDialog` n'est pas défini dans le comportement, Director affiche une boîte de dialogue de personnalisation de comportement reposant sur la liste de propriétés renvoyée par le gestionnaire `on getPropertyDescriptionList`.

Exemple

Le gestionnaire suivant annule les valeurs de comportement définies dans la boîte de dialogue Paramètres du comportement. Les nouvelles valeurs sont répertoriées dans la liste `currentInitializerList`. Normalement, la boîte de dialogue Paramètres permet à l'utilisateur de définir les constantes de masse et de gravité. Cependant, le gestionnaire suivant affecte ces valeurs constantes de paramètres sans afficher de boîte de dialogue :

```
-- Lingo syntax
property mass
property gravitationalConstant

on runPropertyDialog me, currentInitializerList
    --force mass to 10
    currentInitializerList.setaProp(#mass, 10)
    -- force gravitationalConstant to 9.8
    currentInitializerList.setaProp(#gravitationalConstant, 9.8)
    return currentInitializerList
end

// JavaScript syntax
function runPropertyDialog(currentInitializerList) {
    //force mass to 10
    currentInitializerList.setaProp("mass", 10)
    //force gravitationalConstant to 9.8
    currentInitializerList.setaProp("gravitationalConstant", 9.8)
    return(currentInitializerList)
}
```

Voir aussi

on getBehaviorDescription, on getPropertyDescriptionList

on savedLocal

Syntaxe

```
-- Lingo syntax
on savedLocal
    statement(s)
end

// JavaScript syntax
function savedLocal() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; cette propriété permettra d'effectuer des améliorations dans les futures versions de Shockwave Player.

Voir aussi

allowSaveLocal

on sendXML

Syntaxe

```
-- Lingo syntax
on sendXML "sendxmlstring", "window", "postdata"
    statement(s)
end
```

```
// JavaScript syntax
function sendXML(sendxmlstring, window, postdata) {
    statement(s);
}
```

Description

Gestionnaire d'événement ; fonctionne de façon comparable à la méthode de programmation `getURL` également disponible avec l'Xtra Adobe® Flash® Asset. Le gestionnaire `on sendXML` est appelé dans Lingo lorsque la méthode ActionScript *`objetXML`*`.send` est exécutée dans une image-objet Flash ou dans un objet Flash XML.

Dans ActionScript, la méthode *`objetXML`*`.send` transmet deux paramètres, en plus des données XML, dans l'objet XML. Ces paramètres sont les suivants :

- *`url`* URL à laquelle envoyer les données XML. En règle générale, c'est l'URL d'un script serveur qui attend de traiter les données XML.
- *`window`* Fenêtre de navigateur dans laquelle afficher la réponse du serveur.

La méthode ActionScript *`objetXML`*`.send` peut être appelée dans Director, soit par une image-objet Flash, soit par un objet Flash XML global créé dans Lingo. Dans ce cas, le gestionnaire Lingo `on sendXML` est appelé et les mêmes paramètres sont transmis au gestionnaire.

L'instruction Lingo suivante illustre le mode de réception des paramètres par le gestionnaire `on sendXML` :

```
on sendXML me, theURL, targetWindow, XMLdata
```

Ces paramètres sont en corrélation avec le paramètre *`objetXML`*`.send` de la façon suivante :

- `theURL` URL à laquelle envoyer les données XML.
- `targetWindow` Fenêtre de navigateur dans laquelle afficher la réponse du serveur.
- `XMLdata` Données XML de l'objet Flash XML.

En créant un gestionnaire `on sendXML` dans votre animation Director, vous lui permettez de traiter les événements *`objetXML`*`.send` générés dans une image-objet Flash ou dans un objet Flash global.

Les images-objets Flash peuvent également charger des données XML externes ou analyser des données XML internes. L'Xtra Flash Asset gère ces fonctions de la même manière qu'un contenu Flash 5 ou Flash MX dans votre navigateur.

Exemple

La commande Lingo suivante obtient les informations de méthode *`objetXML`*`.send` d'une image-objet Flash, redirige le navigateur vers l'URL, puis transmet les données XML à l'URL :

```
-- Lingo syntax
on sendXML me, theURL, targetWindow, xmlData
    gotoNetPage(theURL, targetWindow)
    postNetText(theURL, xmlData)
end

// JavaScript syntax
function sendXML(theURL, targetWindow, xmlData) {
    gotoNetPage(theURL, targetWindow);
    postNetText(theURL, xmlData);
}
```

on startMovie

Syntaxe

```
-- Lingo syntax
on startMovie
    statement(s)
end

// JavaScript syntax
function startMovie() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées avant que la tête de lecture n'atteigne la première image de l'animation. L'événement `startMovie` se produit après l'événement `prepareFrame` et avant l'événement `enterFrame`.

Un gestionnaire `on startMovie` constitue un emplacement adéquat pour insérer du code Lingo initialisant les images-objets de la première image de l'animation.

Exemple

Le gestionnaire suivant rend les images-objets invisibles au démarrage de l'animation :

```
-- Lingo syntax
on startMovie
    repeat with counter = 10 to 50
        sprite(counter).visible = 0
    end repeat
end startMovie

// JavaScript syntax
function startMovie() {
    for(counter=10;counter<=50;counter++) {
        sprite(counter).visible = 0;
    }
}
```

Voir aussi

on prepareMovie

on stepFrame

Syntaxe

```
-- Lingo syntax
on stepFrame
    statement(s)
end

// JavaScript syntax
function stepFrame() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; fonctionne dans les scripts présents dans la liste `actorList` puisque ce sont les seuls à recevoir des messages `on stepFrame`. Ce gestionnaire d'événement est exécuté lorsque la tête de lecture arrive sur une image ou que la scène est mise à jour.

Un gestionnaire `on stepFrame` constitue un endroit utile pour insérer du code Lingo que vous souhaitez exécuter fréquemment pour une série d'objets spécifique. Affectez les objets à la liste `actorList` lorsque vous souhaitez que les éléments Lingo du gestionnaire `on stepFrame` soient exécutés ; inversement, supprimez ces objets de la liste `actorList` pour éviter l'exécution de ces éléments Lingo. Lorsque les objets figurent dans `actorList`, leurs gestionnaires `on stepFrame` sont exécutés chaque fois que la tête de lecture arrive sur une image ou que la commande `updateStage` est émise.

Le message `stepFrame` est envoyé avant le message `prepareFrame`.

Affectez des objets à `actorList` pour qu'ils puissent répondre aux messages `stepFrame`. Les objets doivent disposer d'un gestionnaire `on stepFrame` pour utiliser cette fonctionnalité intégrée avec `actorList`.

Les commandes `go`, `play` et `updateStage` sont désactivées dans un gestionnaire `on stepFrame`.

Exemple

Si l'objet enfant est affecté à `actorList`, le gestionnaire `on stepFrame` de ce script parent met à jour la position de l'image-objet stockée dans la propriété `mySprite` chaque fois que la tête de lecture arrive sur une image :

```
-- Lingo syntax
property mySprite

on new me, theSprite
    mySprite = theSprite
    return me
end

on stepFrame me
    sprite(mySprite).loc = point(random(640),random(480))
end

// JavaScript syntax
// define a constructor class that contains the mySprite property
function Frame(theSprite) {
    this.mySprite = theSprite;
}

function stepFrame() {
    var myFrame = new Frame(sprite(spriteName).spriteNum);
    sprite(myFrame.mySprite).loc = point(random(640),random(480));
end
```

on stopMovie

Syntaxe

```
-- Lingo syntax
on stopMovie
    statement(s)
end
```

```
// JavaScript syntax
function stopMovie() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées à l'arrêt de l'animation.

Un gestionnaire `on stopMovie` constitue un emplacement adéquat pour insérer du code Lingo destiné à effectuer les tâches de nettoyage, telles que la fermeture des fichiers de ressource, l'élimination des variables globales et la suppression de champs et d'objets, lorsque l'animation est terminée.

Lorsque le gestionnaire `on stopMovie` figure dans une animation dans une fenêtre, il n'est appelé que lorsque l'animation est lue jusqu'à la fin ou débouche sur une autre animation. Il n'est pas appelé lorsque la fenêtre est fermée ou supprimée par le biais de la commande `forget window`.

Exemple

Le gestionnaire suivant crée une variable globale lorsque l'animation s'arrête :

```
-- Lingo syntax
global gCurrentScore

on stopMovie
    gCurrentScore = 0
end

// JavaScript syntax
_global.gCurrentScore;

function stopMovie() {
    _global.gCurrentScore = 0;
}
```

Voir aussi

on prepareMovie

on streamStatus

Syntaxe

```
-- Lingo syntax
on streamStatus URL, state, bytesSoFar, bytesTotal, error
    statement(s)
end

// JavaScript syntax
function streamStatus(URL, state, bytesSoFar, bytesTotal, error) {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; appelé régulièrement pour déterminer la quantité d'un objet déjà téléchargée depuis Internet. Ce gestionnaire n'est appelé que si `tellStreamStatus (TRUE)` a été appelé et que le gestionnaire a été ajouté à un script d'animation.

Le gestionnaire d'événement `on streamStatus` comporte les paramètres suivants :

URL	Affiche l'adresse Internet des données en cours de récupération.
state	Affiche l'état du chargement en cours. Les valeurs possibles sont `Connecting` (connexion), `Started` (commencé), `InProgress` (en cours), `Complete` (terminé) et `Error` (erreur).
bytesSoFar	Affiche le nombre d'octets récupérés depuis le réseau.
bytesTotal	Affiche le nombre total d'octets transférés, si ce chiffre est connu. Cette valeur peut être 0 si le serveur HTTP ne fait pas figurer la longueur du contenu dans l'en-tête MIME.
error	Affiche une chaîne vide ("") si le téléchargement n'est pas terminé, OK s'il a abouti avec succès ou un code d'erreur s'il a échoué.

Ces paramètres sont automatiquement remplis par Director avec des informations concernant l'évolution du téléchargement. Ce gestionnaire est appelé automatiquement par Director et il n'y a aucun moyen de contrôler quand il est appelé la fois suivante. Si des informations relatives à une opération spécifique sont requises, appelez `getStreamStatus()`.

Vous pouvez lancer un téléchargement réseau à l'aide de commandes Lingo, en liant les médias avec une URL ou en utilisant un acteur externe à partir d'une URL. Un gestionnaire `streamStatus` est appelé pour fournir des informations sur tous les transferts à partir du réseau.

Insérez le gestionnaire `streamStatus` dans un script d'animation.

Exemple

Le gestionnaire suivant détermine l'état d'un objet transféré et affiche son URL :

```
-- Lingo syntax
on streamStatus URL, state, bytesSoFar, bytesTotal
    if state = "Complete" then
        put(URL && "download finished")
    end if
end streamStatus

// JavaScript syntax
function streamStatus(URL, state, bytesSoFar, bytesTotal) {
    if (state == "Complete") {
        put(URL + " download finished");
    }
}
```

Voir aussi

`getStreamStatus()`, `tellStreamStatus()`

on timeOut

Syntaxe

```
-- Lingo syntax
on timeOut
    statement(s)
end

// JavaScript syntax
function timeOut() {
```

```
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées lorsque la souris ou le clavier n'a pas été utilisé pendant la durée spécifiée par `timeOutLength`. Insérez toujours un gestionnaire `on timeOut` dans un script d'animation.

Pour que la temporisation produise la même réponse pendant toute l'animation, utilisez `timeoutScript` afin de contrôler le comportement du délai d'inactivité de façon centralisée.

Exemple

Le gestionnaire suivant lit l'animation Boucle lorsque les utilisateurs sont restés inactifs pendant la durée définie par la propriété `timeoutLength`. Il peut être utilisé pour répondre lorsque les utilisateurs quittent l'ordinateur.

```
-- Lingo syntax
on timeOut
    _movie.play("Attract Loop")
end timeOut

// JavaScript syntax
function timeOut() {
    _movie.play("Attract Loop");
}
```

trayIconMouseDoubleClick

Syntaxe

```
-- Lingo syntax
on trayIconMouseDoubleClick
    statement(s)
end

// JavaScript syntax
function trayIconMouseDoubleClick() {
    statement(s);
}
```

Description

Gestionnaire d'événement d'animation et de fenêtre (Microsoft Windows uniquement). Contient les instructions exécutées lorsqu'un utilisateur double-clique sur l'icône de la barre d'état système.

L'événement `trayIconMouseDoubleClick` n'est envoyé au gestionnaire que si la propriété `systemTrayIcon` présente la valeur `TRUE`.

Exemple

Le gestionnaire suivant met une animation en pause lorsqu'un utilisateur double-clique sur l'icône de la barre d'état système.

```
-- Lingo syntax
on trayIconMouseDoubleClick
    _movie.delay(500)
end

// JavaScript syntax
```

```
function trayIconMouseDoubleClick() {
    _movie.delay(500);
}
```

Voir aussi

Animation, systemTrayIcon, trayIconMouseDown, trayIconRightMouseDown, Fenêtre

trayIconMouseDown

Syntaxe

```
-- Lingo syntax
on trayIconMouseDown
    statement(s)
end

// JavaScript syntax
function trayIconMouseDown() {
    statement(s);
}
```

Description

Gestionnaire d'événement d'animation et de fenêtre (Microsoft Windows uniquement). Contient les instructions exécutées lorsqu'un utilisateur clique une fois sur l'icône de la barre d'état système.

L'événement `trayIconMouseDown` n'est envoyé au gestionnaire que si la propriété `systemTrayIcon` présente la valeur `TRUE`.

Exemple

Le gestionnaire suivant met une animation en pause lorsqu'un utilisateur clique après avoir positionné la souris sur l'icône de la barre d'état système.

```
-- Lingo syntax
on trayIconMouseDown
    _movie.delay(500)
end

// JavaScript syntax
function trayIconMouseDown() {
    _movie.delay(500);
}
```

Voir aussi

Animation, systemTrayIcon, trayIconMouseDoubleClick, trayIconRightMouseDown, Fenêtre

trayIconRightMouseDown

Syntaxe

```
-- Lingo syntax
on trayIconRightMouseDown
    statement(s)
end
```

```
// JavaScript syntax
function trayIconRightMouseDown() {
    statement(s);
}
```

Description

Gestionnaire d'événement d'animation et de fenêtre (Microsoft Windows uniquement). Contient les instructions exécutées lorsqu'un utilisateur clique une fois sur l'icône de la barre d'état système.

L'événement `trayIconRightMouseDown` n'est envoyé au gestionnaire que si la propriété `systemTrayIcon` présente la valeur `TRUE`.

Exemple

Le gestionnaire suivant met une animation en pause lorsqu'un utilisateur clique une fois sur l'icône de la barre d'état système.

```
-- Lingo syntax
on trayIconRightMouseDown
    _movie.delay(500)
end

// JavaScript syntax
function trayIconRightMouseDown() {
    _movie.delay(500);
}
```

Voir aussi

Animation, systemTrayIcon, trayIconMouseDoubleClick, trayIconMouseDown, Fenêtre

on zoomWindow

Syntaxe

```
-- Lingo syntax
on zoomWindow
    statement(s)
end

// JavaScript syntax
function zoomWindow() {
    statement(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées lorsqu'une animation dans une fenêtre est redimensionnée. Cette situation se produit lorsque l'utilisateur clique sur le bouton Réduction ou Agrandissement (Windows) ou sur le bouton Zoom (Mac). Le système d'exploitation détermine les dimensions après avoir redimensionné la fenêtre.

Un gestionnaire d'événement `on zoomWindow` constitue un emplacement adéquat pour insérer du code Lingo réorganisant les images-objets lorsque les dimensions de la fenêtre évoluent.

Exemple

Le gestionnaire suivant déplace l'image-objet 3 vers les coordonnées stockées dans la variable `emplacementCentral` lorsque la fenêtre lue par l'animation est redimensionnée :

```
-- Lingo syntax
on zoomWindow
    centerPlace = point(10, 10)
    sprite(3).loc = centerPlace
end

// JavaScript syntax
function zoomWindow() {
    var centerPlace = point(10, 10);
    sprite(3).loc = centerPlace;
}
```

Voir aussi

drawRect, sourceRect, on resizeWindow

Chapitre 11 : Mots-clés

Ce chapitre répertorie dans l'ordre alphabétique tous les mots-clés disponibles dans Adobe® Director®.

Ces mots-clés ne s'appliquent qu'à Lingo. Bien que la syntaxe JavaScript comporte certains mots-clés et constructions ayant la même fonction que les mots-clés Lingo répertoriés ici, ces éléments JavaScript n'apparaissent pas dans ce chapitre. Pour plus d'informations concernant les mots-clés et constructions de la syntaxe JavaScript, reportez-vous au chapitre « Principes de base de la programmation dans Director », page 4.

\ (continuation)

Syntaxe

```
-- Lingo syntax
first part of a statement on this line
second part of the statement
third part of the statement
```

Description

Symbole de continuation ; lorsqu'utilisé comme dernier caractère d'une ligne, indique que l'instruction continue à la ligne suivante. Lingo interprète alors ces lignes comme une seule instruction.

Exemple

L'instruction suivante utilise le caractère \ pour diviser l'instruction en deux lignes :

```
-- Lingo syntax
if sprite("mySprite").member = member("myMember") then _player.alert("The sprite was created
from myMember")
```

case

Syntaxe

```
-- Lingo syntax
case expression of
    expression1: Statement
    expression2: Statement(s)
    expression3, expression4: Statement
    {otherwise: Statement(s)}
end case
```

Description

Mot-clé ; lance une structure de branchements multiples plus facile à rédiger qu'une suite d'instructions `if...then`.

Lingo compare la valeur de *`case expression`* aux expressions des lignes suivantes. Cette comparaison commence au début de ces lignes et continue dans l'ordre jusqu'à ce que Lingo rencontre une expression identique à *`case expression`*.

Le cas échéant, Lingo exécute la ou les instructions correspondantes suivant les deux-points placés après l'expression identique. Si une seule instruction suit l'expression identique, l'instruction peut être placée sur la même ligne que l'expression. Les instructions multiples doivent apparaître sur des lignes en retrait immédiatement après l'expression identique.

Lorsque plusieurs correspondances possibles pourraient entraîner Lingo à exécuter les mêmes instructions, les expressions doivent être séparées par des virgules. La ligne contenant *expression3* et *expression4* est un exemple d'une telle situation.

Lingo suspend sa recherche de correspondance dès qu'il rencontre la première expression correspondant à celle recherchée.

Si l'instruction facultative `otherwise` figure à la fin de la structure `case`, les instructions qui suivent `otherwise` sont exécutées si Lingo ne rencontre aucune expression identique.

Exemple

Le gestionnaire suivant teste la touche sur laquelle l'utilisateur vient d'appuyer et y répond en conséquence.

- Si l'utilisateur a appuyé sur A, l'animation passe à l'image Pomme.
- Si l'utilisateur a appuyé sur B ou C, l'animation exécute la transition demandée et passe à l'image Oranges.
- Si l'utilisateur a appuyé sur n'importe quelle autre touche, l'ordinateur émet un bip sonore.

```
on keyDown
    case (_key.key) of
        "a": _movie.go("Apple")
        "b", "c":
             _movie.puppetTransition(99)
             _movie.go("Oranges")
        otherwise: _sound.beep()
    end case
end keyDown
```

L'instruction case suivante vérifie si le curseur se trouve sur l'image-objet 1, 2 ou 3 et exécute l'élément Lingo approprié :

```
case _movie.rollOver() of
    1: sound(1).play(member("Horn"))
    2: sound(1).play(member("Drum"))
    3: sound(1).play(member("Bongos"))
end case
```

char...of

Syntaxe

```
-- Lingo syntax
textMemberExpression.char[whichCharacter]
char whichCharacter of fieldOrStringVariable
textMemberExpression.char[firstCharacter..lastCharacter]
char firstCharacter to lastCharacter of fieldOrStringVariable
```

Description

Mot-clé ; identifie un caractère ou une plage de caractères dans une sous-chaîne. Une expression de sous-chaîne est n'importe quel caractère, mot, élément ou ligne dans n'importe quelle source de texte (telle que des acteurs champ et des variables) contenant une chaîne.

- Une expression utilisant `whichCharacter` identifie un caractère spécifique.
- Une expression utilisant `firstCharacter` et `lastCharacter` identifie une plage de caractères.

Ces expressions doivent être des nombres entiers spécifiant un caractère ou une plage de caractères dans une sous-chaîne. Les caractères peuvent être des lettres, des nombres, des signes de ponctuation, des espaces et des caractères de contrôle comme Tabulation ou Retour.

Le mot-clé `char...of` peut être testé, mais pas défini. Utilisez la commande `put...into` pour modifier les caractères d'une chaîne.

Exemple

L'instruction suivante affiche le premier caractère de la chaîne $9,00 :

```
//Lingo
put(("$9.00").char[1..1])
-- "$"

// Javascript
trace(("$9.00").substring(0,1))
-- "$"
```

L'instruction suivante affiche la chaîne $9,00 en entier :

```
//Lingo
put(("$9.00").char[1..5])
-- "$9.00"

// Javascript
trace(("$9.00").substring(0))
-- "$9.00"
```

L'instruction suivante modifie les cinq premiers caractères du deuxième mot de la troisième ligne d'un acteur texte :

```
//Lingo
member("quiz").line[3].word[2].char[1..5] = "?????"

// Javascript
var s = member(1).getPropRef("line",3).getProp("word",2)
s=s.replace(s.substring(0),"????")
member(1).getPropRef("line",3).setProp("word",2,s)
```

Voir aussi

mouseMember, mouseItem, mouseLine, mouseWord

end

Syntaxe

```
-- Lingo syntax
end
```

Description

Mot-clé ; marque la fin des gestionnaires et des structures de contrôle à plusieurs lignes.

Exemple

Le gestionnaire mouseDown suivant s'achève par une instruction end mouseDown.

```
on mouseDown
     _player.alert("The mouse was pressed")
end mouseDown
```

end case

Syntaxe

```
-- Lingo syntax
end case
```

Description

Mot-clé ; termine une instruction `case`.

Exemple

Le gestionnaire suivant utilise le mot-clé end case pour terminer l'instruction case :

```
on keyDown
    case (_key.key) of
        "a": _movie.go("Apple")
        "b", "c":
             _movie.puppetTransition(99)
             _movie.go("Oranges")
        otherwise: _sound.beep()
    end case
end keyDown
```

Voir aussi

case

exit

Syntaxe

```
-- Lingo syntax
exit
```

Description

Mot-clé ; indique à Lingo de quitter un gestionnaire et de retourner où le gestionnaire a été appelé. Si le gestionnaire est imbriqué dans un autre gestionnaire, Lingo retourne au gestionnaire principal.

Exemple

La première instruction du script suivant vérifie si le moniteur est en noir et blanc et, le cas échéant, sort du gestionnaire :

```
on setColors
    if _system.colorDepth = 1 then exit
    sprite(1).foreColor = 35
end
```

Voir aussi

abort, halt(), quit(), pass, return (mot-clé)

exit repeat

Syntaxe

```
-- Lingo syntax
exit repeat
```

Description

Mot-clé ; indique à Lingo de quitter une boucle et de passer à l'instruction suivant l'instruction `end repeat`, sans toutefois quitter la méthode ou le gestionnaire actuel.

Le mot-clé `exit repeat` est utile pour sortir d'une boucle de répétition lorsqu'une condition spécifiée, telle que l'égalité de deux valeurs ou la présence d'une valeur donnée dans une variable, existe.

Exemple

Le gestionnaire suivant cherche la position de la première voyelle dans une chaîne représentée par la variable chaîneDeTest. Dès que la première voyelle est trouvée, la commande exit repeat indique à Lingo de quitter la boucle de répétition et de passer à l'instruction return i :

```
on findVowel testString
    repeat with i = 1 to testString.char[testString.char.count]
        if "aeiou" contains testString.char[i] then exit repeat
    end repeat
    return i
end
```

Voir aussi

`repeat while`, `repeat with`

field

Syntaxe

```
field(whichField)
```

Description

Mot-clé ; fait référence à l'acteur champ spécifié par `whichField`.

- Lorsque `whichField` est une chaîne, il est utilisé comme nom d'acteur.
- Lorsque `whichField` est un nombre entier, il est utilisé comme numéro d'acteur.

Les chaînes de caractères et expressions de sous-chaînes peuvent être lues ou placées dans le champ.

Le terme `field` était utilisé dans les versions précédentes de Director et est conservé pour une compatibilité amont. Pour les nouvelles animations, utilisez `member` pour faire référence aux acteurs champs.

Exemple

L'instruction suivante place les caractères 5 à 10 du champ nommé Entrée dans la variable `myKeyword` :

```
myKeyword = field("entry").char[5..10]
```

L'instruction suivante vérifie si l'utilisateur a saisi le mot *bureau* et, le cas échéant, passe à l'image `deskBid` :

```
if member("bid") contains "desk" then _movie.go("deskBid")
```

Voir aussi

`char...of`, `item...of`, `line...of`, `word...of`

global

Syntaxe

```
global variable1 {, variable2} {, variable3}...
```

Description

Mot-clé ; définit une variable comme variable globale pour que les autres gestionnaires ou animations puissent la partager.

Chaque gestionnaire qui examine ou change le contenu d'une variable globale doit utiliser le mot-clé `global` pour identifier la variable comme une variable globale. Autrement, le gestionnaire traite la variable comme une variable locale, même si un autre gestionnaire l'a déclarée comme globale.

Remarque : *pour vérifier que les variables globales sont disponibles dans l'ensemble d'une animation, déclarez et initialisez-les dans le gestionnaire* `prepareMovie`*. Ensuite, si vous quittez l'animation puis revenez à celle-ci à partir d'une autre animation, vos variables globales reprennent leurs valeurs initiales à moins que vous ne vérifiiez d'abord qu'elles ne sont pas déjà définies.*

Une variable globale peut être déclarée dans n'importe quel gestionnaire ou script. Sa valeur peut être utilisée par d'autres gestionnaires ou scripts qui déclarent également la variable comme globale. Si le script change la valeur de la variable, la nouvelle valeur est disponible pour tous les autres gestionnaires traitant la variable comme globale.

Une variable globale est disponible dans n'importe quel script ou animation, quel que soit l'endroit où elle a été d'abord déclarée ; elle n'est pas automatiquement supprimée lorsque vous naviguez vers une autre image, animation ou fenêtre.

Les variables manipulées dans la fenêtre Messages sont automatiquement globales, même si elles ne sont pas explicitement déclarées comme telles.

Les animations comportant du contenu Shockwave® qui sont lues sur Internet ne peuvent pas accéder à des variables globales dans d'autres animations, même si les animations sont lues sur la même page HTML. Les animations peuvent uniquement partager des variables globales si une animation intégrée navigue vers une autre animation et est remplacée par le biais des commandes `goToNetMovie` ou `go movie`.

Exemple

L'exemple suivant attribue à la variable globale PointDeDépart une valeur initiale de 1 si cette variable ne contient pas déjà une valeur. Cela permet une navigation vers l'animation et à partir de celle-ci sans perte des données enregistrées.

```
//Lingo
global gStartingPoint

on prepareMovie
    if voidP(gStartingPoint) then gStartingPoint = 1
end

// Javascript
function prepareMovie(){
    if (_global.gStartingPoint==null) then _global.gStartingPoint = 1
}
```

Voir aussi

showGlobals(), property, gotoNetMovie

if

Syntaxe

```
if logicalExpression then statement
if logicalExpression then statement
else statement
end if
if logicalExpression then
        statement(s)
end if
if logicalExpression then
        statement(s)
else
        statement(s)
end if
if logicalExpression1 then
        statement(s)
else if logicalExpression2 then
        statement(s)
else if logicalExpression3 then
        statement(s)
end if
if logicalExpression1 then
        statement(s)
else logicalExpression2
end if
```

Description

Mot-clé ; la structure `if...then` évalue l'expression logique spécifiée par `logicalExpression`.

- Si la condition présente la valeur `TRUE`, Lingo exécute la ou les instructions qui suivent `then`.
- Si la condition présente la valeur `FALSE`, Lingo exécute la ou les instructions qui suivent `else`. Si aucune instruction ne suit `else`, Lingo quitte la structure `if...then`.
- Toutes les parties de la condition doivent être évaluées, l'exécution ne s'arrêtant pas à la première condition remplie ou non remplie. Un code plus rapide peut donc être créé en imbriquant des instructions `if...then` sur des lignes séparées au lieu de toutes les placer sur la première ligne à évaluer.

Lorsque la condition est une propriété, Lingo vérifie automatiquement si elle a la valeur `TRUE`. Vous n'avez pas besoin d'ajouter explicitement l'expression `= TRUE` après la propriété.

La partie `else` de l'instruction est facultative. Pour utiliser plus d'une instruction *`then`* ou *`else`*, vous devez conclure par la forme `end if`.

La partie `else` correspond toujours à l'instruction `if` précédente ; vous devez donc parfois inclure une instruction `else nothing` pour associer un mot-clé `else` au mot-clé `if` approprié.

Remarque : *une façon rapide de déterminer dans la fenêtre Script si le script est correctement lié est d'appuyer sur la touche de tabulation. Cela force Director à vérifier la fenêtre Script ouverte et afficher la présentation en retrait du contenu. Les différences sont immédiatement visibles.*

Exemple

L'instruction suivante vérifie si l'utilisateur a appuyé sur un retour chariot et, le cas échéant, continue :

```
if the key = RETURN then go the frame + 1
```

Le gestionnaire suivant vérifie si les touches Cmd et Q ont été enfoncées simultanément et, le cas échéant, exécute les instructions suivantes :

```
on keyDown
    if (_key.commandDown) and (_key.key = "q") then
        cleanUp
        quit
    end if
end keyDown
```

Comparez les deux constructions suivantes ainsi que les résultats au niveau des performances. La première construction évalue les deux conditions et doit déterminer la mesure du temps, ce qui prend un certain temps. La seconde construction évalue la première condition ; la seconde condition est uniquement vérifiée si la première condition présente la valeur TRUE.

```
spriteUnderCursor = rollOver()
if (spriteUnderCursor > 25) and MeasureTimeSinceIStarted() then
    _player.alert("You found the hidden treasure!")
end if
```

L'autre construction, plus rapide, serait la suivante :

```
spriteUnderCursor = rollOver()
if (spriteUnderCursor > 25) then
    if MeasureTimeSinceIStarted() then
        _player.alert("You found the hidden treasure!")
    end if
end if
```

Voir aussi

case

INF

Syntaxe

```
-- Lingo syntax
INF
```

Description

Valeur renvoyée ; indique qu'une expression Lingo spécifiée est évaluée comme un nombre infini.

Voir aussi

NAN

item...of

Syntaxe

```
-- Lingo syntax
textMemberExpression.item[whichItem]
item whichItem of fieldOrStringVariable
textMemberExpression.item[firstItem..lastItem]
item firstItem to lastItem of fieldOrStringVariable
```

Description

Mot-clé ; spécifie un élément ou une série d'éléments d'une sous-chaîne. Dans ce contexte, un élément est une série de caractères délimités par le séparateur défini dans la propriété `itemDelimiter`.

Les termes *`whichItem`*, *`firstItem`* et *`lastItem`* doivent être des nombres entiers ou des expressions entières faisant référence à la position des éléments dans la sous-chaîne.

Les sous-chaînes permettent de faire référence à tout caractère, mot, élément ou ligne de n'importe quelle chaîne de texte. Les chaînes possibles sont les acteurs champ et texte et les variables contenant des chaînes.

Lorsque le nombre spécifiant le dernier élément est supérieur à celui correspondant à la position de cet élément dans la sous-chaîne, le dernier élément est spécifié à la place.

Exemple

L'instruction suivante recherche le troisième élément d'une sous-chaîne composée de noms de couleurs et affiche le résultat dans la fenêtre Messages :

```
put("red,yellow,blue green,orange".item[3])
-- "blue green"
```

Le résultat correspond à la sous-chaîne « bleu vert » car tous les caractères placés entre virgules sont pris en compte.

L'instruction suivante détermine les éléments du troisième au cinquième élément de la sous-chaîne. Puisque celle-ci ne contient que quatre éléments, seuls le troisième et le quatrième sont renvoyés. Le résultat apparaît dans la fenêtre Messages.

```
put("red,yellow,blue green,orange".item[3..5])
-- "blue green, orange"
put item 5 of "red, yellow, blue green, orange"
-- ""
```

L'instruction suivante insère l'élément Bureau en quatrième position dans la deuxième ligne de l'acteur champ Tous les devis :

```
member("All Bids").line[2].item[4] = "Desk"
```

Voir aussi

`char...of`, `itemDelimiter`, `number of members`, `word...of`

line...of

Syntaxe

```
-- Lingo syntax
textMemberExpression.line[whichLine]
line whichLine of fieldOrStringVariable
```

```
textMemberExpression.line[firstLine..lastLine]
line firstLine to lastLine of fieldOrStringVariable
```

Description

Mot-clé ; identifie une ligne ou une série de lignes d'une sous-chaîne. Une ligne est constituée d'une série de caractères délimités par des retours de chariot et non par des retours à la ligne automatiques.

Les expressions *whichLine*, *firstLine* et *lastLine* doivent être des nombres entiers spécifiant une ligne de la sous-chaîne.

Les sous-chaînes peuvent représenter n'importe quel caractère, mot, élément ou ligne d'un groupe de caractères. Les sources de texte peuvent être des acteurs champ et des variables contenant des chaînes.

Exemple

L'instruction suivante affecte les quatre premières lignes de la variable Action à l'acteur champ Tâches :

```
member("To Do").text = Action.line[1..4]
```

L'instruction suivante insère le mot et après le deuxième mot de la troisième ligne de la chaîne affectée à la variable Notes :

```
put "and" after Notes.line[3].word[2]
```

Voir aussi

char...of, item...of, word...of, number of members

loop (mot-clé)

Syntaxe

```
-- Lingo syntax
_movie.goLoop()
```

Description

Mot-clé ; fait référence au repère.

Exemple

Le gestionnaire suivant fait boucler l'animation entre le repère précédent et l'image actuelle :

```
on exitFrame
    _movie.goLoop()
end exitFrame
```

me

Syntaxe

```
-- Lingo syntax
me
```

Description

Variable spéciale ; s'utilise à l'intérieur de scripts parents et de comportements pour faire référence à l'objet actuel lorsqu'il est une instance du script parent, du comportement ou d'une variable contenant l'adresse mémoire de l'objet.

Ce terme n'a aucune signification prédéfinie dans Lingo. Le terme `me` est utilisé par convention.

Vous pouvez voir un exemple de `me` dans une animation en consultant l'animation Parent Scripts du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

L'instruction suivante affecte l'objet `myBird1` au script Oiseau. Le mot-clé `me` accepte le script Oiseau et sert à renvoyer ce paramètre.

```
myBird1 = new script("Bird")
```

Voici le gestionnaire `on new` du script Oiseau :

```
on new me
    return me
end
```

Les deux ensembles de gestionnaires suivants forment un script parent. Le premier ensemble utilise `me` pour faire référence à l'objet enfant. Le second ensemble utilise la variable `myAddress` pour faire référence à l'objet enfant. Pour ce qui est du reste, les scripts parents sont les mêmes.

Premier ensemble :

```
property myData

on new me, theData
    myData = theData
    return me
end

on stepFrame me
    ProcessData me
end
```

Second ensemble :

```
property myData

on new myAddress, theData
    myData = theData
    return myAddress
end

on stepFrame myAddress
    ProcessData myAddress
end
```

Voir aussi

`new()`, `ancestor`

menu

Syntaxe

```
-- Lingo syntax
menu: menuName
itemName | script
itemName | script
...
```

or

```
menu: menuName
itemName | script
itemName | script
...
[more menus]
```

Description

Mot-clé ; utilisé avec la commande `installMenu`, il spécifie le contenu des menus personnalisés. Les acteurs champ contiennent des définitions de menus. Faites référence à ces définitions en utilisant le nom ou le numéro de l'acteur.

Le mot-clé `menu` est immédiatement suivi d'un deux-points, d'un espace et du nom du menu. Spécifiez les éléments de ce menu sur les lignes suivantes. Vous pouvez définir un script qui est exécuté lorsque l'utilisateur choisit un élément du menu en plaçant le script après le symbole barre verticale (|). Un nouveau menu est défini par les occurrences suivantes du mot-clé `menu`.

Remarque : *les menus ne sont pas disponibles dans Shockwave Player.*

Sur le Mac, vous pouvez utiliser des caractères spéciaux pour définir des menus personnalisés. Ces caractères spéciaux différencient les majuscules des minuscules. Par exemple, pour qu'un élément de menu apparaisse en gras, la lettre *B* doit être en majuscules.

Des symboles spéciaux doivent suivre le nom de l'élément de menu et précéder le symbole barre verticale (|). Vous pouvez également utiliser plus d'un caractère spécial pour définir un élément de menu. L'utilisation de <B<U, par exemple, définit le style Gras et Souligné.

Evitez de formater les caractères spéciaux des animations qui sont lues sur des plates-formes différentes car Windows® ne prend pas toujours en charge ce formatage.

Symbole	Exemple	Description
`@`	`menu: @`	*Sur le Mac®, crée le symbole Apple® et active les éléments de la barre des menus Mac lorsque vous définissez un menu Pomme.
`!Ã`	`!ÃSélection rapide`	*Sur le Mac, place une coche (Option+v) près du menu.
`<B`	`Gras<B`	*Sur le Mac, attribue le style Gras à l'élément de menu.
`<I`	`Italique<I`	*Sur le Mac, attribue le style Italique.
`<U`	`Souligné<U`	*Sur le Mac, attribue le style Souligné.
`<O`	`Relief<O`	*Sur le Mac, attribue le style Relief.
`<S`	`Ombré<S`	*Sur le Mac, attribue le style Ombré.
`|`	`Ouvrir/O | go to frame "Ouvrir"`	Associe un script à l'élément de menu.
`/`	`Quitter/Q`	Définit un équivalent commande-touche.
`(`	`Enregistrer(`	Désactive l'élément de menu.
`(-`	`(-`	Crée une ligne désactivée dans le menu.

*. identifie les symboles de formatage qui ne fonctionnent que sur le Mac.

Exemple

Voici le texte d'un acteur champ appelé menuPersonnalisé2 qui permet d'indiquer le contenu d'un menu Fichier personnalisé. Pour installer ce menu, utilisez `installMenu member("CustomMenu2")` pendant la lecture de l'animation. L'élément de menu Convertir exécute le gestionnaire personnalisé `convertThis`.

```
menu: File
Open/O | _movie.go("Open")
Close/W | _movie.go("Close")
Convert/C | convertThis
    (-
Quit/Q | _movie.go("Quit")
```

Voir aussi

installMenu, name, number (éléments de menu), checkMark, enabled, script

NAN

Syntaxe

```
-- Lingo syntax
NAN
```

Description

Valeur renvoyée ; indique qu'une expression Lingo spécifiée n'est pas un nombre.

L'instruction suivante tente d'afficher la racine carrée de -1, qui n'est pas un nombre, dans la fenêtre Messages :

```
-- Lingo syntax
put((-1).sqrt) -- NAN
```

Voir aussi

INF

next

Syntaxe

```
-- Lingo syntax
next
```

Description

Mot-clé ; fait référence au repère suivant de l'animation et revient à utiliser l'expression `the marker (+ 1)`.

Exemple

L'instruction suivante envoie la tête de lecture sur le repère suivant de l'animation :

```
go next
```

Le gestionnaire suivant fait passer la tête de lecture au repère suivant du scénario lorsque l'utilisateur appuie sur la touche fléchée Droite et au repère précédent lorsqu'il appuie sur la touche fléchée Gauche :

```
on keyUp
    if (_key.keyCode = 124) then _movie.goNext()
    if (_key.keyCode = 123) then _movie.goPrevious()
end keyUp
```

Voir aussi

loop (mot-clé), goPrevious()

next repeat

Syntaxe

```
-- Lingo syntax
next repeat
```

Description

Mot-clé ; fait passer Lingo à l'étape suivante d'une boucle d'un script. Cette fonction est différente de celle du mot-clé `exit repeat`.

Exampleg

La boucle suivante n'affiche que les nombres impairs dans la fenêtre Messages :

```
repeat with i = 1 to 10
    if (i mod 2) = 0 then next repeat
    put(i)
end repeat
```

on

Syntaxe

```
-- Lingo syntax
on handlerName {argument1}, {arg2}, {arg3} ...
    statement(s)
end handlerName
```

Description

Mot-clé ; indique le début d'un gestionnaire, une suite d'instructions Lingo que vous pouvez exécuter en utilisant le nom du gestionnaire. Un gestionnaire peut accepter des arguments comme valeurs d'entrée et renvoyer une valeur comme résultat d'une fonction.

Les gestionnaires peuvent être définis dans les comportements, les scripts d'animations et les scripts d'acteurs. Le gestionnaire d'un script d'acteur ne peut être appelé que par les autres gestionnaires du même script. Le gestionnaire d'un script d'animation peut être appelé de partout.

Vous pouvez utiliser le même gestionnaire dans plusieurs animations en plaçant son script dans une distribution partagée.

otherwise

Syntaxe

```
-- Lingo syntax
otherwise statement(s)
```

Description

Mot-clé ; précède les instructions exécutées par Lingo si aucune des conditions précédentes d'une instruction `case` n'est remplie.

Ce mot-clé peut s'utiliser pour avertir les utilisateurs d'une entrée hors limites ou d'un type non valide. Il peut aussi s'avérer très utile pour les opérations de débogage pendant le développement.

Exemple

Le gestionnaire suivant teste la touche sur laquelle l'utilisateur vient d'appuyer et y répond en conséquence :

- Si l'utilisateur a appuyé sur A, B ou C, l'animation exécute l'action correspondante qui suit le mot-clé of.
- Si l'utilisateur a appuyé sur une autre touche, l'animation exécute l'instruction qui suit le mot-clé otherwise.

Le cas échéant, l'instruction est un simple message d'alerte.

```
//Lingo
on keyDown
     case (_key.key) of
        "a": _movie.go("Apple")
        "b", "c":
             _movie.puppetTransition(99)
             _movie.go("Oranges")
        otherwise: _player.alert("That is not a valid key.")
    end case
end keyDown

// Javascript
function keyDown()
{
    switch(_key.key)
    {
        case "a":
        _movie.go("Apple");
        break;
        case "b":
        case "c":
        _movie.puppetTransition(99);
        _movie.go("Oranges");
        break;
        default:
        _player.alert("That is not a valid key.");
    }
}
```

property

Syntaxe

```
-- Lingo syntax
property {property1}{, property2} {,property3} {...}
```

Description

Mot-clé ; déclare que les propriétés indiquées par *property1*, *property2*, etc. sont des variables de propriétés.

Déclarez les variables de propriétés au début du script parent ou du script de comportement. L'opérateur `the` permet d'y accéder en dehors de ces scripts.

Remarque : *la propriété `spriteNum` est disponible à tous les comportements et il suffit de la déclarer pour y accéder.*

Vous pouvez faire référence à une propriété dans un script parent ou de comportement sans utiliser le mot-clé `me`. Cependant, pour faire référence à une propriété de l'ancêtre d'un script parent, utilisez la forme `me.property`.

Pour les comportements, les propriétés définies dans un script de comportement sont disponibles aux autres comportements associés à la même image-objet.

Vous pouvez manipuler la propriété d'un objet enfant directement en dehors des scripts parents de cet objet au moyen d'une syntaxe semblable à celle utilisée pour manipuler d'autres propriétés. Par exemple, l'instruction suivante définit la propriété `motionStyle` d'un objet enfant :

```
set the motionStyle of myBouncingObject to #frenetic
```

Utilisez la fonction `count` pour déterminer le nombre de propriétés contenues dans le script parent d'un objet enfant. Récupérez le nom de ces propriétés au moyen de `getPropAt`. Ajoutez des propriétés à un objet au moyen de `setaProp()`.

Vous pouvez voir un exemple de `property` dans une animation en consultant l'animation Parent Scripts du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

L'instruction suivante permet à chaque objet enfant créé à partir d'un script parent unique de posséder ses propres paramètres de position et de vitesse :

```
property location, velocity
```

Le gestionnaire de script parent suivant déclare une propriété pMonNumDimageObjet pour la rendre disponible :

```
-- script Elder
property pMyChannel
on new me, whichSprite
    me.pMyChannel = whichSprite
    return me
end
```

Le script de comportement d'origine définit l'ancêtre et transmet la propriété `spriteNum` à tous les comportements :

```
property spriteNum

property ancestor

on beginSprite me
    ancestor = new script("Elder", spriteNum)
end
```

Voir aussi

`end`, `ancestor`, `spriteNum`

put...after

Syntaxe

```
-- Lingo syntax
put expression after chunkExpression
```

Description

Commande ; évalue une expression Lingo, convertit la valeur en chaîne et insère cette dernière à la fin d'une sous-chaîne spécifiée dans un conteneur, sans remplacer le contenu de ce dernier. Si *chunkExpression* spécifie une sous-chaîne cible qui n'existe pas, la valeur de la chaîne est insérée de manière appropriée dans le conteneur.

Les expressions de sous-chaîne peuvent représenter n'importe quel caractère, mot, élément ou ligne dans un conteneur quelconque. Les conteneurs peuvent être des acteurs champ, des acteurs texte et des variables contenant des chaînes, ainsi que des caractères, des mots, des éléments, des lignes et des plages spécifiés dans des conteneurs.

Exemple

L'instruction suivante ajoute la chaîne « renard chien chat » après le contenu de l'acteur champ Liste d'animaux :

```
put("fox dog cat") after member("Animal List")
```

Le même résultat peut s'obtenir avec l'instruction suivante :

```
put "fox dog cat" after member("Animal List").line[1]
```

Voir aussi

char...of, item...of, line...of, paragraph, word...of, put...before, put...into

put...before

Syntaxe

```
-- Lingo syntax
put expression before chunkExpression
```

Description

Commande ; évalue une expression Lingo, convertit la valeur en chaîne et insère cette dernière avant une sous-chaîne spécifiée dans un conteneur, sans remplacer le contenu de ce dernier. Si *chunkExpression* spécifie une sous-chaîne cible qui n'existe pas, la valeur de la chaîne est insérée de manière appropriée dans le conteneur.

Les expressions de sous-chaîne peuvent représenter n'importe quel caractère, mot, élément ou ligne dans un conteneur quelconque. Les conteneurs peuvent être des acteurs champ, des acteurs texte et des variables contenant des chaînes, ainsi que des caractères, des mots, des éléments, des lignes et des plages spécifiés dans des conteneurs.

Exemple

L'instruction suivante affecte à la variable listeDanimaux la chaîne « renard chien chat », puis insère le mot *élan* avant le deuxième mot de la liste :

```
put "fox dog cat" into animalList
put "elk " before word 2 of animalList
```

Le résultat correspond à la chaîne « renard élan chien chat ».

Le même résultat peut s'obtenir avec la syntaxe suivante :

```
put "fox dog cat" into animalList
```

```
put "elk " before animalList.word[2]
```

Voir aussi

char...of, item...of, line...of, paragraph, word...of, put...after, put...into

put...into

Syntaxe

```
-- Lingo syntax
put expression into chunkExpression
```

Description

Commande ; évalue une expression Lingo, convertit la valeur en une chaîne et insère celle-ci pour remplacer une sous-chaîne spécifiée d'un conteneur. Si *chunkExpression* spécifie une sous-chaîne cible qui n'existe pas, la valeur de la chaîne est insérée de manière appropriée dans le conteneur.

Les expressions de sous-chaîne peuvent représenter n'importe quel caractère, mot, élément ou ligne dans un conteneur quelconque. Les conteneurs peuvent être des acteurs champ, des acteurs texte et des variables contenant des chaînes, ainsi que des caractères, des mots, des éléments, des lignes et des plages spécifiés dans des conteneurs.

Lorsqu'une animation est lue en tant qu'applet, la commande `put...into` remplace tout le texte d'un conteneur, et non uniquement des sous-chaînes de texte.

Pour affecter des valeurs à des variables, utilisez la commande `set`.

Exemple

L'instruction suivante change la seconde ligne de l'acteur champ Critiques en Critique par Olivier Pognon :

```
put "Reviewed by Agnes Gooch" into line 2 of member("Review Comments")
```

Le même résultat peut s'obtenir avec un acteur texte au moyen de la syntaxe suivante :

```
put "Reviewed by Agnes Gooch" into member("Review Comments").line[2]

// Javascript
member("Review Comments").setProp("line",2,"Reviewed by Agnes Gooch")
repeat while:
```

Voir aussi

char...of, item...of, line...of, paragraph, word...of, put...before, put...after, set...to, set...=

repeat while

Syntaxe

```
-- Lingo syntax
repeat while testCondition
    statement(s)
end repeat
```

Description

Mot-clé ; exécute les *statement(s)* tant que la condition spécifiée par *testCondition* a la valeur TRUE. Cette structure peut servir pour des instructions Lingo qui lisent des chaînes jusqu'à ce que la fin d'un fichier soit atteinte, qui vérifient des éléments jusqu'à la fin d'une liste ou qui effectuent une action en boucle jusqu'à ce que l'utilisateur clique sur le bouton de la souris ou le relâche.

Tant qu'il est dans une boucle, Lingo ignore les autres événements. Pour déterminer la touche actuelle dans une boucle, utilisez la propriété keyPressed.

Un seul gestionnaire peut être exécuté à la fois. Si Lingo reste dans une boucle d'une façon prolongée, d'autres événements s'empilent en attente d'évaluation. Les boucles sont donc préférables pour les opérations courtes et rapides ou lorsque vous ne prévoyez aucune action de l'utilisateur.

Si vous devez traiter quelque chose pendant au moins plusieurs secondes, évaluez la fonction dans une boucle avec un compteur quelconque ou testez sa progression.

Si la condition d'arrêt n'est jamais atteinte ou qu'il n'existe aucune sortie de la boucle, vous pouvez forcer Director à s'arrêter avec Ctrl+Alt+point (Windows) ou Cmde+point (Mac).

Exemple

Le gestionnaire suivant lance le compteur en le remettant à 0, puis le fait compter jusqu'à 60 millisecondes :

```
on countTime
    _system.milliseconds
    repeat while _system.milliseconds < 60
    -- waiting for time
    end repeat
end countTime

// Javascript
function countTime()
{
    _system.milliseconds
    while (_system.milliseconds < 60)
    {
        -- waiting for time
    }
}
```

Voir aussi

exit, exit repeat, repeat with, keyPressed()

repeat with

Syntaxe

```
-- Lingo syntax
repeat with counter = start to finish
    statement(s)
end repeat
```

Description

Mot-clé ; exécute le code Lingo spécifié par *statement(s)* autant de fois que spécifié par *counter*. La valeur de *counter* correspond à la différence entre la valeur indiquée par *start* celle indiquée par *finish*. Le compteur augmente de 1 à chaque fois que Lingo parcourt la boucle.

La structure `repeat with` sert à appliquer en continu le même effet à un ensemble d'images-objets ou à calculer une série de nombres à une certaine puissance.

Tant qu'il est dans une boucle, Lingo ignore les autres événements. Pour déterminer la touche actuelle dans une boucle, utilisez la propriété `keyPressed`.

Un seul gestionnaire peut être exécuté à la fois. Si Lingo reste dans une boucle d'une façon prolongée, d'autres événements s'empilent en attente d'évaluation. Les boucles sont donc préférables pour les opérations courtes et rapides ou lorsque vous ne prévoyez aucune action de l'utilisateur.

Si vous devez traiter quelque chose pendant au moins plusieurs secondes, évaluez la fonction dans une boucle avec un compteur ou testez sa progression.

Si la condition d'arrêt n'est jamais atteinte ou qu'il n'existe aucune sortie de la boucle, vous pouvez forcer Director à s'arrêter avec Ctrl+Alt+point (Windows) ou Cmde+point (Mac).

Exemple

Le gestionnaire suivant transforme les images-objets 1 à 30 en esclaves :

```
//Lingo
on puppetize
    repeat with channel = 1 to 30
    _movie.puppetSprite(channel, TRUE)
    end repeat
end puppetize

// Javascript
function puppetize()
{
    for(var channel=1;channel<30;channel++)
    {
    _movie.puppetSprite(channel, true);
    }
}
```

Voir aussi

`exit`, `exit repeat`, `repeat while`, `repeat with...down to`, `repeat with...in list`

repeat with...down to

Syntaxe

```
-- Lingo syntax
repeat with variable = startValue down to endValue
```

Description

Mot-clé ; décompte par incréments de 1 à partir de *`startValue`* jusqu'à *`endValue`*.

Un seul gestionnaire peut être exécuté à la fois. Si Lingo reste dans une boucle d'une façon prolongée, d'autres événements s'empilent en attente d'évaluation. Les boucles sont donc préférables pour les opérations courtes et rapides ou lorsque vous ne prévoyez aucune action de l'utilisateur.

Tant qu'il est dans une boucle, Lingo ignore les autres événements. Pour déterminer la touche actuelle dans une boucle, utilisez la propriété `keyPressed`.

Si vous devez traiter quelque chose pendant au moins plusieurs secondes, évaluez la fonction dans une boucle avec un compteur ou testez sa progression.

Si la condition d'arrêt n'est jamais atteinte ou qu'il n'existe aucune sortie de la boucle, vous pouvez forcer Director à s'arrêter avec Ctrl+Alt+point (Windows) ou Cmde+point (Mac).

Exemple

Le gestionnaire suivant contient une boucle qui décompte de 20 à 15 :

```
on countDown
    repeat with i = 20 down to 15
        sprite(6).member = 10 + i
        _movie.updateStage()
    end repeat
end
```

repeat with...in list

Syntaxe

```
-- Lingo syntax
repeat with variable in someList
```

Description

Mot-clé ; affecte à la variable les valeurs successives issues de la liste spécifiée.

Tant qu'il est dans une boucle, Lingo ignore les autres événements, à l'exception des touches. Pour déterminer la touche actuelle dans une boucle, utilisez la propriété `keyPressed`.

Un seul gestionnaire peut être exécuté à la fois. Si Lingo reste dans une boucle d'une façon prolongée, d'autres événements s'empilent en attente d'évaluation. Les boucles sont donc préférables pour les opérations courtes et rapides ou lorsque vous ne prévoyez aucune action de l'utilisateur.

Si vous devez traiter quelque chose pendant au moins plusieurs secondes, évaluez la fonction dans une boucle avec un compteur ou testez sa progression.

Si la condition d'arrêt n'est jamais atteinte ou qu'il n'existe aucune sortie de la boucle, vous pouvez forcer Director à s'arrêter avec Ctrl+Alt+point (Windows) ou Cmde+point (Mac).

Exemple

L'instruction suivante affiche quatre valeurs dans la fenêtre Messages :

```
//Lingo
repeat with i in [1, 2, 3, 4]
    put(i)
end repeat

// Javascript
var nl = new Array()
nl =[1,2,3,4]
var i
for(i in nl)
{
    trace(nl[i])
}
```

return (mot-clé)

Syntaxe

```
-- Lingo syntax
return expression
```

Description

Mot-clé ; renvoie la valeur *expression* et quitte le gestionnaire. L'argument *expression* peut être une valeur Lingo quelconque.

Lorsque vous appelez un gestionnaire qui sert de fonction définie par l'utilisateur et possède une valeur de renvoi, vous devez entourer de parenthèses les listes d'arguments, même lorsque ces listes sont vides, comme dans le cas du gestionnaire de fonction `diceRoll` illustré sous la fonction `result`.

La fonction du mot-clé `return` est similaire à celle de la commande `exit`, à l'exception près que `return` renvoie également une valeur à l'instruction qui a appelé le gestionnaire. La commande `return` dans un gestionnaire entraîne la sortie immédiate de ce gestionnaire, mais peut renvoyer une valeur au Lingo l'ayant appelé.

L'utilisation de `return` dans des scripts orientés objet peut être difficile à comprendre. Il est plus aisé de commencer par utiliser `return` pour créer des fonctions et sortir de gestionnaires. Vous voyez ensuite que la ligne `return me` dans un gestionnaire `on new` fournit un moyen de passer une référence à un objet créé de façon qu'il puisse être affecté à un nom de variable.

Le mot-clé `return` n'est pas identique à la constante de caractère `RETURN`, qui correspond à un retour de chariot. La fonction dépend du contexte.

Pour récupérer une valeur renvoyée, utilisez des parenthèses après le nom du gestionnaire dans l'instruction d'appel pour indiquer que ce nom de gestionnaire est une fonction.

Vous pouvez voir un exemple de `return (keyword)` dans une animation en consultant l'animation Parent Scripts du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

Le gestionnaire suivant renvoie un multiple aléatoire de cinq compris entre 5 et 100 :

```
//Lingo
on getRandomScore
    theScore = 5 * random(20)
    return theScore
end getRandomScore

// Javascript
function getRandomScore()
{
    theScore = 5 * random(20);
    return theScore;
}
```

Vous appelez ce gestionnaire avec une instruction semblable à la suivante :

```
thisScore = getRandomScore()
```

Dans cet exemple, la variable `thisScore` reçoit la valeur renvoyée par la fonction `getRandomScore()`. Un script parent remplit la même fonction : en renvoyant la référence de l'objet, le nom de la variable du code d'appel fournit un pointeur pour les références ultérieures à cet objet.

Voir aussi

result, RETURN (constante)

set...to, set...=

Syntaxe

```
-- Lingo syntax
lingoProperty = expression
variable = expression
```

Description

Commande ; évalue une expression et affecte le résultat à la propriété spécifiée par *lingoProperty* ou à la variable spécifiée par *variable*.

Exemple

L'instruction suivante donne à l'acteur 3 le nom de Coucher de soleil :

```
member(3).name = "Sunset"
```

L'instruction suivante inverse l'état de la propriété soundEnabled. Si la propriété `soundEnabled` présente la valeur `TRUE` (son activé), cette instruction désactive le son. Si la propriété `soundEnabled` présente la valeur `FALSE` (son désactivé), cette instruction active le son.

```
_sound.soundEnabled = not(_sound.soundEnabled)
```

L'instruction suivante affecte à la variable voyelles la chaîne « aeiou » :

```
vowels = "aeiou"
```

Voir aussi

property

sprite...intersects

Syntaxe

```
-- Lingo syntax
sprite(sprite1).intersects(sprite2)
sprite sprite1 intersects sprite2
```

Description

Mot-clé ; opérateur comparant la position de deux images-objets pour déterminer si le quadrilatère de l'*sprite1* est en contact avec celui de l'*sprite2* (`TRUE`) ou non (`FALSE`).

Si l'encre Dessin seul est appliquée aux deux images-objets, la comparaison porte sur leurs contours, non sur leurs quadrilatères. Le contour d'une image-objet est défini par l'ensemble des pixels autres que blanc formant sa bordure.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 5.

Remarque : *l'opérateur point est requis lorsque l'imageObjet1 n'est pas une expression simple, c'est-à-dire une expression contenant une opération mathématique.*

Exemple

L'instruction suivante vérifie si deux images-objets se croisent et, le cas échéant, modifie le contenu de l'acteur champ Notice pour qu'il contienne le texte « La position est correcte. ».

```
// Lingo Syntax
if sprite i intersects j then put("You placed it correctly.") into member("Notice")

// Javascript
if (sprite(i).intersects(sprite(j)))
{
    member("Notice").text="You placed it correctly";
}
```

Voir aussi

sprite...within, quad

sprite...within

Syntaxe

```
-- Lingo syntax
sprite(sprite1).within(sprite2)
sprite sprite1 within sprite2
```

Description

Mot-clé ; opérateur comparant la position de deux images-objets pour déterminer si le quadrilatère de *sprite1* est entièrement à l'intérieur de celui de *sprite2* (TRUE) ou non (FALSE).

Si l'encre Dessin seul est appliquée aux deux images-objets, la comparaison porte sur leurs contours, non sur leurs quadrilatères. Le contour d'une image-objet est défini par l'ensemble des pixels autres que blanc formant sa bordure.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 5.

Remarque : *l'opérateur point est requis lorsque l'imageObjet1 n'est pas une expression simple, c'est-à-dire une expression contenant une opération mathématique.*

Exemple

L'instruction suivante vérifie si deux images-objets se croisent et, le cas échéant, appelle le gestionnaire Intérieur :

```
//Lingo
if sprite(3).within(2) then doInside

// Javascript
if (sprite(3).within(2))
{
 doInside();
}
```

Voir aussi

sprite...intersects, quad

version

Syntaxe

```
-- Lingo syntax
_player.productVersion
```

Description

Mot-clé ; variable système contenant le numéro de version de Director. La même chaîne apparaît dans la boîte de dialogue Lire les informations du Finder sur le Mac.

Exemple

L'instruction suivante affiche la version de Director dans la fenêtre Messages :

```
put(_player.productVersion)
```

word...of

Syntaxe

```
-- Lingo syntax
member(whichCastMember).word[whichWord]
textMemberExpression.word[whichWord]
chunkExpression.word[whichWord]
word whichWord of fieldOrStringVariable
fieldOrStringVariable. word[whichWord]
textMemberExpression.word[firstWord..lastWord]
member(whichCastMember).word[firstWord..lastWord]
word firstWord to lastWord of chunkExpression
chunkExpression.word[whichWord..lastWord]
```

Description

Expression de sous-chaîne ; spécifie un mot ou une série de mots dans une sous-chaîne. Une sous-chaîne de mots est une suite de caractères délimitée par des espaces. Tout caractère invisible, tel qu'une tabulation ou un retour chariot, est considéré comme un espace.

Les expressions *whichWord*, *firstWord* et *lastWord* doivent avoir pour valeur un nombre entier correspondant à un mot de la sous-chaîne.

Les sous-chaînes peuvent représenter n'importe quel caractère, mot, élément ou ligne d'un groupe de caractères. Les sources de texte peuvent être des acteurs champ et texte et des variables contenant des chaînes.

Vous pouvez voir un exemple de `word...of` dans une animation en consultant l'animation Text du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

Les instructions suivantes affectent à la variable listeDanimaux la chaîne « renard chien chat », puis insèrent le mot élan avant le deuxième mot de la liste :

```
animalList = "fox dog cat"
put "elk" before animalList.word[2]
```

Le résultat correspond à la chaîne « renard élan chien chat ».

L'instruction suivante demande à Director d'afficher le cinquième mot de la même chaîne dans la fenêtre Messages :

```
put "fox elk dog cat".word[5]
```

Cette chaîne ne comportant pas de cinquième mot, la fenêtre Messages affiche deux guillemets droits (""), ce qui indique une chaîne vide.

Voir aussi

`char...of`, `line...of`, `item...of`, `count()`, `number (mots)`

Chapitre 12 : Méthodes

Ce chapitre répertorie dans l'ordre alphabétique toutes les méthodes disponibles dans Director®.

abort

Syntaxe

```
--Lingo syntax
abort

// JavaScript syntax
abort();
```

Description

Commande ; indique à Lingo de sortir du gestionnaire actuel et de tout autre gestionnaire qui l'a appelé sans exécuter les instructions restantes de ce(s) gestionnaire(s). Cette commande diffère du mot-clé `exit` qui revient au gestionnaire à partir duquel le gestionnaire en cours a été appelé.

La commande `abort` ne quitte pas Director.

Paramètres

Aucune.

Exemple

L'instruction suivante indique à Lingo de sortir du gestionnaire actuel et de tout gestionnaire qui l'a appelé si la quantité de mémoire disponible est inférieure à 50 Ko :

```
-- Lingo syntax
if the freeBytes < 50*1024 then abort

// JavaScript syntax
if (_player.freeBytes < 50*1024) {
    abort()
}
```

Voir aussi

`exit`, `halt()`, `quit()`

abs()

Syntaxe

```
--Lingo syntax
abs (numericExpression)

// JavaScript syntax
Math.abs (numericExpression)
```

Description

La fonction `abs()` a plusieurs usages. Elle permet de simplifier le suivi du mouvement de la souris et des images-objets en convertissant leurs coordonnées (qu'elles soient positives ou négatives) en distances (celles-ci sont toujours positives). La fonction `abs()` permet également de traiter les fonctions mathématiques telles que `sqrt()` et `log()`.

En syntaxe JavaScript, utilisez la fonction `abs()` d'un objet mathématique.

Paramètres

`numericExpression` Requis. Nombre entier ou nombre à virgule flottante à partir duquel une valeur absolue est calculée. Si *`numericExpression`* est un nombre entier, sa valeur absolue est également un nombre entier. Si *`numericExpression`* est un nombre à virgule flottante, sa valeur absolue est également un nombre à virgule flottante.

Exemple

L'instruction suivante détermine si la valeur absolue de la différence entre la position en cours de la souris et la valeur de la variable `startV` est supérieure à 30 (puisque vous n'allez pas utiliser un nombre négatif pour exprimer une distance). Le cas échéant, la couleur du premier plan de l'image-objet 6 est modifiée.

```
-- Lingo syntax
if (the mouseV - startV).abs > 30 then sprite(6).forecolor = 95

// JavaScript syntax
if ((_mouse.mouseV - Math.abs(_mouse.startV)) > 30) {
    sprite(6).foreColor = 95;
}
```

activateAtLoc()

Syntaxe

```
-- Lingo syntax
dvdObjRef.activateAtLoc(point(x, y))

// JavaScript syntax
dvdObjRef.activateAtLoc(point(x, y));
```

Description

Méthode de DVD ; active le surlignage de l'élément de menu DVD intégré sous un emplacement de scène spécifié.

Cette méthode renvoie la valeur 0 si l'opération a réussi.

Paramètres

`point(`*`x, y`*`)` Requis. Point de coordonnées de la scène spécifiant l'emplacement de l'élément de menu DVD intégré.

Exemple

L'instruction suivante active le surlignage de l'élément de menu à un emplacement de scène spécifié :

```
-- Lingo syntax
member("movie1").activateAtLoc(point(100, 200))

// JavaScript syntax
member("movie1").activateAtLoc(point(100, 200));
```

Voir aussi

DVD

activateButton()

Syntaxe

```
-- Lingo syntax
dvdObjRef.activateButton()

// JavaScript syntax
dvdObjRef.activateButton();
```

Description

Méthode de DVD ; active le bouton de menu sélectionné.

Cette méthode renvoie la valeur 0 si l'opération a réussi.

Remarque : *Cette méthode n'est pas prise en charge dans Mac®-Intel®.*

Paramètres

Aucune.

Exemple

L'instruction suivante active le bouton de menu sur un acteur spécifié :

```
-- Lingo syntax
sprite(1).member.activateButton()

// JavaScript syntax
sprite(1).member.activateButton();
```

Voir aussi

DVD

add

Syntaxe

```
-- Lingo syntax
linearList.add(value)

// JavaScript syntax
array.push(value)
```

Description

Commande de liste ; pour les listes linéaires uniquement, cette commande ajoute une valeur à une liste linéaire. Dans le cas d'une liste triée, cette valeur est placée dans l'ordre correct. Dans le cas d'une liste non triée, cette valeur est placée à la fin de la liste.

Cette commande, utilisée dans une liste de propriétés provoque une erreur.

Remarque : *Veillez à ne pas confondre la commande* `add` *et l'opérateur* `+` *utilisé pour les additions ou l'opérateur* `&` *utilisé pour concaténer des chaînes.*

Paramètres

value Requis. Valeur à ajouter à la liste linéaire.

Exemple

Les instructions suivantes ajoutent la valeur 2 à la liste intitulée devis. La liste résultante est [3, 4, 1, 2].

```
-- Lingo syntax
bids = [3, 4, 1]
bids.add(2)

// JavaScript syntax
bids = new Array(3,4,1);
bids.push(2);
```

L'instruction suivante ajoute 2 à la liste linéaire triée [1, 4, 5]. Le nouvel élément reste en ordre alphanumérique, la liste étant une liste triée.

```
-- Lingo syntax
bids.add(2)

// JavaScript syntax
bids.push(2);
// to sort the list using JavaScript syntax
bids.sort();
```

Voir aussi

sort

add (texture 3D)

Syntaxe

```
--Lingo syntax
member(whichCastmember).model(whichModel).meshdeform.mesh[index].textureLayer.add()

// JavaScript syntax
member(whichCastmember).model(whichModel).meshdeform.mesh[index].textureLayer.add()
```

Description

Commande 3D de modificateur `meshdeform` ; ajoute une couche de texture vide à la maille du modèle.

Vous pouvez copier les coordonnées de texture entre les couches à l'aide du code suivant :

```
modelReference.meshdeform.texturelayer[a].texturecoordinatelist =
modelReference.meshdeform.texturelayer[b].texturecoordinatelist
```

Paramètres

Aucune.

Exemple

L'instruction suivante crée une nouvelle couche de texture pour la première maille du modèle Oreille.

```
--Lingo syntax
member("Scene").model("Ear").meshdeform.mesh[1].textureLayer.add()

// JavaScript syntax
member("Scene").getprop("model","Ear").meshdeform.mesh[1].textureLayer.add();
```

Voir aussi

meshDeform (modificateur), textureLayer, textureCoordinateList

addAt

Syntaxe

```
list.AddAt(position, value)
```

Description

Commande de liste ; pour les listes linéaires uniquement, cette commande ajoute une valeur à une position spécifique de la liste.

Cette commande, utilisée avec une liste de propriétés provoque une erreur.

Paramètres

position Requis. Nombre entier indiquant la position à laquelle la valeur spécifiée par le paramètre *value* doit être ajoutée à la liste.

value Requis. Valeur à ajouter à la liste.

Exemple

L'instruction suivante ajoute la valeur 8 à la quatrième position de la liste devis qui contient les valeurs [3, 2, 4, 5, 6, 7] :

```
--Lingo
bids = [3, 2, 4, 5, 6, 7]
bids.addAt(4,8)

// Javascript
bids = list(3, 2, 4, 5, 6, 7)
bids.addAt(4,8)
```

La valeur résultante de la liste devis est donc [3, 2, 4, 8, 5, 6, 7].

addBackdrop

Syntaxe

```
-- Lingo syntax
sprite(whichSprite).camera{(index)}.addBackdrop(texture, locWithinSprite, rotation)
member(whichCastmember).camera(whichCamera).addBackdrop(texture, locWithinSprite,
rotation)

// JavaScript syntax
sprite(whichSprite).camera{(index)}.addBackdrop(texture, locWithinSprite, rotation);
member(whichCastmember).camera(whichCamera).addBackdrop(texture, locWithinSprite,
rotation);
```

Description

Commande de caméra 3D ; ajoute un fond à la fin de la liste des fonds de la caméra.

Paramètres

texture Requis. Texture à appliquer au fond.

locWithinSprite Requis. Emplacement 2D auquel le fond s'affiche dans l'image-objet 3D. Cet emplacement est mesuré à partir du coin supérieur gauche de l'image-objet.

rotation Requis. Nombre entier spécifiant le nombre de degrés selon lequel la texture doit pivoter.

Exemple

La première ligne de l'instruction suivante crée la texture Rugueuse à partir de l'acteur Cèdre et l'enregistre dans la variable `t1`. La deuxième ligne applique la texture comme fond à l'emplacement (220, 220) dans l'image-objet 5 avec une rotation de zéro degré. La dernière ligne applique la même texture comme fond pour la caméra 1 de l'acteur Séquence à l'emplacement (20, 20) avec une rotation de 45 degrés.

```
t1 = member("Scene").newTexture("Rough", #fromCastMember,member("Cedar"))
sprite(5).camera.addBackdrop(t1, point(220, 220), 0)
member("Scene").camera[1].addBackdrop(t1, point(20, 20), 45)

// Javascript
var t1 = member("Scene").newTexture("Rough", symbol("fromCastMember"),member("Cedar"))
sprite(5).camera.addBackdrop(t1, point(220, 220), 0) ;
member("Scene").getPropRef("camera",1).addBackdrop(t1, point(20, 20), 45) ;
```

Voir aussi

removeBackdrop

addCamera

Syntaxe

```
-- Lingo syntax
sprite(whichSprite).addCamera(whichCamera, index)
-- JavaScript syntax
sprite(whichSprite).addCamera(whichCamera, index);
```

Description

Commande 3D ; ajoute une caméra à la liste des caméras de l'image-objet. Les vues des différentes caméras s'affichent devant celles des caméras situées aux positions d'*index* inférieures. Vous pouvez définir la propriété `rect` de chaque caméra afin d'afficher plusieurs vues au sein de l'image-objet.

Paramètres

whichCamera Requis. Référence à la caméra à ajouter à la liste des caméras de l'image-objet.

index Requis. Nombre entier spécifiant l'index au niveau duquel *whichCamera* doit être ajoutée à la liste des caméras. Si la valeur *index* est supérieure à la valeur de `cameraCount()`, la caméra est ajoutée à la fin de la liste.

Exemple

L'instruction suivante insère la caméra caméraDeVol en cinquième position de la liste des caméras de l'image-objet 12 :

```
--Lingo syntax
sprite(12).addCamera(member("scene").camera("FlightCam"), 5)

// JavaScript syntax
sprite(12).addCamera(member("scene").getPropRef("camera", i), 5);
// where i is the number index for the camera "FlightCam".
```

Voir aussi

cameraCount(), deleteCamera

addChild

Syntaxe

```
-- Lingo syntax
member(whichCastmember).node(whichParentNode).addChild(member(whichCastmember).node
(whichChildNode) {,#preserveWorld})

// JavaScript syntax
member(whichCastmember).node(whichParentNode).addChild(member(whichCastmember).node
(whichChildNode) {,symbol(preserveWorld)})
```

Description

Commande 3D ; ajoute un nœud à la liste des enfants d'un autre nœud et le supprime de la liste des enfants de son parent précédent.

L'équivalent de cette méthode consisterait à définir la propriété `parent` du nœud enfant sur le nœud parent.

Paramètres

`addMemberRef` Requis. Référence à l'acteur contenant le nœud à ajouter.

`addNodeRef` Requis. Référence au nœud à ajouter. Ce nœud peut être un modèle, un groupe, une caméra ou une lumière.

`symPreserveParentOrWorld` Facultatif. Référence à la caméra à ajouter à la liste des caméras de l'image-objet. Les valeurs possibles sont `#preserveWorld` et `#preserveParent`. Lorsque l'enfant est ajouté et que `#preserveParent` est spécifié, la transformation de l'enfant par rapport à son parent reste la même et l'enfant passe à cette transformation dans l'espace de son nouveau parent. La transformation de l'enfant dans l'univers est recalculée. Lorsque l'enfant est ajouté et que `#preserveWorld` est spécifié, la transformation de l'enfant dans l'univers reste la même et l'enfant ne passe pas à sa transformation dans l'espace de son nouveau parent. Sa transformation par rapport à son parent est recalculée.

Exemple

L'instruction suivante ajoute le modèle Pneu à la liste des enfants du modèle Voiture.

```
-- Lingo syntax
member("3D").model("Car").addChild(member("3D").model("Tire"))

// JavaScript syntax
member("3D").getProp("model" , i ).addChild(member("3D").getProp("model" , j));
// where i is the number index for model "Car" and j is the number index for model "Tire".
```

L'instruction suivante ajoute le modèle Oiseau à la liste des enfants de la caméra maCaméra et utilise l'argument #preserveWorld pour conserver la position du modèle Oiseau dans l'univers.

```
-- Lingo syntax
member("3D").camera("MyCamera").addChild(member("3D").model
("Bird"), #preserveWorld)

// JavaScript syntax
member("3D").getPropRef("camera",j).addChild(member("3D").getProp("model",i),symbol("prese
rveWorld"))
// where i the number index of the model "Bird" and j is the number index of the camera
"MyCamera"
```

Voir aussi

parent, addToWorld, removeFromWorld

addModifier

Syntaxe

```
-- Lingo syntax
member(whichCastmember).model(whichModel).addModifier(#modifierType)

// JavaScript syntax
member(whichCastmember).model(whichModel).addModifier(symbol(modifierType));
```

Description

Commande 3D de modèle ; ajoute un modificateur spécifié au modèle. Il n'existe aucune valeur par défaut pour cette commande.

Paramètres

symbolModType Requis. Symbole spécifiant le modificateur à ajouter. Les modificateurs possibles sont les suivants :

- #bonesPlayer
- #collision
- #inker
- #keyframePlayer
- #lod (niveau de détail)
- #meshDeform
- #sds
- #toon

Pour plus d'informations, consultez les entrées des différents modificateurs.

Exemple

L'instruction suivante ajoute le modificateur toon au modèle Boîte.

```
-- Lingo syntax
member("shapes").model("Box").addModifier(#toon)

// JavaScript syntax
member("shapes").getPropRef("model" , a ).addModifier(symbol("toon"));
// where a is the number index for the "Box" model.
```

Voir aussi

bonesPlayer (modificateur), collision (modificateur), inker (modificateur), keyframePlayer (modificateur), lod (modificateur), meshDeform (modificateur), sds (modificateur), toon (modificateur), getRendererServices(), removeModifier, modifier, modifier[], modifiers

addOverlay

Syntaxe

```
-- Lingo syntax
sprite(whichSprite).camera{(index)}.addOverlay(texture, locWithinSprite, rotation)
member(whichCastmember).camera(whichCamera).addOverlay(texture, locWithinSprite, rotation)

// JavaScript syntax
sprite(whichSprite).camera{(index)}.addOverlay(texture, locWithinSprite, rotation)
member(whichCastmember).camera(whichCamera).addOverlay(texture, locWithinSprite, rotation)
```

Description

Commande 3D de caméra ; ajoute un recouvrement à la fin d'une liste de recouvrements de la caméra.

Paramètres

texture Requis. Texture à appliquer au recouvrement.

locWithinSprite Requis. Emplacement 2D auquel le recouvrement s'affiche dans l'image-objet 3D. Cet emplacement est mesuré à partir du coin supérieur gauche de l'image-objet.

rotation Requis. Nombre entier spécifiant le nombre de degrés selon lequel la texture doit pivoter.

Exemple

La première ligne de l'instruction suivante crée la texture Rugueuse à partir de l'acteur Cèdre et l'enregistre dans la variable `t1`. La deuxième ligne applique la texture comme recouvrement à l'emplacement (220, 220) dans l'image-objet 5 avec une rotation de zéro degré. La dernière ligne de l'instruction applique la même texture comme recouvrement pour la caméra 1 de l'acteur Séquence, au point (20, 20). Une rotation de 45 degrés est appliquée à la texture.

```
-- Lingo syntax
t1 = member("Scene").newTexture("Rough", #fromCastMember,member("Cedar"))
sprite(5).camera.addOverlay(t1, point(220, 220), 0)
member("Scene").camera[1].addOverlay(t1, point(20, 20), 45)

// JavaScript syntax
t1 = member("Scene").newTexture("Rough", symbol("fromCastMember"),member("Cedar"));
sprite(5).camera.addOverlay(t1, point(220, 220), 0);
member("Scene").getPropRef("camera",1).addOverlay(t1, point(20, 20), 45);
```

Voir aussi

removeOverlay

addProp

Syntaxe

```
list.addProp(property, value)
addProp list, property, value
```

Description

Commande de liste de propriétés ; pour les listes de propriétés uniquement, cette commande ajoute une propriété spécifiée ainsi que sa valeur à une liste de propriétés.

Dans le cas d'une liste non triée, cette valeur est placée à la fin de la liste. Dans le cas d'une liste triée, cette valeur est placée dans l'ordre correct.

Si la propriété spécifiée existe déjà dans la liste, les syntaxes Lingo et JavaScript en créent une copie. Pour éviter la présence de propriétés en double, utilisez la commande `setaProp()` pour modifier la propriété correspondant à la nouvelle entrée.

Cette commande, utilisée avec une liste linéaire provoque une erreur.

Paramètres

`property` Requis. Propriété à ajouter à la liste.

`value` Requis. Valeur de la propriété à ajouter à la liste.

Exemple

L'instruction suivante ajoute la propriété `kayne` et sa valeur 3 à la liste de propriétés `bids`, qui contient [#avatar: 4, #soldes: 1]. Cette liste étant triée, la nouvelle entrée est placée en ordre alphabétique :

```
--Lingo
bids.addProp(#kayne, 3)

// Javascript
bids.addProp("kayne",3)
```

La liste qui en résulte est [#avatar: 4, #dupont: 3, #soldes: 1].

L'instruction suivante ajoute l'entrée `kayne` : 7 à la liste `bids`, qui contient désormais [#avatar: 4, #dupont: 3, #soldes: 1]. Cette liste contenant déjà la propriété `kayne`, Lingo en crée une copie :

```
--Lingo
bids.addProp(#kayne, 7)

// Javascript
bids.addProp("kayne",7)
```

La liste qui en résulte est [#avatar: 4, #dupont: 3, #dupont: 7, #soldes: 1].

addToWorld

Syntaxe

```
-- Lingo syntax
member(whichCastmember).model(whichModel).addToWorld()
member(whichCastmember).group(whichGroup).addToWorld()
member(whichCastmember).camera(whichCamera).addToWorld()
member(whichCastmember).light(whichLight).addToWorld()

// JavaScript syntax
member(whichCastmember).model(whichModel).addToWorld()
member(whichCastmember).group(whichGroup).addToWorld()
member(whichCastmember).camera(whichCamera).addToWorld()
memberwhichCastmember).light(whichLight).addToWorld()
```

Description

Commande 3D ; insère le modèle, le groupe, la caméra ou la lumière, dans l'univers 3D de l'acteur comme enfant du groupe Univers.

Lorsqu'un modèle, un groupe, une caméra ou une lumière, est créé ou cloné, il est automatiquement ajouté à l'univers. Utilisez la commande `removeFromWorld` pour extraire un modèle, un groupe, une caméra ou une lumière de l'univers 3D sans le supprimer. Utilisez la commande `isInWorld()` pour vérifier si un modèle, un groupe, une caméra ou une lumière a été ajouté à l'univers ou en a été extrait.

Paramètres

Aucune.

Exemple

L'instruction suivante ajoute le modèle gbCyl à l'univers 3D de l'acteur Séquence.

```
-- Lingo syntax
member("Scene").model("gbCyl").addToWorld()

// JavaScript syntax
member("Scene").getProp("model","gbCyl").addToWorld();
```

Voir aussi

`isInWorld()`, `removeFromWorld`

addVertex()

Syntaxe

```
-- Lingo syntax
memberObjRef.addVertex(indexToAddAt, pointToAddVertex {,[ horizControlLocV,
\ vertControlLocV ], [ horizControlLocH, vertControlLocV ]})

// JavaScript syntax
memberObjRef.addVertex(indexToAddAt, pointToAddVertex {,[ horizControlLocV, vertControlLocV ],
[ horizControlLocH, vertControlLocV ]});
```

Description

Commande de formes vectorielles ; ajoute un nouveau sommet à un acteur forme vectorielle à la position spécifiée.

Les positions horizontale et verticale du nouveau sommet sont déterminées par rapport à l'origine de l'acteur forme vectorielle.

Si vous utilisez les deux derniers paramètres facultatifs, vous pouvez spécifier la position des poignées de contrôle pour le sommet. La position de ces poignées de contrôle doit être donnée relativement à celle du sommet ; par conséquent, si aucune position n'est spécifiée, la poignée est placée sans décalage horizontal ou vertical (valeur 0, 0).

Paramètres

`indexToAddAt` Requis. Nombre entier spécifiant l'index au niveau duquel l'acteur doit être ajouté.

`pointToAddVertex` Requis. Point spécifiant la position à laquelle l'acteur doit être ajouté.

`horizControlLocH` Facultatif. Nombre entier spécifiant l'emplacement de la partie horizontale de la poignée de contrôle horizontale.

`horizControlLocV` Facultatif. Nombre entier spécifiant l'emplacement de la partie verticale de la poignée de contrôle horizontale.

`vertControlLocH` Facultatif. Nombre entier spécifiant l'emplacement de la partie horizontale de la poignée de contrôle verticale.

vertControlLocV Facultatif. Nombre entier spécifiant l'emplacement de la partie verticale de la poignée de contrôle verticale.

Exemple

La ligne suivante ajoute un point de sommet dans la forme vectorielle Archie entre les deux-points de sommet existants, à la position horizontale 25 et verticale 15 :

```
-- Lingo syntax
member("Archie").addVertex(2, point(25, 15))

// JavaScript syntax
member("Archie").addVertex(2, point(25, 15));
```

Voir aussi

vertexList, moveVertex(), deleteVertex(), originMode

alert()

Syntaxe

```
-- Lingo syntax
_player.alert(displayString)

// JavaScript syntax
_player.alert(displayString);
```

Description

Méthode de lecteur ; déclenche l'émission d'un signal sonore par le système et affiche une boîte de dialogue d'alerte contenant une chaîne spécifiée.

Le message d'alerte doit être une chaîne. Si vous souhaitez inclure une variable numérique dans une alerte, convertissez cette variable en chaîne avant de la transmettre à la méthode `alert()`.

Paramètres

displayString Requis. Chaîne représentant le texte affiché dans la boîte de dialogue d'alerte. Cette chaîne peut contenir jusqu'à 255 caractères.

Exemple

L'instruction suivante crée un message d'alerte indiquant qu'aucun lecteur de CD-ROM n'est connecté :

```
-- Lingo syntax
_player.alert("There is no CD-ROM drive connected.")

// JavaScript syntax
_player.alert("There is no CD-ROM drive connected.");
```

L'instruction suivante crée un message d'alerte indiquant qu'un fichier est introuvable :

```
-- Lingo syntax
_player.alert("The file" && QUOTE & filename & QUOTE && "was not found.")

// JavaScript syntax
_player.alert("The file \"" + filename + "\" was not found.");
```

Voir aussi

Lecteur

Alert()

Syntaxe

```
Alert( MUIObject,alertPropertiesList)
```

Description

Cette commande affiche une boîte de dialogue d'alerte créée à partir d'une instance de l'Xtra MUI. Cette fonctionnalité s'ajoute aux messages d'alerte simples créés par la commande alert.

L'Xtra MUI comprend des alertes modales. Ces messages d'alerte peuvent être déplaçables ou non déplaçables. Pour créer un tel message d'alerte, créez un objet Xtra MUI, puis envoyez la commande alert avec une liste incluant des définitions des propriétés alert en tant que second paramètre.

Le tableau ci-dessous présente les propriétés à spécifier, ainsi que leurs valeurs possibles.

Propriété	Valeurs possibles	Spécifie
#buttons	#Ok #OkCancel #AbortRetryIgnore #YesNoCancel #YesNo #RetryCancel	L'ensemble des boutons apparaissant dans le message d'alerte. Les boutons s'affichent suivant l'ordre de dénomination dans chaque symbole.
#default	Numéro ordinal du bouton qui devient le bouton par défaut. Par exemple, si les boutons du message d'alerte sont OK et Annuler, 2 spécifie le bouton Annuler. Pour ne pas spécifier de bouton par défaut, indiquez 0.	Dont le bouton est le bouton par défaut.
#icon	#stop #note #caution #question #error	Le type d'icône apparaissant dans le message d'alerte. Pour ne pas spécifier d'icône, indiquez 0.
#message	chaîne	Message s'affichant dans le message d'alerte.
#movable	TRUE FALSE	Indique si le message d'alerte est déplaçable.
#title	chaîne	Titre du message d'alerte

Vous devez spécifier explicitement chaque propriété de l'alerte. L'Xtra MUI n'offre pas de liste de propriétés de message d'alerte par défaut. Lingo renvoie une valeur pour le bouton cliqué par l'utilisateur.

La taille d'un message d'alerte peut s'approcher de celle de l'écran. Vous pouvez donc afficher une description d'une longueur importante, si vous le souhaitez.

Les instructions suivantes permettent la création et l'affichage d'une boîte de dialogue d'alerte.

- La première instruction crée une instance d'Xtra MUI, qui est l'objet utilisé en tant que boîte de dialogue.
- La seconde instruction établit une liste des propriétés du message d'alerte.
- Les dernières instructions utilisent la commande Alert pour afficher le message d'alerte, et indiquent les boutons cliqués par l'utilisateur.

Exemple

```
-- Lingo syntax
set alertObj = new(xtra "MUI")
set alertInitList = [ #buttons : #YesNo, ¬
#title : "Alert Test", #message : "This shows Yes and No buttons",\
#movable : TRUE]
if objectP ( alertObj ) then
      set result = Alert( alertObj, alertInitList )
      case result of
      1 : -- the user clicked yes
```

```
      2 : -- the user clicked no
      otherwise : -- something is seriously wrong
   end case
end if
```

append

Syntaxe

```
list.append(value)
append list, value
```

Description

Commande de liste ; ajoute, pour les listes linéaires uniquement, la valeur spécifiée à la fin d'une liste linéaire. Cette commande diffère de la commande `add` qui insère une valeur à l'endroit approprié d'une liste triée.

Cette commande, utilisée avec une liste de propriétés, provoque une erreur de script.

Paramètres

`value` Requis. Valeur à ajouter à la fin de la liste linéaire.

Exemple

L'instruction suivante ajoute la valeur 2 à la fin de la liste triée devis, qui contient les valeurs [1, 3, 4], même si cet emplacement ne correspond pas à l'ordre trié de la liste :

```
--Lingo
set bids = [1, 3, 4]
bids.append(2)

// Javascript
bids = list(1, 3, 4)
bids.append(2)
```

La valeur résultante de devis est donc [1, 3, 4, 2].

Voir aussi

`add (texture 3D)`, `sort`

appMinimize()

Syntaxe

```
-- Lingo syntax
_player.appMinimize()

// JavaScript syntax
_player.appMinimize();
```

Description

Méthode de lecteur ; sous Microsoft Windows, entraîne la réduction d'une projection sous la forme d'une icône dans la barre des tâches. Sur Mac, cette méthode entraîne le masquage d'une projection.

Sur Mac, vous pouvez rouvrir une projection masquée à partir du menu de l'application Mac.

Cette méthode se révèle utile pour les projections et animations dans une fenêtre lues sans barre de titre.

Paramètres

Aucune.

Exemple

```
--Lingo syntax
on mouseUp me
    _player.appMinimize()
end


// JavaScript syntax
function mouseUp() {
     _player.appMinimize();
}
```

Voir aussi

Lecteur

atan()

Syntaxe

```
-- Lingo syntax
(number).atan
atan (number)

// JavaScript syntax
Math.atan(number);
```

Description

Fonction mathématique (Lingo uniquement) ; calcule l'arctangente qui correspond à l'angle dont la tangente est un nombre spécifié. Le résultat est une valeur en radians comprise entre pi/2 et +pi/2.

En syntaxe JavaScript, utilisez la fonction `atan()` d'un objet mathématique.

Paramètres

Aucune.

Exemple

L'instruction suivante donne l'arctangente de 1 :

```
(1).atan
```

Le résultat, à quatre chiffres après la virgule, est égal à 0,7854, soit approximativement pi/4.

La plupart des fonctions trigonométriques utilisant les radians, vous devrez peut-être convertir les degrés en radians.

Le gestionnaire suivant vous permet d'effectuer les conversions de degrés en radians :

```
-- Lingo syntax
on DegreesToRads degreeValue
    return degreeValue * PI/180
end
```

```
// JavaScript syntax
function DegreesToRads(degreeValue) {
    return degreeValue * PI/180
}
```

Le gestionnaire affiche le résultat de la conversion de 30 degrés en radians dans la fenêtre Messages :

```
put DegreesToRads(30)
-- 0.5236
```

Voir aussi

cos(), PI, sin()

beep()

Syntaxe

```
-- Lingo syntax
_sound.beep({intBeepCount})

// JavaScript syntax
_sound.beep({intBeepCount});
```

Description

Méthode audio ; entraîne l'émission par l'ordinateur d'un signal sonore qui se répète autant de fois que spécifié par *intBeepCount*. Si vous n'avez pas spécifié *intBeepCount*, le signal sonore ne retentit qu'une seule fois.

- Sous Windows, le signal sonore est le son désigné dans la boîte de dialogue des propriétés sonores.
- Pour le Mac, le signal sonore est le son sélectionné dans la section Alertes du tableau de bord Moniteurs et son.

Si le volume a la valeur 0, le signal sonore est remplacé par un clignotement de la barre de menus.

Paramètres

intBeepCount Facultatif. Nombre entier spécifiant le nombre de fois où l'ordinateur doit émettre un signal sonore.

Exemple

```
-- Lingo syntax
on mouseUp me
    _sound.beep(1)
end mouseUp

// JavaScript syntax
function mouseUp() {
    _sound.beep(1);
}
```

Voir aussi

Son

beginRecording()

Syntaxe

```
-- Lingo syntax
_movie.beginRecording()

// JavaScript syntax
_movie.beginRecording();
```

Description

Méthode d'animation ; démarre une session de création du scénario.

Lorsque vous appelez la méthode `beginRecording()`, la tête de lecture avance automatiquement sur une image et démarre l'enregistrement dans cette dernière. Pour éviter ce comportement et commencer l'enregistrement dans l'image dans laquelle la méthode `beginRecording()` est appelée, insérez une instruction telle que `_movie.go(_movie.frame - 1)` entre les appels des méthodes `beginRecording()` et `endRecording()`.

Vous ne pouvez procéder qu'à une seule session de mise à jour du scénario à la fois dans une animation.

Chaque appel de la méthode `beginRecording()` doit être suivi d'un appel de la méthode `endRecording()` qui met fin à la session de création du scénario.

Paramètres

Aucune.

Exemple

Lorsque vous l'utilisez dans le gestionnaire suivant, le mot-clé `beginRecording` démarre une session de création du scénario qui anime l'acteur Balle en l'affectant à la piste d'image-objet 20, puis en déplaçant l'image-objet horizontalement et verticalement sur une série d'images. Le nombre d'images est déterminé par l'argument `numberOfFrames`.

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    horizontal = 0
    vertical = 100
    repeat with i = 1 to numberOfFrames
        _movie.go(i)
        sprite(20).member = member("Ball")
        sprite(20).locH = horizontal
        sprite(20).locV = vertical
        sprite(20).foreColor = 255
        horizontal = horizontal + 3
        vertical = vertical + 2
        _movie.updateFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    var horizontal = 0;
    var vertical = 100;
    for (var i = 1; i <= numberOfFrames; i++) {
        _movie.go(1);
        sprite(20).member = member("Ball")
```

```
        sprite(20).locH = horizontal;
        sprite(20).locV = vertical;
        sprite(20).foreColor = 255;
        horizontal = horizontal + 3;
        vertical = vertical + 2;
        _movie.updateFrame();
    }
    _movie.endRecording();
}
```

Voir aussi

endRecording(), Animation

bitAnd()

Syntaxe

```
bitAnd(integer1, integer2)
```

Description

Fonction (Lingo uniquement) ; convertit les deux entiers spécifiés en nombres binaires 32 bits et renvoie un nombre binaire dont les chiffres sont des 1 dans les positions dans lesquelles les deux chiffres comportaient des 1 et des 0 pour toutes les autres positions. Le résultat est un nouveau nombre binaire, que Lingo affiche sous la forme d'un entier de base 10.

nombre entier	**Nombre binaire (abrégé)**
6	00110
7	00111
Résultat	
6	00110

En syntaxe JavaScript, utilisez l'opérateur au niveau du bit « & ».

Paramètres

integer1 Requis. Premier nombre entier.

integer2 Requis. Second nombre entier.

Exemple

L'instruction suivante compare les versions binaires des entiers 6 et 7 et renvoie le résultat sous la forme d'un entier :

```
--Lingo
put bitAnd(6, 7)
-- 6

// Javascript
trace ( 6 & 7)
// 6
```

Voir aussi

bitNot(), bitOr(), bitXor()

bitNot()

Syntaxe

```
(integer).bitNot
bitNot(integer)
```

Description

Fonction (Lingo uniquement) ; convertit l'entier spécifié en nombre binaire 32 bits et inverse la valeur de chaque chiffre binaire en remplaçant les 1 par des 0 et les 0 par des 1. Le résultat est un nouveau nombre binaire que Lingo affiche sous la forme d'un entier de base 10.

nombre entier	**Nombre binaire**
1	00000000000000000000000000000001
Résultat	
-2	11111111111111111111111111111110

En syntaxe JavaScript, utilisez l'opérateur au niveau du bit « ~ ».

Paramètres

Aucune.

Exemple

L'instruction suivante inverse la représentation binaire de l'entier 1 et renvoie un nouveau nombre.

```
--Lingo
put (1).bitNot
-- -2

// Javascript
trace(~1)
// -2
```

Voir aussi

bitAnd(), bitOr(), bitXor()

bitOr()

Syntaxe

```
bitOr(integer1, integer2)
```

Description

Fonction (Lingo uniquement) ; convertit les deux entiers spécifiés en nombres binaires 32 bits et renvoie un nombre binaire dont les chiffres sont des 1 dans les positions dans lesquelles les deux chiffres comportaient des 1 et des 0 pour toutes les autres positions. Le résultat est un nouveau nombre binaire, que Lingo affiche sous la forme d'un entier de base 10.

nombre entier	Nombre binaire (abrégé)
5	0101
6	0110
Résultat	
7	0111

En syntaxe JavaScript, utilisez l'opérateur au niveau du bit « | ».

Paramètres

integer1 Requis. Premier nombre entier.

integer2 Requis. Second nombre entier.

Exemple

L'instruction suivante compare les versions binaires des entiers 5 et 6 et renvoie le résultat sous la forme d'un entier :

```
-- Lingo
put bitOr(5, 6)
-- 7

// Javascript
trace(5|6)
// 7
```

Voir aussi

bitNot(), bitAnd(), bitXor()

bitXor()

Syntaxe

```
bitXor(integer1, integer2)
```

Description

Fonction ; convertit les deux entiers spécifiés en nombres binaires 32 bits et renvoie un nombre binaire dont les chiffres sont des 1 dans les positions dans lesquelles les chiffres de nombres donnés ne correspondaient pas, et des 0 pour les positions dans lesquelles les chiffres étaient les mêmes. Le résultat est un nouveau nombre binaire, que Lingo affiche sous la forme d'un entier de base 10.

nombre entier	Nombre binaire (abrégé)
5	0101
6	0110
Résultat	
3	0011

En syntaxe JavaScript, utilisez l'opérateur au niveau du bit « ^ ».

Paramètres

integer1 Requis. Premier nombre entier.

integer2 Requis. Second nombre entier.

Exemple

L'instruction suivante compare les versions binaires des entiers 5 et 6 et renvoie le résultat sous la forme d'un entier :

```
-- Lingo
put bitXor(5, 6)
-- 3

// Javascript
trace(5^6)
// 3
```

Voir aussi

bitNot(), bitOr(), bitAnd()

breakLoop()

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.breakLoop()

// JavaScript syntax
soundChannelObjRef.breakLoop();
```

Description

Méthode de piste audio ; interrompt la lecture en boucle du son mis en boucle dans la piste *soundChannelObjRef* et entraîne sa lecture jusqu'à la limite endTime.

Si aucun son n'est actuellement en boucle, cette méthode n'a pas d'effet.

Paramètres

Aucune.

Exemple

Le gestionnaire suivant arrête la lecture du son mis en boucle dans la piste audio 2 et entraîne sa lecture jusqu'à la fin.

```
-- Lingo syntax
on continueBackgroundMusic
    sound(2).breakLoop()
end

// JavaScript syntax
function continueBackgroundMusic() {
    sound(2).breakLoop();
}
```

Voir aussi

endTime, Piste audio

browserName()

Syntaxe

```
browserName pathName
browserName()
browserName(#enabled, trueOrFalse)
```

Description

Propriété système, commande et fonction ; spécifie le chemin ou l'emplacement du navigateur web. Vous pouvez utiliser l'Xtra FileIO pour afficher une boîte de dialogue permettant à l'utilisateur de spécifier un navigateur de son choix. La méthode `displayOpen()` de l'Xtra FileIO est utile pour afficher une boîte de dialogue d'ouverture.

La forme `browserName()` renvoie le nom du navigateur actuellement spécifié. Si vous placez un nom de chemin, tel que celui trouvé au moyen de l'Xtra FileIO, en tant qu'argument dans la forme `browserName(fullPathToApplication)`, vous pouvez définir la propriété. La forme `browserName(#enabled,`*`trueOrFalse`*`)` détermine si le navigateur spécifié est automatiquement lancé par la commande `goToNetPage`.

Cette commande est utile uniquement lors de la lecture dans une projection ou dans Director et n'a aucun effet pour la lecture dans un navigateur web.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante indique l'emplacement du navigateur Netscape® :

```
browserName "My Disk:My Folder:Netscape"
```

L'instruction suivante affiche le nom du navigateur dans une fenêtre Messages :

```
put browserName()
```

build()

Syntaxe

```
-- Lingo syntax
member(whichCastmember).modelResource(whichModelResource).build()

// JavaScript syntax
member(whichCastmember).modelResource(whichModelResource).build();
```

Description

Commande 3D de maille ; construit une maille. Cette commande n'est utilisée qu'avec les ressources de modèle de type `#mesh`.

Vous devrez utiliser la commande `build()` pour la construction initiale de la maille, après avoir modifié l'une de ses propriétés `face` et avoir utilisé la commande `generateNormals()`.

Paramètres

Aucune.

Exemple

Cet exemple crée une simple ressource de modèle de type #mesh, en spécifie les propriétés, puis l'utilise pour créer un nouveau modèle. Le processus est décrit dans les explications accompagnant l'exemple suivant :

La ligne 1 crée une maille nommée Plan, qui consiste en une face, trois sommets et un maximum de trois couleurs. Le nombre de normales et de coordonnées de textures n'est pas défini. Les normales sont créées par la commande `generateNormals`.

La ligne 2 définit les vecteurs qui sont utilisés comme sommets de Plan.

La ligne 3 affecte les vecteurs aux sommets de la première face de Plan.

La ligne 4 définit les trois couleurs autorisées par la commande newMesh.

La ligne 5 affecte les couleurs à la première face de Plan. La troisième couleur de la liste est appliquée au premier sommet de Plan, la deuxième couleur au deuxième sommet, et la première couleur au troisième sommet. Les couleurs sont étalées sur la première face de Plan en dégradés.

La ligne 6 crée les normales de Plan avec la commande generateNormals().

La ligne 7 appelle la commande build() pour construire la maille.

```
-- Lingo syntax
nm = member("Shapes").newMesh("Plane",1,3,0,3,0)
nm.vertexList = [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)]
nm.face[1].vertices = [1,2,3]
nm.colorList = [rgb(255,255,0), rgb(0, 255, 0), rgb(0,0,255)]
nm.face[1].colors = [3,2,1]
nm.generateNormals(#smooth)
nm.build()
nm = member("Shapes").newModel("TriModel", nm)

// JavaScript syntax
nm = member("Shapes").newMesh("Plane",1,3,0,3,0);
nm.vertexList = [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)];
nm.face[1].vertices = [1,2,3];
nm.colorList = [rgb(255,255,0), rgb(0, 255, 0), rgb(0,0,255)];
nm.face[1].colors = [3,2,1];
nm.generateNormals(#smooth);
nm.build();
nm = member("Shapes").newModel("TriModel", nm);
```

Voir aussi

`generateNormals()`, `newMesh`, `face[ ]`

cacheDocVerify()

Syntaxe

```
-- Lingo syntax
cacheDocVerify #setting
cacheDocVerify()

// JavaScript syntax
cacheDocVerify symbol(setting);
cacheDocVerify();
```

Description

Fonction ; définit la fréquence de rafraîchissement du contenu d'une page web sur la base des informations contenues dans la mémoire cache de la projection.

La forme `cacheDocVerify()` renvoie le paramétrage en cours de la mémoire cache.

La fonction `cacheDocVerify` n'est valide que pour les animations exécutées dans Director ou en tant que projections. Elle n'est pas valide pour les animations comportant du contenu Adobe® Shockwave®, celles-ci utilisant les paramètres réseau du navigateur dans lequel elles sont exécutées.

```
-- Lingo syntax
on resetCache
    current = cacheDocVerify()
    if current = #once then
        alert "Turning cache verification on"
        cacheDocVerify #always
    end if
end

// JavaScript syntax
function resetCache() {
    current = cacheDocVerify();
    if (current == symbol("once")) {
        alert("Turning cache verification on");
        cacheDocVerify(symbol("always"))
    }
}
```

Paramètres

`cacheSetting` Facultatif. Symbole spécifiant la fréquence de rafraîchissement du contenu d'une page Web. Les valeurs possibles sont `#once` (une seule fois, valeur par défaut) et `#always` (autant de fois que nécessaire). La valeur `#once` indique à une animation de télécharger une fois un fichier depuis Internet, puis de l'utiliser depuis la mémoire cache sans en rechercher une version actualisée sur Internet. La valeur `#always` indique à une animation d'essayer de télécharger une version actualisée du fichier chaque fois qu'elle appelle une URL.

Voir aussi

`cacheSize()`, `clearCache`

cacheSize()

Syntaxe

```
-- Lingo syntax
cacheSize Size
cacheSize()

// JavaScript syntax
cacheSize(Size);
cacheSize();
```

Description

Fonction et commande ; définit la taille de la mémoire cache de Director.

La fonction `cacheSize` n'est valide que pour les animations exécutées sous Director ou en tant que projections. Elle n'est pas valide pour les animations comportant du contenu Shockwave, celles-ci utilisant les paramètres réseau du navigateur dans lequel elles sont exécutées.

Paramètres

`newCacheSize` Facultatif. Nombre entier spécifiant la taille du cache en kilo-octets.

Exemple

Le gestionnaire suivant vérifie si le cache du navigateur est défini sur une valeur inférieure à 1 Mo. Le cas échéant, il affiche un message d'alerte et définit la taille de la mémoire cache à 1 Mo :

```
-- Lingo syntax
on checkCache if
    cacheSize()<1000 then
        alert "increasing cache to 1MB"
        cacheSize 1000
    end if
end

// JavaScript syntax
function checkCache() {
    if (cacheSize() < 1000) {
        alert("increasing cache to 1MB");
        cacheSize(1000);
    }
}
```

Voir aussi

`cacheDocVerify()`, `clearCache`

call

Syntaxe

```
call #handlerName, script, {args...}
call (#handlerName, scriptInstance, {args...})
```

Description

Commande ; envoie un message appelant un gestionnaire dans un script spécifié ou dans une liste de scripts.

La commande `call` peut utiliser une variable comme nom du gestionnaire. Les messages transmis à l'aide de `call` ne sont pas transmis aux autres scripts liés à l'image-objet, aux scripts d'acteur, aux scripts d'image ni aux scripts d'animation.

Paramètres

`symHandlerName` Requis. Symbole spécifiant le gestionnaire à activer.

`scriptInstance` Requis. Référence au script ou à la liste de scripts contenant le gestionnaire. Si *`scriptInstance`* est une instance de script unique, un message d'alerte est envoyé si le gestionnaire n'est pas défini dans le script ancêtre du script. Si **`scriptInstance`** est une liste d'instances de script, le message est envoyé à chaque élément de la liste tour à tour ; aucun message d'alerte n'est généré si le gestionnaire n'est pas défini dans le script ancêtre.

`args` Facultatif. Paramètres facultatifs à transmettre au gestionnaire.

Exemple

Le gestionnaire suivant envoie le message augmenterLeCompteur au premier script de comportement lié à l'image-objet 1 :

```
-- Lingo syntax
on mouseDown me
    -- get the reference to the first behavior of sprite 1
    set xref = getAt (the scriptInstanceList of sprite 1,1)
    -- run the bumpCounter handler in the referenced script,
    -- with a parameter
    call (#bumpCounter, xref, 2)
end

// JavaScript syntax
function mouseDown() {
    // get the reference to the first behavior of sprite 1
    xref = getAt(sprite(1).script(1));
    // run the bumpCounter handler in the referenced script
    call(symbol("bumpcounter"), xref, 2);
}
```

L'exemple suivant illustre la façon dont une instruction `call` peut appeler les gestionnaires d'un comportement ou d'un script parent et ceux de son ancêtre.

- Le script suivant est le script parent :

```
-- Lingo syntax
-- script Man
property ancestor

on new me
    set ancestor = new(script "Animal", 2)
    return me
end
on run me, newTool
    put "Man running with "&the legCount of me&" legs"
end
```

- Le script suivant est le script ancêtre :

```
-- script Animal
property legCount

on new me, newLegCount
    set legCount = newLegCount
    return me
end
on run me
    put "Animal running with "& legCount &" legs"
end
on walk me
    put "Animal walking with "& legCount &" legs"
end
```

- Les instructions suivantes utilisent le script parent et le script ancêtre.

L'instruction suivante crée une instance du script parent :

```
set m = new(script "man")
```

L'instruction suivante fait marcher l'homme :

```
call #walk, m
-- "Animal walking with 2 legs"
```

L'instruction suivante fait courir l'homme :

```
set msg = #run
call msg, m
-- "Man running with 2 legs and rock"
```

L'instruction suivante crée une seconde instance du script parent :

```
set m2 = new(script "man")
```

L'instruction suivante envoie un message aux deux instances du script parent :

```
call msg, [m, m2]
```

```
-- "Man running with 2 legs "
-- "Man running with 2 legs "
```

callAncestor

Syntaxe

```
callAncestor handlerName, script, {args...}
```

Description

Commande ; envoie un message au script ancêtre d'un objet enfant.

Les ancêtres peuvent, à leur tour, avoir leurs propres ancêtres.

Lorsque vous utilisez `callAncestor`, le nom du gestionnaire peut être une variable et vous pouvez explicitement contourner les gestionnaires du script principal et accéder directement au script ancêtre.

Paramètres

`symHandlerName` Requis. Symbole spécifiant le gestionnaire à activer.

`scriptInstance` Requis. Référence au script ou à la liste de scripts contenant le gestionnaire. Si *`scriptInstance`* est une instance de script unique, un message d'alerte est envoyé si le gestionnaire n'est pas défini dans le script ancêtre du script. Si *`scriptInstance`* est une liste d'instances de script, le message est envoyé à chaque élément de la liste tour à tour ; aucun message d'alerte n'est généré si le gestionnaire n'est pas défini dans le script ancêtre.

`args` Facultatif. Paramètres facultatifs à transmettre au gestionnaire.

Exemple

L'exemple suivant présente la façon dont une instruction `callAncestor` peut appeler des gestionnaires dans l'ancêtre d'un comportement ou d'un script parent.

- Le script suivant est le script parent :

```
-- script "man"
property ancestor

on new me, newTool
    set ancestor = new(script "Animal", 2)
    return me
end
```

```
on run me
    put "Man running with "&the legCount of me&"legs"
end
```

- Le script suivant est le script ancêtre :

```
-- script "animal"
property legCount

on new me, newLegCount
    set legCount = newLegCount
    return me
end
on run me
    put "Animal running with "& legCount &" legs"
end
on walk me
    put "Animal walking with "& legCount &" legs"
end
```

- Les instructions suivantes utilisent le script parent et le script ancêtre.

  L'instruction suivante crée une instance du script parent :

```
set m = new(script "man")
```

  L'instruction suivante fait marcher l'homme :

```
call #walk, m
-- "Animal walking with 2 legs"
```

  L'instruction suivante fait courir l'homme :

```
set msg = #run
callAncestor msg, m
-- "Animal running with 2 legs"
```

  L'instruction suivante crée une seconde instance du script parent :

```
set m2 = new(script "man")
```

  L'instruction suivante envoie un message au script ancêtre des deux hommes :

```
callAncestor #run,[m,m2]
-- "Animal running with 2 legs"
-- "Animal running with 2 legs"
```

Voir aussi

ancestor, new()

callFrame()

Syntaxe

```
-- Lingo syntax
spriteObjRef.callFrame(flashFrameNameOrNum)

// JavaScript syntax
spriteObjRef.callFrame(flashFrameNameOrNum);
```

Description

Commande ; utilisée pour appeler une série d'actions résidant dans une image d'une image-objet d'animation Flash®.

Cette commande transmet un message au moteur ActionScript de Flash® et déclenche les actions à exécuter dans l'animation Flash.

Paramètres

flashFrameNameOrNum Requis. Chaîne ou nombre spécifiant le nom ou le numéro de l'image à appeler.

Exemple

Cette instruction Lingo lance l'exécution des actions associées à l'image 10 de l'animation Flash dans l'image-objet 1 :

```
-- Lingo syntax
sprite(1).callFrame(10)

// JavaScript syntax
sprite(1).callFrame(10);
```

camera()

Syntaxe

```
member(whichCastMember).camera(whichCamera)
member(whichCastMember).camera[index]
member(whichCastMember).camera(whichCamera).whichCameraProperty
member(whichCastMember).camera[index].whichCameraProperty
sprite(whichSprite).camera{(index)}
sprite(whichSprite).camera{(index)}.whichCameraProperty
```

Description

Elément 3D ; objet à une position de vecteur à partir de laquelle l'univers 3D est observé.

Chaque image-objet possède une liste de caméras. Les vues des différentes caméras de la liste s'affichent au-dessus de celles des caméras en position *index* inférieures. Vous pouvez définir la propriété `rect (caméra)` de chaque caméra afin d'afficher plusieurs vues au sein de l'image-objet.

Les caméras sont enregistrées dans la palette des caméras de l'acteur. Utilisez les commandes `newCamera` et `deleteCamera` pour créer et supprimer les caméras d'un acteur 3D.

La propriété `camera` d'une image-objet est la première caméra de la liste des caméras de l'image-objet. La caméra référencée par `sprite(whichSprite).camera` est la même que `sprite(whichSprite).camera(1)`. Utilisez les commandes `addCamera` et `deleteCamera` pour créer la liste des caméras d'une image-objet 3D.

Exemple

L'instruction suivante affecte à l'image-objet 1 la caméra camArbre de l'acteur Pique-nique.

```
sprite(1).camera = member("Picnic").camera("TreeCam")
```

L'instruction suivante affecte à l'image-objet 1 la caméra 2 de l'acteur Picnic.

```
sprite(1).camera = member("Picnic").camera[2]
```

Voir aussi

`bevelDepth`, `overlay`, `modelUnderLoc`, `spriteSpaceToWorldSpace`, `fog`, `clearAtRender`

cameraCount()

Syntaxe

```
-- Lingo syntax
sprite(whichSprite).cameraCount()

// JavaScript syntax
sprite(whichSprite).cameraCount();
```

Description

Commande 3D ; renvoie le nombre d'éléments de la liste des caméras de l'image-objet.

Paramètres

Aucune.

Exemple

L'instruction suivante indique que l'image-objet 5 contient trois caméras.

```
-- Lingo syntax
put sprite(5).cameraCount()
-- 3

// JavaScript syntax
put(sprite(5).cameraCount());
// 3
```

Voir aussi

addCamera, deleteCamera

cancelIdleLoad()

Syntaxe

```
-- Lingo syntax
_movie.cancelIdleLoad(intLoadTag)

// JavaScript syntax
_movie.cancelIdleLoad(intLoadTag);
```

Description

Méthode d'animation ; annule le chargement de tous les acteurs portant la balise de chargement spécifiée.

Paramètres

intLoadTag Requis. Nombre entier spécifiant un groupe d'acteurs en attente de chargement pendant les périodes d'inactivité de l'ordinateur.

Exemple

L'instruction suivante annule le chargement des acteurs portant la balise de chargement en période d'inactivité numéro 20 :

```
-- Lingo syntax
_movie.cancelIdleLoad(20)
```

```
// JavaScript syntax
_movie.cancelIdleLoad(20);
```

Voir aussi

idleLoadTag, Animation

castLib()

Syntaxe

```
-- Lingo syntax
castLib(castNameOrNum)

// JavaScript syntax
castLib(castNameOrNum);
```

Description

Fonction de niveau supérieur ; renvoie une référence à une bibliothèque de distribution spécifiée.

La bibliothèque de distribution par défaut est la bibliothèque numéro 1. Pour spécifier un acteur dans une bibliothèque de distribution autre que la distribution numéro 1, définissez `castLib()` pour spécifier l'autre bibliothèque de distribution.

Paramètres

`castNameOrNum` Requis. Chaîne spécifiant le nom de la bibliothèque de distribution ou nombre entier indiquant le numéro de la bibliothèque de distribution.

Exemple

L'instruction suivante définit la variable `parts` sur la seconde bibliothèque de distribution :

```
-- Lingo syntax
parts = castLib(2)

// JavaScript syntax
var parts = castLib(2);
```

Voir aussi

Bibliothèque de distribution, castLibNum

channel() (niveau supérieur)

Syntaxe

```
-- Lingo syntax
channel(soundChannelNameOrNum)

// JavaScript syntax
channel(soundChannelNameOrNum);
```

Description

Fonction de niveau supérieur ; renvoie une référence à un objet piste audio.

Paramètres

soundChannelNameOrNum Requis. Chaîne spécifiant le nom d'une piste audio ou nombre entier indiquant la position d'index d'une piste audio.

Exemple

L'instruction suivante définit la variable `newChannel` sur la piste audio 9 :

```
-- Lingo syntax
newChannel = channel(9)

// JavaScript syntax
var newChannel = channel(9);
```

Voir aussi

Piste audio

channel() (son)

Syntaxe

```
-- Lingo syntax
_sound.channel(intChannelNum)

// JavaScript syntax
_sound.channel(intChannelNum);
```

Description

Méthode audio ; renvoie une référence à une piste audio spécifiée.

Cette méthode a la même fonction que la méthode de niveau supérieur `sound()`.

Paramètres

intChannelNum Requis. Nombre entier spécifiant la piste audio à référencer.

Exemple

L'instruction suivante définit la variable `myChannel` sur la piste audio 2 :

```
-- Lingo syntax
myChannel = _sound.channel(2)

// JavaScript syntax
var myChannel = _sound.channel(2);
```

Voir aussi

Son, sound(), Piste audio

chapterCount()

Syntaxe

```
-- Lingo syntax
dvdObjRef.chapterCount({intTitle})

// JavaScript syntax
dvdObjRef.chapterCount({intTitle});
```

Description

Méthode de DVD ; indique le nombre de chapitres disponibles dans un titre.

Paramètres

intTitle Facultatif. Nombre entier indiquant le titre contenant les chapitres à comptabiliser. Si ce paramètre est omis, `chapterCount()` renvoie le nombre de chapitres disponibles dans le titre en cours.

Exemple

L'instruction suivante renvoie le nombre de chapitres du titre en cours :

```
-- Lingo syntax
trace (member(1).chapterCount)-- 17

// JavaScript syntax
trace (member(1). chapterCount);// 17
```

Voir aussi

chapterCount, DVD

charPosToLoc()

Syntaxe

```
--Lingo syntax
memberObjRef.charPosToLoc(nthCharacter)

// JavaScript syntax
memberObjRef.charPosToLoc(nthCharacter);
```

Description

Fonction de champ ; renvoie le point de l'acteur champ entier (et non uniquement la partie affichée sur la scène) qui est le plus proche d'un caractère spécifié. Elle est utile pour déterminer l'emplacement précis de caractères individuels.

Les valeurs de *charPosToLoc* sont exprimées en pixels et commencent à partir du coin supérieur gauche de l'acteur champ. Le paramètre *nthCharacter* est de 1 pour le premier caractère du champ, 2 pour le deuxième caractère, etc.

Paramètres

nthCharacter Requis. Caractère à tester.

Exemple

L'instruction suivante détermine le point au niveau duquel apparaît le cinquantième caractère de l'acteur champ Titre et affecte le résultat à la variable `location` :

```
-- Lingo syntax
location = member("Headline").charPosToLoc(50)

// JavaScript syntax
var location = member("Headline").charPosToLoc(50);
```

chars()

Syntaxe

```
chars(stringExpression, firstCharacter, lastCharacter)
```

Description

Fonction (Lingo uniquement) ; identifie une sous-chaîne de caractères dans une expression.

Les expressions *firstCharacter* et *lastCharacter* doivent spécifier une position dans la chaîne.

Si les expressions *firstCharacter* et *lastCharacter* sont égales, la chaîne ne renvoie qu'un seul caractère. Si la valeur *lastCharacter* est supérieure à la longueur de la chaîne, seule une sous-chaîne allant jusqu'à la longueur de la chaîne est identifiée. Si *lastCharacter* est placé avant *firstCharacter*, cette fonction renvoie la valeur EMPTY.

Vous pouvez voir un exemple d'utilisation de `chars()` dans une animation en consultant l'animation Text du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

En syntaxe JavaScript, utilisez la fonction `substr()` d'un objet chaîne.

Paramètres

stringExpression Requis. Chaîne spécifiant l'expression depuis laquelle une sous-chaîne est renvoyée.

firstCharacter Requis. Nombre entier spécifiant le point de départ de la sous-chaîne.

lastCharacter Requis. Nombre entier spécifiant le point de fin de la sous-chaîne.

Exemple

L'instruction suivante identifie le deuxième caractère du mot *Adobe* :

```
put chars("Adobe", 2, 2)
-- "d"
```

L'instruction suivante identifie les caractères compris entre le deuxième et le cinquième caractère du mot *Adobe*:

```
put chars("Adobe", 2, 5)
-- "dobe"
```

L'instruction suivante tente d'identifier les caractères compris entre le sixième et le vingtième caractère du mot Adobe. Puisque ce mot ne contient que 10 caractères, le résultat ne contient que les caractères compris ente le sixième et le dixième caractère.

```
put chars ("Adobe", 2, 20)
-- "dobe"
```

Voir aussi

char...of, length(), offset() (fonction de chaîne), number (caractères)

charToNum()

Syntaxe

```
(stringExpression).charToNum
charToNum(stringExpression)
```

Description

Fonction (Lingo uniquement) ; renvoie le code ASCII correspondant au premier caractère d'une expression.

La fonction `charToNum()` se révèle particulièrement utile pour tester la valeur ASCII des caractères créés avec des combinaisons de touches, telles que la touche Ctrl et une autre touche alphanumérique.

Director ne fait aucune distinction entre les majuscules et les minuscules si vous utilisez l'opérateur de comparaison égal (=) ; par exemple, l'instruction `put ("M" = "m")` donne le résultat 1 ou `TRUE`.

Pour éviter tout problème, utilisez `charToNum()` pour obtenir le code ASCII d'un caractère, puis utilisez le code ASCII pour faire référence à ce caractère.

En syntaxe JavaScript, utilisez la fonction `charCodeAt()` d'un objet chaîne.

Paramètres

`stringExpression` Requis. Chaîne spécifiant l'expression à tester.

Exemple

L'instruction suivante affiche le code ASCII de la lettre A :

```
put ("A").charToNum
-- 65
```

La comparaison suivante détermine si la lettre saisie est un A majuscule, puis passe à une séquence correcte ou incorrecte du scénario :

```
-- Lingo syntax
on CheckKeyHit theKey
    if (theKey).charToNum = 65 then
        go "Correct Answer"
    else
        go "Wrong Answer"
    end if
end

// JavaScript syntax
function CheckKeyHit(theKey) {
    if (theKey.charToNum() == 65)
        go("Correct Answer");
    } else {
        go("Wrong Answer");
    }
}
```

Voir aussi

`numToChar()`

clearAsObjects()

Syntaxe

```
-- Lingo syntax
clearAsObjects()

// JavaScript syntax
clearAsObjects();
```

Description

Commande ; reconfigure le lecteur Flash global utilisé pour les objets ActionScript et supprime tous les objets ActionScript de la mémoire. Cette commande n'efface ni ne reconfigure les références à ces objets stockées dans Lingo. Les références Lingo restent présentes mais font référence à des objets inexistants. Chaque référence doit être définie sur `VOID` individuellement.

La commande `clearAsObjects()` n'affecte que les objets globaux, tels que le tableau créé dans l'instruction suivante :

```
-- Lingo syntax
myGlobalArray = newObject(#array)

// JavaScript syntax
myGlobalArray = new Array();
```

La commande `clearAsObjects()` n'a aucun effet sur les objets créés dans les références d'images-objets, tels que :

```
myArray = sprite(2).newObject(#array)
```

Paramètres

Aucune.

Exemple

Cette instruction supprime de la mémoire tous les objets ActionScript créés globalement :

```
-- Lingo syntax
clearAsObjects()

// JavaScript syntax
clearAsObjects();
```

Voir aussi

`newObject()`, `setCallback()`

clearCache

Syntaxe

```
clearCache
```

Description

Commande ; vide la mémoire cache réseau de Director.

La commande `clearCache` ne vide que la mémoire cache, qui est distincte de celle du navigateur.

Les fichiers en cours d'utilisation ne sont pas supprimés de la mémoire cache (jusqu'à leur inactivité).

Paramètres

Aucune.

Exemple

Le gestionnaire suivant vide la mémoire cache au lancement de l'animation :

```
-- Lingo syntax
on startMovie
    clearCache
end
```

```
// JavaScript syntax
function startMovie() {
    clearCache();
}
```

Voir aussi

cacheDocVerify(), cacheSize()

clearError()

Syntaxe

```
-- Lingo syntax
memberObjRef.clearError()

// JavaScript syntax
memberObjRef.clearError();
```

Description

Commande Flash ; remet à zéro l'état d'erreur d'un acteur Flash en flux continu.

Si une erreur survient alors qu'un acteur est lu en flux continu dans la mémoire, Director affecte la valeur -1 à la propriété `state` de cet acteur afin d'indiquer qu'une erreur s'est produite. Vous pouvez utiliser la fonction `getError` pour déterminer le type d'erreur survenue, puis utiliser la commande `clearError` pour remettre à zéro l'état d'erreur de l'acteur. Une fois que vous avez éliminé cet état d'erreur, Director tente d'ouvrir l'acteur si l'animation en a encore besoin. La définition des propriétés `pathName`, `linked` et `preload` d'un acteur permet également d'éliminer la condition d'erreur.

Paramètres

Aucune.

Exemple

Le gestionnaire suivant vérifie l'occurrence d'une erreur de type mémoire épuisée pour l'acteur Flash Dali, qui a été transféré en mémoire. Si une telle erreur est survenue, le script utilise la commande `unloadCast` pour essayer de libérer de la mémoire ; il amène ensuite la tête de lecture sur une image de l'animation Artistes de Director dans laquelle l'image-objet de l'animation Flash apparaît pour la première fois, de sorte que Director puisse relancer la lecture de l'animation Flash. Si un autre type d'erreur est survenu, le script passe à une image intitulée Désolé qui explique que l'animation Flash requise ne peut pas être lue.

```
-- Lingo syntax
on CheckFlashStatus
    if (member("Dali").getError() = #memory) then
        member("Dali").clearError()
        member("Dali").unload()
        unloadCast
    else
        _movie.go("Sorry")
    end if
end

// JavaScript syntax
function CheckFlashStatus() {
    var ge = member("Dali").getError();
    if (ge = "memory") {
```

```
        member("Dali").clearError();
        unloadCast;
        _movie.go("Artists");
    } else {
        _movie.go("Sorry");
    }
}
```

Voir aussi

state (Flash, SWA), getError() (Flash, SWA)

clearFrame()

Syntaxe

```
-- Lingo syntax
_movie.clearFrame()

// JavaScript syntax
_movie.clearFrame();
```

Description

Méthode d'animation ; libère toutes les pistes d'image-objet d'une image pendant l'enregistrement du scénario.

Paramètres

Aucune.

Exemple

Le gestionnaire suivant supprime le contenu de chaque image avant de les modifier pendant la création du scénario :

```
-- Lingo syntax
on newScore
    _movie.beginRecording()
    repeat with counter = 1 to 50
        _movie.clearFrame()
        _movie.frameScript = 25
        _movie.updateFrame()
    end repeat
    _movie.endRecording()
end

// JavaScript syntax
function newScore() {
    _movie.beginRecording();
    for (var i = 1; i <= 50; i++) {
        _movie.clearFrame();
        _movie.frameScript = 25;
        _movie.updateFrame();
    }
    _movie.endRecording();
}
```

Voir aussi

beginRecording(), endRecording(), Animation, updateFrame()

clearGlobals()

Syntaxe

```
-- Lingo syntax
_global.clearGlobals()

// JavaScript syntax
_global.clearGlobals();
```

Description

Méthode globale ; définit toutes les variables globales sur `VOID` (Lingo) ou sur `null` (syntaxe JavaScript).

Cette méthode se révèle utile lors de l'initialisation de variables globales ou de l'ouverture d'une nouvelle animation exigeant un nouveau jeu de variables globales.

Paramètres

Aucune.

Exemple

Les gestionnaires suivants définissent toutes les variables globales sur `VOID` (Lingo) ou sur null (JavaScript) :

```
-- Lingo syntax
on mouseDown
    _global.clearGlobals()
end

// JavaScript syntax
function mouseDown() {
    _global.clearGlobals();
}
```

Voir aussi

`Global`

clone

Syntaxe

```
member(whichCastmember).model(whichModel).clone(cloneName)
member(whichCastmember).group(whichGroup).clone(cloneName)
member(whichCastmember).light(whichLight).clone(cloneName)
member(whichCastmember).camera(whichCamera).clone(cloneName)
```

Description

Commande 3D ; crée une copie du modèle, du groupe, de la lumière ou de la caméra, et de tous ses enfants. Le clone possède le même parent que le modèle, le groupe, la lumière ou la caméra à partir duquel il a été cloné.

Le clone d'un modèle utilise la même ressource de modèle et la même liste de matériaux que le modèle d'origine.

Si vous ne spécifiez pas le paramètre *`cloneName`* ou que vous spécifiez `""`, le clone n'est pas pris en compte par la méthode `count`, mais apparaîtra dans la scène.

Paramètres

`cloneName` Requis. Spécifie le nom du nouveau clone.

Exemple

L'instruction suivante crée le clone Théière2 à partir du modèle Théière et renvoie une référence au nouveau modèle.

```
-- Lingo
teapotCopy = member("3D World").model("Teapot").clone("Teapot2")

// Javascript
teapotCopy = member("3D World").getProp("model","Teapot").clone("Teapot2")
```

Voir aussi

cloneDeep, cloneModelFromCastmember, cloneMotionFromCastmember, loadFile()

cloneDeep

Syntaxe

```
member(whichCastmember).model(whichModel).cloneDeep(cloneName)
member(whichCastmember).group(whichGroup).cloneDeep(cloneName)
member(whichCastmember).light(whichLight).cloneDeep(cloneName)
member(whichCastmember).camera(whichCamera).cloneDeep(cloneName)
```

Description

Commande 3D ; crée une copie du modèle, du groupe, de la lumière ou de la caméra, plus tous les éléments suivants :

- les ressources de modèle, matériaux et textures utilisés par le modèle ou groupe d'origine
- les enfants du modèle, du groupe, de la lumière ou de la caméra
- les ressources de modèle, matériaux et textures utilisés par les enfants

Cette méthode utilise davantage de mémoire et demande plus de temps que la commande `clone`.

Paramètres

`cloneName` Requis. Spécifie le nom du nouveau clone.

Exemple

L'instruction suivante crée une copie du modèle Théière et de ses enfants, et des ressources de modèle, des matériaux et des textures utilisés par Théière et ses enfants. La variable `teapotCopy` est une référence au modèle cloné.

```
-- Lingo
teapotCopy = member("3D World").model("Teapot").cloneDeep("Teapot2")

// Javascript
teapotCopy =member("3DWorld").getProp("model","Teapot").cloneDeep("Teapot2")
```

Voir aussi

clone, cloneModelFromCastmember, cloneMotionFromCastmember, loadFile()

cloneModelFromCastmember

Syntaxe

```
member(whichCastmember).cloneModelFromCastmember(newModelName, sourceModelName,
sourceCastmember)
```

Description

Commande 3D ; copie un modèle provenant d'un acteur, le renomme, puis l'insère dans un autre acteur sous la forme d'un enfant de son univers 3D.

Cette commande copie également les enfants de *sourceModelName*, ainsi que les ressources de modèle, les matériaux et les textures, utilisés par le modèle et ses enfants.

Le chargement de l'acteur source doit être terminé pour la bonne exécution de cette commande.

Paramètres

newModelName Requis. Spécifie le nom du modèle nouvellement cloné.

sourceModelName Requis. Spécifie le modèle à cloner.

sourceCastMember Requis. Spécifie l'acteur contenant le modèle à cloner.

Exemple

L'instruction suivante crée une copie du modèle Pluton de l'acteur Séquence et l'insère dans l'acteur Séquence2 avec le nouveau nom Planète. Les enfants de Pluton sont également importés, de même que les ressources de modèle, les matériaux et les textures, utilisés par Pluton et ses enfants.

```
--Lingo
member("Scene2").cloneModelFromCastmember("Planet", "Pluto", member("Scene"))

// Javascript
member("Scene2").cloneModelFromCastmember("Planet", "Pluto", member("Scene"));
```

Voir aussi

cloneMotionFromCastmember, clone, cloneDeep, loadFile()

cloneMotionFromCastmember

Syntaxe

```
member(whichCastmember).cloneMotionFromCastmember(newMotionName, sourceMotionName,
sourceCastmember)
```

Description

Commande 3D ; copie un mouvement provenant d'un acteur, le renomme, puis l'insère dans un autre acteur.

Le chargement de l'acteur source doit être terminé pour la bonne exécution de cette commande.

Paramètres

newMotionName Requis. Spécifie le nom du mouvement nouvellement cloné.

sourceMotionName Requis. Spécifie le mouvement à cloner.

sourceCastMember Requis. Spécifie l'acteur contenant le mouvement à cloner.

Exemple

L'instruction suivante copie le mouvement Marche de l'acteur Parc, nomme la copie marcheBizarre, et la place dans l'acteur gbActeur.

```
--Lingo
member("gbMember").cloneMotionFromCastmember("FunnyWalk", "Walk", member("ParkScene"))
```

```
// Javascript
member("gbMember").cloneMotionFromCastmember("FunnyWalk", "Walk", member("ParkScene"));
```

Voir aussi

map (3D), cloneModelFromCastmember, clone, cloneDeep, loadFile()

close()

Syntaxe

```
-- Lingo syntax
windowObjRef.close()

// JavaScript syntax
windowObjRef.close();
```

Description

Méthode de fenêtre ; ferme une fenêtre.

Toute tentative visant à fermer une fenêtre déjà fermée n'a aucun effet.

Veuillez noter que la fermeture d'une fenêtre ne termine pas l'exécution de l'animation dans la fenêtre et ne la supprime pas non plus de la mémoire. Cette méthode sert simplement à fermer la fenêtre dans laquelle l'animation est en cours de lecture. Pour la rouvrir rapidement, utilisez la méthode `open() (Window)`. Cette procédure assure un accès rapide aux fenêtres qui doivent rester disponibles.

Pour supprimer totalement une fenêtre et la vider de la mémoire, utilisez la méthode `forget()`. Si vous utilisez la méthode `forget()`, assurez-vous qu'aucun élément ne fait référence à l'animation de cette fenêtre, faute de quoi le système génère des erreurs lorsque les scripts tentent de communiquer ou d'utiliser la fenêtre supprimée.

Paramètres

Aucune.

Exemple

L'instruction suivante ferme la fenêtre Panneau située dans le sous-dossier Sources MIAW du dossier de l'animation en cours :

```
-- Lingo syntax
window(_movie.path & "MIAW Sources\Panel").close()

// JavaScript syntax
window(_movie.path + "MIAW Sources\\Panel").close();
```

L'instruction suivante ferme la fenêtre 5 dans la liste `windowList` :

```
-- Lingo syntax
window(5).close()

// JavaScript syntax
window(5).close();
```

Voir aussi

forget() (fenêtre), open() (fenêtre), Fenêtre

closeFile()

Syntaxe

```
-- Lingo syntax
fileioObjRef.closeFile()

// JavaScript syntax
fileioObjRef.closeFile();
```

Description

Méthode FileIO ; ferme un fichier.

Paramètres

Aucune.

Voir aussi

`Fileio`

closeXlib

Syntaxe

```
closeXlib whichFile
```

Description

Commande ; ferme un fichier Xlibrary.

Les Xtras sont enregistrés dans des fichiers Xlibrary. Les fichiers Xlibrary sont les fichiers des ressources contenant les Xtras. Les XCMD et XFCN HyperCard peuvent également être stockés dans des fichiers Xlibrary.

La commande `closeXlib` ne fonctionne pas avec les URL.

Sous Windows, l'extension DLL des Xtras est facultative.

Nous vous recommandons de fermer tout fichier ouvert dès que vous n'en avez plus besoin.

Remarque : *Cette commande n'est pas prise en charge dans Shockwave Player.*

Paramètres

`whichFile` Facultatif. Spécifie le fichier Xlibrary à fermer. Si le fichier *`whichFile`* se trouve dans un autre dossier que celui de l'animation en cours, *`whichFile`* doit spécifier un nom de chemin d'accès. Si vous omettez de spécifier *`whichFile`*, tous les fichiers Xlibrary ouverts sont fermés.

Exemple

L'instruction suivante ferme tous les fichiers Xlibrary ouverts :

```
closeXlib
```

L'instruction suivante ferme le fichier Xlibrary Disque vidéo situé dans le même dossier que l'animation en cours :

```
closeXlib "Video Disc Xlibrary"
```

L'instruction suivante ferme le fichier Xlibrary Transporter Xtras du dossier Nouveaux Xtras, situé dans le même dossier que l'animation. Le disque est identifié par la variable lecteuractuel :

```
closeXlib "@:New Xtras:Transporter Xtras"
```

Voir aussi

Interface(), openXlib

color()

Syntaxe

```
-- Lingo syntax
color(intPaletteIndex)
color(intRed, intGreen, intBlue)

// JavaScript syntax
color(intPaletteIndex);
color(intRed, intGreen, intBlue);
```

Description

Fonction et type de données de niveau supérieur. Renvoie un objet de données couleur à l'aide de valeurs RVB ou de valeurs d'index de palette 8 bits.

L'objet couleur résultant est applicable aux acteurs, aux images-objets et à la scène lorsqu'il y a lieu.

Paramètres

intPaletteIndex Requis en cas d'utilisation de valeurs de palette 8 bits. Nombre entier spécifiant la valeur de palette 8 bits à utiliser. Les valeurs possibles sont comprises entre 0 et 255. Toutes les autres valeurs sont tronquées.

intRed Requis en cas d'utilisation de valeurs RVB. Nombre entier spécifiant le composant de couleur rouge dans la palette en cours. Les valeurs possibles sont comprises entre 0 et 255. Toutes les autres valeurs sont tronquées.

intGreen Requis en cas d'utilisation de valeurs RVB. Nombre entier spécifiant le composant de couleur verte dans la palette en cours. Les valeurs possibles sont comprises entre 0 et 255. Toutes les autres valeurs sont tronquées.

IntBlue Requis en cas d'utilisation de valeurs RVB. Nombre entier spécifiant le composant de couleur bleue dans la palette en cours. Les valeurs possibles sont comprises entre 0 et 255. Toutes les autres valeurs sont tronquées.

Exemple

Les instructions suivantes affichent la couleur de l'image-objet 6 dans la fenêtre Messages, puis définissent la couleur de l'image-objet 6 sur une nouvelle valeur :

```
-- Lingo syntax
put(sprite(6).color) -- paletteIndex(255)
sprite(6).color = color(137)
put(sprite(6).color) -- paletteIndex(137)

// JavaScript syntax
put(sprite(6).color) // paletteIndex(255);
sprite(6).color = color(137);
put(sprite(6).color) // paletteIndex(137);
```

constrainH()

Syntaxe

```
-- Lingo syntax
_movie.constrainH(intSpriteNum, intPosn)

// JavaScript syntax
_movie.constrainH(intSpriteNum, intPosn);
```

Description

Méthode d'animation ; renvoie un nombre entier dont la valeur dépend des coordonnées horizontales des côtés gauche et droit d'une image-objet.

Le nombre entier renvoyé peut prendre trois valeurs.

- Si le paramètre *intPosn* est compris entre les valeurs des coordonnées gauche et droite de l'image-objet, le nombre entier renvoyé est égal à *intPosn*.
- Si le paramètre *intPosn* est inférieur à la valeur de la coordonnée gauche de l'image-objet, le nombre entier renvoyé prend la valeur de cette coordonnée.
- Si le paramètre *intPosn* est supérieur à la valeur de la coordonnée droite de l'image-objet, le nombre entier renvoyé prend la valeur de cette coordonnée.

Cette méthode ne modifie pas les propriétés de l'image-objet.

Les méthodes `constrainH()` et `constrainV()` ne contraignent qu'un seul axe.

Paramètres

`intSpriteNum` Requis. Nombre entier spécifiant l'image-objet dont les coordonnées horizontales sont évaluées par rapport à *intPosn*.

intPosn Requis. Nombre entier à évaluer en fonction des coordonnées horizontales des côtés gauche et droit de l'image-objet identifiée par *intSpriteNum*.

Exemple

Les instructions suivantes vérifient la fonction `constrainH` pour l'image-objet 1 lorsque ses coordonnées gauche et droite sont 40 et 60 :

```
-- Lingo syntax
put(constrainH(1, 20)) -- 40
put(constrainH(1, 55)) -- 55
put(constrainH(1, 100)) -- 60

// JavaScript syntax
put(constrainH(1, 20)); // 40
put(constrainH(1, 55)); // 55
put(constrainH(1, 100)); // 60
```

L'instruction suivante limite les déplacements d'un curseur mobile (image-objet 1) aux bords d'une jauge (image-objet 2) lorsque le pointeur de la souris dépasse les bords de cette dernière :

```
-- Lingo syntax
sprite(1).locH = _movie.constrainH(2, _mouse.mouseH)

// JavaScript syntax
sprite(1).locH = _movie.constrainH(2, _mouse.mouseH);
```

Voir aussi

`constrainV()`, `Animation`

constrainV()

Syntaxe

```
-- Lingo syntax
_movie.constrainV(intSpriteNum, intPosn)

// JavaScript syntax
_movie.constrainV(intSpriteNum, intPosn);
```

Description

Méthode d'animation ; renvoie un nombre entier dont la valeur dépend des coordonnées verticales des côtés supérieur et inférieur d'une image-objet.

Le nombre entier renvoyé peut prendre trois valeurs.

- Si le paramètre *intPosn* est compris entre les valeurs des coordonnées supérieure et inférieure de l'image-objet, le nombre entier renvoyé est égal à *intPosn*.
- Si le paramètre *intPosn* est inférieur à la valeur de la coordonnée supérieure de l'image-objet, le nombre entier renvoyé prend la valeur de cette coordonnée.
- Si le paramètre *intPosn* est supérieur à la valeur de la coordonnée inférieure de l'image-objet, le nombre entier renvoyé prend la valeur de cette coordonnée.

Cette méthode ne modifie pas les propriétés de l'image-objet.

Les méthodes `constrainV()` et `constrainH()` ne contraignent qu'un seul axe.

Paramètres

intSpriteNum Requis. Nombre entier identifiant l'image-objet dont les coordonnées verticales sont évaluées par rapport à *intPosn*.

intPosn Requis. Nombre entier à évaluer en fonction des coordonnées verticales des côtés gauche et droit de l'image-objet identifiée par *intSpriteNum*.

Exemple

Les instructions suivantes vérifient la fonction constrainV de l'image-objet 1 lorsque ses coordonnées supérieure et inférieure sont 40 et 60 :

```
-- Lingo syntax
put(constrainV(1, 20)) -- 40
put(constrainV(1, 55)) -- 55
put(constrainV(1, 100)) -- 60

// JavaScript syntax
put(constrainV(1, 20)); // 40
put(constrainV(1, 55)); // 55
put(constrainV(1, 100)); // 60
```

L'instruction suivante limite les déplacements d'un curseur mobile (image-objet 1) aux bords d'une jauge (image-objet 2) lorsque le pointeur de la souris dépasse les bords de cette dernière :

```
-- Lingo syntax
sprite(1).locV = _movie.constrainV(2, _mouse.mouseH)

// JavaScript syntax
sprite(1).locV = _movie.constrainV(2, _mouse.mouseH);
```

Voir aussi

constrainH(), Animation

copyPixels()

Syntaxe

```
-- Lingo syntax
imageObjRef.copyPixels(sourceImgObj, destRectOrQuad, sourceRect {, paramList})

// JavaScript syntax
imageObjRef.copyPixels(sourceImgObj, destRectOrQuad, sourceRect {, paramList});
```

Description

Méthode d'image. Copie le contenu d'un rectangle d'un objet image existant dans un nouvel objet image.

Lors de la copie de pixels d'une zone d'un acteur vers une autre zone du même acteur, il est recommandé de copier d'abord les pixels dans un objet image dupliqué avant de les recopier dans l'acteur d'origine. La copie directe d'une zone à l'autre dans la même image est déconseillée.

Pour simuler l'encre Dessin seul avec `copyPixels()`, créez un objet Dessin seul avec `createMatte()`, puis transmettez cet objet en tant que paramètre `#maskImage` de `copyPixels()`.

Vous pouvez voir un exemple d'utilisation de quad dans une animation en consultant l'animation Quad du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Lors de l'utilisation de la méthode copyPixel, si la propriété useAlpha est définie sur False pour l'image source ou de destination, la destination ne comporte aucune information alpha. Si vous souhaitez que la destination intègre un contenu alpha, la propriété useAlpha doit présenter la valeur True pour la source et pour la destination.

Paramètres

`sourceImgObj` Requis. Référence à l'objet image source à partir duquel les pixels sont copiés.

`destRectOrQuad` Requis en cas de copie de pixels dans un rectangle de coordonnées d'écran ou un quadrilatère à virgule flottante. Rectangle ou quadrilatère dans lequel les pixels sont copiés.

`sourceRect` Requis. Rectangle source à partir duquel les pixels sont copiés.

`paramList` Facultatif. Liste de paramètres utilisable pour manipuler les pixels copiés avant leur insertion dans *`destRect`* ou *`destQuad`*. La liste de propriétés peut contenir la totalité ou une partie des paramètres suivants.

Propriété	Usage et effet
#color	Couleur du premier plan à appliquer pour les effets de colorisation. La valeur par défaut est le noir.
#bgColor	Couleur d'arrière-plan à appliquer pour les effets de colorisation ou comme transparence d'arrière-plan. La valeur par défaut est le blanc.
#ink	Type d'encre à appliquer aux pixels copiés. Il peut s'agir d'un symbole d'encre ou de la valeur numérique correspondante. L'encre par défaut est `#copy`.

Propriété	Usage et effet
#blendLevel	Degré d'opacité (transparence) à appliquer aux pixels copiés. La plage de valeurs s'étend de 0 à 225. La valeur par défaut est 255 (opaque). L'utilisation d'une valeur inférieure à 255 définit le paramètre `#ink` sur `#blend` ou sur `#blendTransparent` s'il était initialement défini sur `#backgroundTransparent`. #blendLevel pourrait également être remplacé par `#blend` ; dans ce cas, utilisez une valeur comprise entre 0 et 100.
#dither	Valeur `TRUE` ou `FALSE` déterminant si les pixels copiés sont tramés s'ils sont placés dans le *`destRect`* des images 8 et 16 bits. La valeur par défaut est `FALSE` qui convertit directement les pixels copiés dans la palette de couleurs de *`imageObjRef`*.
#useFastQuads	Valeur `TRUE` ou `FALSE` déterminant si les calculs de quadrilatère sont effectués à l'aide de la méthode de Director, plus rapide mais moins précise, lors de la copie de pixels dans *`destQuad`*. Définissez la valeur `TRUE` pour utiliser des quadrilatères pour de simples opérations de rotation et d'inclinaison. Définissez la valeur `FALSE` pour des quadrilatères aléatoires, tels que ceux utilisés pour les transformations de perspective. La valeur par défaut est `FALSE`.
#maskImage	Spécifie un objet Masque ou Dessin seul créé avec les fonctions `creatMask()` ou `createMatte()` afin d'être utilisé en tant que masque pour les pixels à copier. Cette opération permet de reproduire les effets des encres d'image-objet Masque et Dessin seul. Notez que si l'image source comporte une couche alpha et que sa propriété `useAlpha` est définie sur `TRUE`, la couche alpha est utilisée et l'encre Masque ou Dessin seul spécifiée est ignorée. La valeur par défaut est pas de masque.
#maskOffset	Point indiquant le montant de décalage x et y à appliquer au masque spécifié par `#maskImage`. Le décalage est calculé par rapport au coin supérieur gauche de l'image source. Le décalage par défaut est (0, 0).

Exemple

L'instruction suivante copie toute l'image de l'acteur Joyeux dans le rectangle de l'acteur Fleur. Si les acteurs sont de tailles différentes, l'image de l'acteur Joyeux est redimensionnée pour s'ajuster au rectangle de l'acteur Fleur.

```
-- Lingo
member("flower").image.copyPixels(member("Happy").image,member("flower").image.rect,
member("Happy").image.rect)

// JavaScript syntax
member("flower").image.copyPixels(member("Happy").image,member("flower").image.rect,
member("Happy").image.rect);
```

L'instruction suivante copie une partie de l'image de l'acteur Joyeux dans une partie de l'acteur Fleur. La partie de l'image copiée de l'acteur Joyeux est située dans le rectangle (0, 0, 200, 90). Elle est collée dans le rectangle (20, 20, 100, 40) à l'intérieur de l'image de l'acteur Fleur. La portion copiée de l'acteur Joyeux est redimensionnée pour s'adapter aux dimensions du rectangle dans lequel elle est collée.

```
-- Lingo
member("flower").image.copyPixels(member("Happy").image,
rect(20,20,100,40),rect(0,0,200,90))

// JavaScript syntax
member("flower").image.copyPixels(member("Happy").image,
rect(20,20,100,40),rect(0,0,200,90))
```

L'instruction suivante copie entièrement l'image de l'acteur Joyeux dans un rectangle à l'intérieur de l'image de l'acteur Fleur. Le rectangle dans lequel l'image copiée de l'acteur Joyeux est collée est de taille identique à celle du rectangle de l'acteur Joyeux, de sorte que l'image copiée ne doive pas être redimensionnée. Le niveau d'opacité de l'image copiée est de 50, elle est donc semi-tranparente, révélant la portion de l'acteur Fleur sur laquelle elle est collée.

```
-- Lingo
member("flower").image.copyPixels(member("Happy").image, rect(90,110,290,310),
member("Happy").image.rect, [#blendLevel: 50])
```

```
// JavaScript syntax
member("flower").image.copyPixels(member("Happy").image, rect(90,110,290,310),
member("Happy").image.rect, \propList(symbol("blendLevel"),50))
```

Voir aussi

color(), image()

copyToClipBoard()

Syntaxe

```
-- Lingo syntax
memberObjRef.copyToClipBoard()

// JavaScript syntax
memberObjRef.copyToClipBoard();
```

Description

Méthode d'acteur ; copie un acteur spécifié dans le Presse-papiers.

L'appel de cette méthode n'exige pas que la fenêtre Distribution soit active.

Cette méthode se révèle utile pour la copie d'acteurs entre animations ou applications.

Paramètres

Aucune.

Exemple

L'instruction suivante copie l'acteur `chair` dans le Presse-papiers :

```
-- Lingo syntax
member("chair").copyToClipBoard()

// JavaScript syntax
member("chair").copyToClipBoard();
```

L'instruction suivante copie l'acteur numéro 5 dans le Presse-papiers :

```
--- Lingo syntax
member(5).copyToClipBoard()

// JavaScript syntax
member(5).copyToClipBoard();
```

Voir aussi

Acteur, pasteClipBoardInto()

cos()

Syntaxe

```
(angle).cos
cos (angle)
```

Description

Fonction (Lingo uniquement) ; calcule le cosinus de l'angle spécifié, qui doit être exprimé en radians.

En syntaxe JavaScript, utilisez la fonction `cos()` d'un objet mathématique.

Paramètres

`angle` Requis. Nombre entier spécifiant l'angle à tester.

Exemple

L'instruction suivante calcule le cosinus de PI divisé par 2 et l'affiche dans la fenêtre Messages :

```
put (PI/2).cos
```

Voir aussi

`atan()`, `PI`, `sin()`

count()

Syntaxe

```
-- Lingo syntax
list.count
object.count

// JavaScript syntax
list.count;
object.count;
```

Description

Fonction ; renvoie le nombre d'entrées d'une liste linéaire ou de propriétés, le nombre de propriétés d'un script parent sans compter les propriétés d'un script ancêtre ou les sous-chaînes d'une expression texte telles que caractères, lignes ou mots.

La commande `count` fonctionne avec les listes linéaires et de propriétés, les objets créés avec des scripts parents et la propriété `globals`.

Vous pouvez voir un exemple d'utilisation de `count()` dans une animation en consultant l'animation Text du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

Aucune.

Exemple

L'instruction suivante affiche la valeur 3, qui correspond au nombre d'entrées :

```
--Lingo syntax
put([10,20,30].count) -- 3

// JavaScript syntax
put(list(10,20,30).count); // 3
```

Voir aussi

`globals`

createFile()

Syntaxe

```
-- Lingo syntax
fileioObjRef.createFile(stringFileName)

// JavaScript syntax
fileioObjRef.createFile(stringFileName);
```

Description

Méthode FileIO ; crée un fichier spécifié.

Paramètres

stringFileName Requis. Chaîne spécifiant le chemin d'accès et le nom du fichier à créer.

Exemple

L'exemple suivant crée un fichier appelé « xtra.txt » dans c:\.

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.createFile("c:\xtra.txt")

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.createFile("c:\xtra.txt");
```

Voir aussi

Fileio

createMask()

Syntaxe

```
imageObject.createMask()
```

Description

Cette fonction crée et renvoie un objet masque à utiliser avec la fonction `copyPixels()`.

Les objets masques ne sont pas des objets images. Ils ne sont utiles qu'avec la fonction `copyPixels()` pour la reproduction de l'effet d'encre Masque. Si vous envisagez d'utiliser plusieurs fois la même image en tant que masque, il est conseillé de créer l'objet masque et de l'enregistrer dans une variable pour une utilisation ultérieure.

Exemple

L'instruction suivante copie entièrement l'image de l'acteur Joyeux dans un rectangle à l'intérieur de l'acteur Carré marron. L'acteur Dégradé2 est utilisé en tant que masque avec l'image copiée. Le masque est décalé de 10 pixels vers le haut et vers la gauche du rectangle dans lequel l'image Joyeux est collée.

```
member("brown square").image.copyPixels(member("Happy").image, rect(20, 20, 150, 108),
member("Happy").rect,[#maskImage:member("gradient2").image.createMask(),
maskOffset:point(-10, -10)])
```

Voir aussi

copyPixels(), createMatte(), ink

createMatte()

Syntaxe

```
imageObject.createMatte({alphaThreshold})
```

Description

Cette fonction crée et renvoie un objet dessin seul que vous pouvez utiliser avec `copyPixels()` pour reproduire l'effet d'encre Dessin seul. L'objet dessin seul est créé à partir de la couche alpha de l'objet image spécifiée. Le paramètre facultatif *`alphaThreshold`* exclut de l'encre Dessin seul tous les pixels dont la valeur de couche alpha est inférieure à ce seuil. Il n'est utilisé qu'avec les images 32 bits contenant une couche alpha. La valeur de *`alphaThreshold`* doit être comprise entre 0 et 255.

Les objets dessin seul ne sont pas des objets images. Ils ne sont utiles qu'avec la fonction `copyPixels()`. Si vous envisagez d'utiliser plusieurs fois la même image en tant que Dessin seul, il est conseillé de créer l'objet dessin seul et de l'enregistrer dans une variable pour une utilisation ultérieure.

Exemple

L'instruction suivante crée un nouvel objet Dessin seul à partir de la couche alpha de l'objet image imageTest et ignore les pixels dont les valeurs alpha sont inférieures à 50 %.

```
newMatte = testImage.createMatte(128)
```

Voir aussi

`copyPixels()`, `createMask()`

crop() (image)

Syntaxe

```
-- Lingo syntax
imageObjRef.crop(rectToCropTo)

// JavaScript syntax
imageObjRef.crop(rectToCropTo);
```

Description

Méthode d'image. Renvoie un nouvel objet image contenant une copie d'un objet image source recadrée dans un rectangle donné.

L'appel de la méthode `crop()` n'altère pas l'objet image source.

Le nouvel objet image n'appartient plus à aucun acteur et n'est plus associé à la scène. Pour affecter la nouvelle image à un acteur, définissez la propriété `image` de cet acteur.

Paramètres

`rectToCropTo` Requis. Rectangle dans lequel la nouvelle image est recadrée.

Exemple

L'instruction suivante indique à Lingo de recadrer toute image-objet faisant référence à l'acteur vidéo numérique Entrevue.

```
-- Lingo
Dot syntax:
member("Interview").crop = TRUE
Verbose syntax:
set the crop of member "Interview" to TRUE

// Javascript
member("Interview").crop=true
```

Voir aussi

image (image), image(), rect (image)

crop() (bitmap)

Syntaxe

```
-- Lingo syntax
memberObjRef.crop()

// JavaScript syntax
memberObjRef.crop();
```

Description

Commande bitmap ; permet de recadrer un acteur bitmap à une taille spécifique.

La commande `crop` permet de recadrer des acteurs existants ou, en combinaison avec l'image de la scène, de saisir un instantané, puis de le recadrer à la taille souhaitée pour l'afficher.

Le point d'alignement reste inchangé de sorte que le bitmap n'est pas déplacé par rapport à sa position d'origine.

Paramètres

`rectToCropTo` Requis. Spécifie le rectangle en fonction duquel un acteur est recadré.

Exemple

L'instruction suivante affecte un acteur bitmap existant à un instantané de la scène, puis recadre l'image résultante dans un rectangle égal à l'image-objet 10 :

```
-- Lingo syntax
stageImage = (_movie.stage).image
spriteImage = stageImage.crop(sprite(10).rect)
member("sprite snapshot").image = spriteImage

// JavaScript syntax
var stageImage = (_movie.stage).image;
var spriteImage = stageImage.crop(sprite(10).rect);
member("sprite snapshot").image = spriteImage;
```

Voir aussi

picture (acteur)

cross

Syntaxe

```
vector1.cross(vector2)
```

Description

Méthode de vecteur 3D ; renvoie un vecteur perpendiculaire à *vector1* et *vector2*.

Exemple

Dans l'exemple suivant, pos1 est un vecteur sur l'axe des x et pos2 est un vecteur sur l'axe des y. La valeur renvoyée par `pos1.cross(pos2)` est `vector( 0.0000, 0.0000, 1.00000e4 )`, qui est perpendiculaire à pos1 et pos2.

```
ppos1 = vector(100, 0, 0)
pos2 = vector(0, 100, 0)

-- Lingo
put pos1.cross(pos2)
-- vector( 0.0000, 0.0000, 1.00000e4 )

// Javascript
trace(pos1.cross(pos2))
// vector( 0.0000, 0.0000, 1.00000e4 )
```

Voir aussi

crossProduct(), perpendicularTo

crossProduct()

Syntaxe

```
vector1.crossProduct(vector2)
```

Description

Méthode de vecteur 3D ; renvoie un vecteur perpendiculaire à *vector1* et *vector2*.

Exemple

Dans l'exemple suivant, pos1 est un vecteur sur l'axe des x et pos2 est un vecteur sur l'axe des y. La valeur renvoyée par `pos1.crossProduct(pos2)` est `vector( 0.0000, 0.0000, 1.00000e4 )`, qui est perpendiculaire à pos1 et pos2.

```
pos1 = vector(100, 0, 0)
pos2 = vector(0, 100, 0)

-- Lingo
put pos1.crossProduct(pos2)
-- vector( 0.0000, 0.0000, 1.00000e4 )

// Javascript
trace(pos1.crossProduct(pos2))
// vector( 0.0000, 0.0000, 1.00000e4 )
```

Voir aussi

perpendicularTo, cross

cursor()

Syntaxe

```
-- Lingo syntax
_player.cursor(intCursorNum)
_player.cursor(cursorMemNum, maskMemNum)
_player.cursor(cursorMemRef)

// JavaScript syntax
_player.cursor(intCursorNum);
_player.cursor(cursorMemNum, maskMemNum);
_player.cursor(cursorMemRef);
```

Description

Méthode de lecteur ; change l'acteur ou le curseur intégré utilisé comme curseur et reste en vigueur jusqu'à ce que vous la désactiviez en lui affectant la valeur 0.

• Utilisez la syntaxe `_player.cursor(cursorMemNum, maskMemNum)` pour spécifier le numéro d'acteur à utiliser comme curseur et son masque facultatif. La zone référencée du curseur correspond au point d'alignement de l'acteur.

L'acteur spécifié doit être un acteur 1 bit. Si l'acteur est plus grand que 16 x 16 pixels, Director le recadre pour le transformer en carré de 16 x 16 pixels, à partir de l'angle supérieur gauche de l'image. La zone référencée du curseur correspond toujours au point d'alignement de l'acteur.

• Utilisez la syntaxe `_player.cursor(cursorMemRef)` pour les curseurs personnalisés proposés par l'Xtra Cursor.

Remarque : *L'Xtra Cursor accepte différents types de bibliothèques de distribution comme curseurs, mais non les acteurs texte.*

• Utilisez la syntaxe `_player.cursor(intCursorNum)` pour spécifier les curseurs par défaut fournis par le système. Le terme *intCursorNum* doit être l'un des nombres entiers suivants :

Valeur	Description
-1, 0	Flèche
1	I
2	Croix
3	Croix épaisse
4	Montre (Mac) ou sablier (Windows)
5	Nord Sud Est Ouest (NSEO)
6	Nord Sud (NS)
200	Vide (masque le curseur)
254	Aide
256	Crayon
257	Gomme
258	Sélection
259	Pot de peinture

Valeur (Suite)	**Description (Suite)**
260	Main
261	Outil Rectangle
262	Outil Rectangle arrondi
263	Outil Cercle
264	Outil Ligne
265	Outil Texte enrichi
266	Outil Champ de texte
267	Outil Bouton
268	Outil Case à cocher
269	Outil Bouton radio
270	Outil Placement
271	Outil Point d'alignement
272	Lasso
280	Doigt
281	Pipette
282	Attente souris appuyée 1
283	Attente souris appuyée 2
284	Taille verticale
285	Taille horizontale
286	Taille diagonale
290	Main fermée
291	Interdiction
292	Copie (main fermée)
293	Flèche inversée
294	Rotation
295	Inclinaison
296	Double flèche horizontale
297	Double flèche verticale
298	Double flèche Sud-ouest Nord-est
299	Double flèche Nord-ouest Sud-est
300	Pinceau Enduire/Lisser
301	Aérographe
302	Zoom avant

Valeur (Suite)	Description (Suite)
303	Zoom arrière
304	Annulation du zoom
305	Début de forme
306	Ajout de point
307	Fermeture de forme
308	Zoom de caméra
309	Déplacement de caméra
310	Rotation de caméra
457	Personnalisé

Pendant les événements système tels que le chargement d'un fichier, le système d'exploitation peut afficher le curseur montre, puis revenir au curseur pointeur en rendant le contrôle à l'application. Cette opération annule les paramètres de la commande `cursor` de l'animation précédente. Pour utiliser la commande `cursor()` au début de toute nouvelle animation chargée dans une présentation utilisant un curseur personnalisé pour plusieurs animations, stockez donc tout numéro de ressource de curseur spécial sous la forme d'une variable globale qui reste en mémoire entre les animations.

Les commandes de curseur peuvent être interrompues par un Xtra ou tout autre élément externe. Lorsque le curseur est défini sur une valeur dans Director et qu'un Xtra ou un élément externe en prend le contrôle, le rétablissement de sa valeur d'origine n'a aucun effet, Director ne percevant pas que sa valeur a changé. Pour résoudre ce problème, attribuez explicitement au curseur une troisième valeur, puis rendez-lui sa valeur d'origine.

Paramètres

`intCursorNum` Requis en cas d'utilisation d'un nombre entier pour identifier un curseur. Nombre entier spécifiant le curseur intégré à utiliser comme curseur.

`cursorMemNum` Requis en cas d'utilisation d'un numéro d'acteur et de son masque facultatif pour identifier le curseur. Nombre entier spécifiant le numéro d'acteur à utiliser comme curseur.

`maskMemNum` Requis en cas d'utilisation d'un numéro d'acteur et de son masque facultatif pour identifier le curseur. Nombre entier spécifiant le numéro de masque de *`cursorMemNum`*.

`cursorMemRef` Requis en cas d'utilisation d'une référence d'acteur pour identifier le curseur. Référence à l'acteur à utiliser comme curseur.

Exemple

L'instruction suivante change le curseur en curseur montre sur un Mac et en sablier sous Windows si la valeur de la variable `status` est égale à 1 :

```
-- Lingo syntax syntax
if (status = 1) then
    _player.cursor(4)
end if

// JavaScript syntax
if (status == 1) {
    _player.cursor(4);
}
```

Le gestionnaire suivant vérifie que l'acteur affecté à la variable est un acteur 1 bit et, le cas échéant, l'utilise comme curseur :

```
-- Lingo syntax syntax
on myCursor(someMember)
    if (member(someMember).depth = 1) then
        _player.cursor(someMember)
    else
        _sound.beep()
    end if
end

// JavaScript syntax
function myCursor(someMember) {
    if (member(someMember).depth == 1) {
        _player.cursor(someMember);
    }
    else {
        _sound.beep();
    }
}
```

Voir aussi

Lecteur

date() (formats)

Syntaxe

```
-- Lingo syntax syntax
date({stringFormat})
date({intFormat})
date({intYearFormat, intMonthFormat, intDayFormat})

// JavaScript syntax
Date({"month dd, yyyy hh:mm:ss"});
Date({"month dd, yyyy"});
Date({yy,mm,dd,hh,mm,ss});
Date({yy,mm,dd});
Date({milliseconds});
```

Description

Fonction et type de données de niveau supérieur. Crée une instance d'objet date au format standard que vous pouvez utiliser avec d'autres instances d'objet date dans des opérations arithmétiques ou pour manipuler les dates d'une plate-forme à l'autre, ainsi que dans des formats de pays différents.

Les objets date Lingo diffèrent des objets date JavaScript ; par conséquent, les objets date Lingo ne peuvent pas être créés à l'aide de la syntaxe JavaScript et les objets date JavaScript ne peuvent pas être créés dans la syntaxe Lingo.

Pour créer un objet date JavaScript, utilisez la syntaxe `new Date()`. La syntaxe JavaScript fait la distinction entre les majuscules et les minuscules. Par exemple, l'utilisation d'une commande `new date()` entraîne une erreur d'exécution.

Lorsque vous créez une date avec Lingo, utilisez des années à quatre chiffres, des mois à deux chiffres et des jours à deux chiffres. Les expressions suivantes renvoient toutes un objet date équivalent au 21 octobre 2004.

Format de date	Syntaxe
chaîne	date("20041021")
entier	date(20041021)
Séparation par des virgules	date(2004, 10, 21)

Les différentes propriétés de l'objet date renvoyé sont les suivantes :

Propriété	Description
#year	Nombre entier représentant l'année
#month	Nombre entier représentant le mois de l'année
#day	Nombre entier représentant le jour du mois

Les opérations d'addition et de soustraction sur la date sont interprétées comme l'addition et la soustraction de jours.

Paramètres

stringFormat Facultatif en cas de création d'un objet date Lingo. Chaîne spécifiant le nouvel objet date.

intFormat Facultatif en cas de création d'un objet date Lingo. Nombre entier spécifiant le nouvel objet date.

intYearFormat Facultatif en cas de création d'un objet date Lingo. Nombre entier spécifiant l'année à quatre chiffres du nouvel objet date.

intMonthFormat Facultatif en cas de création d'un objet date Lingo. Nombre entier spécifiant le mois à deux chiffres du nouvel objet date.

intDayFormat Facultatif en cas de création d'un objet date Lingo. Nombre entier spécifiant le jour à deux chiffres du nouvel objet date.

month Facultatif en cas de création d'un objet date en syntaxe JavaScript. Chaîne spécifiant le mois du nouvel objet date. Les valeurs possibles sont comprises entre 0 (janvier) et 11 (décembre).

dd Facultatif en cas de création d'un objet date en syntaxe JavaScript. Nombre entier à deux chiffres spécifiant le jour du nouvel objet date. Les valeurs possibles sont comprises entre 0 (dimanche) et 6 (samedi).

yyyy Facultatif en cas de création d'un objet date en syntaxe JavaScript. Nombre entier à quatre chiffres spécifiant l'année du nouvel objet date.

hh Facultatif en cas de création d'un objet date en syntaxe JavaScript. Nombre entier à deux chiffres spécifiant l'heure du nouvel objet date. Les valeurs possibles sont comprises entre 0 (minuit) et 23 (23h00).

mm Facultatif en cas de création d'un objet date en syntaxe JavaScript. Nombre entier à deux chiffres spécifiant la minute du nouvel objet date. Les valeurs possibles sont comprises entre 0 et 59.

ss Facultatif en cas de création d'un objet date en syntaxe JavaScript. Nombre entier à deux chiffres spécifiant les secondes du nouvel objet date. Les valeurs possibles sont comprises entre 0 et 59.

yy Facultatif en cas de création d'un objet date en syntaxe JavaScript. Nombre entier à deux chiffres spécifiant l'année du nouvel objet date. Les valeurs possibles sont comprises entre 0 et 99.

milliseconds Facultatif en cas de création d'un objet date en syntaxe JavaScript. Nombre entier spécifiant les millisecondes du nouvel objet date. Les valeurs possibles sont comprises entre 0 et 999.

Exemple

Les instructions suivantes créent et déterminent le nombre de jours séparant deux dates :

```
-- Lingo syntax syntax
myBirthday = date(19650712)
yourBirthday = date(19450529)
put("There are" && abs(yourBirthday - myBirthday) && "days between our birthdays.")

// JavaScript syntax
var myBirthday = new Date(1965, 07, 12);
var yourBirthday = new Date(1945, 05, 29);
put("There are " + Math.abs(((yourBirthday - myBirthday)/1000/60/60/24)) + " days between
our birthdays.");
```

Les instructions suivantes accèdent à une propriété individuelle d'une date :

```
-- Lingo syntax syntax
myBirthday = date(19650712)
put("I was born in month number" && myBirthday.month)

// JavaScript syntax
var myBirthday = new Date(1965, 07, 12);
put("I was born in month number " + myBirthday.getMonth());
```

date() (système)

Syntaxe

```
-- Lingo syntax
_system.date({yyyymmdd})

// JavaScript syntax
_system.date({yyyymmdd});
```

Description

Méthode système ; renvoie la date en cours dans l'horloge système.

Le format de date utilisé par Director varie selon le format de date défini sur l'ordinateur.

- Sous Windows, vous pouvez personnaliser l'affichage de la date dans le Panneau de configuration Paramètres régionaux. Windows enregistre le format de date court dans le fichier System.ini. Utilisez cette valeur pour déterminer ce que les parties de la date au format court indiquent.
- Sur le Mac, vous pouvez personnaliser l'affichage de la date dans le tableau de bord Date et heure.

Paramètres

yyyymmdd Facultatif. Nombre spécifiant l'année à quatre chiffres (*yyyy*), le mois à deux chiffres (*mm*) et le jour à deux chiffres (*dd*) de la date renvoyée.

Exemple

L'instruction suivante détermine si la date en cours est le 1er janvier en vérifiant si les quatre premiers caractères de la date sont 1/1. Si c'est le 1er janvier, le message d'alerte Bonne année ! apparaît:

```
-- Lingo syntax
if (_system.date().char[1..4] = "1/1/") then
    _player.alert("Happy New Year!")
end if
```

```
// JavaScript syntax
if (_system.date().toString().substr(0, 4) == "1/1/") {
    _player.alert("Happy New Year!");
}
```

Voir aussi

Système

delay()

Syntaxe

```
-- Lingo syntax
_movie.delay(intTicks)

// JavaScript syntax
_movie.delay(intTicks);
```

Description

Méthode d'animation ; immobilise la tête de lecture pendant une durée donnée.

La seule activité de souris et de clavier possible à ce moment-là consiste à arrêter l'animation en appuyant sur Ctrl+Alt+point (Windows) ou sur Cmd+point (Mac). Dans la mesure où elle augmente la durée des différentes images, la méthode `delay()` est utile pour contrôler la cadence de lecture d'une séquence d'images.

La méthode `delay()` n'est applicable que lorsque la tête de lecture est en mouvement. Toutefois, lorsque la méthode `delay()` est effective, les gestionnaires continuent de s'exécuter ; seule la tête de lecture s'arrête, et non l'exécution des scripts. Insérez les scripts utilisant `delay()` dans un gestionnaire `enterFrame` ou `exitFrame`.

Pour simuler une interruption dans un gestionnaire lorsque la tête de lecture est immobile, utilisez la propriété `milliseconds` de l'objet système et attendez le laps de temps spécifié avant de quitter l'image.

Paramètres

`entBattements` Requis. Nombre entier spécifiant le nombre de battements pendant lequel la tête de lecture doit rester immobilisée. Chaque battement correspond à un 60ème de seconde.

Exemple

Le gestionnaire suivant interrompt l'animation pendant 2 secondes lorsque l'utilisateur appuie sur une touche :

```
-- Lingo syntax
on keyDown
    _movie.delay(2*60)
end

// JavaScript syntax
function keyDown() {
    _movie.delay(2*60)
}
```

Le gestionnaire suivant, qui peut être inséré dans un script d'image, retarde l'animation d'un nombre de battements aléatoire :

```
-- Lingo syntax
on keyDown
    if (_key.key = "x") then
```

```
        _movie.delay(random(180))
    end if
end

// JavaScript syntax
function keyDown() {
    if (_key.key == "x") {
        _movie.delay(random(180));
    }
}
```

Voir aussi

endFrame, milliseconds, Animation

delete()

Syntaxe

```
delete chunkExpression
```

Description

Une expression de sous-chaîne peut être un caractère, un mot, un élément ou une ligne d'un conteneur de caractères.

Paramètres

Aucune.

Voir aussi

Fileio

Exemple

```
delete member(1).line[1]
```

L'instruction suivante supprime la première ligne de l'acteur texte.

```
delete member(1).word[1]
```

L'instruction suivante supprime le premier mot de l'acteur texte.

```
delete member(1).char[1]
```

L'instruction suivante supprime le premier caractère de l'acteur texte.

deleteFile()

Syntaxe

```
-- Lingo syntax
fileioObjRef.deleteFile()

// JavaScript syntax
fileioObjRef.deleteFile();
```

Description

Méthode FileIO ; supprime un fichier.

Paramètres

Aucune.

Voir aussi

Fileio

deleteAt

Syntaxe

```
list.deleteAt(number)
deleteAt list, number
```

Description

Commande de liste ; supprime un élément d'une liste linéaire ou de propriétés.

La commande `deleteAt` vérifie que l'élément se trouve dans la liste ; si vous tentez de supprimer un objet ne figurant pas dans la liste, Director affiche un message d'erreur.

Paramètres

number Requis. Spécifie la position de l'élément à supprimer dans la liste.

Exemple

L'instruction suivante supprime le deuxième élément de la liste designers, qui contient [dupont, avatar, soldes] :

```
--Lingo
designers = ["gee", "kayne", "ohashi"]
designers.deleteAt(2)

// Javascript
Designers = list("gee","kayne","ohashi");
Designers.deleteAt(2);
```

Le résultat est la liste [Jean, Marc].

Le gestionnaire suivant vérifie si un objet se trouve dans la liste avant d'essayer de le supprimer :

```
on myDeleteAt theList, theIndex
    if theList.count < theIndex then
        beep

    else
        theList.deleteAt(theIndex)
    end if
end
```

Voir aussi

addAt

deleteCamera

Syntaxe

```
member(whichCastmember).deleteCamera(cameraName)
member(whichCastmember).deleteCamera(index)
```

```
sprite(whichSprite).deleteCamera(cameraOrIndex)
```

Description

Commande 3D ; dans un acteur, cette commande supprime la caméra de l'acteur et de l'univers 3D. Les enfants de la caméra sont retirés de l'univers 3D mais ne sont pas supprimés.

Il est impossible de supprimer la caméra par défaut de l'acteur.

Dans une image-objet, cette commande supprime la caméra de la liste des caméras de l'image-objet. La caméra n'est pas supprimée de l'acteur.

Paramètres

cameraNameOrNum Requis. Chaîne ou nombre entier spécifiant le nom ou la position d'index de la caméra à supprimer.

Exemple

L'instruction suivante supprime deux caméras de l'acteur Pièce : tout d'abord la caméra Caméra06, puis la caméra 3.

```
member("Room").deleteCamera("Camera06")
member("Room").deleteCamera(1)
```

L'instruction suivante supprime deux caméras de la liste des caméras de l'image-objet 5 : tout d'abord la deuxième caméra de la liste, puis la caméra Caméra06.

```
sprite(5).deleteCamera(2)
sprite(5).deleteCamera(member("Room").camera("Camera06"))
```

Voir aussi

newCamera, addCamera, cameraCount()

deleteFrame()

Syntaxe

```
-- Lingo syntax
_movie.deleteFrame()

// JavaScript syntax
_movie.deleteFrame();
```

Description

Méthode d'animation ; supprime l'image en cours et fait de l'image suivante la nouvelle image en cours uniquement lors d'une session de création du scénario.

Paramètres

Aucune.

Exemple

Le gestionnaire suivant vérifie si l'image-objet de la piste 10 de l'image actuelle a dépassé le bord droit d'une scène de 640 x 480 pixels et, le cas échéant, supprime l'image :

```
-- Lingo syntax
on testSprite
    _movie.beginRecording()
    if (sprite(10).locH > 640) then
        _movie.deleteFrame()
```

```
    end if
    _movie.endRecording()
end

// JavaScript syntax
function testSprite() {
    _movie.beginRecording();
    if (sprite(10).locH > 640) {
        _movie.deleteFrame();
    }
    _movie.endRecording();
}
```

Voir aussi

beginRecording(), endRecording(), Animation, updateFrame()

deleteGroup

Syntaxe

```
member(whichCastmember).deleteGroup(whichGroup)
member(whichCastmember).deleteGroup(index)
```

Description

Commande 3D ; supprime le groupe de l'acteur et de l'univers 3D. Les enfants du groupe sont retirés de l'univers 3D mais ne sont pas supprimés.

Il est impossible de supprimer le groupe Univers, qui est le groupe par défaut.

Paramètres

groupNameOrNum Requis. Chaîne ou nombre entier spécifiant le nom ou la position d'index du groupe à supprimer.

Exemple

La première ligne de cet exemple supprime le groupe Factice16 de l'acteur Séquence. La deuxième ligne supprime le troisième groupe de Séquence.

```
member("Scene").deleteGroup("Dummy16")
member("Scene").deleteGroup(3)
```

Voir aussi

newGroup, child (3D), parent

deleteLight

Syntaxe

```
member(whichCastmember).deleteLight(whichLight)
member(whichCastmember).deleteLight(index)
```

Description

Commande 3D ; supprime la lumière de l'acteur et de l'univers 3D. Les enfants de la lumière sont retirés de l'univers 3D mais ne sont pas supprimés.

Paramètres

lightNameOrNum Requis. Chaîne ou nombre entier spécifiant le nom ou la position d'index de la lumière à supprimer.

Exemple

Les exemples suivants suppriment les lumières de l'acteur Pièce.

```
member("Room").deleteLight("ambientRoomLight")
member("Room").deleteLight(6)
```

Voir aussi

newLight

deleteModel

Syntaxe

```
member(whichCastmember).deleteModel(whichModel)
member(whichCastmember).deleteModel(index)
```

Description

Commande 3D ; supprime le modèle de l'acteur et de l'univers 3D. Les enfants du modèle sont retirés de l'univers 3D mais ne sont pas supprimés.

Paramètres

modelNameOrNum Requis. Chaîne ou nombre entier spécifiant le nom ou la position d'index du modèle à supprimer.

Exemple

La première ligne de cet exemple supprime le modèle Lecteur3 de l'acteur gbUnivers. La deuxième ligne supprime le neuvième modèle de gbUnivers.

```
member("gbWorld").deleteModel("Player3")
member("gbWorld").deleteModel(9)
```

Voir aussi

newModel

deleteModelResource

Syntaxe

```
member(whichCastmember).deleteModelResource(whichModelResource)
member(whichCastmember).deleteModelResource(index)
```

Description

Commande 3D ; supprime la ressource de modèle de l'acteur et de l'univers 3D.

Les modèles qui utilisent la ressource de modèle supprimée deviennent invisibles car ils perdent leur géométrie, mais ne sont ni supprimés ni retirés de l'univers.

Paramètres

resourceNameOrNum Requis. Chaîne ou nombre entier spécifiant le nom ou la position d'index de la ressource de modèle à supprimer.

Exemple

Les exemples suivants suppriment deux ressources de modèle de l'acteur scèneDeRue.

```
member("StreetScene").deleteModelResource("HouseB")
member("StreetScene").deleteModelResource(3)
```

Voir aussi

newModelResource, newMesh

deleteMotion

Syntaxe

```
member(whichCastmember).deleteMotion(whichMotion)
member(whichCastmember).deleteMotion(index)
```

Description

Commande 3D ; supprime le mouvement de l'acteur.

Paramètres

modelNameOrNum Requis. Chaîne ou nombre entier spécifiant le nom ou la position d'index du mouvement à supprimer.

Exemple

La première ligne de l'exemple suivant supprime le mouvement Saut de l'acteur scèneDePique-nique. La deuxième ligne supprime le cinquième mouvement de scèneDePique-nique.

```
member("PicnicScene").deleteMotion("BackFlip")
member("PicnicScene").deleteMotion(5)
```

Voir aussi

newMotion(), removeLast()

deleteOne

Syntaxe

```
list.deleteOne(value)
deleteOne list, value
```

Description

Commande de liste ; supprime une valeur d'une liste linéaire ou de propriétés. Pour une liste de propriétés, `deleteOne` supprime également la propriété associée à la valeur supprimée. Si la valeur apparaît à plusieurs endroits de la liste, `deleteOne` ne supprime que la première occurrence.

L'utilisation de cette commande pour supprimer une propriété reste sans effet.

Lorsque vous ajoutez un filtre à l'aide de la méthode add ou append de la liste de filtres, un double du filtre est créé et ajouté à la liste. Les méthodes telles que deleteOne, getPos, findPos et getOne utilisent la valeur exacte de la liste et non la valeur dupliquée.

Dans ces cas précis, vous pouvez utiliser la commande deleteOne de la façon suivante :

```
f = filter(#glowfilter)
sprite(1).filterlist.append(f)
f = sprite(1).filterlist[1]-- here we get the actual value added to the list.
sprite(1).filterlist.deleteOne(f)
```

La troisième ligne du script ajoute la référence de la valeur de filtre à la liste.

Paramètres

value Requis. Valeur à supprimer de la liste.

Exemple

La première instruction suivante crée une liste contenant les jours mardi, mercredi et vendredi. La seconde instruction suivante supprime le nom mercredi de la liste.

```
--Lingo
days = ["Tuesday", "Wednesday", "Friday"]
days.deleteOne("Wednesday")
put days

// Javascript
days = list("Tuesday", "Wednesday", "Friday");
days.deleteOne("Wednesday");
trace(days);
```

L'instruction `put days` entraîne l'affichage du résultat dans la fenêtre Messages :

```
-- ["Tuesday", "Friday"].
```

deleteProp

Syntaxe

```
list.deleteProp(item)
deleteProp list, item
```

Description

Commande de liste ; supprime l'élément spécifié de la liste indiquée.

• Pour les listes linéaires, remplacez *item* par le numéro correspondant à la position de l'élément à supprimer dans la liste. La commande `deleteProp` dans les listes linéaires produit le même effet que la commande `deleteAt`. Un chiffre supérieur au nombre d'éléments de la liste entraîne une erreur de script.

• Pour les listes de propriétés, remplacez *item* par le nom de la propriété à supprimer. La suppression d'une propriété supprime également la valeur qui lui est associée. Si la liste contient la même propriété à plusieurs endroits, seule la première occurrence de la propriété est supprimée.

Paramètres

item Requis. Elément à supprimer de la liste.

Exemple

L'instruction suivante supprime la propriété `color` de la liste [#height:100, #width: 200, #color: 34, #ink: 15], appelée attributsDimageObjet :

```
spriteAttributes.deleteProp(#color)
```

Le résultat est la liste [#height:100, #width: 200, #ink: 15].

Voir aussi

deleteAt

deleteShader

Syntaxe

```
member(whichCastmember).deleteShader(whichShader)
member(whichCastmember).deleteShader(index)
```

Description

Commande 3D ; retire le matériau de l'acteur.

Paramètres

shaderNameOrNum Requis. Chaîne ou nombre entier spécifiant le nom ou la position d'index du matériau à supprimer.

Exemple

La première ligne de cet exemple supprime le matériau Route de l'acteur scèneDeRue. La deuxième ligne supprime le troisième matériau de scèneDeRue.

```
-- Lingo
member("StreetScene").deleteShader("Road")
member("StreetScene").deleteShader(3)

// Javascript
member("StreetScene").deleteShader("Road");
member("StreetScene").deleteShader(3);
```

Voir aussi

newShader, shaderList

deleteTexture

Syntaxe

```
member(whichCastmember).deleteTexture(whichTexture)
member(whichCastmember).deleteTexture(index)
```

Description

Commande 3D ; supprime la texture de l'acteur.

Paramètres

textureNameOrNum Requis. Chaîne ou nombre entier spécifiant le nom ou la position d'index de la texture à supprimer.

Exemple

La première ligne de cet exemple supprime la texture Ciel de l'acteur scèneDePique-nique. La deuxième ligne supprime la cinquième texture de scèneDePique-nique.

```
-- Lingo
member("PicnicScene).deleteTexture("Sky")
member("PicnicScene").deleteTexture(5)

// Javascript
member("PicnicScene).deleteTexture("Sky");
member("PicnicScene").deleteTexture(5)
```

Voir aussi

newTexture

deleteVertex()

Syntaxe

```
-- Lingo syntax
memberObjRef.deleteVertex(indexToRemove)

// JavaScript syntax
memberObjRef.deleteVertex(indexToRemove);
```

Description

Commande de forme vectorielle ; supprime un sommet existant d'un acteur forme vectorielle à la position d'index spécifiée.

Paramètres

indexToRemove Requis. Nombre entier spécifiant la position d'index du sommet à supprimer.

Exemple

La ligne suivante supprime le deuxième point de sommet de la forme vectorielle Archie :

```
-- Lingo syntax
member("Archie").deleteVertex(2)

// JavaScript syntax
member("Archie").deleteVertex(2);
```

Voir aussi

addVertex(), moveVertex(), originMode, vertexList

displayOpen()

Syntaxe

```
-- Lingo syntax
```

```
fileioObjRef.displayOpen()

// JavaScript syntax
fileioObjRef.displayOpen();
```

Description

Méthode FileIO ; affiche une boîte de dialogue d'ouverture.

Cette méthode renvoie le chemin d'accès complet et le nom du fichier sélectionné.

Paramètres

Aucune.

Voir aussi

Fileio

displaySave()

Syntaxe

```
-- Lingo syntax
fileioObjRef.displaySave(stringTitle, stringFileName)

// JavaScript syntax
fileioObjRef.displaySave(stringTitle, stringFileName);
```

Description

Méthode FileIO ; affiche une boîte de dialogue d'enregistrement.

Cette méthode renvoie le chemin d'accès complet et le nom du fichier enregistré.

Paramètres

stringTitle Requis. Chaîne spécifiant le titre affiché dans la boîte de dialogue d'enregistrement.

stringFileName Requis. Chaîne spécifiant le chemin d'accès complet et le nom du fichier à enregistrer.

Voir aussi

Fileio

do

Syntaxe

```
do stringExpression
```

Description

Commande ; évalue une chaîne et exécute le résultat sous la forme d'une instruction de script. Cette commande est pratique pour évaluer des expressions saisies par l'utilisateur et pour exécuter des commandes placées dans des variables chaînes, des champs, des listes et des fichiers.

L'utilisation de variables locales non initialisées dans une commande `do` entraîne une erreur de compilation. Il est conseillé d'initialiser les variables locales en avance.

Remarque : *Cette commande ne permet pas la déclaration de variables globales, celles-ci devant toujours être déclarées en avance.*

La commande `do` fonctionne avec des chaînes de plusieurs lignes ou d'une seule ligne.

Paramètres

stringExpression Requis. Chaîne à évaluer.

Exemple

L'instruction suivante exécute l'instruction placée entre guillemets :

```
do "beep 2"
do commandList[3]
```

doneParsing()

Syntaxe

```
parserObject.doneParsing()
```

Description

Fonction ; renvoie `1 (TRUE)` lorsque l'analyseur a achevé l'analyse d'un document avec `parseURL()`. La valeur renvoyée est `0 (FALSE)` jusqu'à la fin de l'analyse.

Paramètres

Aucune.

Voir aussi

`parseURL()`

dot()

Syntaxe

```
vector1.dot(vector2)
```

Description

Méthode 3D de vecteur ; renvoie la somme des produits des composants x, y et z de deux vecteurs. Si les deux vecteurs sont normalisés, le résultat de la méthode `dot` correspond au cosinus de l'angle entre les deux vecteurs.

Pour arriver manuellement à la valeur dot de deux vecteurs, multipliez le composant x de `vector1` par le composant x de `vector2`, multipliez le composant y de `vector1` par le composant y de `vector2`, multipliez le composant z de `vector1` par le composant z de `vector2`, puis additionnez les trois produits.

Cette méthode est identique à la fonction `dotProduct()`.

Paramètres

`vector2` Requis. Second vecteur à partir duquel une somme est renvoyée.

Exemple

Dans l'exemple suivant, l'angle entre les vecteurs `pos5` et `pos6` est de 45 degrés. La fonction getNormalized renvoie les valeurs normalisées de pos5 et de pos6 et les stocke dans les variables `norm1` et `norm2`. La valeur dot de `norm1` et de `norm2` est égale à 0,7071, ce qui constitue le cosinus de 45 degrés.

```
pos5 = vector(100, 100, 0)
pos6 = vector(0, 100, 0)
put pos5.angleBetween(pos6)
-- 45.0000
norm1 = pos5.getNormalized()
put norm1
-- vector( 0.7071, 0.7071, 0.0000 )
norm2 = pos6.getNormalized()
put norm2
-- vector( 0.0000, 1.0000, 0.0000 )
put norm1.dot(norm2)
-- 0.7071
```

Voir aussi

`dotProduct()`, `getNormalized`, `normalize`

dotProduct()

Syntaxe

```
vector1.dotProduct(vector2)
```

Description

Méthode 3D de vecteur ; renvoie la somme des produits des composants x, y et z de deux vecteurs. Si les deux vecteurs sont normalisés, la valeur `dotproduct` correspond au cosinus de l'angle entre les deux vecteurs.

Pour arriver manuellement à la valeur dot de deux vecteurs, multipliez le composant x de `vector1` par le composant x de `vector2`, multipliez le composant y de `vector1` par le composant y de `vector2`, multipliez le composant z de `vector1` par le composant z de `vector2`, puis additionnez les trois produits.

Cette méthode est identique à la fonction `dot()`.

Paramètres

`vector2` Requis. Second vecteur à partir duquel une somme est renvoyée.

Exemple

Dans l'exemple suivant, l'angle entre les vecteurs `pos5` et `pos6` est de 45 degrés. La fonction getNormalized renvoie les valeurs normalisées de `pos5` et de `pos6` et les stocke dans les variables `norm1` et `norm2`. La valeur `dotProduct` de `norm1` et de `norm2` est égale à 0,7071, ce qui constitue le cosinus de 45 degrés.

```
pos5 = vector(100, 100, 0)
pos6 = vector(0, 100, 0)
put pos5.angleBetween(pos6)
-- 45.0000
norm1 = pos5.getNormalized()
put norm1
-- vector( 0.7071, 0.7071, 0.0000 )
norm2 = pos6.getNormalized()
put norm2
-- vector( 0.0000, 1.0000, 0.0000 )
```

```
put norm1.dotProduct(norm2)
-- 0.7071
```

Voir aussi

dot(), getNormalized, normalize

downloadNetThing

Syntaxe

```
downloadNetThing URL, localFile
```

Description

Commande ; copie un fichier depuis Internet vers un fichier placé sur le disque local, tandis que la lecture de l'animation actuelle continue. Utilisez `netDone` pour vérifier si le téléchargement est terminé.

Les animations Director en mode auteur et les projections prennent en charge la commande `downLoadNetThing`, ce qui n'est pas le cas de Shockwave Player. Cela empêche les utilisateurs de copier accidentellement des fichiers à partir d'Internet.

Bien que plusieurs opérations réseau puissent avoir lieu en même temps, l'exécution de plus de quatre opérations simultanées réduit généralement les performances de façon inacceptable.

Ni la taille de la mémoire cache de l'animation Director, ni le paramétrage de l'option Vérifier les documents n'affectent le comportement de la commande `downloadNetThing`.

Paramètres

`URL` Requis. URL d'un objet téléchargeable : par exemple, un serveur FTP ou HTTP, une page HTML, un acteur externe, une animation Director ou un graphique.

`localFile` Requis. Chemin d'accès et nom du fichier sur le disque local.

Exemple

Les instructions suivantes téléchargent un acteur externe à partir d'une URL vers le dossier de Director et font de ce fichier l'acteur externe intitulé Distribution importante :

```
downLoadNetThing("http://www.cbDeMille.com/Thousands.cst",
the\applicationPath&"Thousands.cst")
castLib("Cast of Thousands").fileName = the applicationPath&"Thousands.cst"
```

Voir aussi

importFileInto(), netDone(), preloadNetThing()

draw()

Syntaxe

```
-- Lingo syntax
imageObjRef.draw(x1, y1, x2, y2, colorObjOrParamList)
imageObjRef.draw(point(x, y), point(x, y), colorObjOrParamList)
imageObjRef.draw(rect, colorObjOrParamList)

// JavaScript syntax
```

```
imageObjRef.draw(x1, y1, x2, y2, colorObjOrParamList);
imageObjRef.draw(point(x, y), point(x, y), colorObjOrParamList);
imageObjRef.draw(rect, colorObjOrParamList);
```

Description

Méthode d'image. Dessine une ligne ou une forme vide d'une couleur spécifiée dans une zone rectangulaire d'un objet image donné.

Cette méthode renvoie une valeur de 1 si aucune erreur n'est détectée.

En cas d'absence de liste de paramètres, la méthode `draw()` dessine une ligne de 1 pixel entre le premier et le second point donnés ou entre le coin supérieur gauche et le coin inférieur droit du rectangle donné.

Pour optimiser les performances avec des images 8 bits (ou d'une valeur inférieure), l'objet couleur doit contenir une valeur de couleur indexée. Pour les images 16 ou 32 bits, utilisez une valeur de couleur rvb.

Pour remplir une zone unie, utilisez la méthode `fill()`.

Paramètres

x1 Requis en cas de dessin d'une ligne à l'aide de coordonnées `x` et `y`. Nombre entier spécifiant la coordonnée `x` du début de la ligne.

x1 Requis en cas de dessin d'une ligne à l'aide de coordonnées `x` et `y`. Nombre entier spécifiant la coordonnée `y` du début de la ligne.

x2 Requis en cas de dessin d'une ligne à l'aide de coordonnées `x` et `y`. Nombre entier spécifiant la coordonnée `x` de la fin de la ligne.

x2 Requis en cas de dessin d'une ligne à l'aide de coordonnées `x` et `y`. Nombre entier spécifiant la coordonnée `y` de la fin de la ligne.

colorObjOrParamList Requis. Objet couleur ou liste de paramètres spécifiant la couleur de la ligne ou de la bordure de la forme. Vous pouvez utiliser la liste de paramètres plutôt qu'un objet couleur unique pour spécifier les propriétés suivantes.

Propriété	Description
#shapeType	Valeur de symbole de `#oval`, `#rect`, `#roundRect` ou `#line`. La valeur par défaut est #line.
#lineSize	Epaisseur du trait à utiliser pour le dessin de la forme.
#color	Objet couleur déterminant la couleur de la bordure de la forme.

`point(x, y), point(x, y)` Requis en cas de dessin d'une ligne à l'aide de points. Deux-points spécifiant les points de début et de fin de la ligne.

rect Requis en cas de dessin d'une forme. Rectangle spécifiant la zone rectangulaire dans laquelle une forme est dessinée.

Exemple

L'instruction suivante trace une ligne rouge foncé de 1 pixel entre le point (20, 20) et le point (20, 60) dans l'image de la scène.

```
-- Lingo
objImage = _movie.stage.image
objImage.draw(point(20, 20), point(20, 60), rgb(255, 0 ,0))
```

```
// Javascript
var objImage = _movie.stage.image ;
objImage.draw(point(20, 20), point(20, 60), color(255, 0 ,0)) ;
```

Voir aussi

color(), copyPixels(), fill(), image(), setPixel()

duplicate() (image)

Syntaxe

```
-- Lingo syntax
imageObjRef.duplicate()

// JavaScript syntax
imageObjRef.duplicate();
```

Description

Méthode d'image. Crée et renvoie une copie d'une image donnée.

La nouvelle image est complètement indépendante de l'originale et n'est liée à aucun acteur. Si vous projetez d'apporter un grand nombre de modifications à une image, il est recommandé d'en effectuer une copie indépendante d'un acteur.

Paramètres

Aucune.

Exemple

L'instruction suivante crée un objet image à partir de l'image de l'acteur Surface lunaire et place le nouvel objet image dans la variable `imageTemporaire` :

```
workingImage = member("Lunar Surface").image.duplicate()
```

Voir aussi

image()

duplicate() (fonction de liste)

Syntaxe

```
(oldList).duplicate()
duplicate(oldList)
```

Description

Fonction de liste ; renvoie la copie d'une liste et copie des listes imbriquées (éléments de listes qui sont eux-mêmes des listes) et leur contenu. Cette fonction se révèle utile pour enregistrer le contenu en cours d'une liste.

Remarque : *filterlist() ne peut pas être reproduit à l'aide de cette fonction.*

Lorsque vous affectez une liste à une variable, la variable contient une référence à la liste et non la liste même. Cela signifie que les modifications apportées à la copie affectent également l'original.

Vous pouvez voir un exemple d'utilisation de `duplicate() (list function)` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

ancienneListe Requis. Spécifie la liste à dupliquer.

Exemple

L'instruction suivante copie la liste ClientsDuJour et l'affecte à la variable `ListeDesClients` :

```
-- Lingo
CustomersToday = [1, 3, 5]
CustomerRecord = CustomersToday.duplicate()

// Javascript

CustomersToday = list(1, 3, 5);
CustomerRecord = CustomersToday.duplicate();
```

Voir aussi

`image()`

duplicate() (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.duplicate({intPosn})

// JavaScript syntax
memberObjRef.duplicate({intPosn});
```

Description

Méthode d'acteur ; effectue une copie d'un acteur spécifié.

Il est préférable d'utiliser cette méthode en mode création plutôt que pendant l'exécution, car elle crée un autre acteur en mémoire, ce qui risque d'entraîner des problèmes de mémoire.

Utilisez cette méthode pour enregistrer de façon permanente les modifications d'acteur avec le fichier.

Paramètres

intPosn Facultatif. Nombre entier spécifiant la fenêtre Distribution pour l'acteur dupliqué. Si vous omettez ce paramètre, l'acteur dupliqué est inséré dans la première position de la fenêtre Distribution ouverte.

Exemple

L'instruction suivante copie l'acteur Bureau et le place dans la première position vide de la fenêtre Distribution :

```
-- Lingo syntax
member("Desk").duplicate()

// JavaScript syntax
member("Desk").duplicate();
```

L'instruction suivante copie l'acteur Bureau et le place à la position 125 de la fenêtre Distribution :

```
-- Lingo syntax
member("Desk").duplicate(125)

// JavaScript syntax
member("Desk").duplicate(125);
```

Voir aussi

Acteur

duplicateFrame()

Syntaxe

```
-- Lingo syntax
_movie.duplicateFrame()

// JavaScript syntax
_movie.duplicateFrame();
```

Description

Méthode d'animation ; copie l'image en cours et son contenu, insère la copie après l'image en cours, puis fait de cette copie l'image en cours. Cette méthode n'est utilisable qu'en phase de création du scénario.

Cette méthode remplit la même fonction que la méthode `insertFrame()`.

Paramètres

Aucune.

Exemple

Lorsqu'elle est utilisée dans le gestionnaire suivant, la commande `duplicateFrame` crée une série d'images dans lesquelles l'acteur Balle de la distribution externe Jouets est affecté à la piste d'image-objet 20. Le nombre d'images est déterminé par l'argument `numberOfFrames`.

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    sprite(20).member = member("Ball", "Toys")
    repeat with i = 0 to numberOfFrames
        _movie.duplicateFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    sprite(20).member = member("Ball", "Toys");
    for (var i = 0; i <= numberOfFrames; i++) {
        _movie.duplicateFrame();
    }
    _movie.endRecording();
}
```

Voir aussi

insertFrame(), Animation

enableHotSpot()

Syntaxe

```
-- Lingo syntax
spriteObjRef.enableHotSpot(hotSpotID, trueOrFalse)

// JavaScript syntax
spriteObjRef.enableHotSpot(hotSpotID, trueOrFalse);
```

Description

Commande QTVR (QuickTime® VR) ; détermine si une zone référencée dans une image-objet QTVR est activée (`TRUE`) ou désactivée (`FALSE`).

Paramètres

`hotSpotID` Requis. Spécifie la zone référencée dans l'image-objet QTVR à tester.

`trueOrFalse` Requis. Valeur `TRUE` ou `FALSE` spécifiant si l'image-objet QTVR est activée.

endRecording()

Syntaxe

```
-- Lingo syntax
_movie.endRecording()

// JavaScript syntax
_movie.endRecording();
```

Description

Méthode d'animation ; met fin à une session de mise à jour du scénario.

Vous pouvez reprendre le contrôle des pistes du scénario à l'aide d'un script après avoir appelé la méthode `endRecording()`.

Paramètres

Aucune.

Exemple

Lorsqu'il est utilisé dans le gestionnaire suivant, le mot-clé `endRecording` met fin à la session de création du scénario :

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    horizontal = 0
    vertical = 100
    repeat with i = 1 to numberOfFrames
        _movie.go(i)
        sprite(20).member = member("Ball")
        sprite(20).locH = horizontal
        sprite(20).locV = vertical
        sprite(20).foreColor = 255
        horizontal = horizontal + 3
        vertical = vertical + 2
```

```
        _movie.updateFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    var horizontal = 0;
    var vertical = 100;
    for (var i = 1; i <= numberOfFrames; i++) {
        _movie.go(1);
        sprite(20).member = member("Ball");
        sprite(20).locH = horizontal;
        sprite(20).locV = vertical;
        sprite(20).foreColor = 255;
        horizontal = horizontal + 3;
        vertical = vertical + 2;
        _movie.updateFrame();
    }
    _movie.endRecording();
}
```

Voir aussi

beginRecording(), Animation, updateFrame()

erase()

Syntaxe

```
-- Lingo syntax
memberObjRef.erase()

// JavaScript syntax
memberObjRef.erase();
```

Description

Méthode d'acteur ; supprime un acteur spécifié et laisse son emplacement vide dans la fenêtre Distribution.

Pour obtenir de meilleurs résultats, utilisez cette méthode en phase de création et non dans les projections. L'utilisation de cette méthode dans les projections risque d'entraîner des problèmes de mémoire.

Paramètres

Aucune.

Exemple

L'instruction suivante supprime l'acteur Engrenage de la distribution Matériel :

```
-- Lingo syntax
member("Gear", "Hardware").erase()

// JavaScript syntax
member("Gear", "Hardware").erase();
```

Le gestionnaire suivant supprime les acteurs numérotés entre les valeurs `start` et `finish` :

```
-- Lingo syntax
```

```
on deleteMember start, finish
    repeat with i = start to finish
        member(i).erase()
    end repeat
end deleteMember

// JavaScript syntax
function deleteMember(start, finish) {
    for (var i=start; i<=finish; i++) {
        member(i).erase();
    }
}
```

Voir aussi

Acteur, new()

error()

Syntaxe

```
-- Lingo syntax
fileioObjRef.error(intError)

// JavaScript syntax
fileioObjRef.error(intError);
```

Description

Méthode FileIO ; renvoie un message d'erreur spécifié.

Paramètres

intErreur Requis. Nombre entier spécifiant l'erreur. Les valeurs possibles sont 0 (« OK ») ou 1 (« Echec d'allocation mémoire »). Toutes les autres valeurs renvoient « Erreur inconnue ».

Exemple

L'exemple suivant crée un fichier intitulé c:\vérif.txt. En cas d'erreur lors de la création du fichier, cette méthode renverrait l'erreur ci-après.

```
-- Lingo syntax
objFileio = new xtra("fileio")
isok = objFileio.createFile("c:\check.txt")
if ( isok = 0 ) then
    alert( " file creation successful")
else
    alert( " file creation failed " & objFileio.error(isok))
end if

// JavaScript syntax
var objFileio = new xtra("fileio");
isok = objFileio.createFile("c:\check.txt");
if ( isok == 0 )
{
    _player.alert( " file creation successful");
}
else
{
    _player.alert( " file creation failed " + objFileio.error(isok));
}
```

Voir aussi

Fileio

externalEvent()

Syntaxe

```
externalEvent "string"
```

Description

Commande ; envoie une chaîne au navigateur web pour qu'il l'interprète comme une instruction de script, permettant la lecture d'une animation ou la communication avec la page HTML dans laquelle elle est intégrée.

Cette commande fonctionne uniquement pour les animations dans des navigateurs.

Remarque : *La commande* `externalEvent` *ne produit pas de valeur de renvoi. Il n'existe aucune façon immédiate permettant de déterminer si le navigateur a traité l'événement ou l'a ignoré. Utilisez* `on EvalScript` *dans le navigateur pour renvoyer un message à l'animation.*

Paramètres

`string` Requis. Chaîne à envoyer au navigateur. Cette chaîne doit être indiquée dans un langage de programmation pris en charge par le navigateur.

Exemple

Les instructions suivantes utilisent `externalEvent` dans l'environnement de programmation LiveConnect qui est pris en charge par Netscape 3.x et ses versions ultérieures.

LiveConnect évalue la chaîne transmise par `externalEvent` comme un appel de fonction. Les auteurs utilisant JavaScript doivent définir et nommer cette fonction dans l'en-tête HTML. Dans l'animation, le nom et les paramètres de la fonction sont définis sous la forme d'une chaîne dans `externalEvent`. Les paramètres devant être interprétés par le navigateur web comme des chaînes distinctes, chaque paramètre est encadré de guillemets droits simples.

Instructions dans le code HTML :

```
function MyFunction(parm1, parm2) {
    //script here
}
```

Instructions dans un script dans l'animation :

```
externalEvent ("MyFunction('parm1','parm2')")
```

Les instructions suivantes utilisent `externalEvent` dans l'environnement de programmation ActiveX utilisé par Internet Explorer® sous Windows®. ActiveX interprète `externalEvent` comme un événement et traite cet événement et son paramètre de chaîne de la même façon qu'un événement `onClick` dans un objet bouton.

- Instructions dans le code HTML :

```
Sub
NameOfShockwaveInstance_externalEvent(aParam)
    'script here
End Sub
```

Vous pouvez également définir un script pour l'événement :

```
<SCRIPT FOR="NameOfShockwaveInstance"
EVENT="externalEvent(aParam)"
```

```
LANGUAGE="VBScript">
 'script here
</SCRIPT>
```

Dans l'animation, incluez la fonction et les paramètres dans la chaîne pour la méthode `externalEvent` :

```
externalEvent ("MyFunction ('parm1','parm2')")
```

Voir aussi

on EvalScript

extrude3D

Syntaxe

```
member(whichTextCastmember).extrude3D(member(which3dCastmember))
```

Description

Commande 3D ; crée une ressource de modèle `#extruder` dans un acteur 3D à partir du texte d'un acteur texte.

Cette opération n'équivaut pas à utiliser la propriété 3D `displayMode` d'un acteur texte.

Pour créer un modèle avec extrude3D :

1. Créez une ressource de modèle `#extruder` dans un acteur 3D :

   ```
   textResource = member("textMember").extrude3D(member("3DMember"))
   ```

2. Créez un modèle à l'aide de la ressource de modèle créée à l'étape 1 :

   ```
   member("3DMember").newModel("myText", textResource)
   ```

Paramètres

`which3dCastmember` Requis. Acteur dans lequel une ressource de modèle `#extruder` est créée.

Exemple

Dans l'exemple suivant, Logo est un acteur texte et Séquence est un acteur 3D. La première ligne crée une ressource de modèle dans Séquence, qui est une version 3D du texte de Logo. La seconde ligne utilise cette ressource de modèle pour créer un modèle nommé logo3D.

```
-- Lingo
myTextModelResource =member("Logo").extrude3d(member("Scene"))
member("Scene").newModel("3dLogo", myTextModelResource)

// Javascript
myTextModelResource =member("Logo").extrude3d(member("Scene"));
member("Scene").newModel("3dLogo", myTextModelResource);
```

Voir aussi

bevelDepth, bevelType, displayFace, smoothness, tunnelDepth, displayMode

externalParamName()

Syntaxe

```
-- Lingo syntax
```

```
_player.externalParamName(paramNameOrNum)

// JavaScript syntax
_player.externalParamName(paramNameOrNum);
```

Description

Méthode de lecteur ; renvoie le nom d'un paramètre spécifié dans la liste de paramètres externes d'une balise HTML <EMBED> ou <OBJECT>.

En cas de spécification d'un paramètre par son nom, cette méthode renvoie tous les noms de paramètre correspondant à *paramNameOrNum*. Elle ne distingue pas les majuscules des minuscules. Si aucun nom de paramètre correspondant n'est trouvé, cette méthode renvoie la valeur `VOID` (Lingo) ou `null` (syntaxe JavaScript).

En cas de spécification d'un paramètre par son numéro, cette méthode renvoie le nom de paramètre figurant à la position *paramNameOrNum* dans la liste de paramètres. Si aucune position de paramètre correspondante n'est trouvée, cette méthode renvoie la valeur `VOID` ou `null`.

Cette méthode n'est valide que pour les animations avec contenu Shockwave exécutées dans un navigateur. Elle ne peut pas être utilisée avec des animations Director ou des projections.

La liste suivante décrit les paramètres externes prédéfinis utilisables.

Paramètre	**Définition**
swAudio	Chaîne spécifiant l'emplacement d'un fichier Shockwave Audio à lire avec l'animation. Cette valeur est une URL complète.
swBackColor	Valeur de couleur destinée à modifier la propriété de couleur de la scène de l'animation. Cette valeur est un nombre entier compris entre 0 et 255. Utilisez une valeur de 0 à 255 pour les animations en couleur 8 bits et de 0 à 15 pour les animations en couleur 4 bits.
swBanner	Chaîne spécifiant le texte à utiliser sous la forme d'un bandeau dans l'animation.
swColor	Valeur de couleur à utiliser pour la modification de la couleur d'un objet spécifique. Cette valeur est un nombre entier compris entre 0 et 255. Utilisez une valeur de 0 à 255 pour les animations en couleur 8 bits et de 0 à 15 pour les animations en couleur 4 bits.
swForeColor	Nouvelle valeur de couleur du premier plan. Le texte apparaissant dans les acteurs champ est rendu dans la couleur du premier plan active. Cette valeur est un nombre entier compris entre 0 et 255. Utilisez une valeur de 0 à 255 pour les animations en couleur 8 bits et de 0 à 15 pour les animations en couleur 4 bits.
swFrame	Valeur de chaîne correspondant au nom affecté à une image donnée de l'animation.
swList	Liste séparée par des virgules répertoriant les éléments analysables à l'aide d'un script. Les valeurs de cette liste peuvent être des paires touche/valeur, des éléments booléens, des nombres entiers ou des chaînes.
swName	Nom, tel qu'un nom d'utilisateur, à afficher ou à utiliser dans l'animation.
swPassword	Mot de passe, éventuellement combiné à la propriété `swName`, à utiliser dans l'animation.
swPreloadTime	Nombre entier spécifiant le nombre de secondes d'un fichier audio à précharger avant que la lecture du son correspondant ne commence. Utilisée avec Shockwave Audio, cette valeur permet d'améliorer les performances de lecture en augmentant la quantité de données audio déjà téléchargées avant le début de la lecture.
swSound	Valeur de chaîne pouvant spécifier le nom d'un son dans l'animation Director à lire ou indiquant si un son doit être lu ou non.
swText	Valeur de chaîne spécifiant le texte à utiliser dans l'animation.

Paramètre (Suite)	Définition (Suite)
swURL	URL pouvant spécifier l'emplacement d'une autre animation comportant du contenu Shockwave ou d'un fichier Shockwave Audio.
swVolume	Nombre entier (l'utilisation d'une valeur entre 0 et 10 est recommandée) utilisé pour contrôler le niveau de volume du son sortant de l'animation. La valeur 0 indique que le son est désactivé (absence de son) et la valeur 10 correspond au volume maximal.
`sw1` à `sw9`	Neuf propriétés supplémentaires pour les paramètres définis par l'auteur.

Paramètres

`paramNameOrNum` Requis. Chaîne spécifiant le nom du paramètre à renvoyer ou nombre entier indiquant la position d'index du nom de paramètre à renvoyer.

Exemple

L'instruction suivante place la valeur d'un paramètre externe donné dans la variable `myVariable` :

```
-- Lingo syntax
if (_player.externalParamName("swURL") = "swURL") then
    myVariable = _player.externalParamValue("swURL")
end if

// JavaScript syntax
if (_player.externalParamName("swURL") == "swURL") {
    var myVariable = _player.externalParamName("swURL");
}
```

Voir aussi

`externalParamValue()`, `Animation`

externalParamValue()

Syntaxe

```
-- Lingo syntax
_player.externalParamValue(paramNameOrNum)

// JavaScript syntax
_player.externalParamValue(paramNameOrNum);
```

Description

Renvoie la valeur d'un paramètre spécifié dans la liste de paramètres externes d'une balise HTML <EMBED> ou <OBJECT>.

En cas de spécification d'une valeur de paramètre par son nom, cette méthode renvoie la valeur du premier paramètre dont le nom correspond à *`paramNameOrNum`*. Elle ne distingue pas les majuscules des minuscules. Si aucune valeur de paramètre correspondante n'est trouvée, cette méthode renvoie la valeur `VOID` (Lingo) ou `null` (syntaxe JavaScript).

En cas de spécification d'une valeur de paramètre par son index, cette méthode renvoie la valeur du paramètre figurant à la position *`paramNameOrNum`* dans la liste de paramètres. Si aucune position de paramètre correspondante n'est trouvée, cette méthode renvoie la valeur `VOID` ou `null`.

Cette méthode n'est valide que pour les animations avec contenu Shockwave exécutées dans un navigateur. Elle ne peut pas être utilisée avec des animations Director ou des projections.

La liste suivante décrit les paramètres externes prédéfinis utilisables.

Paramètre	Définition
swAudio	Chaîne spécifiant l'emplacement d'un fichier Shockwave Audio à lire avec l'animation. Cette valeur est une URL complète.
swBackColor	Valeur de couleur destinée à modifier la propriété de couleur de la scène de l'animation. Cette valeur est un nombre entier compris entre 0 et 255. Utilisez une valeur de 0 à 255 pour les animations en couleur 8 bits et de 0 à 15 pour les animations en couleur 4 bits.
swBanner	Chaîne spécifiant le texte à utiliser sous la forme d'un bandeau dans l'animation.
swColor	Valeur de couleur à utiliser pour la modification de la couleur d'un objet spécifique. Cette valeur est un nombre entier compris entre 0 et 255. Utilisez une valeur de 0 à 255 pour les animations en couleur 8 bits et de 0 à 15 pour les animations en couleur 4 bits.
swForeColor	Nouvelle valeur de couleur du premier plan. Le texte apparaissant dans les acteurs champ est rendu dans la couleur du premier plan active. Cette valeur est un nombre entier compris entre 0 et 255. Utilisez une valeur de 0 à 255 pour les animations en couleur 8 bits et de 0 à 15 pour les animations en couleur 4 bits.
swFrame	Valeur de chaîne correspondant au nom affecté à une image donnée de l'animation.
swList	Liste séparée par des virgules répertoriant les éléments analysables à l'aide d'un script. Les valeurs de cette liste peuvent être des paires touche/valeur, des éléments booléens, des nombres entiers ou des chaînes.
swName	Nom, tel qu'un nom d'utilisateur, à afficher ou à utiliser dans l'animation.
swPassword	Mot de passe, éventuellement combiné à la propriété `swName`, à utiliser dans l'animation.
swPreloadTime	Nombre entier spécifiant le nombre de secondes d'un fichier audio à précharger avant que la lecture du son correspondant ne commence. Utilisée avec Shockwave Audio, cette valeur permet d'améliorer les performances de lecture en augmentant la quantité de données audio déjà téléchargées avant le début de la lecture.
swSound	Valeur de chaîne pouvant spécifier le nom d'un son dans l'animation Director à lire ou indiquant si un son doit être lu ou non.
swText	Valeur de chaîne spécifiant le texte à utiliser dans l'animation.
swURL	URL pouvant spécifier l'emplacement d'une autre animation Shockwave ou d'un fichier Shockwave Audio.
swVolume	Nombre entier (l'utilisation d'une valeur entre 0 et 10 est recommandée) utilisé pour contrôler le niveau de volume du son sortant de l'animation. La valeur 0 indique que le son est désactivé (absence de son) et la valeur 10 correspond au volume maximal.
`sw1` à `sw9`	Neuf propriétés supplémentaires pour les paramètres définis par l'auteur.

Paramètres

`paramNameOrNum` Requis. Chaîne spécifiant le nom de la valeur de paramètre à renvoyer ou nombre entier indiquant la position d'index de la valeur de paramètre à renvoyer.

Exemple

L'instruction suivante place la valeur d'un paramètre externe donné dans la variable `myVariable` :

```
-- Lingo syntax
if (_player.externalParamName("swURL") = "swURL") then
    myVariable = _player.externalParamValue("swURL")
end if

// JavaScript syntax
```

```
if (_player.externalParamName("swURL") == "swURL") {
    var myVariable = _player.externalParamName("swURL");
}
```

Voir aussi

externalParamName(), Animation

extractAlpha()

Syntaxe

```
imageObject.extractAlpha()
```

Description

Cette fonction copie la couche alpha de l'image 32 bits donnée et la renvoie sous forme d'un nouvel objet image. Il en résulte une image 8 bits faite de niveaux de gris, représentant la couche alpha.

Cette fonction se révèle utile pour le sous-échantillonnage des images 32 bits avec des couches alpha.

Exemple

L'instruction suivante place la couche alpha de l'image de l'acteur 1 dans la variable `mainAlpha` :

```
-- Lingo
mainAlpha = member(1).image.extractAlpha()

// Javascript
mainAlpha = member(1).image.extractAlpha();
```

Voir aussi

setAlpha()

fadeIn()

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.fadeIn({intMilliseconds})

// JavaScript syntax
soundChannelObjRef.fadeIn({intMilliseconds});
```

Description

Méthode de piste audio ; règle immédiatement le `volume` d'une piste audio sur zéro, puis rétablit le volume en cours pendant un nombre de millisecondes donné.

Le paramètre de balance actuel est conservé pour l'ensemble du fondu.

Paramètres

`intMilliseconds` Facultatif. Nombre entier spécifiant le nombre de millisecondes pendant lequel le volume est rétabli à sa valeur d'origine. Si aucune valeur n'est spécifiée, la valeur par défaut est de 1 000 millisecondes (1 seconde).

Exemple

L'instruction Lingo suivante amplifie le son de la piste 3 pendant 3 secondes à partir du début de l'acteur `introMusic2` :

```
-- Lingo syntax
sound(3).play(member("introMusic2"))
sound(3).fadeIn(3000)

// JavaScript syntax
sound(3).play(member("introMusic2"));
sound(3).fadeIn(3000);
```

Voir aussi

`fadeOut()`, `fadeTo()`, `pan`, `Piste audio`, `volume (Windows Media)`

fadeOut()

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.fadeOut({intMilliseconds})

// JavaScript syntax
soundChannelObjRef.fadeOut({intMilliseconds});
```

Description

Méthode de piste audio ; abaisse progressivement le `volume` d'une piste audio jusqu'au volume zéro pendant un nombre de millisecondes donné.

Le paramètre de balance actuel est conservé pour l'ensemble du fondu.

Paramètres

`intMilliseconds` Facultatif. Nombre entier spécifiant le nombre de millisecondes pendant lequel le volume est réduit à zéro. Si aucune valeur n'est spécifiée, la valeur par défaut est de 1 000 millisecondes (1 seconde).

Exemple

L'instruction suivante diminue le son de la piste 3 pendant 5 secondes :

```
-- Lingo syntax
sound(3).fadeOut(5000)

// JavaScript syntax
sound(3).fadeOut(5000);
```

Voir aussi

`fadeIn()`, `fadeTo()`, `pan`, `Piste audio`, `volume (Windows Media)`

fadeTo()

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.fadeTo(intVolume {, intMilliseconds})
```

```
// JavaScript syntax
soundChannelObjRef.fadeTo(intVolume {, intMilliseconds});
```

Description

Méthode de piste audio ; remplace progressivement le `volume` d'une piste audio par un volume spécifié pendant un nombre de millisecondes donné.

Le paramètre de balance actuel est conservé pour l'ensemble du fondu.

Vous pouvez voir un exemple d'utilisation de `fadeTo()` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

`intVolume` Requis. Nombre entier spécifiant le niveau de volume de remplacement. Les valeurs possibles du volume indiqué par *`entVolume`* sont comprises entre 0 et 255.

`intMilliseconds` Facultatif. Nombre entier spécifiant le nombre de millisecondes pendant lequel le volume est remplacé par *`intVolume`*. Si aucune valeur n'est spécifiée, la valeur par défaut est de 1 000 millisecondes (1 seconde).

Exemple

L'instruction suivante fait passer le volume de la piste audio 4 à 150, sur une période de 2 secondes. Il peut s'agir d'une augmentation ou d'une diminution de volume, en fonction du volume initial de la piste audio 4 au moment de départ du fondu.

```
-- Lingo syntax
sound(4).fadeTo(150, 2000)

// JavaScript syntax
sound(4).fadeTo(150, 2000);
```

Voir aussi

`fadeIn()`, `fadeOut()`, `pan`, `Piste audio`, `volume (Windows Media)`

fileName()

Syntaxe

```
-- Lingo syntax
fileioObjRef.fileName()

// JavaScript syntax
fileioObjRef.fileName();
```

Description

Méthode FileIO ; renvoie le chemin d'accès complet et le nom d'un fichier ouvert.

Vous devez avoir ouvert un fichier à l'aide de la méthode `openFile()` avant d'utiliser la méthode `fileName()` pour obtenir le nom de ce fichier.

Paramètres

Aucune.

Exemple

L'instruction suivante crée un fichier, puis imprime l'emplacement de ce dernier.

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.createFile(_player.ApplicationPath)
put objFileio.fileName()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.createFile(_player.ApplicationPath);
trace(objFileio.fileName());
```

Voir aussi

Fileio , openFile()

fileOpen()

Syntaxe

```
FileOpen(MUIObject, string)
```

Description

Cette fonction affiche une boîte de dialogue d'ouverture de fichier standard, fournie par une instance de l'Xtra MUI.

Le deuxième paramètre spécifie une chaîne qui s'affiche dans le champ modifiable, lors de l'ouverture de la boîte de dialogue. L'utilisateur peut spécifier le fichier à ouvrir en saisissant son nom dans le champ modifiable. Lorsque l'utilisateur clique sur un bouton, le texte est renvoyé.

- Si l'utilisateur clique sur Annuler, le texte renvoyé est identique à la valeur passée.
- Si l'utilisateur clique sur OK, le texte renvoyé est un emplacement spécifique à la plate-forme.

Paramètres

Aucune.

Exemple

Ces instructions permettent la création et l'affichage d'une boîte de dialogue d'ouverture de fichier standard.

- La première instruction crée une instance d'Xtra MUI, qui est l'objet utilisé en tant que boîte de dialogue.
- La deuxième instruction attribue une chaîne à la variable fileString, utilisée ultérieurement en tant que deuxième paramètre de la commande FileOpen.
- La troisième instruction utilise la commande FileOpen pour générer une boîte de dialogue d'ouverture de fichier.
- Les dernières instructions vérifient que la chaîne d'origine envoyée avec la commande FileOpen est identique à celle renvoyée lorsque l'utilisateur a cliqué sur un bouton. Si les valeurs sont différentes, cela signifie que l'utilisateur a sélectionné un fichier à ouvrir.

```
-- Lingo syntax
set aMuiObj = new (xtra "MUI")

set fileString = "Open this file"

set result = fileOpen(aMuiObj, fileString)

-- Check to see if the dialog was canceled

if (result <> fileString) then
   -- The dialog wasn't canceled, do something with the new path data.
```

```
else
   put "ERROR - fileOpen requires a valid aMuiObj"
end if
```

Voir aussi

Fileio , openFile()

fileSave()

Syntaxe

```
FileSave( MUIObject, string, message )
```

Description

Cette fonction affiche une boîte de dialogue d'enregistrement de fichier standard qui enregistre le fichier actuel. La boîte de dialogue est créée depuis une instance de l'Xtra MUI.

- Le paramètre de chaîne spécifie la chaîne qui s'affiche dans le champ de nom de fichier de la boîte de dialogue. L'utilisateur peut utiliser ce champ afin de saisir un nouveau nom pour le fichier. Lorsque l'utilisateur clique sur un bouton, Lingo renvoie une valeur pour la chaîne qui inclut le contenu du champ. Si l'utilisateur clique sur Annuler, la chaîne renvoyée est identique à la chaîne d'origine.
- Le paramètre de message est la chaîne qui s'affiche au-dessus du champ modifiable de la boîte de dialogue.

Paramètres

Aucune.

Exemple

Ces instructions permettent la création et l'affichage d'une boîte de dialogue d'enregistrement de fichier.

- La première instruction crée une instance d'Xtra MUI, qui est l'objet utilisé en tant que boîte de dialogue.
- La deuxième instruction attribue une chaîne à la variable fileString, utilisée ultérieurement en tant que deuxième paramètre de la commande FileSave.
- La troisième instruction utilise la commande FileSave pour générer une boîte de dialogue d'enregistrement de fichier.
- Les dernières instructions vérifient que le résultat obtenu lorsque l'utilisateur clique sur un bouton est identique à la chaîne envoyée lors de l'ouverture de la boîte de dialogue. Si le résultat est différent, cela signifie que l'utilisateur a cliqué sur un autre bouton qu'Annuler.

```
-- Lingo syntax
set aMuiObj = new (xtra "MUI")

set fileString = "Save this file"

set result = fileSave(aMuiObj, fileString, "with this prompt")

-- Check to see if the dialog was canceled

if (result <> fileString) then
   -- The dialog wasn't canceled, do something with the new path data.
end if
```

fill()

Syntaxe

```
-- Lingo syntax
imageObjRef.fill(left, top, right, bottom, colorObjOrParamList)
imageObjRef.fill(point(x, y), point(x, y), colorObjOrParamList)
imageObjRef.fill(rect, colorObjOrParamList)

// JavaScript syntax
imageObjRef.fill(left, top, right, bottom, colorObjOrParamList);
imageObjRef.fill(point(x, y), point(x, y), colorObjOrParamList);
imageObjRef.fill(rect, colorObjOrParamList);
```

Description

Méthode d'image. Remplit une zone rectangulaire avec une couleur spécifiée dans un objet image donné.

Cette méthode renvoie la valeur 1 si aucune erreur n'est détectée, et la valeur zéro en cas d'erreur.

Pour optimiser les performances avec des images 8 bits (ou d'une valeur inférieure), l'objet couleur doit contenir une valeur de couleur indexée. Pour les images 16 ou 32 bits, utilisez une valeur de couleur RVB.

Paramètres

left Requis en cas de remplissage d'une zone spécifiée par des coordonnées. Nombre entier spécifiant le côté gauche de la zone à remplir.

top Requis en cas de remplissage d'une zone spécifiée par des coordonnées. Nombre entier spécifiant le côté supérieur de la zone à remplir.

right Requis en cas de remplissage d'une zone spécifiée par des coordonnées. Nombre entier spécifiant le côté droit de la zone à remplir.

bottom Requis en cas de remplissage d'une zone spécifiée par des coordonnées. Nombre entier spécifiant le côté inférieur de la zone à remplir.

colorObjOrParamList Requis. Objet couleur ou liste de paramètres spécifiant la couleur utilisée pour remplir la zone. Vous pouvez utiliser la liste de paramètres plutôt qu'un objet couleur unique pour spécifier les propriétés suivantes.

Propriété	Description
#shapeType	Valeur de symbole de `#oval`, `#rect`, `#roundRect` ou `#line`. La valeur par défaut est #line.
#lineSize	Epaisseur du trait à utiliser pour le dessin de la forme.
#color	Objet couleur déterminant la couleur de remplissage de la zone.
#bgColor	Objet couleur déterminant la couleur de la bordure de la zone.

`point(x, y), point(x, y)` Requis en cas de remplissage d'une zone à l'aide de points. deux-points spécifiant les coins supérieur gauche et inférieur droit de la zone à remplir par rapport au coin supérieur gauche de l'objet image donné.

rect Requis en cas de remplissage d'une zone à l'aide d'un rectangle. Rectangle spécifiant la zone rectangulaire à remplir.

Exemple

L'instruction suivante remplit une zone spécifiée par deux-points correspondant aux coins supérieur gauche et inférieur droit de la zone à remplir par rapport au coin supérieur gauche de l'objet image donné, ici l'objet scène.

```
-- Lingo
objImage = _movie.stage.image
objImage.fill(point(20, 20), point(30, 60), rgb(255, 0 ,0))

// Javascript
var objImage = _movie.stage.image ;
objImage.fill(point(20, 20), point(30, 60), color(255, 0 ,0)) ;
```

Voir aussi

color(), draw(), image()

filter()

Syntaxe

```
filter(filter symbol)
```

Description

Méthode Filtres de bitmaps, utilisée pour la création de filtres de bitmaps et leur utilisation sur une image-objet.

Exemple

La première instruction définit la variable dénommée monFiltre sur le filtre de flou. La ligne suivante définit le filtre de flou sur sprite(1).

```
--Lingo syntax
MyFilter=filter(#BlurFilter)
sprite(1).filterlist.append(MyFilter)

// JavaScript syntax
var MyFilter = filter(symbol("BlurFilter"));
sprite(1).filterlist.append(MyFilter);
```

Voir aussi

Filtres de bitmaps dans le mode d'emploi de Director.

findLabel()

Syntaxe

```
-- Lingo syntax
spriteObjRef.findLabel(whichLabelName)

// JavaScript syntax
spriteObjRef.findLabel(whichLabelName);
```

Description

Fonction : cette fonction renvoie le numéro d'image (dans l'animation Flash) associé au libellé demandé.

Un 0 est renvoyé si le libellé n'existe pas ou si cette partie de l'animation Flash n'est pas encore transférée en mémoire.

Paramètres

whichLabelName Requis. Spécifie le libellé d'image à rechercher.

Exemple

L'instruction suivante renvoie le numéro d'image (dans l'animation Flash de l'acteur(1)) associé au libellé « ObtenirImage ».

```
-- Lingo syntax
sprite(1).findLabel("GetMe")

// JavaScript syntax
sprite(1).findLabel("GetMe");
```

findEmpty()

Syntaxe

```
-- Lingo syntax
castObjRef.findEmpty({memberObjRef})

// JavaScript syntax
castObjRef.findEmpty({memberObjRef});
```

Description

Méthode de bibliothèque de distribution ; affiche la prochaine position d'acteur vide ou la position qui suit un acteur spécifié.

Cette méthode n'est disponible que dans la bibliothèque de distribution en cours.

Paramètres

memberObjRef Facultatif. Référence à l'acteur après lequel la prochaine position d'acteur vide apparaît. Si ce paramètre est omis, la prochaine position d'acteur vide est affichée.

Exemple

L'instruction suivante recherche le premier acteur vide à partir de l'acteur 100 :

```
-- Lingo syntax
trace(castLib(1).findEmpty(member(100)))

// JavaScript syntax
trace(castLib(1).findEmpty(member(100)));
```

Voir aussi

Bibliothèque de distribution, Acteur

findPos

Syntaxe

```
list.findPos(property)
findPos(list, property)
```

Description

Commande de liste ; identifie la position d'une propriété dans une liste de propriétés.

L'utilisation de `findPos` avec des listes linéaires renvoie un nombre fictif si la valeur de *`property`* est un nombre et une erreur de script si la valeur de *`property`* est une chaîne.

La commande `findPos` remplit la même fonction que la commande `findPosNear`, à l'exception du fait que `findPos` présente la valeur `VOID` lorsque la propriété spécifiée ne figure pas dans la liste.

Lorsque vous ajoutez un filtre à l'aide de la méthode add ou append de la liste de filtres, un double du filtre est créé et ajouté à la liste. Les méthodes telles que deleteOne, getPos, findPos et getOne utilisent la valeur exacte de la liste et non la valeur dupliquée.

Dans ces cas précis, vous pouvez utiliser la commande findPos de la façon suivante :

```
f = filter(#glowfilter)
sprite(1).filterlist.append(f)
f = sprite(1).filterlist[1]-- here we get the actual value added to the list.
sprite(1).filterlist.findPos(f)
```

La troisième ligne du script ci-dessus ajoute la référence de la valeur de filtre à la liste.

Paramètres

`property` Requis. Propriété dont la position est identifiée.

Exemple

L'instruction suivante identifie la position de la propriété `c` dans la liste Réponses, composée de `[#a:10, #b:12, #c:15, #d:22]`:

```
-- Lingo
Answers.findPos(#c)

// Javascript
Answers.findPos("c");
```

Le résultat est 3 car `c` constitue la troisième propriété de la liste.

Voir aussi

`findPosNear`, `sort`

findPosNear

Syntaxe

```
sortedList.findPosNear(valueOrProperty)
findPosNear(sortedList, valueOrProperty)
```

Description

Commande de liste ; pour les listes triées uniquement, identifie la position d'un élément dans une liste triée spécifiée.

La commande `findPosNear` fonctionne uniquement avec les listes triées. Remplacez *`valueOrProperty`* par une valeur pour les listes linéaires triées et par une propriété pour les listes de propriétés triées.

La commande `findPosNear` est semblable à la commande `findPos`, à l'exception du fait que, lorsque la propriété spécifiée ne figure pas dans la liste, la commande `findPosNear` identifie la position de la valeur portant le nom alphanumérique le plus similaire. Cette commande est pratique pour trouver le nom le plus proche dans un répertoire de noms triés.

Paramètres

`valueOrProperty` Requis. Valeur ou propriété dont la position est identifiée.

Exemple

L'instruction suivante identifie la position d'une propriété dans la liste triée Réponses, composée de `[#Nile:2, #Pharaoh:4, #Raja:0]`:

```
Answers.findPosNear(#Ni)
```

Le résultat est égal à 1, Ni étant le plus proche de `Nile` qui constitue la première propriété de la liste.

Voir aussi

`findPos`

finishIdleLoad()

Syntaxe

```
-- Lingo syntax
_movie.finishIdleLoad(intLoadTag)

// JavaScript syntax
_movie.finishIdleLoad(intLoadTag);
```

Description

Méthode d'animation ; demande le chargement de tous les acteurs comportant la balise de chargement spécifiée.

Paramètres

`intLoadTag` Requis. Nombre entier spécifiant la balise de chargement des acteurs à charger.

Exemple

L'instruction suivante termine le chargement de tous les acteurs possédant la balise de chargement 20 :

```
-- Lingo syntax
_movie.finishIdleLoad(20)

// JavaScript syntax
_movie.finishIdleLoad(20);
```

Voir aussi

`idleHandlerPeriod`, `idleLoadDone()`, `idleLoadMode`, `idleLoadPeriod`, `idleLoadTag`, `idleReadChunkSize`, `Animation`

flashToStage()

Syntaxe

```
-- Lingo syntax
```

```
spriteObjRef.flashToStage(pointInFlashMovie)

// JavaScript syntax
spriteObjRef.flashToStage(pointInFlashMovie);
```

Description

Fonction ; renvoie la coordonnée de la scène Director correspondant à une coordonnée spécifiée dans une image-objet d'animation Flash. Cette fonction accepte la coordonnée de piste et d'animation Flash et renvoie la coordonnée de scène Director en valeurs de points Director : par exemple, point (300, 300).

Les coordonnées d'animation Flash sont mesurées en pixels d'animation Flash, qui sont déterminés par la taille d'origine d'une animation lors de sa création dans Flash. Afin de calculer les coordonnées de l'animation Flash, le point (0, 0) d'une animation Flash est toujours son angle supérieur gauche. La propriété `originPoint` de l'acteur est utilisée uniquement pour la rotation et la mise à l'échelle, et non pour le calcul des coordonnées d'une animation.

La fonction `flashToStage` et la fonction `stageToFlash` correspondante se révèlent utiles pour déterminer la coordonnée d'une animation Flash directement située sur une coordonnée spécifique de la scène Director. Pour Flash et Director, le point (0, 0) est le coin supérieur gauche de la scène Flash ou Director. Ces coordonnées peuvent ne pas coïncider sur la scène Director si une image-objet Flash est étirée, mise à l'échelle ou a pivoté.

Paramètres

`pointInFlashMovie` Requis. Point de l'image-objet d'animation Flash dont les coordonnées sont renvoyées.

Exemple

Le gestionnaire suivant accepte une valeur de point et une référence d'image-objet comme paramètre, puis donne à la coordonnée supérieure gauche de l'image-objet spécifiée le point indiqué dans une image-objet d'animation Flash de la piste 10 :

```
-- Lingo syntax
on snapSprite(whichFlashPoint, whichSprite)
    sprite(whichSprite).loc = sprite(1).FlashToStage(whichFlashPoint)
    _movie.updatestage()
end

// JavaScript syntax
function snapSprite(whichFlashPoint, whichSprite) {
    sprite(whichSprite).loc = sprite(1).FlashToStage(whichFlashPoint);
    _movie.updateStage();
}
```

Voir aussi

`stageToFlash()`

float()

Syntaxe

```
(expression).float
float (expression)
```

Description

Fonction (Lingo uniquement) ; convertit une expression en nombre à virgule flottante. Le nombre de chiffres après la virgule (pour l'affichage uniquement, les calculs n'étant pas affectés) est défini à l'aide de la propriété `floatPrecision`.

En syntaxe JavaScript, utilisez la fonction `parseFloat()`.

Paramètres

`expression` Requis. Expression à convertir en nombre à virgule flottante.

Exemple

L'instruction suivante convertit le nombre entier 1 en 1 à virgule flottante :

```
put (1).float
-- 1.0
```

Les opérations mathématiques peuvent être effectuées à l'aide de float. Si l'un des termes correspond à une valeur flottante, toute l'opération est effectuée avec `float` :

```
put 2 + 2
-- 4
put (2).float + 2
-- 4.0
the floatPrecision = 4
put 22/7
-- 3
put (22).float / 7
-- 3.1429"
```

Voir aussi

`floatPrecision`, `ilk()`

floatP()

Syntaxe

```
(expression).floatP
floatP(expression)
```

Description

Fonction (Lingo uniquement) ; indique si une expression est un nombre à virgule flottante (1 ou `TRUE`) ou non (0 ou `FALSE`).

Le caractère *P* de `floatP` signifie *prédicat*.

Paramètres

`expression` Requis. Expression à tester.

Exemple

L'instruction suivante vérifie si 3.0 est un nombre à virgule flottante. La fenêtre Messages affiche le nombre 1 pour indiquer que c'est le cas (`TRUE`).

```
put (3.0).floatP
-- 1
```

L'instruction suivante vérifie si 3 est un nombre à virgule flottante. La fenêtre Messages affiche le nombre 0 pour indiquer que ce n'est pas le cas (`FALSE`).

```
put (3).floatP
-- 0
```

Voir aussi

float(), ilk(), integerP(), objectP(), stringP(), symbolP()

flushInputEvents()

Syntaxe

```
-- Lingo syntax
_player.flushInputEvents()

// JavaScript syntax
_player.flushInputEvents();
```

Description

Méthode de lecture, purge les événements clavier ou souris de la file d'attente de messages de Director.

Cette commande se révèle généralement utile lorsqu'un script exécute une boucle assez longue et que l'auteur souhaite s'assurer que les opérations effectuées au clavier ou à l'aide de la souris ne sont pas transmises.

Cette méthode n'est effective qu'au moment de l'exécution et n'a aucun effet en phase de création.

Paramètres

Aucune.

Exemple

L'instruction suivante désactive les événements souris et clavier lors de l'exécution d'une boucle de répétition :

```
-- Lingo syntax
repeat with i = 1 to 10000
    _player.flushInputEvents()
    sprite(1).loc = sprite(1).loc + point(1, 1)
end repeat

// JavaScript syntax
for (var i = 1; i <= 10000; i++) {
    _player.flushInputEvents();
    sprite(1).loc = sprite(1).loc + point(1, 1);
}
```

Voir aussi

on keyDown, on keyUp, on mouseDown (gestionnaire d'événement), on mouseUp (gestionnaire d'événement), Lecteur

forget() (fenêtre)

Syntaxe

```
-- Lingo syntax
windowObjRef.forget()

// JavaScript syntax
windowObjRef.forget();
```

Description

Méthode de fenêtre ; indique au script de fermer une fenêtre et d'arrêter la lecture de cette dernière lorsqu'elle n'est plus utilisée et qu'aucune autre variable n'y fait référence.

L'appel de la méthode `forget()` sur une fenêtre supprime également la référence à cette fenêtre de la liste `windowList`.

Lorsque la méthode `forget()` est appelée, la fenêtre et l'animation dans une fenêtre disparaissent sans appeler les gestionnaires `stopMovie`, `closeWindow` ou `deactivateWindow`.

S'il existe de nombreuses références globales à l'animation dans une fenêtre, la fenêtre ne répond pas à la commande `forget()`.

Paramètres

Aucune.

Exemple

L'instruction suivante indique à Lingo de supprimer la fenêtre Tableau de commande lorsque l'animation ne l'utilise plus :

```
-- Lingo syntax
window("Control Panel").forget()

// JavaScript syntax
window("Control Panel").forget();
```

Voir aussi

`close()`, `open() (fenêtre)`, `Fenêtre`, `windowList`

forget() (temporisation)

Syntaxe

```
timeout("timeoutName").forget()
forget(timeout("timeoutName"))
```

Description

Cette fonction d'objet de temporisation supprime un objet de temporisation de la liste `timeoutList` et l'empêche d'envoyer d'autres événements de temporisation.

Paramètres

Aucune.

Exemple

L'instruction suivante supprime l'objet de temporisation Réveil de la liste `timeoutList` :

```
-- Lingo
timeout("AlarmClock").forget()

// Javascript
timeout("AlarmClock").forget();
```

Voir aussi

`timeout()`, `timeoutHandler`, `timeoutList`, `new()`

framesToHMS()

Syntaxe

```
framesToHMS(frames, tempo, dropFrame, fractionalSeconds)
```

Description

Fonction ; convertit le nombre d'images spécifié en durée équivalente en heures, minutes et secondes. Cette fonction est pratique pour prévoir la durée de lecture réelle d'une animation ou pour contrôler un appareil de lecture vidéo.

Le résultat est une chaîne spécifiée sous la forme `sHH:MM:SS.FFD`, où :

s	Un caractère est utilisé si le temps est inférieur à zéro ou un espace si le temps est supérieur ou égal à zéro.
HH	Heures.
MM	Minutes.
SS	Secondes.
FF	Indique une fraction de seconde si *`fractionalSeconds`* présente la valeur `TRUE` ou `images` si *`fractionalSeconds`* présente la valeur `FALSE`.
D	Un « `d` » est utilisé si *`dropFrame`* présente la valeur `TRUE` ; un espace est utilisé si *`dropFrame`* présente la valeur `FALSE`.

Paramètres

`frames` Requis. Expression entière spécifiant le nombre d'images.

`tempo` Requis. Expression entière spécifiant la cadence en images par seconde.

`dropFrame` Requis. Compense la cadence NTSC couleur qui n'est pas exactement de 30 images par seconde et n'est utile que pour une cadence de 30 images par seconde. Normalement, ce paramètre présente la valeur `FALSE`.

`fractionalSeconds` Requis. Détermine si les images résiduelles sont converties au centième de seconde le plus proche (`TRUE`) ou renvoyées sous la forme d'un nombre d'images entier (`FALSE`).

Exemple

L'instruction suivante convertit une animation de 2710 images, 30 images par seconde. Les arguments `dropFrame` et `fractional-Seconds` sont tous les deux désactivés :

```
put framesToHMS(2710, 30, FALSE, FALSE)
-- " 00:01:30.10 "
```

Voir aussi

`HMStoFrames()`

frameReady() (animation)

Syntaxe

```
-- Lingo syntax
_movie.frameReady({intFrameNum})
_movie.frameReady(frameNumA, frameNumB)

// JavaScript syntax
_movie.frameReady({intFrameNum});
```

```
_movie.frameReady(frameNumA, frameNumB);
```

Description

Méthode d'animation ; pour les animations, projections et animations Director comportant du contenu Shockwave, détermine si les acteurs d'une image ou d'une série d'images ont été téléchargés.

Cette méthode renvoie la valeur `TRUE` si les acteurs spécifiés ont été téléchargés, et la valeur `FALSE` dans le cas contraire.

Pour visualiser une démonstration de la méthode `frameReady()` dans une animation Director, consultez l'exemple d'animation Shockwave en flux continu dans l'Aide de Director.

Paramètres

`intFrameNum` Facultatif en cas de vérification si les acteurs d'une image spécifique ont été téléchargés. Nombre entier spécifiant l'image individuelle à tester. Si ce paramètre est omis, `frameReady()` détermine si les acteurs utilisés dans n'importe quelle image d'un scénario ont été téléchargés.

`frameNumA` Requis en cas de vérification si les acteurs d'une série d'images ont été téléchargés. Nombre entier spécifiant la première image de la série.

`frameNumB` Requis en cas de vérification si les acteurs d'une série d'images ont été téléchargés. Nombre entier spécifiant la dernière image de la série.

Exemple

L'instruction suivante détermine si les acteurs de l'image 20 sont téléchargés et prêts à être affichés :

```
-- Lingo syntax
on exitFrame
    if (_movie.frameReady(20)) then
        _movie.go(20)
    else
        _movie.go(1)
    end if
end

// JavaScript syntax
function exitFrame() {
    if (_movie.frameReady(20)) {
        _movie.go(20);
    }
    else {
        _movie.go(1);
    }
}
```

Le script d'image suivant vérifie si l'image 25 d'une image-objet d'animation Flash dans la piste 5 peut être affichée. Dans le cas contraire, le script maintient la tête de lecture en boucle sur l'image en cours de l'animation Director. Lorsque l'image 25 peut être affichée, le script démarre l'animation et laisse la tête de lecture passer à l'image suivante de l'animation Director.

Voir aussi

`mediaReady`, `Animation`

frameStep()

Syntaxe

```
-- Lingo syntax
dvdObjRef.frameStep(intFrames)

// JavaScript syntax
dvdObjRef.frameStep(intFrames);
```

Description

Méthode de DVD ; avance la lecture du nombre d'images spécifié à partir de la position en cours lorsque la lecture a été interrompue.

La lecture vers l'arrière n'est pas prise en charge par le logiciel système Windows ou Mac pour la lecture de DVD.

Paramètres

`intFrames` Requis. Nombre entier spécifiant le nombre d'images duquel la lecture doit être avancée.

Exemple

L'instruction suivante avance la lecture de 100 images :

```
-- Lingo syntax
member("drama").frameStep(100)

// JavaScript syntax
member("drama").frameStep(100);
```

Voir aussi

DVD

freeBlock()

Syntaxe

```
the freeBlock
```

Description

Fonction ; indique la taille du plus grand bloc de mémoire disponible, en octets. Un kilo-octet (ko) correspond à 1 024 octets. Un méga-octet (Mo) correspond à 1 024 kilo-octets. Le chargement d'un acteur nécessite un bloc libre au moins aussi grand que l'acteur.

Paramètres

Aucune.

Exemple

L'instruction suivante détermine si le plus grand bloc de mémoire disponible est inférieur à 10 ko et affiche un message d'erreur si c'est le cas :

```
-- Lingo syntax
if (the freeBlock < (10 * 1024)) then alert "Not enough memory!"

// JavaScript syntax
if (freeBlock < (10 * 1024)) {
```

```
    alert("Not enough memory!")
}
```

Voir aussi

freeBytes(), memorySize, ramNeeded(), size

freeBytes()

Syntaxe

```
the freeBytes
```

Description

Fonction ; indique le nombre total d'octets de mémoire disponible, qui ne forme pas forcément un bloc continu. Un kilo-octet (ko) correspond à 1 024 octets. Un méga-octet (Mo) correspond à 1 024 kilo-octets.

Cette fonction diffère de la fonction `freeBlock` car elle indique la totalité de la mémoire disponible, et non uniquement la mémoire contiguë.

Sur le Mac, la sélection de l'option Utiliser la mémoire temporaire du système dans les préférences générales de Director ou dans la boîte de dialogue Options d'une projection indique à la fonction `freeBytes` de renvoyer toute la mémoire disponible pour l'application. Cette quantité est égale à la quantité affectée à l'application affichée dans sa boîte de dialogue Lire les informations, et à la valeur Mémoire disponible affichée dans la boîte de dialogue A propos de votre Mac.

Paramètres

Aucune.

Exemple

L'instruction suivante vérifie si plus de 200 ko de mémoire est disponible et lit une animation couleur si c'est le cas :

```
if (the freeBytes > (200 * 1024)) then play movie "colorMovie"
```

Voir aussi

freeBlock(), memorySize, objectP(), ramNeeded(), size

generateNormals()

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).generateNormals(style)
```

Description

Commande 3D de ressource de modèle `#mesh` ; calcule les vecteurs `normal` pour chaque sommet de la maille.

Si le paramètre *`style`* présente la valeur `#flat`, chaque sommet reçoit une normale pour chaque face à laquelle il appartient. Les trois sommets d'une face partagent la même normale. Par exemple, si les sommets de `face[1]` reçoivent tous `normal[1]`, que les sommets de `face[2]` reçoivent tous `normal[2]` et que les deux faces partagent `vertex[8]`, la normale de `vertex[8]` est `normal[1]` pour `face[1]` et `normal[2]` pour `face[2]`. L'utilisation du paramètre `#flat` résulte en une délimitation très claire des faces d'une maille.

Si le paramètre *style* présente la valeur #smooth, chaque sommet ne reçoit qu'une seule normale, quel que soit le nombre de faces auquel il appartient, les trois sommets d'une face pouvant avoir différentes normales. Chaque normale de sommet est la moyenne des normales de toutes les faces le partageant. L'utilisation du paramètre #smooth résulte en une apparence plus adoucie des faces d'une maille, à l'exception des bords extérieurs qui restent nets.

Une normale de sommet est un vecteur de direction indiquant la direction « vers l'avant » d'un sommet. Si la normale de sommet pointe vers la caméra, les couleurs affichées dans la région de la maille contrôlée par cette normale sont déterminées par le matériau. Si la normale de sommet pointe dans la direction opposée à la caméra, la zone de la maille contrôlée par cette normale n'est pas visible.

Si vous utilisez la commande generateNormals(), vous devrez utiliser la commande build() pour reconstruire la maille.

Paramètres

style Requis. Symbole spécifiant le style du sommet.

Exemple

L'instruction suivante calcule les normales de sommet de la ressource de modèle mailleDeSol. Le paramètre *style* présente la valeur #smooth afin que chaque sommet de la maille ne reçoive qu'une seule normale.

```
-- Lingo
member("Room").modelResource("FloorMesh").generateNormals(#smooth)

// Javascript
member("Room").getProp("modelResource", "index (of the
modelresource)").generateNormals("smooth");
```

Voir aussi

build(), face[ ], normalList, normals, flat

getaProp

Syntaxe

```
propertyList.propertyName
getaProp(list, item)
list[listPosition]
propertyList [ #propertyName ]
propertyList [ "propertyName" ]
```

Description

Commande de liste ; pour les listes linéaires et de propriétés, identifie la valeur associée à l'élément spécifié par *item*, *listPosition* ou *propertyName* dans la liste spécifiée par *list*.

- Lorsque la liste est linéaire, remplacez *item* par le numéro correspondant à la position de l'élément dans cette liste, indiquée par *listPosition*. Le résultat est la valeur située à cette position.
- Lorsque la liste est une liste de propriétés, remplacez *item* par une propriété de la liste comme dans *propertyName*. Le résultat est la valeur associée à cette propriété.

La commande getaProp renvoie VOID si la valeur spécifiée n'est pas dans la liste.

Lorsqu'elle est utilisée avec des listes linéaires, la commande getaProp remplit la même fonction que la commande getAt.

Paramètres

itemNameOrNum Requis. Pour les listes linéaires, nombre entier spécifiant la position d'index de la valeur dans la liste à renvoyer ; pour les listes de propriétés, symbole (Lingo) ou chaîne (syntaxe JavaScript) spécifiant la propriété dont la valeur est renvoyée.

Exemple

L'instruction suivante identifie la valeur associée à la propriété `#joe` dans la liste de propriétés âges, composée de [#jean:10, #joe:12, #sophie:15, #barbara:22] :

```
put getaProp(ages, #joe)
```

Le résultat est 12 car il s'agit de la valeur associée à la propriété `#joe`.

Le même résultat peut être obtenu avec des crochets d'accès dans la même liste :

```
put ages[#joe]
```

Le résultat est de nouveau 12.

Pour obtenir la valeur située à une certaine position dans la liste, vous pouvez également utiliser des crochets d'accès. Pour obtenir la troisième valeur de la liste associée à la troisième propriété, utilisez la syntaxe suivante :

```
put ages[3]
-- 15
```

Remarque : *Contrairement à la commande getAProp pour laquelle la valeur VOID est renvoyée lorsqu'une propriété n'existe pas, une erreur de script se produit si la propriété n'existe pas et qu'un crochet d'accès est utilisé.*

Voir aussi

`getAt`, `getOne()`, `getProp()`, `setaProp`, `setAt`

getAt

Syntaxe

```
getAt(list, position)
list [position]
```

Description

Commande de liste ; identifie l'élément à une position spécifiée d'une liste spécifique. Si la liste contient moins d'éléments que la position spécifiée, une erreur de script a lieu.

La commande `getAt` fonctionne avec les listes linéaires et de propriétés. Cette commande remplit la même fonction que la commande `getaProp` pour les listes linéaires.

Elle est pratique pour extraire une liste d'une autre liste, comme `deskTopRectList`.

Paramètres

list Requis. Spécifie la liste à laquelle appartient l'élément.

position Requis. Spécifie la position d'index de l'élément dans la liste.

Exemple

L'instruction suivante entraîne l'affichage dans la fenêtre Messages du troisième élément de la liste Réponses, composée de [10, 12, 15, 22] :

```
put getAt(answers, 3)
-- 15
```

Le même résultat peut être renvoyé à l'aide de crochets d'accès :

```
put answers[3]
-- 15
```

L'exemple suivant extrait la première entrée d'une liste de deux entrées indiquant le nom, le service et le numéro d'identification des employés. Le second élément de la liste extraite est ensuite renvoyé, identifiant le service dans lequel la première personne de la liste est employée. Le format de la liste est [["Denis", "Conseil", 510], ["Sophie", "Distribution", 973]] et la liste est appelée listeInfosEmployés

```
firstPerson = getAt(employeeInfoList, 1)
put firstPerson
-- ["Dennis", "consulting", 510]
firstPersonDept = getAt(firstPerson, 2)
put firstPersonDept
-- "consulting"
```

Il est également possible d'imbriquer des commandes `getAt` sans affecter de valeurs aux variables dans les étapes intermédiaires. Ce format peut se révéler plus difficile à lire et à rédiger, mais contient moins de texte.

```
firstPersonDept = getAt(getAt(employeeInfoList, 1), 2)
put firstPersonDept
 -- "consulting"
```

Vous pouvez également utiliser les crochets d'accès :

```
firstPerson = employeeInfoList[1]
put firstPerson
-- ["Dennis", "consulting", 510]
firstPersonDept = firstPerson[2]
put firstPersonDept
-- "consulting"
```

De même qu'avec la commande getAt, les crochets peuvent être imbriqués :

```
firstPersonDept = employeeInfoList[1][2]
```

Voir aussi

getaProp, setaProp, setAt

getError() (Flash, SWA)

Syntaxe

```
-- Lingo syntax
memberObjRef.getError()

// JavaScript syntax
memberObjRef.getError();
```

Description

Fonction ; pour les acteurs Shockwave Audio (SWA) ou Flash, indique si une erreur est survenue pendant le transfert de l'acteur en mémoire et renvoie une valeur.

Les acteurs Shockwave Audio peuvent prendre les valeurs entières `getError()` suivantes et les messages `getErrorString()` correspondants :

valeur getError()	message getErrorString()
0	OK
1	mémoire
2	réseau
3	périphérique de lecture
99	autre

Les valeurs `getError` possibles des acteurs animation Flash sont les suivantes :

- `FALSE` : aucune erreur n'est survenue.
- `#memory` : la mémoire est insuffisante pour le chargement de l'acteur.
- `#fileNotFound` : le fichier contenant les éléments de l'acteur est introuvable.
- `#network` : une erreur réseau a empêché le chargement de l'acteur.
- `#fileFormat` : le fichier a été trouvé mais semble être d'un type incorrect ou une erreur est survenue à la lecture du fichier.
- `#other` : une autre erreur est survenue.

Lorsqu'une erreur survient pendant le transfert de l'acteur en mémoire, Director affecte la valeur -1 à la propriété d'état de l'acteur. Utilisez la fonction `getError` pour déterminer le type d'erreur.

Paramètres

Aucune.

Exemple

Le gestionnaire suivant utilise `getError` pour déterminer si une erreur impliquant l'acteur Shockwave Audio Norma Desmond parle est survenue et, le cas échéant, affiche la chaîne d'erreur appropriée dans un champ :

```
-- Lingo syntax
on exitFrame
    if member("Norma Desmond Speaks").getError() <> 0 then
        member("Display Error Name").text = member("Norma Desmond \
Speaks").getErrorString()
    end if
end

// JavaScript syntax
function exitFrame() {
var memNor = member("Norma Desmond Speaks").getError();
    if (memNor != 0) {
        member("Display Error Name").text = member("Norma Desmond Speaks").getErrorString();
    }
}
```

Le gestionnaire suivant vérifie si une erreur est survenue pour un acteur Flash intitulé Dali lors de son transfert en mémoire. Si une erreur est survenue et qu'il s'agit d'une erreur de mémoire, le script utilise la commande `unloadCast` pour tenter de libérer de la mémoire, puis fait ensuite passer la tête de lecture à une image appelée Artistes de l'animation Director, dans laquelle l'image-objet d'animation Flash apparaît en premier, de façon à ce que Director puisse de nouveau essayer de charger et de lire l'animation Flash. Si un autre type d'erreur est survenu, le script passe à une image intitulée Désolé qui explique que l'animation Flash requise ne peut pas être lue.

```
-- Lingo syntax
```

```
on CheckFlashStatus
errorCheck = member("Dali").getError()
    if errorCheck <> 0 then
        if errorCheck = #memory then
            member("Dali").clearError()
            unloadCast()
            _movie.go("Artists")
        else
            _movie.go("Sorry")
        end if
    end if
end

// JavaScript syntax
function CheckFlashStatus() {
var errorCheck = member("Dali").getError();
    if (errorCheck != 0) {
        if (errorCheck == "memory") {
            member("Dali").clearError();
            unloadCast();
            _movie.go("Artists");
        } else {
            _movie.go("Sorry");
        }
    }
}
```

Voir aussi

clearError(), getErrorString(), state (Flash, SWA)

getError() (XML)

Syntaxe

```
parserObject.getError()
```

Description

Fonction ; renvoie une chaîne d'erreur descriptive associée à un numéro d'erreur (comprenant le numéro de la ligne et de la colonne de l'emplacement de l'erreur dans le code XML). S'il n'existe aucune erreur, cette fonction renvoie la valeur <VOID>.

Paramètres

Aucune.

Exemple

Les instructions suivantes vérifient une erreur après l'analyse d'une chaîne contenant des données XML :

```
-- Lingo
errCode = parserObject.parseString(member("XMLtext").text)
errorString = parserObject.getError()
if voidP(errorString) then
    -- Go ahead and use the XML in some way
else
    alert "Sorry, there was an error " & errorString
    -- Exit from the handler
    exit
end if
```

```
// Javascript
errCode = parserObject.parseString(member("XMLtext").text);
errorString = parserObject.getError();
if (errorString != null)
{
    // Go ahead and use the XML in some way
}else{
    _player.alert("Sorry, there was an error " + errorString);
    // Exit from the handler
}
```

getErrorString()

Syntaxe

```
-- Lingo syntax
memberObjRef.getErrorString()

// JavaScript syntax
memberObjRef.getErrorString();
```

Description

Fonction ; pour les acteurs Shockwave Audio (SWA), renvoie la chaîne de message d'erreur correspondant à la valeur de l'erreur renvoyée par la fonction `getError()`.

Les valeurs entières `getError()` possibles et les messages `getErrorString()` correspondants sont les suivants :

valeur getError()	message getErrorString()
0	OK
1	mémoire
2	réseau
3	périphérique de lecture
99	autre

Paramètres

Aucune.

Exemple

Le gestionnaire suivant utilise `getError()` pour déterminer si une erreur est survenue pour l'acteur Shockwave Audio Norma Desmond parle et, le cas échéant, utilise `getErrorString` pour obtenir le message d'erreur et l'affecter à un acteur champ :

```
-- Lingo syntax
on exitFrame
    if member("Norma Desmond Speaks").getError() <> 0 then
        member("Display Error Name").text = member("Norma Desmond Speaks").getErrorString()
    end if
end

// JavaScript syntax
function exitFrame() {
```

```
    var memNor = member("Norma Desmond Speaks").getError();
    if (memNor != 0) {
        member("Display Error Name").text = member("Norma Desmond Speaks").getErrorString();
    }
}
```

Voir aussi

getError() (Flash, SWA)

getFinderInfo()

Syntaxe

```
-- Lingo syntax
fileioObjRef.getFinderInfo()

// JavaScript syntax
fileioObjRef.getFinderInfo();
```

Description

Méthode FileIO (Mac uniquement) ; renvoie les informations du Finder relatives à un fichier ouvert.

Vous devez avoir ouvert un fichier à l'aide de la méthode `openFile()` avant d'utiliser la méthode `getFinderInfo()` pour obtenir les informations du Finder concernant ce fichier.

Paramètres

Aucune.

Exemple

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.getFinderInfo()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.getFinderInfo() ;
```

Voir aussi

Fileio, openFile()

getFlashProperty()

Syntaxe

```
-- Lingo syntax
spriteObjRef.getFlashProperty(targetName, symProp)

// JavaScript syntax
spriteObjRef.getFlashProperty(targetName, symProp);
```

Description

Cette fonction permet à Lingo d'invoquer la fonction script d'action Flash `getProperty()` dans l'image-objet Flash donnée. Cette fonction script d'action Flash est utilisée pour obtenir la valeur des propriétés des recadrages ou des niveaux dans une animation Flash. Cette fonction est similaire au test des propriétés d'images-objets dans Director.

Pour obtenir une propriété globale de l'image-objet Flash, transmettez une ligne vide en tant que *`targetName`*. Ces propriétés globales Flash peuvent être testées : `#focusRect` et `#spriteSoundBufferTime`.

Consultez la documentation de Flash pour plus d'informations sur ces propriétés.

Paramètres

`targetName` Requis. Chaîne spécifiant le nom du clip ou du niveau de l'animation dont vous souhaitez obtenir la propriété dans l'image-objet Flash donnée.

`symProp` Requis. Symbole spécifiant le nom de la propriété à obtenir. Les valeurs possibles sont : `#posX`, `#posY`, `#scaleX`, `#scaleY`, `#visible`, `#rotate`, `#alpha`, `#name`, `#width`, `#height`, `#target`, `#url`, `#dropTarget`, `#totalFrames`, `#currentFrame`, `#cursor` et `#lastframeLoaded`.

Exemple

L'instruction suivante obtient la valeur de la propriété `#rotate` du clip Etoile dans l'acteur Flash de l'image-objet 3 :

```
-- Lingo syntax
sprite(3).setFlashProperty("Star", #rotate)
sprite(3).getFlashProperty("Star")

// JavaScript syntax
sprite(3).setFlashProperty("Star", symbol("rotate"));
sprite(3).getFlashProperty("Star");
```

getFrameLabel()

Syntaxe

```
sprite(whichFlashSprite).getFrameLabel(whichFlashFrameNumber)
getFrameLabel(sprite whichFlashSprite, whichFlashFrameNumber)
```

Description

Fonction ; renvoie le libellé d'image d'une animation Flash associé au nom de numéro demandé. Si le libellé n'existe pas ou que cette partie de l'animation Flash n'est pas encore transférée en mémoire, cette fonction renvoie une chaîne vide.

Paramètres

`whichFlashFrameNumber` Requis. Spécifie le numéro d'image associé au libellé d'image.

Exemple

Le gestionnaire suivant vérifie si le repère de l'image 15 de l'animation Flash lue dans l'image-objet 1 porte le nom « Lions ». Le cas échéant, l'animation Director passe à l'image Lions. Dans le cas contraire, l'animation Director reste dans l'image en cours et la lecture de l'animation Flash se poursuit.

```
-- Lingo syntax
on exitFrame
    if sprite(1).getFrameLabel(15) = "Lions" then
        go "Lions"
    else
```

```
        go the frame
    end if
end

// JavaScript syntax
function exitFrame() {
    if (sprite(1).getFrameLabel(15) == "Lions") {
        _movie.go("Lions");
    } else {
        _movie.go(_movie.frame);
    }
}
```

getHardwareInfo()

Syntaxe

```
getRendererServices().getHardwareInfo()
```

Description

Méthode 3D `rendererServices` ; renvoie une liste de propriétés contenant les informations relatives à la carte vidéo de l'utilisateur. La liste contient les propriétés suivantes :

`#present` est une valeur booléenne indiquant si l'ordinateur est équipé de matériel d'accélération vidéo.

`#vendor` indique le nom du fabricant de la carte vidéo.

`#model` indique le modèle de la carte vidéo.

`#version` indique la version du pilote vidéo.

`#maxTextureSize` est une liste linéaire contenant la largeur et hauteur maximale d'une texture, en pixels. Les textures dépassant cette taille sont sous-échantillonnées à la taille maximum. Vous pouvez éviter des erreurs de sous-échantillonnage en créant des textures de différentes tailles et en choisissant celles qui ne dépassent pas la valeur `#maxTextureSize` lors de l'exécution.

`#supportedTextureRenderFormats` est une liste linéaire des formats de texture pris en charge par la carte vidéo. Pour plus d'informations, reportez-vous à l'entrée `textureRenderFormat`.

`#textureUnits` indique le nombre d'unités de texture disponibles pour la carte.

`#depthBufferRange` est une liste linéaire de résolutions binaires sur lesquelles la propriété `depthBufferDepth` peut être définie.

`#colorBufferRange` est une liste linéaire de résolutions binaires sur lesquelles la propriété `colorBufferDepth` peut être définie.

Exemple

L'instruction suivante affiche une liste de propriétés détaillée concernant le matériel de l'utilisateur :

```
-- Lingo
put getRendererServices().getHardwareInfo()
-- [#present: 1, #vendor: "NVIDIA Corporation", #model: "32MB DDR NVIDIA GeForce2 GTS
(Dell)", #version: "4.12.01.0532", #maxTextureSize: [2048, 2048],
#supportedTextureRenderFormats: [#rgba8888, #rgba8880, #rgba5650, #rgba5551, #rgba5550,
#rgba4444], #textureUnits: 2, #depthBufferRange: [16, 24], #colorBufferRange: [16, 32]]

// Javascript
```

```
trace(getRendererServices().getHardwareInfo())
//<[#present: 1, #vendor: "NVIDIA Corporation", #model: "32MB DDR NVIDIA GeForce2 GTS
(Dell)", #version: "4.12.01.0532", #maxTextureSize: [2048, 2048],
#supportedTextureRenderFormats: [#rgba8888, #rgba8880, #rgba5650, #rgba5551, #rgba5550,
#rgba4444], #textureUnits: 2, #depthBufferRange: [16, 24], #colorBufferRange: [16, 32]]>
```

Voir aussi

`getRendererServices()`

getHotSpotRect()

Syntaxe

```
-- Lingo syntax
spriteObjRef.getHotSpotRect(hotSpotID)

// JavaScript syntax
spriteObjRef.getHotSpotRect(hotSpotID);
```

Description

Fonction QuickTime VR ; renvoie un rectangle de délimitation approximatif pour une zone référencée. Si la zone référencée n'existe pas ou n'est pas visible sur la scène, cette fonction renvoie rect(0, 0, 0, 0). Si la zone référencée est partiellement visible, cette fonction renvoie le rectangle de délimitation pour la partie visible.

Paramètres

`hotSpotID` Requis. Spécifie la zone référencée pour laquelle un rectangle de délimitation est renvoyé.

GetItemPropList

Syntaxe

```
GetItemPropList(MUIObject)
```

Description

Cette fonction renvoie une liste des propriétés prédéfinies de l'Xtra MUI pour les composants, dans une boîte de dialogue générale. Elle peut s'avérer utile lors de la définition de nouveaux composants dans une boîte de dialogue générale. Utilisez la fonction GetItemPropList pour obtenir une liste complète des propriétés et des valeurs, puis modifiez individuellement les propriétés si nécessaire.

Les propriétés et leurs valeurs sont présentées dans le tableau ci-dessous.

Propriété	Valeur par défaut
#value	0
#type	#checkBox
#attributes	[],
#title	"titre"
#tip	"conseil" (compatible avec les versions ultérieures de l'Xtra MUI)
#locH	20

Propriété	Valeur par défaut
#locV	24
#width	200
#height	210
#enabled	1

Exemple

Ces instructions définissent le début d'une fenêtre de boîte de dialogue.

- La première instruction crée une instance d'Xtra MUI, qui est l'objet utilisé en tant que boîte de dialogue.
- La deuxième instruction attribue une liste des paramètres de composant de boîte de dialogue par défaut à la variable tempItemProps.
- Grâce à la troisième instruction qui lui attribue le type #windowBegin, le composant devient le début de la boîte de dialogue.

```
--Lingo Syntax
set aMuiObj = new (Xtra "MUI")
set tempItemProps = GetItemPropList(aMuiObj)
set the type of tempItemProps = #windowBegin
```

getLast()

Syntaxe

```
list.getLast()
getLast(list)
```

Description

Fonction de liste ; identifie la dernière valeur d'une liste linéaire ou de propriétés spécifiée par *`list`*.

Paramètres

Aucune.

Exemple

L'instruction suivante identifie le dernier élément (22) de la liste Réponses, composée de [10, 12, 15, 22] :

```
put Answers.getLast()
```

L'instruction suivante identifie le dernier élément (850) de la liste Devis, composée de [#avatar:750, #dupont:600, #soldes:850] :

```
put Bids.getLast()
```

getLatestNetID

Syntaxe

```
getLatestNetID
```

Description

Cette fonction renvoie un identifiant pour la dernière opération réseau entamée.

L'identificateur renvoyé par `getLatestNetID` peut être utilisé en tant que paramètre dans les fonctions `netDone`, `netError` et `netAbort` pour identifier la dernière opération réseau.

Remarque : *Cette fonction est destinée à la compatibilité en amont. Il est recommandé d'utiliser l'ID réseau renvoyé par une fonction réseau de Lingo plutôt que par* `getLatestNetID`*. Toutefois, si vous utilisez* `getLatestNetID`*, faites-le immédiatement après la commande netLingo.*

Paramètres

Aucune.

Exemple

Le script suivant affecte l'identifiant réseau d'une opération `getNetText` à l'acteur champ Résultat de façon à permettre la récupération ultérieure des résultats de cette opération :

```
on startOperation
    global gNetID
    getNetText("url")
    set gNetID = getLatestNetID()
end
on checkOperation
    global gNetID
    if netDone(gNetID) then
        put netTextResult into member "Result"
    end if
end
```

Voir aussi

`netAbort`, `netDone()`, `netError()`

getLength()

Syntaxe

```
-- Lingo syntax
fileioObjRef.getLength()

// JavaScript syntax
fileioObjRef.getLength();
```

Description

Méthode FileIO ; renvoie la longueur d'un fichier ouvert.

Vous devez avoir ouvert un fichier à l'aide de la méthode `openFile()` avant d'utiliser la méthode `getLength()` pour obtenir la longueur de ce fichier.

Paramètres

Aucune.

Exemple

L'exemple suivant ouvre le fichier c:\vérif.txt et vérifie la longueur de ce fichier.

```
-- Lingo syntax
```

```
objFileio = new xtra("fileio")
objFileio.openFile("c:\check.txt",2)
put objFileio.getLength()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\check.txt",2);
trace(objFileio.getLength())
```

Voir aussi

Fileio, openFile()

getNetText()

Syntaxe

```
getNetText(URL {, serverOSString} {, characterSet})
getNetText(URL, propertyList {, serverOSString} {, characterSet})
```

Description

Fonction ; démarre la récupération de texte à partir d'un fichier situé sur un serveur HTTP ou FTP ou lance une requête CGI.

La première syntaxe présentée entame la récupération du texte. Vous pouvez soumettre des requêtes CGI HTTP de cette manière et devez les encoder correctement dans l'adresse URL. La seconde syntaxe comprend une liste de propriétés et soumet une requête CGI, fournissant le codage URL correct.

Utilisez le paramètre facultatif *propertyList* pour utiliser une liste de propriétés pour les requêtes CGI. La liste de propriétés est codée dans l'URL et l'URL envoyée est `(urlstring & "?" & encodedproplist)`.

Utilisez le paramètre facultatif *serverOSString* pour coder tout caractère de retour dans *propertyList*. Ce paramètre présente la valeur `UNIX` par défaut, mais peut être défini sur `Win` ou sur `Mac` et convertit les retours chariot de l'argument *propertyList* en ceux utilisés par le serveur. Pour la plupart des applications, ce paramètre n'est pas nécessaire, les sauts de ligne n'étant généralement pas utilisés dans les réponses de formulaires.

Le paramètre facultatif *characterSet* ne s'applique que si l'utilisateur exécute Director sur un système `shift-JIS` (Japonais). Les jeux de caractères possibles sont `JIS`, `EUC`, `ASCII` et `AUTO`. Lingo convertit les données récupérées de `shift-JIS` dans le jeu de caractères spécifié. Avec le paramètre AUTO, le jeu de caractères essaie de déterminer le jeu de caractères du texte récupéré et de le convertir au jeu de caractères de la machine locale. Le paramètre par défaut est ASCII.

Utilisez `netDone` pour savoir quand l'opération `getNetText` est terminée et `netError` pour savoir si elle a réussi. Utilisez `netTextResult` pour renvoyer le texte récupéré par `getNetText`.

Cette fonction est utilisable avec des URL relatives.

Vous pouvez voir un exemple d'utilisation de `getNextText()` dans une animation en consultant l'animation Forms and Post du dossier Learning/Lingo, lui-même dans le dossier de Director.

Paramètres

URL Requis. URL du fichier contenant le texte à obtenir.

propertyList Facultatif. Spécifie une liste de propriétés utilisée pour les requêtes CGI.

serverOSString Facultatif. Spécifie le codage des caractères renvoyés dans *propertyList*.

characterSet Facultatif. Spécifie le paramétrage des caractères.

Exemple

Le script suivant récupère du texte à partir de l'URL http://grandServeur.fr/exemple.txt et met à jour l'acteur champ sur lequel la souris se trouve lorsque l'utilisateur clique dessus :

```
property spriteNum

property theNetID

on mouseUp me
    theNetID = getNetText ("http://BigServer.com/sample.txt")
end

on exitFrame me
    if netDone(theNetID) then
        sprite(spriteNum).member.text = netTextResult(theNetID)
    end if
end
```

L'exemple suivant récupère les résultats d'une requête CGI :

```
getNetText("http://www.yourserver.com/cgi-bin/query.cgi?name=Bill")
```

L'exemple suivant est semblable au précédent, mais utilise une liste de propriétés pour soumettre une requête CGI et effectue automatiquement le codage URL :

```
getNetText("http://www.yourserver.com/cgi-bin/query.cgi", [#name:"Bill"])
```

Voir aussi

netDone(), netError(), netTextResult()

getNormalized

Syntaxe

```
getNormalized(vector)
vector.getNormalized()
```

Description

Méthode 3D de vecteur ; copie le vecteur et divise les composants x, y et z de la copie par la longueur du vecteur d'origine. Le vecteur résultant est long d'une unité de l'univers.

Cette méthode renvoie la copie et n'affecte pas le vecteur d'origine. Pour normaliser le vecteur d'origine, utilisez la commande `normalize`.

Exemple

L'instruction suivante enregistre la valeur normalisée du vecteur monVecteur dans la variable `Norm`. Norm a pour valeur vector(-0,1199, 0,9928, 0,0000) et 1 pour magnitude.

```
-- Lingo
MyVec = vector(-209.9019, 1737.5126, 0.0000)
Norm = MyVec.getNormalized()
put Norm
-- vector( -0.1199, 0.9928, 0.0000 )
put Norm.magnitude
-- 1.0000
```

```
// Javascript
MyVec = vector(-209.9019, 1737.5126, 0.0000);
Norm = MyVec.getNormalized();
trace(Norm);
// vector( -0.1199, 0.9928, 0.0000 )
trace(Norm.magnitude);
// 1.0000
```

Voir aussi

normalize

getNthFileNameInFolder()

Syntaxe

```
getNthFileNameInFolder(folderPath, fileNumber)
```

Description

Méthode d'animation ; renvoie un nom de fichier situé dans un dossier en fonction du chemin d'accès et du numéro spécifiés dans ce dossier. Pour être détectées par la fonction `getNthFileNameInFolder`, les animations Director doivent être définies comme étant visibles dans la structure des dossiers. Sur le Mac, d'autres types de fichiers peuvent être détectés, qu'ils soient visibles ou non. Si cette fonction renvoie une chaîne vide, vous avez spécifié un nombre supérieur au nombre de fichiers dans le dossier.

La fonction `getNthFileNameInFolder` ne peut pas être utilisée avec des URL.

Pour spécifier d'autres noms de dossier, utilisez l'opérateur `@ pathname` ou le chemin d'accès complet défini dans le format de la plate-forme sur laquelle l'animation est exécutée. Par exemple :

- Sous Windows, utilisez un chemin tel que C:/Director/Animations.
- Sur le Mac, utilisez un chemin d'accès tel que DisqueDur:Director:Animations**.** Pour rechercher des fichiers se trouvant sur le bureau du Mac, utilisez le chemin DisqueDur:Dossier.
- Cette fonction n'est pas disponible dans Shockwave Player.

Paramètres

`folderPath` Requis. Spécifie le chemin d'accès du dossier contenant le fichier.

`fileNumber` Requis. Spécifie la position d'index du fichier dans le dossier.

Exemple

Le gestionnaire suivant renvoie une liste de noms de fichiers dans le dossier du chemin actuel. Pour appeler cette fonction, utilisez des parenthèses, comme dans `put currentFolder()`.

```
-- Lingo
on currentFolder
    fileList = [ ]
    repeat with i = 1 to 100
        n = getNthFileNameInFolder(the moviePath, i)
        if n = EMPTY then exit repeat
        fileList.append(n)
    end repeat
    return fileList
end currentFolder
```

```
// Javascript
Function currentFolder()
{
    fileList = list();
    var i=0;
    while(i<100)
    {
        n= getNthFileNameInFolder(_movie.path,i);
        if ( n==null)
        {
            I=101;
        }
        Else
        {
            fileList.append(n);
        }
        I++;
    }
    Return fileList;
}
```

Voir aussi

@ (chemin d'accès), Animation

getOne()

Syntaxe

```
list.getOne(value)
getOne(list, value)
```

Description

Fonction de liste ; identifie la position (liste linéaire) ou la propriété (liste de propriétés) associée à une valeur de la liste.

Pour les valeurs contenues plus d'une fois dans la liste, seule la première occurrence est affichée. La commande `getOne` renvoie le résultat 0 lorsque la valeur spécifiée ne figure pas dans la liste.

Lorsqu'elle est utilisée avec des listes linéaires, la commande `getOne` remplit les mêmes fonctions que la commande `getPos`.

Lorsque vous ajoutez un filtre à l'aide de la méthode add ou append de la liste de filtres, un double du filtre est créé et ajouté à la liste. Les méthodes telles que deleteOne, getPos, findPos et getOne utilisent la valeur exacte de la liste et non la valeur dupliquée.

Dans ces cas précis, vous pouvez utiliser la commande getOne de la façon suivante :

```
f = filter(#glowfilter)
sprite(1).filterlist.append(f)
f = sprite(1).filterlist[1]-- here we get the actual value added to the list.
sprite(1).filterlist.getOne(f)
```

La troisième ligne du script ci-dessus ajoute la référence de la valeur de filtre à la liste.

Paramètres

`value` Requis. Spécifie la valeur associée à la position ou à la propriété.

Exemple

L'instruction suivante identifie la position de la valeur 12 dans la liste linéaire Réponses, composée de [10, 12, 15, 22] :

```
-- Lingo
put Answers.getOne(12)

// Javascript
trace(Answers.getOne(12));
```

Le résultat est égal à 2 car 12 est la deuxième valeur de la liste.

L'instruction suivante identifie la propriété associée à la valeur 12 dans la liste de propriétés Réponses, composée de `[#a:10, #b:12, #c:15, #d:22]`:

```
-- Lingo
put Answers.getOne(12)

// Javascript
trace(Answers.getOne(12));
```

Le résultat est #b, correspondant à la propriété associée à la valeur 12.

Voir aussi

`getPos()`

getOSDirectory()

Syntaxe

```
-- Lingo syntax
getOSDirectory()

// JavaScript syntax
getOSDirectory();
```

Description

Fonction ; renvoie le chemin d'accès complet du Dossier système (Mac) ou du répertoire Windows (Windows).

Paramètres

Aucune.

Exemple

L'instruction suivante place le répertoire du système d'exploitation de l'ordinateur. Dans ce cas précis, il s'agit de c:\windows.

```
-- Lingo
Put getOSDirectory()

// Javascript
trace(getOSDirectory());
```

Voir aussi

`Fileio`

getPixel()

Syntaxe

```
-- Lingo syntax
imageObjRef.getPixel(x, y {, #integer})
imageObjRef.getPixel(point(x, y) {, #integer})

// JavaScript syntax
imageObjRef.getPixel(x, y {, #integer});
imageObjRef.getPixel(point(x, y) {, #integer});
```

Description

Méthode d'image. Renvoie une couleur indexée ou RVB du pixel à un point spécifié d'une image donnée.

L'index des lignes et colonnes de l'image renvoyée démarre à la valeur 0. Par conséquent, pour accéder au pixel supérieur gauche d'une image, spécifiez son emplacement sous la forme `(0,0)` et non sous la forme `(1,1)`. Si une image donnée comporte une hauteur de *h* pixels et une largeur de *h* pixels, pour accéder au pixel inférieur droit de cette image, spécifiez son emplacement sous la forme `(h,1)`, `(h,1)`.

Cette méthode renvoie la valeur 0 si le pixel spécifié est situé à l'extérieur de l'image donnée.

Pour définir un grand nombre de pixels sur la couleur d'un autre pixel, il est plus rapide de les définir en tant que nombres bruts (en utilisant le paramètre facultatif `#entier`). Les valeurs de couleur brutes sont également pratiques en ce sens qu'elles contiennent à la fois des informations de couche alpha et de couleurs dans les images 32 bits. Les informations de couche alpha peuvent être extraites du nombre brut en le divisant le nombre par 2^8+8+8.

Paramètres

x Requis en cas de spécification d'un pixel à l'aide de coordonnées `x` et `y`. Nombre entier spécifiant la coordonnée `x` du pixel.

y Requis en cas de spécification d'un pixel à l'aide de coordonnées `x` et `y`. Nombre entier spécifiant la coordonnée `y` du pixel.

#entier Facultatif. Symbole spécifiant le nombre brut de la valeur de couleur renvoyée.

`point(x, y)` Requis en cas de spécification d'un pixel à l'aide d'un point. Point spécifiant le point du pixel.

Exemple

Les instructions suivantes définissent la couleur du pixel au point (20, 20) dans l'acteur Image de la scène.

```
-- Lingo
objImage = _movie.stage.image
objImage.getPixel(20, 20)
put (objImage)

-- Javascript
var objImage = _movie.stage.image ;
objImage.getPixel(20, 20) ;
put (objImage) ;
```

Voir aussi

color(), image(), power(), setPixel()

getPlayList()

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.getPlayList()

// JavaScript syntax
soundChannelObjRef.getPlayList();
```

Description

Méthode de piste audio ; renvoie une copie de la liste des sons mis en attente pour une piste audio.

La liste renvoyée n'inclut pas le son en cours de lecture et n'est pas modifiable directement. Vous devez utiliser `setPlayList()`.

La liste de lecture est une liste linéaire de listes de propriétés. Chaque liste de propriétés correspond à un acteur son placé en file d'attente. Chaque son placé en file d'attente peut spécifier ces propriétés :

Propriété	Description
#member	Acteur à placer en file d'attente. Cette propriété doit être spécifiée. Toutes les autres sont facultatives.
#startTime	Position temporelle de départ de la lecture du son, en millisecondes. La valeur par défaut est le début du son. Pour plus d'informations, reportez-vous à l'entrée `startTime`.
#endTime	Position temporelle de fin de la lecture du son, en millisecondes. La valeur par défaut est la fin du son. Pour plus d'informations, reportez-vous à l'entrée `endTime`.
#loopCount	Nombre de répétitions d'une boucle défini avec `#loopStartTime` et `#loopEndTime`. La valeur par défaut est 1. Pour plus d'informations, reportez-vous à l'entrée `loopCount`.
#loopStartTime	Position temporelle de départ de la boucle, en millisecondes. Pour plus d'informations, reportez-vous à l'entrée `loopStartTime`.
#loopEndTime	Position temporelle de fin de la boucle, en millisecondes. Pour plus d'informations, reportez-vous à l'entrée `loopEndTime`.
#preloadTime	Quantité de son à placer en mémoire tampon avant la lecture, en millisecondes. Pour plus d'informations, reportez-vous à l'entrée `preloadTime`.

Paramètres

Aucune.

Exemple

Ce gestionnaire place deux sons en file d'attente dans la piste audio 2, démarre leur lecture, puis affiche la liste de lecture dans la fenêtre Messages. La liste de lecture ne contient que le second son placé en file d'attente, le premier son étant déjà en cours de lecture.

```
-- Lingo syntax
on playMusic
    sound(2).queue(member("Chimes"))
    sound(2).queue([#member:member("introMusic"), #startTime:3000, #endTime:10000,
#loopCount:5, #loopStartTime:8000, #loopEndTime:8900])
    put(sound(2).getPlayList())
    sound(2).play()
end playMusic

// JavaScript syntax
```

```
function playMusic() {
    sound(2).queue(member("Chimes"));
    sound(2).queue(propList("member",member("introMusic"), "startTime",3000,
"endTime",10000, "loopCount",5, "loopStartTime",8000, "loopEndTime",8900));
    put(sound(2).getPlayList());
sound(2).play();
}
```

Voir aussi

endTime, loopCount, loopEndTime, loopStartTime, Acteur, member, preLoadTime, queue(), setPlayList(), Piste audio, startTime

getPosition()

Syntaxe

```
-- Lingo syntax
fileioObjRef.getPosition()

// JavaScript syntax
fileioObjRef.getPosition();
```

Description

Méthode FileIO ; renvoie la position d'un fichier.

Paramètres

Aucune.

Exemple

L'instruction suivante ouvre le fichier c:\xtra.txt et obtient la position en cours de ce dernier.

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
put objFileio.getPosition()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
trace(objFileio.getPosition());
```

Voir aussi

Fileio

getPref()

Syntaxe

```
-- Lingo syntax
_player.getPref(stringPrefName)

// JavaScript syntax
_player.getPref(stringPrefName);
```

Description

Méthode de lecteur ; récupère le contenu du fichier spécifié.

Lorsque vous utilisez cette méthode, remplacez *stringPrefName* par le nom d'un fichier créé par la méthode `setPref()`. Si ce fichier n'existe pas, la méthode `getPref()` renvoie la valeur `VOID` (Lingo) ou `null` (syntaxe JavaScript).

Le nom de fichier utilisé pour *stringPrefName* doit être un nom de fichier valide et non un chemin d'accès complet ; Director fournit le chemin. Le chemin vers le fichier est géré par Director. Les seules extensions de fichier valides pour *stringPrefName* sont .txt et .htm, toute autre extension étant rejetée.

N'utilisez pas cette méthode pour accéder à des médias en lecture seule ou verrouillés.

Remarque : *Dans un navigateur, les données écrites par `setPref()` ne sont pas confidentielles. Toute animation comportant du contenu Shockwave est en mesure de lire ces informations et de les charger sur un serveur. Les informations confidentielles ne doivent pas être stockées à l'aide de la méthode `setPref()`.*

Vous pouvez voir un exemple d'utilisation de `getPref()` dans une animation en consultant l'animation Read and Write Text du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

stringPrefName Requis. Chaîne spécifiant le fichier dont le contenu est récupéré.

Exemple

Le gestionnaire suivant récupère le contenu du fichier Test et en affecte le texte au champ Score :

```
-- Lingo syntax
on mouseUp
    theText = _player.getPref("Test")
    member("Total Score").text = theText
end

// JavaScript syntax
function mouseUp() {
    var theText = _player.getPref("Test");
    member("Total Score").text = theText;
}
```

Voir aussi

Lecteur, setPref()

getPos()

Syntaxe

```
list.getPos(value)
getPos(list, value)
```

Description

Fonction de liste ; identifie la position d'une valeur dans une liste. Lorsque la valeur spécifiée n'est pas dans la liste, la commande `getPos` renvoie la valeur 0.

Pour les valeurs contenues plus d'une fois dans la liste, seule la première occurrence est affichée. Cette commande remplit la même fonction que la commande `getOne` lorsqu'elle est utilisée pour les listes linéaires.

Lorsque vous ajoutez un filtre à l'aide de la méthode add ou append de la liste de filtres, un double du filtre est créé et ajouté à la liste. Les méthodes telles que deleteOne, getPos, findPos et getOne utilisent la valeur exacte de la liste et non la valeur dupliquée.

Dans ces cas précis, vous pouvez utiliser la méthode getPos de la façon suivante :

```
f = filter(#glowfilter)
sprite(1).filterlist.append(f)
f = sprite(1).filterlist[1]-- here we get the actual value added to the list.
sprite(1).filterlist.getPos(f)
```

La troisième ligne du script ci-dessus récupère la référence de la valeur de filtre ajoutée à la liste.

Paramètres

value Requis. Spécifie la valeur associée à la position.

Exemple

L'instruction suivante identifie la position de la valeur 12 dans la liste Réponses, composée de [#a:10, #b:12, #c:15, #d:22] :

```
-- Lingo
put Answers.getPos(12)

// Javascript
trace(Answers.getPos(12));
```

Le résultat est égal à 2 car 12 est la deuxième valeur de la liste.

Voir aussi

`getOne()`

getPref()

Syntaxe

```
getPref(prefFileName)
```

Description

Fonction ; récupère le contenu du fichier spécifié.

Lorsque vous utilisez cette fonction, remplacez *prefFileName* par le nom d'un fichier créé par la fonction `setPref`. Si ce fichier n'existe pas, `getPref` renvoie `VOID`.

Le nom de fichier utilisé pour *prefFileName* doit être un nom de fichier valide et non un chemin d'accès complet ; Director fournit le chemin. Le chemin vers le fichier est géré par Director. Les seules extensions de fichier valides pour *prefFileName* sont .txt et .htm, toute autre extension étant rejetée.

N'utilisez pas cette commande pour accéder à des médias en lecture seule ou verrouillés.

Remarque : *Dans un navigateur, les données écrites par* `setPref` *ne sont pas confidentielles. Toute animation comportant du contenu Shockwave est en mesure de lire ces informations et de les charger sur un serveur. Les informations confidentielles ne doivent pas être stockées à l'aide de la commande* `setPref`*.*

Vous pouvez voir un exemple d'utilisation de `getPref()` dans une animation en consultant l'animation Read and Write Text du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

prefFileName Requis. Spécifie le fichier dont le contenu est récupéré.

Exemple

Le gestionnaire suivant récupère le contenu du fichier Test et en affecte le texte au champ Score :

```
on mouseUp
    theText = getPref("Test")
    member("Total Score").text = theText
end
```

Voir aussi

setPref()

getProp()

Syntaxe

```
getProp(list, property)
list.property
```

Description

Fonction de liste de propriétés ; identifie la valeur associée à une propriété dans une liste de propriétés.

Quasiment identique à la commande `getaProp`, la commande `getProp` affiche un message d'erreur si la propriété spécifiée ne figure pas dans la liste ou si vous spécifiez une liste linéaire.

Paramètres

list Requis. Spécifie la liste de propriétés dont la valeur *property* est récupérée.

property Requis. Spécifie la propriété à laquelle la valeur identifiée est associée.

Exemple

L'instruction suivante identifie la valeur associée à la propriété #c dans la liste de propriétés Réponses, composée de [#a:10, #b:12, #c:15, #d:22] :

```
-- Lingo
getProp(Answers, #c)

/ Javascript
Answers.getProp("c");
```

Le résultat est 15 car il s'agit de la valeur associée à #c.

Voir aussi

getOne()

getPropAt()

Syntaxe

```
list.getPropAt(index)
getPropAt(list, index)
```

Description

Fonction de liste de propriétés ; pour les listes de propriétés uniquement, identifie le nom de propriété associé à une position spécifiée dans une liste de propriétés. Si l'élément spécifié ne figure pas dans la liste ou si vous utilisez `getPropAt()` avec une liste linéaire, une erreur de script se produit.

Paramètres

`index` Requis. Spécifie la position d'index de la propriété dans la liste de propriétés.

Exemple

L'instruction suivante identifie la deuxième propriété dans la liste de propriétés Réponses, composée de `[#a:10, #b:12, #c:15, #d:22]`:

```
-- Lingo
put Answers.getPropAt(2)
-- #b

// Javascript
trace(Answers.getPropAt(2))
// b
```

getRendererServices()

Syntaxe

```
getRendererServices()
getRendererServices().whichGetRendererServicesProperty
```

Description

Commande 3D ; renvoie l'objet `rendererServices`. Cet objet contient les informations sur le matériel et les propriétés qui affectent tous les acteurs et images-objets 3D.

L'objet `rendererServices` comporte les propriétés suivantes :

- `renderer` indique le traceur par ligne logiciel utilisé pour le rendu de toutes les images-objets 3D.
- `rendererDeviceList` renvoie une liste des traceurs par ligne logiciels présents sur le système de l'utilisateur. Les valeurs possibles sont `#openGL`, `#directX5_2`, `#directX7_0`, `#directx9` et `#software`. La valeur de `renderer` doit être l'une de ces valeurs. Cette propriété peut être testée, mais pas définie.
- `textureRenderFormat` indique le format de pixel utilisé par le moteur de rendu. Les valeurs possibles sont `#rgba8888`, `#rgba8880`, `#rgba5650`, `#rgba5550`, `#rgba5551` et `#rgba4444`. Les quatre chiffres de chaque symbole indiquent le nombre de bits utilisés pour chaque composante rouge, verte, bleue et alpha.
- `depthBufferDepth` indique le codage binaire du tampon de sortie du matériel.
- `colorBufferDepth` indique le codage binaire du tampon des couleurs. Cette propriété peut être testée, mais pas définie.
- `modifiers` est une liste linéaire des modificateurs disponibles et utilisables par les modèles des acteurs 3D. Les valeurs possibles sont `#collision`, `#bonesPlayer`, `#keyframePlayer`, `#toon`, `#lod`, `#meshDeform`, `#sds`, `#inker` et les modificateurs reposant sur des Xtras d'autres éditeurs. Cette propriété peut être testée, mais pas définie.
- `primitives` est une liste linéaire des types de primitives utilisables dans la création de ressources de modèle. Les valeurs possibles sont `#sphere`, `#box`, `#cylinder`, `#plane`, `#particle` et les types de primitives reposant sur des Xtras d'autres éditeurs. Cette propriété peut être testée, mais pas définie.

Remarque : *Pour plus d'informations, consultez les entrées des différentes propriétés.*

Paramètres

Aucune.

Exemple

L'instruction suivante récupère l'objet rendererServices ainsi que les informations concernant le moteur de rendu.

```
-- Lingo
Ro = getRendererServices()
Put Ro.renderer
-- #directX7_0

// Javascript
Var ro = getRendererServices();
trace(ro.renderer);
// #directX7_0
```

Voir aussi

`renderer`, `preferred3dRenderer`, `active3dRenderer`, `rendererDeviceList`

getStreamStatus()

Syntaxe

```
getStreamStatus(netID)
getStreamStatus(URLString)
```

Description

Fonction ; renvoie une liste de propriétés correspondant au format utilisé pour la fonction `tellStreamStatus` disponible globalement et utilisable avec des appels d'images-objets ou d'objets. La liste contient les chaînes suivantes :

#URL	Chaîne contenant l'emplacement de l'URL utilisée pour le début des opérations réseau.
#state	Chaîne composée de Connecting (connexion), Started (démarrée), InProgress (en cours), Complete (terminée), Error (erreur) ou NoInformation (aucune information) ; cette dernière chaîne s'applique lorsque l'ID réseau est tellement ancien que les informations d'état n'existent plus ou que l'URL spécifiée dans `URLString` n'a pas été trouvée dans la mémoire cache.
#bytesSoFar	Nombre d'octets récupérés jusqu'à maintenant à partir du réseau.
#bytesTotal	Nombre total d'octets dans le flux, s'il est connu. Cette valeur peut être zéro si le serveur HTTP n'indique pas la longueur du contenu dans l'en-tête MIME.
#error	Chaîne contenant "" (`EMPTY`) si le téléchargement n'est pas terminé, OK s'il a abouti ou un code d'erreur s'il s'est achevé par une erreur.

Par exemple, vous pouvez démarrer une opération réseau avec `getNetText()` et suivre sa progression avec `getStreamStatus()`.

Paramètres

`netID` Requis. Opération réseau représentant le flux de texte concerné par la méthode.

Exemple

L'instruction suivante affiche dans la fenêtre Messages l'état d'un téléchargement démarré avec `getNetText()` et l'identifiant réseau résultant placé dans la variable `netID` :

```
-- Lingo
put getStreamStatus(netID)
-- [#URL: "www.adobe.com", #state: "InProgress", #bytesSoFar: 250, #bytesTotal: 50000,
#error: EMPTY]

// Javascript
trace(getStreamStatus(netID))
// <[#URL: "www.adobe.com", #state: "InProgress", #bytesSoFar: 250, #bytesTotal: 50000,
#error: EMPTY]>
```

Voir aussi

`on streamStatus`, `tellStreamStatus()`

getURL()

Syntaxe

```
GetUrl(MUIObject, message, MovableOrNot)
```

Description

Cette fonction affiche une boîte de dialogue de saisie d'URL, et renvoie l'URL saisie par l'utilisateur.

- *message* spécifie le message qui s'affiche dans le champ de saisie d'URL. Lorsque la boîte de dialogue s'ouvre, cette chaîne est envoyée en tant que valeur prédéfinie. Lorsque l'utilisateur clique sur un bouton, Lingo renvoie la chaîne saisie par l'utilisateur. Si l'utilisateur clique sur Annuler, la chaîne renvoyée est identique à la valeur d'origine.
- Sur Macintosh, *MovableOrNot* spécifie si la boîte de dialogue est déplaçable. Si la valeur est TRUE, la boîte de dialogue est déplaçable. Si la valeur est FALSE, la boîte de dialogue est non déplaçable. La boîte de dialogue de saisie d'URL est toujours déplaçable sous Windows.

Exemple

Ces instructions affichent une boîte de dialogue de saisie d'URL.

- La première instruction crée une instance d'Xtra MUI, qui est l'objet utilisé en tant que boîte de dialogue.
- La deuxième instruction utilise la fonction getURL pour afficher une boîte de dialogue de saisie d'URL déplaçable, et attribue la boîte de dialogue au résultat de la variable. Le message « Saisissez une URL ici » s'affiche dans le champ de saisie d'URL de la boîte de dialogue.
- Les dernières instructions vérifient que le résultat obtenu lorsque l'utilisateur clique sur un bouton est identique à la chaîne envoyée lors de l'ouverture de la boîte de dialogue. Si le résultat est différent, cela signifie que l'utilisateur a saisi une URL et cliqué sur OK.

```
-- Lingo
set MUIObj = new (xtra "Mui")

set result = GetUrl(MUIObj, "Enter a URL", TRUE )

if objectP ( MUIObj) then
   set result = GetUrl( MUIObj, "Enter a URL", TRUE )
   if ( result <> "Enter a URL" ) then
      goToNetPage result
   end if
```

```
end if
```

getVal()

Syntaxe

```
<float> Matrix.getVal(whichRow, whichColumn)
```

Description

Méthode de matrice ; récupère la valeur de l'élément spécifié dans la matrice indiquée.

Paramètres

whichRow Requis. Numéro de ligne de l'élément dont la valeur est lue.

whichColumn Requis. Numéro de colonne de l'élément dont la valeur est lue.

Exemple

La fonction suivante utilise la méthode getVal() d'une matrice pour convertir une matrice en liste linéaire.

```
--Lingo
on matrixToList(mat)
  rows = mat.numRows
  cols = mat.numColumns
  matrixList = []
  repeat with i = 1 to rows
    repeat with j = 1 to cols
      matrixList.append(mat.getVal(i,j))
    end repeat
  end repeat
  return matrixList
end
```

```
//Java Script
function matrixToList(mat)
{
  rows = mat.numRows;
  cols = mat.numColumns;
  matrixList = list();
  for( i = 1; i <= rows; i++)
  {
    for( j = 1; j <= cols; j++)
      matrixList.append(mat.getVal(i,j));
}
  return matrixList;
}
```

Voir aussi

setVal(), numRows(), numColumns(), matrixAddition(), matrixMultiply(), matrixMultiplyScalar(), matrixTranspose(), newMatrix()

getVariable()

Syntaxe

```
-- Lingo syntax
```

```
spriteObjRef.getVariable(variableName {, returnValueOrReference})

// JavaScript syntax
spriteObjRef.getVariable(variableName {, returnValueOrReference});
```

Description

Cette fonction renvoie la valeur actuelle de la variable donnée de l'image-objet donnée. Les variables Flash ont été introduites dans la version 4 de Flash.

Cette fonction peut être utilisée de deux façons.

La définition du paramètre facultatif *returnValueOrReference* sur TRUE (valeur par défaut) renvoie la valeur en cours de la variable sous forme de chaîne. La définition du paramètre *returnValueOrReference* sur FALSE renvoie la valeur littérale en cours de la variable Flash.

Si la valeur de la variable Flash est une référence d'objet, vous devez définir le paramètre *returnValueOrReference* sur FALSE pour que la valeur renvoyée soit traitée en tant que référence d'objet. Si elle est renvoyée en tant que chaîne, la chaîne ne constitue pas une référence d'objet valide.

Paramètres

variableName Requis. Spécifie le nom de la variable dont la valeur est renvoyée.

returnValueOrReference Facultatif. Spécifie si la valeur renvoyée est une chaîne (TRUE) ou une référence d'objet (FALSE).

Exemple

L'instruction suivante définit la variable tValue sur la valeur de la chaîne de la variable Flash appelée gOtherVar dans l'animation Flash au niveau de l'image-objet 3 :

```
-- Lingo syntax
tValue = sprite(3).getVariable("gOtherVar", TRUE)
put(tValue) -- "5"

// JavaScript syntax
var tValue = sprite(3).getVariable("gOtherVar", true);
trace(tValue); // 5
```

L'instruction suivante définit la variable tObject de manière à référencer le même objet que la variable appelée gVar dans l'animation Flash au niveau de l'image-objet 3 :

```
-- Lingo syntax
tObject = sprite(3).getVariable("gVar",FALSE)
// JavaScript syntax
var tObject = sprite(3).getVariable("gVar",0);
```

L'instruction suivante renvoie la valeur de la variable currentURL de l'acteur Flash de l'image-objet 3 et l'affiche dans la fenêtre Messages :

```
-- Lingo syntax
put(sprite(3).getVariable("currentURL"))

// JavaScript syntax
trace(sprite(3).getVariable("currentURL"));
```

Voir aussi

setVariable()

GetWidgetList()

Syntaxe

```
GetWidgetList(MUIObject)
```

Description

Cette fonction renvoie une liste linéaire des symboles des types de composants de boîte de dialogue générale pris en charge par une instance de l'Xtra MUI.

Exemple

Cette instruction affiche une liste de contrôles pris en charge par MUIObject, instance de l'Xtra MUI :

```
-- Lingo
put GetWidgetList(MUIObject)

-- [#dividerV, #dividerH, #bitmap, #checkBox, #radioButton, #PopupList, #editText,
#WindowBegin, #WindowEnd, #GroupHBegin, #GroupHEnd, #GroupVBegin, #GroupVEnd, #label,
#IntegerSliderH, #FloatSliderH, #defaultPushButton, #cancelPushButton, #pushButton,
#toggleButton]
```

GetWindowPropList

Syntaxe

```
GetWindowPropList(MUIObject)
```

Description

Cette fonction renvoie une liste des paramètres prédéfinis de l'XtraMUI pour la fenêtre d'une boîte de dialogue générale.

Lors de la définition d'une nouvelle boîte de dialogue générale, utilisez la fonction GetWindowPropList pour obtenir une liste complète des propriétés de la boîte de dialogue et de leurs valeurs, puis modifiez individuellement les propriétés si nécessaire. Très pratique, cette technique permet la compatibilité avec les versions ultérieures de l'Xtra MUI, susceptibles de posséder des propriétés supplémentaires.

Les propriétés de la fenêtre et leurs valeurs prédéfinies renvoyées par la fonction GetWindowPropList sont indiquées dans le tableau ci-dessous.

Propriété	Valeur prédéfinie
#type	#normal
#name	"window"
#callback	"nothing"
#mode	#data
#xPosition	100
#yPosition	120
#width	200
#height	210
#modal	1

Propriété	Valeur prédéfinie
#toolTips	0
#closeBox	1
#canZoom	0

Exemple

Ces instructions définissent une nouvelle boîte de dialogue générale. La première instruction attribue une liste de propriétés prédéfinies à la variable thePropList. Les instructions suivantes personnalisent la boîte de dialogue en modifiant les paramètres ci-dessous :

```
-- Lingo
set thePropList = GetWindowPropList(muiObject)

set the name of thePropList = "Picture Window"
set the callback of thePropList = "theWindowCallback"
set the mode of thePropList = #data
set the modal of thePropList = TRUE
set the closeBox of thePropList = FALSE
```

getWorldTransform()

Syntaxe

```
member(whichCastmember).node(whichNode).getWorldTransform()
member(whichCastmember).node(whichNode).getWorldTransform().position
member(whichCastmember).node(whichNode).getWorldTransform().rotation
member(whichCastmember).node(whichNode).getWorldTransform().scale
```

Description

Commande 3D ; renvoie la transformation par rapport à l'univers du modèle, du groupe, de la caméra ou de la lumière représenté par `node`.

La propriété `transform` du nœud est calculée en fonction de la transformation du parent du nœud et est donc relative au parent. La commande `getWorldTransform()` calcule la transformation du nœud par rapport à l'origine de l'univers 3D et est donc relative à l'univers.

Utilisez `(`*`whichCastmember`*`).node(`*`whichNode`*`).getWorldTransform()` pour rechercher la propriété de position de la transformation du nœud par rapport à l'univers. Vous pouvez également utiliser `worldPosition` comme raccourci de `getWorldTransform().position`.

Utilisez `member(`*`whichCastmember`*`).node(`*`whichNode`*`).getWorldTransform()` pour rechercher la propriété de rotation de la transformation du nœud par rapport à l'univers.

Utilisez `member(`*`whichCastmember`*`).node(`*`whichNode`*`).getWorldTransform().scale` pour rechercher la propriété d'échelle de la transformation du nœud par rapport à l'univers.

Ces propriétés peuvent être testées, mais pas définies.

Exemple

L'instruction suivante indique la transformation relative à l'univers du modèle Boîte, suivi de ses propriétés de position et de rotation.

```
put member("3d world").model("Box").getworldTransform()
```

```
-- transform(1.000000,0.000000,0.000000,0.000000, 0.000000,1.000000,0.000000,0.000000,
0.000000,0.000000,1.000000,0.000000, - 94.144844,119.012825,0.000000,1.000000)
put member("3d world").model("Box"). getworldTransform().position
-- vector(-94.1448, 119.0128, 0.0000)
put member("3d world").model("Box"). getworldTransform().rotation
--vector(0.0000, 0.0000, 0.0000)

// Javascript
trace(member("3d world").getProp("model","Box").getworldTransform()
// <transform(1.000000,0.000000,0.000000,0.000000, 0.000000,1.000000,0.000000,0.000000,
0.000000,0.000000,1.000000,0.000000, -94.144844,119.012825,0.000000,1.000000)>
trace(member("3d world").getProp("model","Box"). getworldTransform().position
// <vector(-94.1448, 119.0128, 0.0000)>
trace(member("3d world").getProp("model","Box"). getworldTransform().rotation
//<vector(0.0000, 0.0000, 0.0000)>
```

Voir aussi

worldPosition, transform (propriété)

go()

Syntaxe

```
-- Lingo syntax
_movie.go(frameNameOrNum {, movieName})

// JavaScript syntax
_movie.go(frameNameOrNum {, movieName});
```

Description

Méthode d'animation ; place la tête de lecture sur une image spécifiée d'une animation spécifique.

Cette méthode est utilisable pour indiquer à la tête de lecture d'effectuer une boucle vers le repère précédent et se révèle utile pour garder la tête de lecture dans la même partie de l'animation pendant que le script reste actif.

Il est préférable d'utiliser des libellés de repère pour *frameNameOrNum* plutôt que des numéros d'image car la modification d'une animation peut entraîner le changement des numéros d'image. L'utilisation du libellé des repères facilite également la lecture des scripts.

L'appel de la méthode `go()` avec le paramètre *movieName* charge l'image 1 de l'animation. Si la méthode `go()` est appelée à partir d'un gestionnaire, le gestionnaire dans lequel elle est placée poursuit son exécution.

Lorsque vous spécifiez une animation à lire, indiquez son chemin d'accès si l'animation se trouve dans un dossier différent mais, afin d'empêcher un échec possible du chargement, n'incluez pas l'extension du fichier de l'animation (.dir, .dxr ou .dcr).

Pour accéder plus facilement à une animation au niveau d'une URL, utilisez la méthode `downloadNetThing()` pour télécharger le fichier de l'animation sur un disque local, puis utilisez la commande `go()` avec le paramètre *movieName* pour accéder à cette animation sur le disque local.

La méthode `goLoop()` envoie la tête de lecture sur le marqueur précédent d'une animation, ce qui constitue un moyen pratique pour garder la tête de lecture dans la même partie de l'animation pendant que la syntaxe Lingo ou JavaScript reste active.

Les paramètres suivants sont réinitialisés au chargement d'une animation : propriétés `beepOn` et `constraint` ; `keyDownScript`, `mouseDownScript` et `mouseUpScript` ; propriétés d'image-objet `cursor` et `immediate` ; méthodes `cursor()` et `puppetSprite()` et menus personnalisés. Toutefois, la propriété `timeoutScript` n'est pas réinitialisée au chargement d'une animation.

Paramètres

`frameNameOrNum` Requis. Chaîne spécifiant le libellé de repère de l'image sur laquelle la tête de lecture vient se placer ou nombre entier spécifiant le numéro de cette image.

`movieName` Facultatif. Chaîne indiquant l'animation contenant l'image spécifiée par *`frameNameOrNum`*. Cette valeur doit indiquer un fichier d'animation ; si l'animation se trouve dans un autre dossier, *`movieName`* doit également spécifier le chemin d'accès.

Exemple

L'instruction suivante place la tête de lecture sur le point de repère intitulé Début :

```
-- Lingo syntax
_movie.go("start")

// JavaScript syntax
_movie.go("start");
```

L'instruction suivante fait passer la tête de lecture au repère Memory dans l'animation intitulée Mon animation :

```
-- Lingo syntax
_movie.go("Memory", "Noh Tale to Tell")

// JavaScript syntax
_movie.go("Memory", "Noh Tale to Tell");
```

Le gestionnaire suivant indique à l'animation d'effectuer une boucle sur l'image en cours. Ce gestionnaire se révèle utile pour demander à l'animation d'attendre dans une image tout en répondant aux événements.

```
-- Lingo syntax
on exitFrame
    _movie.go(_movie.frame)
end

// JavaScript syntax
function exitFrame() {
    _movie.go(_movie.frame);
}
```

Voir aussi

`downloadNetThing`, `goLoop()`, `Animation`

goLoop()

Syntaxe

```
-- Lingo syntax
_movie.goLoop()

// JavaScript syntax
_movie.goLoop();
```

Description

Méthode d'animation ; place la tête de lecture sur le repère précédent dans l'animation, correspondant soit au repère situé avant l'image en cours si cette dernière n'a pas de repère, soit au repère de l'image en cours si celle-ci en comporte un.

Si aucun repère ne se trouve à gauche de la tête de lecture, la tête de lecture se place sur :

- Le prochain repère à droite si l'image actuelle ne possède pas de repère.
- L'image actuelle si celle-ci possède un repère.
- L'image 1 si l'animation ne contient aucun repère.

Paramètres

Aucune.

Exemple

L'instruction suivante demande à l'animation d'effectuer une boucle entre l'image en cours et le repère précédent :

```
-- Lingo syntax
_movie.goLoop()

// JavaScript syntax
_movie.goLoop();
```

Voir aussi

go(), goNext(), goPrevious(), Animation

goNext()

Syntaxe

```
-- Lingo syntax
_movie.goNext()

// JavaScript syntax
_movie.goNext();
```

Description

Méthode d'animation ; place la tête de lecture sur le repère suivant dans l'animation.

Si aucun repère ne se trouve à droite de la tête de lecture, celle-ci se place sur le dernier repère de l'animation ou sur l'image 1 si l'animation ne contient aucun repère.

Paramètres

Aucune.

Exemple

L'instruction suivante envoie la tête de lecture sur le repère suivant de l'animation :

```
-- Lingo syntax
_movie.goNext()

// JavaScript syntax
_movie.goNext();
```

Voir aussi

go(), goLoop(), goPrevious(), Animation

goPrevious()

Syntaxe

```
-- Lingo syntax
_movie.goPrevious()

// JavaScript syntax
_movie.goPrevious();
```

Description

Méthode d'animation ; place la tête de lecture sur le repère précédent dans l'animation.

Ce repère est situé soit deux repères avant l'image en cours si cette dernière ne comporte pas de repère, soit juste avant l'image en cours si celle-ci comporte un repère.

Si aucun repère ne se trouve à gauche de la tête de lecture, la tête de lecture se place sur un des éléments suivants :

- Le prochain repère à droite si l'image actuelle ne possède pas de repère.
- L'image actuelle si celle-ci possède un repère.
- L'image 1 si l'animation ne contient aucun repère.

Paramètres

Aucune.

Exemple

L'instruction suivante envoie la tête de lecture sur le repère précédent dans l'animation :

```
-- Lingo syntax
_movie.goPrevious()

// JavaScript syntax
_movie.goPrevious();
```

Voir aussi

go(), goLoop(), goNext(), Animation

goToFrame()

Syntaxe

```
-- Lingo syntax
spriteObjRef.goToFrame(frameNameOrNum)

// JavaScript syntax
spriteObjRef.goToFrame(frameNameOrNum);
```

Description

Commande ; lit une image-objet d'animation Flash à partir de l'image identifiée par le paramètre *frameNumber*. Vous pouvez identifier l'image par un nombre entier indiquant le numéro d'image ou par une chaîne indiquant le nom du libellé. L'utilisation de la commande `goToFrame` produit le même effet que la définition de la propriété `frame` d'une image-objet d'animation Flash.

Exemple

Le gestionnaire suivant passe à différents points d'une animation Flash dans la piste 5. Il accepte un paramètre indiquant l'image à laquelle passer.

```
-- Lingo syntax
on Navigate(whereTo)
    sprite(5).goToFrame(whereTo)
end

// JavaScript syntax
function Navigate(whereTo) {
    sprite(5).goToFrame(whereTo);
}
```

gotoNetMovie

Syntaxe

```
gotoNetMovie URL
gotoNetMovie (URL)
```

Description

Commande ; récupère et lit une nouvelle animation comportant du contenu Shockwave à partir d'un serveur HTTP ou FTP. L'exécution de l'animation actuelle continue jusqu'à ce que la nouvelle animation soit disponible.

Seules les URL sont supportées comme paramètres valides. L'URL peut spécifier un nom de fichier ou un repère dans une animation. Les URL relatives fonctionnent si l'animation se trouve sur un serveur Internet, mais vous devez inclure l'extension avec le nom de fichier.

Lorsque vous réalisez des tests sur un disque ou réseau local, les médias doivent se trouver dans un répertoire appelé dswmedia.

Si une opération `gotoNetMovie` est en cours et que vous envoyez une seconde commande `gotoNetMovie` avant la fin de la première, la seconde commande annule la première.

Paramètres

URL Requis. Spécifie l'URL du contenu Shockwave à lire.

Exemple

Dans l'instruction suivante, l'URL indique un nom de fichier Director :

```
gotoNetMovie "http://www.yourserver.com/movies/movie1.dcr"
```

Dans l'instruction suivante, l'URL indique un repère dans un nom de fichier :

```
gotoNetMovie "http://www.yourserver.com/movies/buttons.dcr#Contents"
```

Dans l'instruction suivante, la commande `gotoNetMovie` est utilisée comme une fonction. Cette fonction renvoie l'identifiant réseau pour l'opération.

```
myNetID = gotoNetMovie ("http://www.yourserver.com/movies/buttons.dcr#Contents")
```

gotoNetPage

Syntaxe

```
gotoNetPage "URL", {"targetName"}
```

Description

Commande ; ouvre une animation comportant du contenu Shockwave ou un autre fichier MIME dans le navigateur.

Seules les URL sont supportées comme paramètres valides. Les URL relatives fonctionnent si l'animation se trouve sur un serveur HTTP ou FTP.

Dans l'environnement de création, la commande `gotoNetPage` lance le navigateur favori s'il est activé. Dans les projections, cette commande tente de lancer le navigateur favori, défini à l'aide de la boîte de dialogue des préférences réseau ou de la commande `browserName`. Si aucune commande n'a été utilisée pour définir le navigateur favori, la commande `goToNetPage` tente de trouver un navigateur sur l'ordinateur.

Paramètres

`URL` Requis. Spécifie l'URL de l'animation comportant du contenu Shockwave ou du fichier MIME à lire.

`targetName` Facultatif. Paramètre HTML identifiant la fenêtre ou le cadre dans lequel la page est chargée.

- Si *`targetName`* est une fenêtre ou un cadre dans le navigateur, la méthode `gotoNetPage` en remplace le contenu.
- Si *`targetName`* n'est pas un cadre ou une fenêtre ouvert(e), `goToNetPage` ouvre une nouvelle fenêtre. L'utilisation de la chaîne `"_new"` provoque systématiquement l'ouverture d'une nouvelle fenêtre.
- Si *`targetName`* est omis, `gotoNetPage` remplace la page en cours, où qu'elle se trouve.

Exemple

Le script suivant charge le fichier nouvellePage.html dans le cadre ou la fenêtre intitulé(e) frwin. S'il existe une fenêtre ou un cadre intitulé(e) frwin, cette fenêtre ou ce cadre est utilisé(e). Si la fenêtre frwin n'existe pas, une nouvelle fenêtre intitulée frwin est créée.

```
on keyDown
 gotoNetPage "Newpage.html", "frwin"
end
```

Le gestionnaire suivant ouvre une nouvelle fenêtre, quelle que soit la fenêtre ouverte par le navigateur :

```
on mouseUp
 gotoNetPage "Todays_News.html", "_new"
end
```

Voir aussi

`browserName()`, `netDone()`

group()

Syntaxe

```
member(whichCastmember).group(whichGroup)
member(whichCastmember).group[index]
```

Description

Elément 3D ; nœud de l'univers 3D qui a un nom, une transformation, un parent et des enfants, mais aucune autre propriété.

Chaque acteur 3D est associé à un groupe par défaut nommé Univers et qui ne peut pas être supprimé. La hiérarchie parente de tous les modèles, lumières, caméras et groupes qui existent dans l'univers 3D s'achève dans `group("world")`.

Exemple

L'instruction suivante indique que le quatrième groupe de l'acteur nouveauMartien est le groupe Direct01.

```
-- Lingo
put member("newAlien").group[4]

// Javascript
put member("newAlien").getPropRef("group",4) ;
-- group("Direct01")
```

Voir aussi

`newGroup`, `deleteGroup`, `child (3D)`, `parent`

halt()

Syntaxe

```
-- Lingo syntax
_movie.halt()

// JavaScript syntax
_movie.halt();
```

Description

Méthode d'animation ; quitte le gestionnaire en cours ainsi que tout autre gestionnaire ayant appelé cette méthode et arrête l'animation pendant la phase de création ou quitte la projection pendant son exécution.

Paramètres

Aucune.

Exemple

L'instruction suivante vérifie si la quantité de mémoire disponible est inférieure à 50 Ko et, le cas échéant, quitte tous les gestionnaires qui l'ont appelée, puis arrête l'animation :

```
-- Lingo syntax
if (_system.freeBytes < (50*1024)) then
    _movie.halt()
end if

// JavaScript syntax
if (_system.freeBytes < (50*1024)) {
    _movie.halt();
}
```

Voir aussi

`Animation`

handler()

Syntaxe

```
scriptObject.handler(#handlerSymbol)
```

Description

Cette fonction renvoie la valeur `TRUE` si l'`scriptObject` donné contient un gestionnaire spécifié, et la valeur `FALSE` dans le cas contraire. L'objet script doit être un script parent, un objet enfant ou un comportement.

Paramètres

`symHandler` Requis. Spécifie le nom du gestionnaire.

Exemple

Le code suivant appelle un gestionnaire sur un objet, uniquement si ce gestionnaire existe :

```
-- Lingo
if spiderObject.handler(#pounce) = TRUE then
    spiderObject.pounce()
end if

// Javascript
if (spiderObject.handler(symbol("pounce")) == true)
{
    spiderObject.pounce();
}
```

Voir aussi

`handlers()`, `new()`, `rawNew()`, `script()`

handlers()

Syntaxe

```
scriptObject.handlers()
```

Description

Cette fonction renvoie une liste linéaire des gestionnaires dans l'*`scriptObject`* donné. Chaque nom de gestionnaire est présenté sous forme de symbole dans la liste. Cette fonction est pratique pour le débogage des animations.

Vous ne pouvez pas accéder directement aux gestionnaires d'un acteur script. Vous devez y accéder par l'intermédiaire de la propriété `script` de l'acteur.

Paramètres

Aucune.

Exemple

L'instruction suivante affiche la liste des gestionnaires de l'objet enfant voitureRouge dans la fenêtre Messages :

```
put RedCar.handlers()
-- [#accelerate, #turn, #stop]
```

L'instruction suivante affiche la liste des gestionnaires de l'acteur script parent ScriptParentDeVoiture dans la fenêtre Messages :

```
put member("CarParentScript").script.handlers()
-- [#accelerate, #turn, #stop]
```

Voir aussi

handler(), script()

hilite (commande)

Syntaxe

```
fieldChunkExpression.hilite()
hilite fieldChunkExpression
```

Description

Commande ; sélectionne l'expression de sous-chaîne spécifiée dans l'image-objet champ, qui peut être n'importe quelle sous-chaîne que Lingo vous permet de définir, telle qu'un caractère, un mot ou une ligne. Sur le Mac, la couleur de surlignage est définie dans le tableau de bord Couleur.

Paramètres

Aucune.

Exemple

L'instruction suivante sélectionne le troisième mot dans l'acteur champ Commentaires, qui contient la chaîne Pensée du jour :

```
-- Lingo
member("Comments").word[4].hilite()

// Javascript
member("Comments").getPropRef("word",4).hilite() ;
```

Voir aussi

char...of, item...of, line...of, word...of, delete(), mouseChar, mouseLine, mouseWord, field, selection() (fonction), selEnd, selStart

hitTest()

Syntaxe

```
-- Lingo syntax
spriteObjRef.hitTest(point)

// JavaScript syntax
spriteObjRef.hitTest(point);
```

Description

Fonction ; indique la partie d'une animation Flash se trouvant directement sur un emplacement spécifique de la scène Director. L'emplacement de la scène Director est exprimé en valeur de points Director : par exemple, point(100,50). La fonction `hitTest` renvoie les valeurs suivantes :

- `#background` : l'emplacement de scène spécifié se situe dans l'arrière-plan de l'image-objet d'animation Flash.
- `#normal` : l'emplacement de scène spécifié se situe dans un objet rempli.

- `#button` : l'emplacement de scène spécifié se situe dans la zone active d'un bouton.
- `#editText` : l'emplacement de scène spécifié se situe dans un champ de texte modifiable Flash.

Paramètres

`point` Requis. Spécifie le point à tester.

Exemple

Le script d'image suivant vérifie si la souris se trouve au-dessus d'un bouton dans une image-objet animation Flash dans la piste 5 et, le cas échéant, le script définit un champ de texte utilisé pour afficher un message d'état :

```
-- Lingo syntax
on exitFrame
    if sprite(5).hitTest(_mouse.mouseLoc) = #button then
        member("Message Line").text = "Click here to play the movie."
        _movie.updatestage()
    else
        member("Message Line").text = ""
    end if
    _movie.go(_movie.frame)
end

// JavaScript syntax
function exitFrame() {
    var hT = sprite(5).hitTest(_mouse.mouseLoc);
    if (hT.toString() == "#button")
    {
        member("Message Line").text = "Click here to play the movie.";
        _movie.updatestage();
    } else {
        member("Message Line").text = "";
    }
    _movie.go(_movie.frame)
}
```

HMStoFrames()

Syntaxe

```
HMStoFrames(hms, tempo, dropFrame, fractionalSeconds)
```

Description

Fonction ; convertit les animations mesurées en heures, minutes et secondes au nombre d'images équivalent ou convertit un nombre d'heures, minutes et secondes en temps si vous attribuez à l'argument *`tempo`* la valeur 1 (1 image = 1 seconde).

Paramètres

`hms` Requis. Expression de chaîne spécifiant le temps sous la forme `sHH:MM:SS.FFD`, où :

s	Un caractère est utilisé si le temps est inférieur à zéro ou un espace si le temps est supérieur ou égal à zéro.
HH	Heures.
MM	Minutes.

SS	Secondes.
FF	Indique une fraction de seconde si `fractions` présente la valeur `TRUE` ou des images si `fractions` présente la valeur `FALSE`.
D	Un d est utilisé si *`dropFrame`* présente la valeur `TRUE` , un espace est utilisé si *`dropFrame`* présente la valeur `FALSE`.

`tempo` Requis. Spécifie la cadence en images par seconde.

`dropFrame` Requis. Expression logique déterminant si l'image est compensée (`TRUE`) ou non (`FALSE`). Si la chaîne *`hms`* se termine par *d*, le temps est traité comme une image compensée, quelle que soit la valeur de l'argument *`compensé`*.

`fractionalSeconds` Requis. Expression logique déterminant la signification des nombres après les secondes ; il peut s'agir de fractions de secondes arrondies au centième de seconde le plus proche (`TRUE`) ou du nombre d'images résiduelles (`FALSE`).

Exemple

L'instruction suivante détermine le nombre d'images dans une animation d'une minute, 30,1 secondes lorsque la cadence est de 30 images par seconde. Les arguments `dropFrame` et `fractionalSeconds` ne sont pas utilisés.

```
put HMStoFrames(" 00:01:30.10 ", 30, FALSE, FALSE)
-- 2710
```

L'instruction suivante convertit 600 secondes en minutes :

```
put framesToHMS(600, 1,0,0)
-- " 00:10:00.00 "
```

L'instruction suivante convertit une heure et demie en secondes :

```
put HMStoFrames("1:30:00", 1,0,0)
-- 5400
```

Voir aussi

`framesToHMS()`

hold()

Syntaxe

```
-- Lingo syntax
spriteObjRef.hitTest(point)

// JavaScript syntax
spriteObjRef.hitTest(point);
```

Description

Commande Flash ; arrête une image-objet animation Flash lue dans l'image actuelle, sans interrompre la lecture audio.

Paramètres

Aucune.

Exemple

Le script d'image suivant arrête la lecture des images-objets animation Flash dans les pistes 5 à 10 sans interrompre la lecture audio de ces pistes :

```
-- Lingo syntax
on enterFrame
    repeat with i = 5 to 10
        sprite(i).hold()
    end repeat
end


// JavaScript syntax
function enterFrame() {
    var i = 5;
    while (i < 11) {
        sprite(i).hold();
        i++;
    }
}
```

Voir aussi

playRate

identity()

Syntaxe

```
member(whichCastmember).model(whichModel).transform.identity()
member(whichCastmember).group(whichGroup).transform.identity()
member(whichCastmember).camera(whichCamera).transform.identity()
sprite(whichSprite).camera{(index)}.transform.identity()
member(whichCastmember).light(whichLight).transform.identity()
transformReference.identity()
```

Description

Commande 3D ; définit la transformation en tant que transformation d'identité, correspondant à `transform(1.0000,0.0000,0.0000,0.0000, 0.0000,1.0000,0.0000,0.0000, 0.0000,0.0000,1.0000,0.0000, 0.0000,0.0000,0.0000,1.0000)`.

La propriété `position` de la transformation d'identité est `vector(0, 0, 0)`.

La propriété `rotation` de la transformation d'identité est `vector(0, 0, 0)`.

La propriété `scale` de la transformation d'identité est `vector(1, 1, 1)`.

La transformation d'identité est relative au parent.

Paramètres

Aucune.

Exemple

L'instruction suivante fait de la transformation du modèle Boîte la transformation d'identité.

```
-- Lingo
member("3d world").model("Box").transform.identity()
```

```
// Javascript
member("3d world").getProp("model",1).transform.identity()
```

Le script suivant suppose que le modèle Boîte est le premier modèle disponible.

Voir aussi

transform (propriété), getWorldTransform()

idleLoadDone()

Syntaxe

```
-- Lingo syntax
_movie.idleLoadDone(intLoadTag)

// JavaScript syntax
_movie.idleLoadDone(intLoadTag);
```

Description

Méthode d'animation ; indique si tous les acteurs présentant la balise spécifiée ont été chargés (TRUE) ou s'ils sont encore en attente de chargement (FALSE).

Paramètres

intLoadTag Requis. Nombre entier spécifiant la balise de chargement des acteurs à tester.

Exemple

L'instruction suivante vérifie si tous les acteurs dont la balise de chargement est 20 ont été chargés et, le cas échéant, lit l'animation Kiosque :

```
-- Lingo syntax
if (_movie.idleLoadDone(20)) then
    _movie.play(1, "Kiosk")
end if

// JavaScript syntax
if (_movie.idleLoadDone(20)) {
    _movie.play(1, "Kiosk");
}
```

Voir aussi

idleHandlerPeriod, idleLoadMode, idleLoadPeriod, idleLoadTag, idleReadChunkSize, Animation

ignoreWhiteSpace()

Syntaxe

```
XMLparserObject.ignoreWhiteSpace(trueOrFalse)
```

Description

Commande XML ; spécifie si l'analyseur doit ignorer ou conserver les espaces blancs dans les listes Lingo. Lorsque la commande ignoreWhiteSpace() présente la valeur TRUE (valeur par défaut), l'analyseur ignore les espaces blancs. Lorsque cette commande présente la valeur FALSE, l'analyseur conserve les espaces blancs et les traite comme des données réelles.

Lorsqu'un élément possède des balises initiales et finales distinctes, telles que `<exemple> </exemple>`, les caractères de l'élément sont ignorés si (et uniquement si) il n'est composé que d'espaces blancs. En cas de présence d'un espace autre qu'un espace blanc ou si `ignoreWhiteSpace()` présente la valeur `FALSE`, un nœud CDATA contenant le même texte y compris l'espace blanc, est créé.

Paramètres

`trueOrFalse` Requis. Valeur spécifiant si l'analyseur doit ignorer l'espace blanc (`TRUE`) ou non (`FALSE`).

Exemple

Les instructions Lingo suivantes laissent la commande `ignoreWhiteSpace()` définie sur sa valeur par défaut de `TRUE` et convertissent le document XML donné en une liste. L'élément `<exemple>` ne comporte aucun enfant dans la liste.

```
XMLtext = "<sample> </sample>"
parserObj.parseString(XMLtext)
theList = parserObj.makelist()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES": [:]]]]
```

Les instructions Lingo suivantes définissent `ignoreWhiteSpace()` sur la valeur `FALSE` et convertissent le document XML donné en une liste. L'élément `<exemple>` comporte à présent un enfant contenant un espace.

```
XMLtext = "<sample> </sample>"
parserObj.ignorewhitespace(FALSE)
parserObj.parseString(XMLtext)
theList = parserObj.makelist()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES": [:], "!CHARDATA":
" "]]]
```

Les instructions Lingo suivantes laissent la commande `ignoreWhiteSpace()` définie sur sa valeur par défaut de `TRUE` et analysent le document XML donné. Il n'existe qu'un seul nœud enfant de la balise <exemple> et qu'un seul nœud enfant de la balise `<inf>`.

```
XMLtext = "<sample> <sub> phrase 1 </sub></sample>"
parserObj.parseString(XMLtext)
theList = parserObj.makeList()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES": [:], "sub":
["!ATTRIBUTES": [:], "!CHARDATA": " phrase 1 "]]]]
```

Les instructions Lingo suivantes définissent `ignoreWhiteSpace()` sur la valeur `FALSE` et analysent le document XML donné. Il existe à présent deux nœuds enfants de la balise `<exemple>`, le premier étant un espace.

```
XMLtext = "<sample> <sub> phrase 1 </sub></sample>"
gparser.ignoreWhiteSpace(FALSE)
gparser.parseString(XMLtext)
theList = gparser.makeList()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES": [:], "!CHARDATA":
" ", "sub": ["!ATTRIBUTES": [:], "!CHARDATA": " phrase 1 "]]]]
```

ilk()

Syntaxe

```
ilk(object)
```

```
ilk(object, type)
```

Description

Fonction ; indique le type d'un objet.

Le tableau suivant présente la valeur renvoyée pour chaque type d'objet reconnu par `ilk()` :

Type d'objet	**ilk(objet) renvoie**	**ilk(objet, type) renvoie 1 uniquement si le type =**	**Exemple**
liste linéaire	`#list`	`#list` ou `#linearlist`	`ilk ([1,2,3])`
liste de propriétés	`#proplist`	`#list` ou `#proplist`	`ilk ([#Alui: 1234, #Aelle: 7890])`
entier	`#integer`	`#integer` ou `#number`	`ilk (333)`
valeur à virgule flottante	`#float`	`#float` ou `#number`	`ilk (123,456)`
chaîne	`#string`	`#string`	`ilk ("asdf")`
rect	`#rect`	`#rect` ou `#list`	`ilk (sprite(1).rect)`
point	`#point`	`#point` ou `#list`	`ilk (sprite(1).loc)`
couleur	`#color`	`#color`	`ilk (sprite(1).color)`
date	`#date`	`#date`	`ilk (the systemdate)`
symbole	`#symbol`	`#symbol`	`ilk (#bonjour)`
void	`#void`	`#void`	`ilk (void)`
image	`#picture`	`#picture`	`ilk (member (2).picture)`
instance de script parent	`#instance`	`#object`	`ilk (new (script "blabla"))`
instance d'Xtra	`#instance`	`#object`	`ilk (new (xtra "fileio"))`
acteur	`#member`	`#object` ou `#member`	`ilk (member 1)`
Xtra	`#xtra`	`#object` ou `#xtra`	`ilk (xtra "fileio")`
script	`#script`	`#object` ou `#script`	`ilk (script "blabla")`
castlib	`#castlib`	`#object` ou `#castlib`	`ilk (castlib 1)`
image-objet	`#sprite`	`#object` ou `#sprite`	`ilk (sprite 1)`
son	`#instance` ou `#sound` (lorsque l'Xtra de contrôle audio est absent)	`#instance` ou #sound	`ilk (sound "tralala")`
fenêtre	`#window`	`#object` ou `#window`	`ilk (the stage)`
médias	`#media`	`#object` ou `#media`	`ilk (member (2).media)`
temporisation	`#timeout`	`#object` ou `#timeout`	`ilk (timeOut("compteurIntervalle") )`
image	`#image`	`#object` ou `#image`	ilk ((the stage).image)

Paramètres

object Requis. Spécifie l'objet à tester.

type Facultatif. Spécifie le type auquel *object* est comparé. Si l'objet est du type spécifié, la fonction `ilk()` renvoie la valeur `TRUE`. Si l'objet n'est pas du type spécifié, la fonction `ilk()` renvoie la valeur `FALSE`.

Exemple

L'instruction `ilk` suivante identifie le type de l'objet appelé Devis :

```
Bids = [:]
put ilk( Bids )
-- #proplist
```

L'instruction ilk suivante teste si la variable `Total` est une liste et affiche le résultat dans la fenêtre Messages :

```
Total = 2+2
put ilk( Total, #list )
-- 0
```

Dans ce cas précis, étant donné que la variable `Total` ne constitue pas une liste, la fenêtre Messages affiche 0, l'équivalent numérique de FALSE.

L'exemple suivant teste une variable intitulée `myVariable` et vérifie qu'elle correspond à un objet de date avant de l'afficher dans la fenêtre Messages :

```
myVariable = the systemDate
if ilk(myVariable, #date) then put myVariable
-- date( 1999, 2, 19 )
```

ilk (3D)

Syntaxe

```
ilk(object)
ilk(object,type)
object.ilk
object.ilk(type)
```

Description

Fonction Lingo ; indique le type d'un objet.

Le tableau suivant présente la valeur renvoyée pour chaque type d'objet 3D reconnu par `ilk()`. Le dictionnaire Lingo principal contient une liste des valeurs renvoyées pour les objets autres que 3D.

Type d'objet	ilk(objet) renvoie	ilk(objet, type) uniquement si type =
services de rendu	#renderer	#renderer
ressource de modèle	#modelResource, #plane, #box, #sphere, #cylinder, #particle, #mesh	Identique à ilk(objet), à l'exception de #modelResource qui est équivalent aux ressources générées par un fichier *.w3d importé
modèle	#model	#model
mouvement	#motion	#motion ou #list

Type d'objet	ilk(objet) renvoie	ilk(objet, type) uniquement si type =
matériau	#shader	#shader ou #list
texture	#texture	#texture ou #list
groupe	#group	#group
camera	#camera	#camera
données de collision	#collisiondata	#collisiondata
vecteur	#vector	#vector
transformation	#transform	#transform

Paramètres

object Requis. Spécifie l'objet à tester.

type Facultatif. Spécifie le type auquel *object* est comparé. Si l'objet est du type spécifié, la fonction `ilk()` renvoie la valeur `TRUE`. Si l'objet n'est pas du type spécifié, la fonction `ilk()` renvoie la valeur `FALSE`.

Exemple

L'instruction suivante indique que monObjet est un objet de mouvement.

```
put MyObject.ilk
-- #motion
```

L'instruction suivante vérifie si monObjet est un objet de mouvement. La valeur renvoyée, 1, indique qu'il s'agit bien d'un tel objet.

```
put MyObject.ilk(#motion)
-- 1
```

Voir aussi

tweenMode

image()

Syntaxe

```
-- Lingo syntax
image(intWidth, intHeight, intBitDepth)

// JavaScript syntax
image(intWidth, intHeight, intBitDepth);
```

Description

Fonction de niveau supérieur ; crée et renvoie une nouvelle image présentant les dimensions spécifiées.

Si vous créez une image à l'aide de la fonction de niveau supérieur `image()`, cette nouvelle image constitue un jeu autonome de données d'image indépendant de toutes les autres images. Par conséquent, les modifications apportées à une autre image n'ont aucun effet sur la nouvelle image.

Si vous faites référence à une image en définissant une variable comme étant égale à une image source, telle qu'un acteur ou l'image de la scène, cette variable contient une référence à l'image source. Par conséquent, toute modification apportée à l'image dans l'objet source ou dans la variable est reflétée dans l'autre image.

Pour éviter que ceci se produise et pour créer une copie d'une image indépendante de l'image source, utilisez la méthode `duplicate()`. La méthode `duplicate()` renvoie une copie d'une image source qui hérite toutes les valeurs de l'image source, mais qui n'est pas liée à cette dernière. De cette façon, toute modification apportée à l'image source ou à la nouvelle copie de l'image source n'a aucun effet sur l'autre image.

Si vous créez un objet image en faisant référence à un acteur, ce nouvel objet contient une référence à l'image de l'acteur. Toute modification de l'image est reflétée dans l'acteur et dans les images-objets créées à partir de cet acteur.

Lorsque vous créez un objet image, son arrière-plan prend la couleur blanche par défaut (color(255, 255, 255)) et la couche alpha est complètement opaque (color(0, 0, 0)).

La couleur de couche alpha avec une transparence de 100 % est le blanc (color(255, 255, 255)) , tandis que la couleur de couche alpha avec une opacité de 100 % est le noir (color(0, 0, 0)).

Vous pouvez voir un exemple d'utilisation de la fonction `image()` dans une animation en consultant l'animation Imaging du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

`intWidth` Requis. Nombre entier spécifiant la largeur de la nouvelle image.

`intHeight` Requis. Nombre entier spécifiant la hauteur de la nouvelle image.

`intBitDepth` Requis. Nombre entier spécifiant le codage binaire de la nouvelle image. Les valeurs possibles sont 1, 2, 4, 8, 16 ou 32.

Exemple

L'exemple suivant crée une image 8 bits de 200 pixels de large sur 200 pixels de haut.

```
-- Lingo syntax
objImage = image(200, 200, 8)

// JavaScript syntax
var objImage = image(200, 200, 8);
```

L'exemple suivant crée une image en faisant référence à l'image de la scène.

```
-- Lingo syntax
objImage = _movie.stage.image

// JavaScript syntax
var objImage = _movie.stage.image;
```

Voir aussi

duplicate() (image), fill(), image (image)

importFileInto()

Syntaxe

```
-- Lingo syntax
memberObjRef.importFileInto(fileOrUrlString, propertyList)

// JavaScript syntax
memberObjRef.importFileInto(fileOrUrlString, propertyList);
```

Description

Méthode d'acteur ; remplace le contenu d'un acteur spécifié par un fichier spécifique.

La méthode `importFileInto()` se révèle utile dans les cas suivants.

• A la fin du développement d'une animation, utilisez-la pour intégrer des médias externes liés pour pouvoir les modifier pendant le projet.

• Lors de la création d'un scénario en syntaxe Lingo ou JavaScript pendant la création de l'animation, utilisez-la pour affecter un contenu aux nouveaux acteurs.

• Lors du téléchargement de fichiers à partir d'Internet, elle permet de télécharger le fichier à partir d'une URL spécifique et de définir le nom de fichier du média lié.

> ***Remarque :*** *Pour importer un fichier à partir d'une URL, il est généralement plus judicieux d'utiliser* `preloadNetThing()` *pour télécharger le fichier sur un disque local avant d'importer le fichier depuis le disque local. L'utilisation de* `preloadNetThing()` *réduit également les risques de problèmes de téléchargement.*

• Utilisez cette méthode pour importer des documents RTF et HTML dans des acteurs texte en gardant le format et les liens intacts.

L'utilisation de la méthode `importFileInto()` dans les projections peut consommer rapidement la mémoire disponible ; il est donc recommandé de réutiliser les mêmes acteurs pour les données importées.

Dans Director et les projections, `importFileInto()` télécharge automatiquement le fichier. Dans Shockwave Player, appelez la méthode `preloadNetThing()` et attendez la fin du téléchargement avant d'utiliser `importFileInto()` avec le fichier.

Paramètres

`fileOrUrlString` Requis. Chaîne spécifiant le fichier destiné à remplacer le contenu de l'acteur.

Le paramètre *`propertyList`* est facultatif et prend en charge les propriétés suivantes :

#dither : 0 ; ne pas tramer (valeur par défaut)

#dither : différent de zéro ; tramer

#trimWhiteSpace : 0 ; ne pas supprimer l'espace blanc sur les bords extérieurs de l'image

#trimWhiteSpace : différent de zéro ; supprimer l'espace blanc (valeur par défaut)

#linked : 0 ; importer en tant qu'acteur interne (valeur par défaut)

#linked : différent de zéro ; importer en tant qu'acteur lié

#remapImageToStage : 0 ; utiliser le codage de l'image

#remapImageToStage : différent de zéro ; convertir l'image en fonction du codage de la scène (valeur par défaut)

Exemple

Le gestionnaire suivant affecte une URL contenant un fichier GIF à la variable `tempURL`, puis utilise la commande `importFileInto` pour importer le fichier de l'URL dans un nouvel acteur bitmap :

```
-- Lingo syntax
on exitFrame
    tempURL = "http://www.dukeOfUrl.com/crown.gif"
    _movie.newMember(#bitmap).importFileInto(tempURL)
end

// JavaScript syntax
```

```
function exitFrame() {
    var tempURL = "http://www.dukeOfUrl.com/crown.gif";
    _movie.newMember("bitmap").importFileInto(tempURL);
}
```

L'instruction suivante remplace le contenu de l'acteur son Mémoire par le fichier audio Vent :

```
-- Lingo syntax
member("Memory").importFileInto("Wind.wav")

// JavaScript syntax
member("Memory").importFileInto("Wind.wav");
```

Les instructions suivantes téléchargent un fichier externe depuis une URL dans le dossier de Director, puis importent ce fichier dans l'acteur son Norma Desmond parle :

```
-- Lingo syntax

downLoadNetThing("http://www.cbDeMille.com/Talkies.AIF", _player.applicationPath &
"Talkies.AIF")
member("Norma Desmond Speaks").importFileInto(_player.applicationPath & "Talkies.AIF")


// JavaScript syntax

downLoadNetThing("http://www.cbDeMille.com/Talkies.AIF",_player.applicationPath +
"Talkies.AIF");
member("Norma Desmond Speaks").importFileInto(_player.applicationPath +"Talkies.AIF");
```

Voir aussi

downloadNetThing, fileName (fenêtre), Acteur, preloadNetThing()

Initialize

Syntaxe

```
Initialize (MUIObject, initialPropertyList)
```

Description

Cette commande définit une boîte de dialogue générale à partir d'une instance de l'Xtra MUI. initialPropertyList est une liste de propriétés qui spécifie les emplacements à partir desquels Director obtient des définitions pour les attributs de la boîte de dialogue.

- La liste de propriétés associée à la propriété #windowPropList est la liste qu'utilise Director pour définir les attributs de la boîte de dialogue globale.
- La liste linéaire associée à la propriété #windowItemList est la liste qu'utilise Director pour définir les composants individuels. Chaque élément de la liste est une liste de propriétés définissant un composant.

Exemple

Cette instruction initialise une nouvelle boîte de dialogue générale créée à partir de MUIObject, qui est une instance de l'Xtra MUI. La liste aWindowPropList contient des définitions pour la boîte de dialogue globale. La liste aWindowItemList contient des définitions pour les composants individuels de la boîte de dialogue.

```
-- Lingo
Initialize(MUIObj, [#windowPropList:aWindowPropList, \
#windowItemList:aWindowItemList])
```

insertBackdrop

Syntaxe

```
sprite(whichSprite).camera{(index)}.insertBackdrop(index, texture, locWithinSprite, rotation)
member(whichCastmember).camera(whichCamera).insertBackdrop(index, texture,
locWithinSprite, rotation)
```

Description

Commande 3D de caméra ; ajoute un fond à la liste des fonds de la caméra à une position spécifiée dans la liste.

Paramètres

index Requis. Spécifie la position d'index à laquelle le fond est ajouté dans la liste des fonds de la caméra.

texture Requis. Spécifie la texture du fond ajouté.

locWithinSprite Requis. Emplacement 2D auquel le fond s'affiche dans l'image-objet 3D. Cet emplacement est mesuré à partir du coin supérieur gauche de l'image-objet.

rotation Facultatif. Spécifie la rotation du fond ajouté.

Exemple

La première ligne de l'exemple suivant crée une texture nommée Cèdre. La deuxième ligne insère cette texture à la première position de la liste des fonds de la caméra de l'image-objet 5. Le fond est placé au point (300, 120), mesuré à partir du coin supérieur gauche de l'image-objet. La rotation est de 45 degrés.

```
-- Lingo
t1 = member("scene").texture("Cedar")
sprite(5).camera.insertBackdrop(1, t1, point(300, 120), 45)

// Javascript
Var t1= member("scene").getProp("texture",1);
Sprite(5).getPropRef("camera",1).insertBackDrop(1,t1,point(300,120),45);
```

Voir aussi

removeBackdrop, bevelDepth, overlay

insertFrame()

Syntaxe

```
-- Lingo syntax
_movie.insertFrame()

// JavaScript syntax
_movie.insertFrame();
```

Description

Méthode d'animation ; duplique l'image en cours ainsi que son contenu.

L'image copiée est insérée après l'image actuelle et devient alors l'image actuelle.

Cette méthode n'est utilisable que lors d'une session d'enregistrement du scénario et remplit la même fonction que la méthode `duplicateFrame()`.

Paramètres

Aucune.

Exemple

Le gestionnaire suivant crée une image dans laquelle l'acteur transition Brouillard est affecté à la piste des transitions, suivi d'un jeu d'images vides. L'argument `numberOfFrames` définit ce nombre d'images.

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    _movie.frameTransition = member("Fog").number
    _movie.go(_movie.frame + 1)
    repeat with i = 0 to numberOfFrames
        _movie.insertFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    _movie.frameTransition = member("Fog").number;
    _movie.go(_movie.frame + 1);
    for (var i = 0; i <= numberOfFrames; i++) {
        _movie.insertFrame();
    }
    _movie.endRecording();
}
```

Voir aussi

`duplicateFrame()`, `Animation`

insertOverlay

Syntaxe

```
sprite(whichSprite).camera{(index)}.insertOverlay(index, texture, locWithinSprite,
rotation)
member(whichCastmember).camera(whichCamera).insertOverlay(index, texture, locWithinSprite,
rotation)
```

Description

Commande 3D de caméra ; ajoute un recouvrement à la liste des recouvrements de la caméra à une position spécifiée dans la liste.

Paramètres

`index` Requis. Spécifie la position d'index à laquelle le recouvrement est ajouté dans la liste des recouvrements de la caméra.

`texture` Requis. Spécifie la texture du recouvrement ajouté.

`locWithinSprite` Requis. Emplacement 2D auquel le recouvrement s'affiche dans l'image-objet 3D. Cet emplacement est mesuré à partir du coin supérieur gauche de l'image-objet.

`rotation` Facultatif. Spécifie la rotation du recouvrement ajouté.

Exemple

La première ligne de l'exemple suivant crée une texture nommée Cèdre. La deuxième ligne insère cette texture à la première position de la liste des recouvrements de la caméra de l'image-objet 5. Le recouvrement est placé au point (300, 120), mesuré à partir du coin supérieur gauche de l'image-objet. La rotation est de 45 degrés.

```
-- Lingo
t1 = member("scene").texture("Cedar")
sprite(5).camera.insertOverlay(1, t1, point(300, 120), 45)

// Javascript
Var t1=member("scene").getProp("texture",1);
Sprite(5).getPropRef("camera",1).insertOverlay(1,t1,point(300,120),45);
```

Voir aussi

removeOverlay, overlay, bevelDepth

inside()

Syntaxe

```
point.inside(rectangle)
inside(point, rectangle)
```

Description

Fonction ; indique si un point spécifié se trouve à l'intérieur d'un rectangle spécifié (TRUE) ou à l'extérieur du rectangle (FALSE).

Paramètres

rectangle Requis. Spécifie le rectangle contenant le point à tester.

Exemple

L'instruction suivante indique si le point Centre se trouve à l'intérieur du rectangle Zone et affiche le résultat dans la fenêtre Messages :

```
-- Lingo
put Center.inside(Zone)

// Javascript
trace(Center. inside(Zone));
```

Voir aussi

map(), mouseH, mouseV, point()

installMenu

Syntaxe

```
installMenu whichCastMember
```

Description

Commande ; installe le menu défini dans l'acteur champ spécifié par *whichCastMember*. Ces menus personnalisés n'apparaissent que pendant la lecture de l'animation. Pour les supprimer, utilisez la commande `installMenu` sans argument ou avec la valeur 0 comme argument. Cette commande ne fonctionne pas avec les menus hiérarchiques.

Pour plus de détails sur la définition d'éléments de menus dans un acteur champ, reportez-vous au mot-clé `menu`.

Il est conseillé d'éviter la modification fréquente des menus, celle-ci affectant les ressources du système.

Sous Windows, si le menu ne tient pas à l'écran, seule une partie de ce menu est affichée. Sur le Mac, il est possible de faire défiler les menus qui ne tiennent pas à l'écran.

Remarque : *Les menus ne sont pas disponibles dans Shockwave Player.*

Paramètres

fieldMemberObjRef Facultatif. Spécifie l'acteur champ dans lequel un menu est installé.

Exemple

L'instruction suivante installe le menu défini dans l'acteur champ 37 :

```
installMenu 37
```

L'instruction suivante installe le menu défini dans l'acteur champ Barre de menus :

```
installMenu member "Menubar"
```

L'instruction suivante désactive les menus installés par la commande installMenu :

```
installMenu 0
```

Voir aussi

menu

integer()

Syntaxe

```
(numericExpression).integer
integer(numericExpression)
```

Description

Fonction (Lingo uniquement) ; arrondit la valeur d'une expression au nombre entier le plus proche.

Vous pouvez appliquer un nombre entier en tant que chaîne à l'aide de la fonction `string()`.

En syntaxe JavaScript, utilisez la fonction `parseInt()`.

Paramètres

numericExpression Requis. Nombre à arrondir sous la forme d'un nombre entier.

Exemple

L'instruction suivante arrondit le nombre 3,75 au nombre entier le plus proche :

```
put integer(3.75)
-- 4
```

L'instruction suivante arrondit la valeur entre parenthèses. Cette opération permet d'obtenir une valeur utilisable pour la propriété d'image-objet `locH` qui requiert un nombre entier :

```
sprite(1).locH = integer(0.333 * stageWidth)
```

Voir aussi

`float()`, `string()`

integerP()

Syntaxe

```
expression.integerP
(numericExpression).integerP
integerP(expression)
```

Description

Fonction (Lingo uniquement) ; indique si une expression spécifiée peut être évaluée en tant que nombre entier (`1 ou TRUE`) ou non (`0 ou FALSE`). Le *P* de la fonction `integerP` signifie *prédicat*.

Paramètres

`expression` Requis. Expression à tester.

Exemple

L'instruction suivante vérifie si le chiffre 3 peut être évalué en tant que nombre entier et affiche 1 (`TRUE`) dans la fenêtre Messages :

```
put(3).integerP
-- 1
```

L'instruction suivante vérifie si le nombre 3 peut être évalué en tant qu'entier. Puisque le chiffre 3 est entouré de guillemets droits, il n'est pas considéré comme un entier et la valeur 0 (`FALSE`) s'affiche dans la fenêtre Messages :

```
put("3").integerP
-- 0
```

L'instruction suivante vérifie si la valeur numérique de la chaîne de l'acteur champ Saisie est un entier et, dans le cas contraire, affiche un message d'erreur :

```
if field("Entry").value.integerP = FALSE then alert "Please enter an integer."
```

Voir aussi

`floatP()`, `integer()`, `ilk()`, `objectP()`, `stringP()`, `symbolP()`

Interface()

Syntaxe

```
xtra("XtraName").Interface()
Interface(xtra "XtraName")
```

Description

Fonction ; renvoie une chaîne délimitée par des retours chariot décrivant l'Xtra et fournissant une liste de ses méthodes. Cette fonction remplace la fonction `mMessageList` devenue obsolète.

Paramètres

Aucune.

Exemple

L'instruction suivante affiche les données produites par cette fonction utilisée avec l'Xtra QuickTime Asset dans la fenêtre Messages :

```
-- Lingo
put Xtra("QuickTimeSupport").Interface()

// Javascript
trace(xtra("QuickTimeSupport").Interface());
```

interpolate()

Syntaxe

```
transform1.interpolate(transform2,percentage)
```

Description

Méthode 3D `transform` ; renvoie une copie de *`transform1`* créée par l'interpolation entre la position et la rotation de *`transform1`* et la position et la rotation de *`transform2`* en fonction du pourcentage spécifié. La *`transform1`* d'origine n'est pas affectée. Pour interpoler *`transform1`*, utilisez `interpolateTo()`.

Pour effectuer l'interpolation manuellement, multipliez la différence des deux nombres par le pourcentage. Par exemple, l'interpolation de 4 à 8 par 50 % donne 6.

Exemple

Dans l'exemple suivant, tBoîte est la transformation du modèle nommé Boîte et tSphère est la transformation du modèle nommé Sphère. La troisième ligne de l'exemple interpole une copie de la transformation de Boîte à mi-chemin jusqu'à la transformation de Sphère.

```
-- Lingo
tBox = member("3d world").model("Box").transform
tSphere = member("3d world").model("Sphere").transform
tNew = tBox.interpolate(tSphere, 50)

// Javascript
var tBox = member("3d world").getPropRef("model",a).transform;
// where a is the number index for the model "Box"
var tSphere = member("3d world").getPropRef("model", i).transform;
// where i is the number index for the model "Sphere"
var tNew = tBox.interpolate(tSphere, 50);
```

Voir aussi

`interpolateTo()`

interpolateTo()

Syntaxe

```
transform1.interpolateTo(transform2, percentage)
```

Description

Méthode 3D `transform` ; modifie *`transform1`* en effectuant une interpolation entre la position et la rotation de *`transform1`* et la position et la rotation d'une nouvelle transformation en fonction d'un pourcentage spécifié. La *`transform1`* d'origine est modifiée. Pour interpoler une copie de *`transform1`*, utilisez la fonction `interpolate()`.

Pour effectuer l'interpolation manuellement, multipliez la différence des deux nombres par le pourcentage. Par exemple, l'interpolation de 4 à 8 par 50 % donne 6.

Paramètres

`transform2` Requis. Spécifie la transformation avec laquelle une transformation donnée est interpolée.

`percentage` Requis. Spécifie le pourcentage de rotation de *`transform2`*.

Exemple

Dans l'exemple suivant, tBoîte est la transformation du modèle nommé Boîte et tSphère est la transformation du modèle nommé Sphère. La troisième ligne de l'exemple interpole la transformation de Boîte à mi-chemin jusqu'à la transformation de Sphère.

```
-- Lingo
tBox = member("3d world").model("Box").transform
tSphere = member("3d world").model("Sphere").transform
tBox.interpolateTo(tSphere, 50)

// Javascript
var tBox = member("3d world").getPropRef("model", i).transform;
// where i the number index for the model "Box"
var tSphere = member("3d world").getPropRef("model",j).transform;
// where j is the number index for the model "Sphere"
tBox.interpolateTo(tSphere, 50);
```

Voir aussi

`interpolate()`

intersect()

Syntaxe

```
rectangle1. Intersect(rectangle2)
intersect(rectangle1, rectangle2)
```

Description

Fonction ; détermine le rectangle formé par l'intersection de deux rectangles.

Paramètres

`transform2` Requis. Spécifie le second rectangle du test d'intersection.

Exemple

L'instruction affecte la variable `newRectangle` au rectangle formé par l'intersection du rectangle Outils avec le rectangle Rampe :

```
-- Lingo
newRectangle = toolKit.intersect(Ramp)
```

```
// Javascript
newRectangle = toolKit.intersect(Ramp);
```

Voir aussi

map(), rect(), union()

inverse()

Syntaxe

```
member(whichCastmember).model(whichModel).transform.inverse()
member(whichCastmember).group(whichGroup).transform.inverse()
member(whichCastmember).camera(whichCamera).transform.inverse()
sprite(whichSprite).camera{(index)}.transform.inverse()
member(whichCastmember).light(whichLight).transform.inverse()
transformReference.inverse()
```

Description

Méthode 3D `transform` ; renvoie une copie de la transformation dont les propriétés de position et de rotation ont été inversées.

Cette méthode ne change pas la transformation d'origine. Pour inverser la transformation d'origine, utilisez la fonction `invert()`.

Paramètres

Aucune.

Exemple

L'instruction suivante inverse une copie de la transformation du modèle Fauteuil.

```
-- Lingo
boxInv = member("3d world").model("Chair").transform.inverse()

// Javascript
var boxInv = member("3d world").getPropRef("model", a).transform.inverse();
// where a the number index for the model "Chair"
```

Voir aussi

invert()

invert()

Syntaxe

```
member(whichCastmember).model(whichModel).transform.invert()
member(whichCastmember).group(whichGroup).transform.invert()
member(whichCastmember).camera(whichCamera).transform.invert()
sprite(whichSprite).camera{(index)}.transform.invert()
member(whichCastmember).light(whichLight).transform.invert()
transformReference.invert()
```

Description

Méthode 3D `transform` ; inverse les propriétés de position et de rotation de la transformation.

Cette méthode change la transformation d'origine. Pour inverser une copie de la transformation d'origine, utilisez la fonction `inverse()`.

Paramètres

Aucune.

Exemple

```
-- Lingo
member("3d world").model("Box").transform.invert()

// Javascript
member("3d world").getPropRef("model", a).transform.invert();
// where a the number index for the model "Box"
```

Voir aussi

`inverse()`

isBusy()

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.isBusy()

// JavaScript syntax
soundChannelObjRef.isBusy();
```

Description

Méthode de piste audio ; détermine si un son est en cours de lecture (`TRUE`) ou non (`FALSE`) dans une piste audio.

Assurez-vous que la tête de lecture a bougé avant d'utiliser la méthode `isBusy()` pour vérifier la piste audio. Si cette fonction continue de renvoyer la valeur `FALSE` après la lecture présumée d'un son, ajoutez la méthode `updateStage()` pour démarrer la lecture du son avant que la tête de lecture ne recommence à bouger.

Cette méthode fonctionne avec les pistes audio occupées par des acteurs son réels. Les composants audio QuickTime, Flash et Shockwave Player gérant le son différemment, cette méthode ne fonctionne avec ces types de médias.

Il est conseillé d'utiliser la propriété `status` d'une piste audio plutôt que la méthode `isBusy()`. La propriété `status` se révèle plus appropriée dans la plupart des cas.

Paramètres

Aucune.

Exemple

L'instruction suivante vérifie si un son est lu dans la piste audio 1 et, le cas échéant, effectue une lecture en boucle dans l'image. Cela permet au son d'être lu en entier avant que la tête de lecture ne passe à une autre image.

```
-- Lingo syntax
if (sound(1).isBusy()) then
    _movie.go(_movie.frame)
end if

// JavaScript syntax
if (sound(1).isBusy()) {
```

```
    _movie.go(_movie.frame);
}
```

Voir aussi

status, Piste audio

isInWorld()

Syntaxe

```
member(whichCastmember).model(whichModel).isInWorld()
member(whichCastmember).camera(whichCamera).isInWorld()
member(whichCastmember).light(whichLight).isInWorld()
member(whichCastmember).group(whichGroup).isInWorld()
```

Description

Commande 3D ; renvoie la valeur TRUE si la hiérarchie parente du modèle, de la caméra, de la lumière ou du groupe s'achève dans l'univers. Si la commande isInWorld présente la valeur TRUE, le modèle, la caméra, la lumière ou le groupe fonctionne dans l'univers 3D de l'acteur.

Les modèles, caméras, lumières et groupes peuvent être enregistrés dans un acteur 3D mais ne peuvent pas être utilisés dans son univers 3D. Utilisez les commandes addToWorld et removeFromWorld pour ajouter et supprimer des modèles, des caméras, des lumières et des groupes de l'univers 3D de l'acteur.

Paramètres

Aucune.

Exemple

L'instruction suivante indique que le modèle Théière existe dans l'univers 3D de l'acteur scèneDeTable.

```
--Lingo
put member("TableScene").model("Teapot").isInWorld()

// Javascript
put member("TableScene").getPropRef("model", a).isInWorld() ;
// where a is the member index for the Teapot model.
-- 1
```

Voir aussi

addToWorld, removeFromWorld, child (3D)

isPastCuePoint()

Syntaxe

```
-- Lingo syntax
spriteObjRef.isPastCuePoint(cuePointID)

// JavaScript syntax
spriteObjRef.isPastCuePoint(cuePointID);
```

Description

Fonction ; détermine si une piste d'image-objet ou une piste audio est passée par un point de repère spécifié dans ses médias. Cette fonction peut être utilisée avec les fichiers audio (WAV, AIFF, SND, SWA, AU), QuickTime ou Xtra supportant les points de repère.

Remplacez *spriteNum* ou *channelNum* par une piste d'image-objet ou une piste audio. Les sons Shockwave Audio (SWA) peuvent apparaître sous forme d'images-objets dans les pistes d'images-objets, mais sont lus dans les pistes audio. Il est conseillé de faire référence aux images-objets audio SWA par le numéro de leur piste d'image-objet plutôt que par celui de leur piste audio.

Remplacez *cuePointID* par une référence à un point de repère :

- Si *cuePointID* est un entier, `isPastCuePoint` renvoie la valeur 1 si le point de repère a été dépassé et la valeur 0 dans le cas contraire.
- Si *cuePointID* est un nom, `isPastCuePoint` renvoie le nombre de points de repère dépassés portant ce nom.

Si la valeur spécifiée pour *cuePointID* n'existe pas dans l'image-objet ou le son, la fonction renvoie la valeur 0.

Le nombre renvoyé par `isPastCuePoint` repose sur la position absolue de l'image-objet dans son média. Par exemple, lorsqu'un son dépasse le point de repère Principal, puis se met en boucle et dépasse de nouveau ce point de repère, `isPastCuePoint` renvoie 1 au lieu de 2.

Lorsque le résultat de `isPastCuePoint` est traité comme un opérateur booléen, la fonction renvoie la valeur `TRUE` si un point identifié par *cuePointID* a été dépassé et la valeur `FALSE` dans le cas contraire.

Paramètres

cuePointID Requis. Chaîne ou nombre entier spécifiant le nom ou le numéro du point de repère spécifié.

Exemple

L'instruction suivante lit un son jusqu'au troisième passage sur le point de repère Fin du chœur :

```
-- Lingo syntax
if (sound(1).isPastCuePoint("Chorus End")=3) then
 sound(1).stop()
end if

// JavaScript syntax
var ce = sound(1).isPastCuePoint("Chorus End");
if (ce = 3) {
 sound(1).stop();
}
```

L'exemple suivant affiche des informations dans l'acteur Champ 2 concernant la lecture de la musique sur la piste audio 1. Si le point de repère apogée n'a pas encore été atteint, le texte de Champ 2 est « Début du morceau ». Sinon, le texte est Fin du morceau.

```
--- Lingo syntax
if not sound(1).isPastCuePoint("climax") then
    member("field 2").text = "This is the beginning of the piece."
else
    member("field 2").text = "This is the end of the piece."
end if

// JavaScript syntax
var cmx = sound(1).isPastCuePoint("climax");
if (cmx != 1) {
    member("field 2").text = "This is the beginning of the piece.";
} else {
```

```
    member("field 2").text = "This is the end of the piece.";
}
```

ItemUpdate()

Syntaxe

```
ItemUpdate(MUIObject, itemNumber, itemInputPropList)
```

Description

Cette commande met à jour un composant dans une boîte de dialogue générale. Elle peut s'avérer utile dans le cas d'une mise à jour d'une boîte de dialogue qui répond aux actions de l'utilisateur, lorsque celle-ci est affichée.

- itemNumber représente le numéro de l'élément mis à jour.
- itemInputPropList représente la liste des nouvelles propriétés de l'élément.

La commande ItemUpdate peut être utilisée pour de nombreuses applications, comme l'activation ou la désactivation de boutons, le changement des plages de valeurs d'une fenêtre contextuelle, la mise à jour de la position d'une glissière ou encore la mise à jour d'éléments de texte modifiables si l'utilisateur saisit une valeur incorrecte.

Vous pouvez choisir de mettre à jour des éléments individuels d'une boîte de dialogue selon l'action effectuée par l'utilisateur, l'interaction avec celui-ci ou l'affichage de données sous-jacentes. Outre la propriété #value, vous pouvez mettre à jour toutes les autres propriétés d'un élément, excepté son type. Définissez la hauteur, la largeur, les propriétés locH et locV sur -1 pour conserver leurs valeurs actuelles.

Exemple

Ces instructions mettent à jour le composant de boîte de dialogue dont le numéro est itemNum.

- La première instruction obtient les définitions du composant dans la liste globale des définitions d'élément.
- Les deuxième et troisième instructions modifient le type du composant et les propriétés d'attribut.
- La dernière instruction utilise la commande ItemUpdate pour mettre à jour les paramètres du composant.

```
--Lingo
set baseItemList = getAt ( theItemList, itemNum )
set the type of baseItemList = #IntegerSliderH
set the attributes of baseItemList = [#valueRange :[#min:1, #max:8, #increment:1,
#jump:1, #acceleration:1]
ItemUpdate(MUIObj, itemNum, baseItemList)
```

```
on smileyUpdate
   -- declare globals
   global smileyIndex, gMuiSmileDialObj, itemNumSmile, itemNumSlide, smileItemList

-- validate dialog object
if ( objectP ( gMuiSmileDialObj ) ) then
   -- get a list to put in new/updated values
   set baseItemList = duplicate ( getAt ( smileItemList,itemNumSmile ) )
   -- metrics can be set to -1, this "keeps them the same"
   -- instead of updating.
   -- could also be set to a new value if you
   -- wanted to resize the item or relocate it.
   set the width of baseItemList = -1 -- keep previous
   set the height of baseItemList = -1 -- keep previous
   set the locH of baseItemList = -1 -- keep previous
   set the locV of baseItemList = -1 -- keep previous
```

```
   -- in this particular case, the value is
   -- the only thing that's changing
   set the value of baseItemList = string(smileyIndex)
   -- member name

   -- tell the dialog to update the item number
   -- with the new item list
   ItemUpdate(gMuiSmileDialObj, itemNumSmile, baseItemList)
   end if
end
```

keyPressed()

Syntaxe

```
-- Lingo syntax
_key.keyPressed({keyCodeOrCharacter})

// JavaScript syntax
_key.keyPressed({keyCodeOrCharacter});
```

Description

Méthode de touche ; renvoie la chaîne de caractères affectée à la dernière touche sélectionnée par l'utilisateur ou indique en option si une touche spécifiée a été pressée.

Si le paramètre *keyCodeOrCharacter* est omis, cette méthode renvoie la chaîne de caractères affectée à la dernière touche sélectionnée. Si l'utilisateur n'a appuyé sur aucune touche, cette méthode renvoie une chaîne vide.

Si le paramètre *keyCodeOrCharacter* est utilisé pour spécifier une touche, cette méthode renvoie la valeur `TRUE` si l'utilisateur appuie sur la touche indiquée ou la valeur `FALSE` dans le cas contraire.

Cette méthode est mise à jour lorsque l'utilisateur appuie sur des touches dans une boucle `repeat` (Lingo) ou `for` (syntaxe JavaScript). Ceci constitue un avantage par rapport à la fonction `key` qui n'est pas mise à jour dans une boucle `repeat` ou `for`.

Pour tester la correspondance des caractères des différentes touches sur des claviers distincts, utilisez l'animation Clavier et Lingo.

Paramètres

keyCodeOrCharacter Facultatif. Code de touche ou chaîne de caractères ASCII à tester.

Exemple

L'instruction suivante vérifie si l'utilisateur a appuyé sur la touche Entrée sous Windows ou sur la touche Retour du Mac et, le cas échéant, exécute le gestionnaire `updateData` :

```
-- Lingo syntax
if (_key.keyPressed(RETURN)) then
    updateData
end if

// JavaScript syntax
if (_key.keyPressed(36)) {
    updateData();
}
```

L'instruction suivante utilise le code de la touche *a* pour vérifier si elle est enfoncée et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
if (_key.keyPressed(0)) then
    put("The key is down")
end if

// JavaScript syntax
if (_key.keyPressed(0)) {
    put("The key is down");
}
```

L'instruction suivante utilise les chaînes ASCII pour vérifier si les touches *a* et *b* sont enfoncées et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
if (_key.keyPressed("a") and _key.keyPressed("b")) then
    put("Both keys are down")
end if

// JavaScript syntax
if (_key.keyPressed("a") && _key.keyPressed("b")) {
    put("Both keys are down");
}
```

Voir aussi

Touche, key, keyCode

label()

Syntaxe

```
-- Lingo syntax
_movie.label(stringMarkerName)

// JavaScript syntax
_movie.label(stringMarkerName);
```

Description

Méthode d'animation ; indique l'image associée à un libellé de repère.

Le paramètre *stringMarkerName* doit correspondre à un libellé figurant dans l'animation en cours. Dans le cas contraire, cette méthode renvoie la valeur 0.

Paramètres

stringMarkerName Requis. Chaîne spécifiant le nom du libellé de repère associé à une image.

Exemple

L'instruction suivante place la tête de lecture sur la dixième image après l'image intitulée Départ :

```
-- Lingo syntax
_movie.go(_movie.label("Start") + 10)

// JavaScript syntax
_movie.go(_movie.label("Start") + 10);
```

Voir aussi

frameLabel, go(), labelList, Animation

last()

Syntaxe

```
the last chunk of ( chunkExpression )
the last chunk in (chunkExpression)
```

Description

Fonction ; identifie la dernière sous-chaîne d'une expression de sous-chaîne.

Les expressions de sous-chaînes peuvent correspondre à tout caractère, mot, élément ou ligne extrait d'un conteneur de caractères. Les conteneurs peuvent être des acteurs champ, des variables contenant des chaînes, et les caractères, mots, lignes et plages spécifiés dans les conteneurs.

Paramètres

chunkExpression Requis. Spécifie l'expression de sous-chaîne contenant la dernière sous-chaîne.

Exemple

L'instruction suivante identifie le dernier mot de la chaîne « Adobe, la société multimédia » et affiche le résultat dans la fenêtre Messages :

```
-- Lingo
put the last word of "Adobe, the multimedia company"
```

Le résultat est le mot *company*.

L'instruction suivante identifie le dernier caractère de la chaîne « Adobe, la société multimédia » et affiche le résultat dans la fenêtre Messages :

```
put last char("Adobe, the multimedia company")
```

Le résultat est la lettre *y*.

Voir aussi

char...of, word...of

lastClick()

Syntaxe

```
the lastClick
```

Description

Fonction ; renvoie le nombre de battements (1 battement = 1/60ème de seconde) écoulés depuis le dernier clic de la souris.

Cette fonction peut être testée, mais pas définie.

Paramètres

Aucune.

Exemple

L'instruction suivante vérifie si 10 secondes se sont écoulées depuis le dernier clic de la souris et, le cas échéant, fait passer la tête de lecture sur le repère Pas de clic :

```
if the lastClick > 10 * 60 then go to "No Click"
```

Voir aussi

lastEvent(), lastKey, lastRoll, milliseconds

lastEvent()

Syntaxe

```
the lastEvent
```

Description

Fonction ; renvoie le nombre de battements (1 battement = 1/60ème de seconde) écoulés depuis la dernière fois où l'utilisateur a appuyé sur le bouton de la souris, a survolé un élément avec la souris ou a appuyé sur une touche.

Paramètres

Aucune.

Exemple

L'instruction suivante vérifie si 10 secondes se sont écoulées depuis la dernière fois où l'utilisateur a appuyé sur le bouton de la souris, a survolé un élément avec la souris ou a appuyé sur une touche. Si c'est le cas, la tête de lecture est placée sur le repère Aide :

```
if the lastEvent > 10 * 60 then go to "Help"
```

Voir aussi

lastClick(), lastKey, lastRoll, milliseconds

length()

Syntaxe

```
string.length
length(string)
```

Description

Fonction ; renvoie le nombre de caractères de la chaîne spécifiée par *string*, y compris les espaces et les caractères de contrôle tels que Tabulation et Retour.

Paramètres

Aucune.

Exemple

L'instruction suivante affiche le nombre de caractères de la chaîne "Macro"&"media" :

```
put ("Macro" & "media").length
-- 10
```

L'instruction suivante vérifie si le contenu de l'acteur champ Nom de fichier contient plus de 31 caractères et, le cas échéant, affiche un message d'erreur :

```
-- Lingo syntax
if member("Filename").text.length > 31 then
    alert "That filename is too long."
```

```
end if

// JavaScript syntax
if (member("Filename").text.length > 31) {
    _player.alert("That filename is too long.");
}
```

Voir aussi

chars(), offset() (fonction de chaîne)

light()

Syntaxe

```
member(whichCastmember).light(whichLight)
member(whichCastmember).light[index]
member(whichCastmember).light(whichLight).whichLightProperty
member(whichCastmember).light[index].whichLightProperty
```

Description

Elément 3D ; objet à une position de vecteur à partir de laquelle la lumière émane.

Pour obtenir la liste complète des propriétés et commandes de lumière, reportez-vous aux rubriques Utilisation de Director dans l'Aide de Director.

Exemple

L'exemple suivant indique les deux façons de faire référence à une lumière. La première ligne utilise une chaîne entre parenthèses et la seconde utilise un nombre entre crochets. La chaîne correspond au nom de la lumière et le nombre à la position de la lumière dans la liste des lumières de l'acteur.

```
-- Lingo
thisLight = member("3D World").light("spot01")
thisLight = member("3D World").light[2]

// Javascript
thisLight = member("3D World").light[1];
```

Voir aussi

newLight, deleteLight

lineHeight()

Syntaxe

```
-- Lingo syntax
memberObjRef.lineHeight(lineNumber)

// JavaScript syntax
memberObjRef.lineHeight(lineNumber);
```

Description

Fonction ; renvoie la hauteur, en pixels, d'une ligne spécifique d'un acteur champ spécifié.

Paramètres

lineNumber Requis. Nombre entier spécifiant la ligne à mesurer.

Exemple

L'instruction suivante détermine la hauteur, en pixels, de la première ligne de l'acteur champ Nouvelles du jour et affecte le résultat à la variable `headline` :

```
--Lingo syntax
headline = member("Today's News").lineHeight(1)

// JavaScript syntax
var headline = member("Today's News").lineHeight(1);
```

linePosToLocV()

Syntaxe

```
-- Lingo syntax
memberObjRef.linePosToLocV(lineNumber)

// JavaScript syntax
memberObjRef.linePosToLocV(lineNumber);
```

Description

Fonction ; renvoie la distance, en pixels, d'une ligne spécifique à partir du bord supérieur de l'acteur champ spécifié.

Paramètres

lineNumber Requis. Nombre entier spécifiant la ligne à mesurer.

Exemple

L'instruction suivante mesure la distance, en pixels, entre la deuxième ligne de l'acteur champ Nouvelles du jour et le bord supérieur de l'acteur, puis affecte le résultat à la variable `startOfString` :

```
--Lingo syntax
startOfString = member("Today's News").linePosToLocV(2)

// JavaScript syntax
var startOfString = member("Today's News").linePosToLocV(2);
```

linkAs()

Syntaxe

```
castMember.linkAs()
```

Description

Fonction d'acteur script ; ouvre une boîte de dialogue d'enregistrement, permettant d'enregistrer le contenu du script dans un fichier externe. L'acteur script est alors lié à ce fichier.

Les scripts liés sont importés dans l'animation lorsque vous les enregistrez en tant que projection ou qu'animation comportant du contenu Shockwave. Ceci diffère des autres médias liés, qui restent externes à l'animation, à moins que vous ne les importiez explicitement.

Paramètres

Aucune.

Exemple

L'instruction suivante, saisie dans la fenêtre Messages, ouvre une boîte de dialogue d'enregistrement, permettant d'enregistrer le script Mouvement aléatoire comme fichier externe :

```
-- Lingo
member("Random Motion").linkAs()
importFileInto, linked

// Javascript
member("Random Motion").linkAs() ;
```

list()

Syntaxe

```
-- Lingo syntax
list()
[]
list(stringValue1, stringValue2, ...)
[stringValue1, stringValue2, ...]

// JavaScript syntax
list();
list(stringValue1, stringValue2, ...);
```

Description

Fonction de niveau supérieur ; crée une liste linéaire.

Lors de la création d'une liste à l'aide de la syntaxe `list()`, avec ou sans paramètres, l'index des valeurs de la liste commence par 1.

Lors de la création d'une liste à l'aide de la syntaxe `[]`, avec ou sans paramètres, l'index des valeurs de la liste commence par 0.

La longueur maximale d'une ligne de script exécutable est de 256 caractères. La fonction `list()` ne permet pas de créer des listes volumineuses. Pour créer une liste incluant un important volume de données, indiquez ces données entre crochets ([]), insérez les données dans un champ, puis affectez le champ à une variable. Le contenu de cette variable correspondra à une liste de ces données.

Paramètres

`strigValue1, stringValue2 ...` Facultatif. Liste de chaînes spécifiant les valeurs initiales de la liste.

Exemple

L'instruction suivante associe la variable `Designers` à une liste linéaire contenant les noms Jean, Pierre et Marc :

```
-- Lingo syntax
designers = list("Gee", "Kayne", "Ohashi") -- using list()
designers = ["Gee", "Kayne", "Ohashi"] -- using brackets

// JavaScript syntax
var designers = list("Gee", "Kayne", "Ohashi");
```

Voir aussi
propList()

listP()

Syntaxe
```
listP(item)
```

Description
Fonction ; indique si un élément spécifié est une liste, un rectangle ou un point (1 ou TRUE) ou non (0 ou FALSE).

Paramètres
item Requis. Spécifie l'élément à tester.

Exemple
L'instruction suivante vérifie si la liste contenue dans la variable designers est une liste, un rectangle ou un point et affiche le résultat dans la fenêtre Messages :

```
put listP(designers)
```

Le résultat est 1, équivalent numérique de TRUE.

Voir aussi
ilk(), objectP()

loadFile()

Syntaxe
```
member(whichCastmember).loadFile(fileName {, overwrite, generateUniqueNames})
```

Description
Commande d'acteur 3D ; importe les éléments d'un fichier *.w3d dans un acteur.

La propriété state de l'acteur doit présenter la valeur -1 (erreur) ou 4 (chargé) avant l'utilisation de la commande loadFile.

Paramètres
fileName Requis. Spécifie le fichier *.w3d contenant les éléments à importer.

overwrite Facultatif. Indique si les éléments du fichier *.w3d remplacent ceux de l'acteur (TRUE) ou s'ils sont ajoutés à ceux de l'acteur (FALSE). La valeur par défaut de *overwrite* est TRUE.

generateUniqueNames Facultatif. Si ce paramètre présente la valeur TRUE, tout élément du fichier *.w3d portant le même nom qu'un élément correspondant de l'acteur est renommé. Si ce paramètre présente la valeur FALSE, les éléments de l'acteur sont remplacés par les éléments correspondants du même nom dans le fichier *.w3d. La valeur par défaut de *generateUniqueNames* est TRUE.

Exemple

L'instruction suivante importe le contenu du fichier Camion.w3d dans l'acteur Route. Le contenu de Camion.w3d est ajouté au contenu de Route. Si certains objets importés ont le même nom que des objets déjà présents dans Route, Director leur donne un nouveau nom.

```
-- Lingo
member("Roadway").loadFile("Truck.W3d", FALSE, TRUE)

// Javascript
member("Roadway").loadFile("Truck.W3d",false,true);
```

L'instruction suivante importe le contenu du fichier Chevy.w3d dans l'acteur Route. Chevy.w3d se trouve dans un dossier Modèles, un niveau au-dessous de l'animation. Le contenu de Route est remplacé par celui de Chevy.w3d. Le troisième paramètre n'a aucune importance étant donné que le deuxième paramètre présente la valeur `TRUE`.

```
-- Lingo
member("Roadway").loadFile(the moviePath & "Models\Chevy.W3d", TRUE, TRUE)

// Javascript
member("Roadway").loadFile(_movie.Path + "Models\Chevy.W3d",true,true);
```

Voir aussi

`state (3D)`

locToCharPos()

Syntaxe

```
-- Lingo syntax
memberObjRef.locToCharPos(location)

// JavaScript syntax
memberObjRef.locToCharPos(location);
```

Description

Fonction ; renvoie le numéro du caractère d'un acteur champ spécifié occupant la position la plus proche d'un point dans le champ.

La valeur 1 correspond au premier caractère de la chaîne, la valeur 2 au second caractère de cette chaîne, et ainsi de suite.

Paramètres

`location` Requis. Point dans l'acteur champ. La valeur du paramètre *`location`* est un point calculé par rapport au coin supérieur gauche de l'acteur champ.

Exemple

L'instruction suivante détermine le caractère le plus proche du point situé à 100 pixels sur la droite et à 100 pixels en dessous du coin supérieur gauche de l'acteur champ Nouvelles du jour. Elle affecte ensuite le résultat à la variable `PageDesign`.

```
--Lingo syntax
pageDesign = member("Today's News").locToCharPos(point(100, 100))

// JavaScript syntax
var pageDesign = member("Today's News").locToCharPos(point(100, 100));
```

locVToLinePos()

Syntaxe

```
-- Lingo syntax
memberObjRef.locVToLinePos(locV)

// JavaScript syntax
memberObjRef.locVToLinePos(locV);
```

Description

Fonction ; renvoie le numéro de la ligne de caractères apparaissant à une position verticale spécifiée.

Paramètres

locV Requis. Spécifie la position verticale de la ligne de caractères. Cette valeur représente le nombre de pixels à partir du haut de l'acteur champ et non depuis la partie de l'acteur champ affichée sur la scène.

Exemple

L'instruction suivante détermine la ligne de caractères apparaissant à 150 pixels du haut de l'acteur champ Nouvelles du jour et affecte le résultat à la variable `pageBreak` :

```
--Lingo syntax
pageBreak = member("Today's News").locVToLinePos(150)

// JavaScript syntax
var pageBreak = member("Today's News").locVToLinePos(150);
```

log()

Syntaxe

```
log(number)
```

Description

Fonction mathématique (Lingo uniquement) ; calcule le logarithme naturel d'un nombre spécifié.

En syntaxe JavaScript, utilisez la fonction `log()` d'un objet mathématique.

Paramètres

number Requis. Nombre dont le logarithme naturel est calculé. Ce nombre doit être une valeur décimale supérieure à 0.

Exemple

L'instruction suivante affecte le logarithme naturel de 10,5 à la variable `Answer`.

```
-- Lingo
Answer = log(10.5)

// Javascript
Answer = Math.log(10.5);
```

Exemple

L'instruction suivante calcule le logarithme naturel de la racine carrée de la valeur `Nombre` et en affecte le résultat à la variable `Réponse` :

```
-- Lingo
Answer = log(Number.sqrt)

// Javascript
Answer = Math.log(Math.sqrt(Number));
```

makeList()

Syntaxe

```
--Lingo syntax
parserObject.makeList()

// JavaScript syntax
parserObject.makeList();
```

Description

Fonction ; renvoie une liste de propriétés basée sur le document XML analysé à l'aide de `parseString()` ou de `parseURL()`.

Paramètres

Aucune.

Exemple

Le gestionnaire suivant analyse un document XML et renvoie la liste résultante :

```
-- Lingo syntax
on ConvertToList xmlString
    parserObject = new(xtra "xmlparser")
    errorCode = parserObject.parseString(xmlString)
    errorString = parserObject.getError()
    if voidP(errorString) then
        parsedList = parserObject.makeList()
    else
        alert "Sorry, there was an error" && errorString
        exit
    end if
    return parsedList
end

// JavaScript syntax
function ConvertToList(xmlString) {
    parserObject = new Xtra("xmlparser"); // check syntax
    errorCode = parserObject.parseString(xmlString);
    errorString = parserObject.getError();
    if (voidP(errorString)) {
        parsedList = parserObject.makeList();
    } else {
        alert("Sorry, there was an error" + errorString);
        return false;
    }
    return parsedList;
}
```

Voir aussi

`makeSubList()`

makeScriptedSprite()

Syntaxe

```
-- Lingo syntax
spriteChannelObjRef.makeScriptedSprite({memberObjRef, loc})

// JavaScript syntax
spriteChannelObjRef.makeScriptedSprite({memberObjRef, loc});
```

Description

Méthode de piste d'image-objet ; demande qu'une piste d'image-objet contrôlée par le scénario soit désormais contrôlée par un script et, en option, place une image-objet d'un acteur spécifié à un emplacement spécifique de la scène.

Pour rendre au scénario le contrôle de la piste d'image-objet contrôlée par un script, appelez la méthode `removeScriptedSprite()`.

Paramètres

`memberObjRef` Facultatif. Référence à l'acteur à partir duquel une image-objet contrôlée par un script est créée. Lorsque ce paramètre est le seul spécifié, l'image-objet est placée au centre de la scène.

`loc` Facultatif. Point spécifiant l'emplacement de la scène où l'image-objet contrôlée par un script est placée.

Exemple

L'instruction suivante crée une image-objet contrôlée par un script dans la piste d'image-objet 5 à partir de l'acteur champ cerf-volant, puis la place à un point spécifique de la scène :

```
--- Lingo syntax
channel(5).makeScriptedSprite(member("kite"), point(35, 70))

// JavaScript syntax
channel(5).makeScriptedSprite(member("kite"), point(35, 70));
```

Voir aussi

`removeScriptedSprite()`, `Piste d'image-objet`

makeSubList()

Syntaxe

```
XMLnode.makeSubList()
```

Description

Fonction ; renvoie une liste de propriétés d'un nœud enfant de la même façon que `makeList()` renvoie la racine d'un document XML sous la forme d'une liste.

Paramètres

Aucune.

Exemple

Avec le code XML suivant :

```
<e1>
```

```
        <tagName attr1="val1" attr2="val2"/>
        <e2> element 2</e2>
        <e3>element 3</e3>
</e1>
```

L'instruction suivante renvoie une liste de propriétés constituée du contenu du premier enfant de la balise <e1> :

```
put gparser.child[ 1 ].child[ 1 ].makeSubList()
-- ["tagName": ["!ATTRIBUTES": ["attr1": "val1", "attr2": "val2"]]]
```

Voir aussi

makeList()

map()

Syntaxe

```
map(targetRect, sourceRect, destinationRect)
map(targetPoint, sourceRect, destinationRect)
```

Description

Fonction ; permet de positionner et de dimensionner un rectangle ou un point en fonction du rapport existant entre un rectangle source et un rectangle cible.

La relation de la valeur *targetRect* avec la valeur *sourceRect* contrôle la relation du résultat de la fonction avec *destinationRect*.

Paramètres

targetRect Requis. Rectangle cible de la relation.

targetPoint Requis. Point cible de la relation.

sourceRect Requis. Rectangle source de la relation.

destinationRect Requis. Rectangle de destination.

Exemple

Dans le comportement suivant, toutes les images-objets ont déjà été définies comme déplaçables. L'image-objet 2b contient un petit bitmap. L'image-objet 1s est une image-objet de forme rectangulaire, suffisamment grande pour contenir facilement l'image-objet 2b. L'image-objet 4b est une version plus grande du bitmap compris dans l'image-objet 2b. L'image-objet 3s est une version plus grande de la forme comprise dans l'image-objet 1s. Le déplacement des images-objets 2b ou 1s entraîne le déplacement de l'image-objet 4b. Lorsque vous faites glisser l'image-objet 2b, ses mouvements sont copiés par l'image-objet 4b. Lorsque vous faites glisser l'image-objet 1s, l'image-objet 4b se déplace dans la direction opposée. Le redimensionnement de l'image-objet 2b ou 1s produira des résultats intéressants.

```
on exitFrame
    sprite(4b).rect = map(sprite(2b).rect, sprite(1s).rect, sprite(3s).rect)
    go the frame
end
```

lingo_map.dcr

map (3D)

Syntaxe

```
member(whichCastmember).motion(whichMotion).map(whichOtherMotion {, boneName})
```

Description

Commande de mouvement 3D ; plaque un mouvement spécifié sur le mouvement en cours, puis l'applique à un segment ainsi qu'à tous ses enfants. Cette commande remplace tout mouvement précédemment plaqué au segment spécifié et à ses enfants. Cette commande ne modifie pas la liste de lecture d'un modèle.

Paramètres

whichOtherMotion Requis. Chaîne spécifiant le nom du mouvement à plaquer.

boneName Facultatif. Chaîne spécifiant le nom du segment auquel le mouvement plaqué est appliqué. Si ce paramètre est omis, le segment racine est utilisé.

Exemple

L'instruction suivante plaque le mouvement Plafond sur le mouvement Assis à partir du segment Cou. Le modèle s'assied et regarde le plafond simultanément.

```
member("Restaurant").motion("SitDown").map("LookUp", "Neck")
```

Voir aussi

motion(), duration (3D), cloneMotionFromCastmember

mapMemberToStage()

Syntaxe

```
sprite(whichSpriteNumber). mapMemberToStage(whichPointInMember)
mapMemberToStage(sprite whichSpriteNumber, whichPointInMember)
```

Description

Fonction ; utilise l'image-objet et le point spécifiés pour renvoyer un point équivalent dans les dimensions de la scène. Cette fonction prend correctement en compte les transformations en cours de l'image-objet à l'aide de `quad` ou le rectangle si celui-ci n'est pas transformé.

Elle est idéale pour déterminer si l'utilisateur a cliqué sur une zone spécifique d'un acteur, même si son image-objet sur la scène a été transformée de manière importante.

Si le point de la scène spécifié ne se trouve pas à l'intérieur de l'image-objet, la valeur `VOID` est renvoyée.

Paramètres

whichPointInMember Requis. Point à partir duquel un point équivalent est renvoyé.

Exemple

L'instruction suivante utilise l'image-objet(1) et le point(10, 10) pour renvoyer un point équivalent situé dans les dimensions de la scène.

```
-- Lingo
sprite(1). mapMemberToStage(point(10,10))

// Javascript
```

```
sprite(1). mapMemberToStage(point(10,10)) ;
```

Voir aussi

map(), mapStageToMember()

mapStageToMember()

Syntaxe

```
sprite(whichSpriteNumber). mapStageToMember(whichPointOnStage)
mapStageToMember(sprite whichSpriteNumber, whichPointOnStage)
```

Description

Fonction ; utilise l'image-objet et le point spécifiés pour renvoyer un point équivalent dans les dimensions de l'acteur. Cette fonction prend correctement en compte les éventuelles transformations en cours de l'image-objet à l'aide de `quad` ou le rectangle si celui-ci n'est pas transformé.

Elle est idéale pour déterminer si l'utilisateur a cliqué sur une zone spécifique d'un acteur, même si son image-objet sur la scène a été transformée de manière importante.

Si le point de la scène spécifié ne se trouve pas à l'intérieur de l'image-objet, cette fonction renvoie la valeur `VOID`.

Paramètres

`whichPointOnStage` Requis. Point à partir duquel un point équivalent est renvoyé.

Voir aussi

map(), mapMemberToStage()

marker()

Syntaxe

```
-- Lingo syntax
_movie.marker(markerNameOrNum)

// JavaScript syntax
_movie.marker(markerNameOrNum);
```

Description

Méthode d'animation ; renvoie le numéro d'image des repères situés avant ou après l'image en cours.

Cette méthode est utile pour mettre en œuvre un bouton Suivant ou Précédent ou pour définir une boucle d'animation.

Si le paramètre *`markerNameOrNum`* est un nombre entier, il peut correspondre à tout nombre entier positif ou négatif ou à la valeur 0. Par exemple :

- `marker(2)` : renvoie le numéro d'image du deuxième repère après l'image en cours.
- `marker(1)` : renvoie le numéro d'image du premier repère après l'image en cours.
- `marker(0)` : renvoie le numéro de l'image en cours si celle-ci contient un repère ou, si elle n'en contient pas, le numéro d'image du repère précédent.

- `marker(-1)` : renvoie le numéro d'image du premier repère précédant le repère (0).
- `marker(-2)` : renvoie le numéro d'image du second repère précédant le repère (0).

Si le paramètre *markerNameOrNum* est une chaîne, `marker()` renvoie le numéro de la première image dont le libellé de repère correspond à la chaîne.

Paramètres

markerNameOrNum Requis. Chaîne spécifiant un libellé de repère ou nombre entier spécifiant un numéro de repère.

Exemple

L'instruction suivante place la tête de lecture au début de l'image en cours si celle-ci contient un repère ; dans le cas contraire, elle place la tête de lecture au repère précédent :

```
-- Lingo syntax
_movie.go(_movie.marker(0))

// JavaScript syntax
_movie.go(_movie.marker(0));
```

L'instruction suivante attribue à la variable `nextMarker` la valeur du repère suivant dans le scénario :

```
-- Lingo syntax
nextMarker = _movie.marker(1)

// JavaScript syntax
nextMarker = _movie.marker(1);
```

Voir aussi

frame, frameLabel, go(), label(), markerList, Animation

matrixAddition()

Syntaxe

```
<Matrix> matrixAddition(matrix1, matrix2)
```

Description

Fonction globale ; additionne deux matrices et renvoie le résultat sous la forme d'une matrice. matrice1 et matrice2 doivent avoir la même dimension.

Paramètres

matrix1 Requis. Première matrice.

matrix2 Requis. Seconde matrice.

Exemple

L'extrait de code suivant crée deux matrices, puis les additionne à l'aide de la méthode matrixAddition().

```
--Lingo
mat1 = newMatrix(2, 2, [1,2,3,4])
mat2 = newMatrix(2, 2, [5,6,7,8])
mat3 = matrixAddition(mat1, mat2)
put(mat3.getVal(2,2))
-- 12.0000

//Java Script
```

```
mat1 = newMatrix(2, 2, list(1,2,3,4));
mat2 = newMatrix(2, 2, list(5,6,7,8));
mat3 = matrixAddition(mat1, mat2);
put(mat3.getVal(2,2));
// 12.0000
```

Voir aussi

getVal(), setVal(), numRows(), numColumns(), matrixMultiply(), matrixMultiplyScalar(), matrixTranspose(), newMatrix()

matrixMultiply()

Syntaxe

```
<Matrix> matrixMultiply(matrix1, matrix2)
```

Description

Fonction globale ; multiplie deux matrices et renvoie le résultat sous la forme d'une matrice. Le nombre de colonnes de matrice1 doit être identique au nombre de lignes de matrice2.

Paramètres

matrix1 Requis. Première matrice.

matrix2 Requis. Seconde matrice.

Exemple

L'extrait de code suivant crée deux matrices, puis les multiplie à l'aide de la méthode matrixMultiply().

```
--Lingo
mat1 = newMatrix(2, 2, [1,2,3,4])
mat2 = newMatrix(2, 2, [5,6,7,8])
mat3 = matrixMultiply(mat1, mat2)
put(mat3.getVal(2,2))

-- 50.0000


//Java Script
mat1 = newMatrix(2, 2, list(1,2,3,4));
mat2 = newMatrix(2, 2, list(5,6,7,8));
mat3 = matrixMultiply(mat1, mat2);
put(mat3.getVal(2,2));

// 50.0000
```

Voir aussi

getVal(), setVal(), numRows(), numColumns(), matrixAddition(), matrixMultiplyScalar(), matrixTranspose(), newMatrix()

matrixMultiplyScalar()

Syntaxe

```
<Matrix> matrixMultiplyScalar(matrix, scalarMultiplier)
```

Description

Fonction globale ; multiplie chaque élément de la matrice par une valeur scalaire donnée et renvoie le résultat sous la forme d'une matrice.

Paramètres

matrix Requis. Matrice impliquée dans la multiplication scalaire.

scalarMultiplier Requis. Valeur scalaire par laquelle chaque matrice doit être multipliée.

Exemple

L'extrait de code suivant crée deux matrices, puis les multiplie à l'aide de la méthode matrixMultiplyScalar().

```
--Lingo
mat1 = newMatrix(2, 2, [1,2,3,4])
mat2 = matrixMultiplyScalar(mat1, 10)
put(mat2.getVal(2,2))

-- 40.0000

//Java Script
mat1 = newMatrix(2, 2, list(1,2,3,4));
mat2 = matrixMultiplyScalar(mat1, 10);
put(mat2.getVal(2,2));

// 40.0000
```

Voir aussi

getVal(), setVal(), numRows(), numColumns(), matrixAddition(), matrixMultiply(), matrixTranspose(), newMatrix()

matrixTranspose()

Syntaxe

```
<Matrix> matrixTranspose(matrix)
```

Description

Fonction globale ; calcule la transposée d'une matrice donnée et renvoie le résultat sous la forme d'une autre matrice.

Paramètres

matrice Requis. Matrice à transposer.

Exemple

L'extrait de code suivant crée une matrice, puis la transpose à l'aide de la méthode matrixTranspose().

```
--Lingo
mat1 = newMatrix(2, 2, [1,2,3,4])
mat2 = matrixTranspose(mat1)
put(mat1.getVal(1,2))
put(mat2.getVal(1,2))

-- 2.0000
-- 3.0000

//Java Script
mat1 = newMatrix(2, 2, list(1,2,3,4));
mat2 = matrixTranspose(mat1);
put(mat1.getVal(1,2));
```

```
put(mat2.getVal(1,2));

// 2.0000
// 3.0000
```

Voir aussi

getVal(), setVal(), numRows(), numColumns(), matrixAddition(), matrixMultiply(), matrixMultiplyScalar(), newMatrix()

max()

Syntaxe

```
list.max()
max(list)
max(value1, value2, value3, ...)
```

Description

Fonction (Lingo uniquement) ; renvoie la valeur maximale de la liste spécifiée ou d'une série de valeurs donnée.

La fonction `max` est également utilisable avec des caractères ASCII, de la même manière que les opérateurs < et > peuvent être utilisés avec des chaînes.

Paramètres

value1, value2, value3, ... Facultatif. Liste de valeurs au sein de laquelle la valeur maximale est choisie.

Exemple

Le gestionnaire suivant affecte à la variable `Winner` la valeur maximale de la liste `Bids`, qui est composée de `[#Martin:600, #Dupont:750, #Lajoie:230]`. Le résultat est ensuite inséré dans le contenu de l'acteur champ Félicitations.

```
-- Lingo syntax
on findWinner Bids
    Winner = Bids.max()
    member("Congratulations").text =
    "You have won, with a bid of $" & Winner &"!"
end

// JavaScript syntax
function findWinner(Bids) {
    Winner = Bids.max();
    member("Congratulations").text = "You have won, with a bid of $" +
    Winner + "!");
}
```

maximize()

Syntaxe

```
-- Lingo syntax
windowObjRef.maximize()

// JavaScript syntax
windowObjRef.maximize();
```

Description

Méthode de fenêtre ; agrandit une fenêtre.

Utilisez cette méthode en cas de création de barres de titre personnalisées.

Paramètres

Aucune.

Exemple

Les instructions suivantes agrandissent la fenêtre Artistes si elle ne l'est pas déjà.

```
-- Lingo syntax
if (window("Artists").sizeState <> #maximized) then
    window("Artists").maximize()
end if

// JavaScript syntax
if (window("Artists").sizeState != symbol("maximized")) {
    window("Artists").maximize();
}
```

Voir aussi

`minimize()`, Fenêtre

mci

Syntaxe

```
mci "string"
```

Description

Commande ; pour Windows uniquement, transmet les chaînes spécifiées par *`string`* à l'interface de contrôle des médias de Windows (MCI) pour le contrôle des extensions multimédia.

Remarque : *Microsoft ne recommande plus l'utilisation de l'interface MCI de 16 bits. Vous devrez peut-être utiliser des Xtras de fabricants tiers pour remplacer cette fonctionnalité.*

Paramètres

`string` Requis. Chaîne transmise à l'interface MCI.

Exemple

L'instruction suivante demande à la commande `play cdaudio from 200 to 600 track 7` de ne lire le CD audio que lorsque l'animation est lue sous Windows :

```
mci "play cdaudio from 200 to 600 track 7"
```

member()

Syntaxe

```
-- Lingo syntax
member(memberNameOrNum {, castNameOrNum})
```

```
// JavaScript syntax
member(memberNameOrNum {, castNameOrNum});
```

Description

Fonction de niveau supérieur ; crée une référence à un acteur et, en option, spécifie la bibliothèque de distribution contenant cet acteur.

Si elle est utilisée avec les paramètres *memberNameOrNum* et *castNameOrNum*, la méthode `member()` constitue une référence spécifique à une bibliothèque de distribution et à un acteur de cette dernière :

```
trace(sprite(1).member);
// (member 1 of castLib 1)
```

Cette méthode diffère de la propriété d'image-objet `spriteNum` qui est toujours un nombre entier désignant une position dans une bibliothèque de distribution, mais qui ne spécifie pas cette bibliothèque :

```
trace(sprite(2).spriteNum);
// 2
```

Le numéro d'un acteur constitue également une référence absolue à un acteur spécifique d'une bibliothèque de distribution particulière :

```
trace(sprite(3).member.number)
// 3
```

Paramètres

memberNameOrNum Requis. Chaîne spécifiant le nom de l'acteur à référencer ou nombre entier spécifiant la position d'index de l'acteur à référencer.

castNameOrNum Facultatif. Chaîne spécifiant le nom de la bibliothèque de distribution à laquelle appartient l'acteur ou nombre entier spécifiant la position d'index de cette bibliothèque. Si ce paramètre est omis, la fonction `member()` recherche dans toutes les bibliothèques de distribution jusqu'à ce qu'une correspondance soit trouvée.

Exemple

Les instructions suivantes définissent la variable `title property` à l'acteur Avions situé dans la bibliothèque de `displayTemplate`.

```
-- Lingo syntax
memWings = member("Planes", "Transportation")

// JavaScript syntax
var memWings = member("Planes", "Transportation");
```

Voir aussi

Acteur, Image-objet, spriteNum

mergeDisplayTemplate()

Syntaxe

```
-- Lingo syntax
_movie.mergeDisplayTemplate(propList)

// JavaScript syntax
_movie.mergeDisplayTemplate(propList);
```

Description

Méthode d'animation ; fusionne simultanément un nombre arbitraire de propriétés de modèle d'affichage dans un jeu existant de propriétés de modèle d'affichage.

Paramètres

propList Requis. Liste de propriétés contenant les propriétés de modèle d'affichage à fusionner dans le jeu existant de propriétés de modèle d'affichage. Dans Lingo, le paramètre *propList* peut correspondre à une liste de paires nom/valeur ou symbole/valeur séparées par des virgules. En syntaxe JavaScript, *propList* peut uniquement désigner une liste de paires nom/valeur séparées par des virgules.

Exemple

L'instruction suivante fusionne une valeur de la liste des propriétés `title` dans l'ensemble de propriétés `displayTemplate` :

```
-- Lingo syntax
_movie.mergeDisplayTemplate(propList(#title, "Welcome!"))

// JavaScript syntax
_movie.mergeDisplayTemplate(propList("title", "Welcome!"))
```

Voir aussi

appearanceOptions, displayTemplate, Animation, propList(), titlebarOptions

mergeProps()

Syntaxe

```
-- Lingo syntax
windowObjRef.mergeProps(propList)

// JavaScript syntax
windowObjRef.mergeProps(propList);
```

Description

Méthode de fenêtre. Fusionne simultanément un nombre arbitraire de propriétés de fenêtre dans le jeu de propriétés de fenêtre existant.

Paramètres

propList Requis. Jeu de propriétés de fenêtre à fusionner dans le jeu de propriétés de fenêtre existant. Les propriétés sont spécifiées par les propriétés `appearanceOptions` et `titlebarOptions`.

- Dans Lingo, le paramètre *propList* peut correspondre à une liste de paires nom/valeur ou symbole/valeur séparées par des virgules.
- En syntaxe JavaScript, *propList* peut uniquement désigner une liste de paires nom/valeur séparées par des virgules.

Exemple

L'instruction suivante définit différentes propriétés pour la fenêtre Voitures.

```
-- Lingo syntax
window("Cars").mergeProps([#title:"Car pictures", #resizable:FALSE,
#titlebarOptions:[#closebox:TRUE, #icon:member(2)], #appearanceOptions:[#border:#line,
#shadow:TRUE]])
```

```
// JavaScript syntax
window("Cars").mergeProps(propList("title","Car pictures",
"resizable",false,"titlebarOptions",propList("closebox",true, "icon",member(2)),
"appearanceOptions",propList("border","line", "shadow",true)));
```

Voir aussi

appearanceOptions, titlebarOptions, Fenêtre

mesh (propriété)

Syntaxe

```
member(whichCastmember).model(whichModel).meshdeform.mesh[index].meshProperty
```

Description

Commande 3D ; permet d'accéder aux propriétés de maille des modèles auxquels le modificateur `meshDeform` est associé. Lorsqu'elle est utilisée en tant que `mesh.count`, cette commande renvoie le nombre total de mailles du modèle référencé.

Les propriétés de chacune des mailles auxquelles il est possible d'accéder sont les suivantes :

- `colorList` permet d'obtenir ou de définir la liste des couleurs utilisées par la maille spécifiée.
- `vertexList` permet d'obtenir ou de définir la liste des sommets utilisés par la maille spécifiée.
- `normalList` vous permet d'obtenir ou de définir la liste des vecteurs de normales utilisés par la maille spécifiée.
- `textureCoordinateList` permet d'obtenir ou de définir les coordonnées de texture utilisées par la première couche de texture de la maille spécifiée. Pour obtenir ou définir les coordonnées de texture pour toute autre couche de texture de la maille spécifiée, utilisez `meshdeform.mesh[`*index*`].texturelayer[`*index*`].textureCoordinateList`.
- `textureLayer[index]` permet d'obtenir ou de définir l'accès aux propriétés de la couche de texture spécifiée.
- `face[index]` permet d'obtenir ou de définir les sommets, les normales, les coordonnées de texture, les couleurs et les matériaux utilisés par les faces de la maille spécifiée.
- `face.count` permet d'obtenir le nombre total des faces qui se trouvent à l'intérieur de la maille spécifiée.

Remarque : *Pour plus d'informations sur ces propriétés, reportez-vous aux entrées correspondantes (répertoriées sous la section « Voir aussi » de cette entrée).*

Paramètres

Aucune.

Exemple

Le code Lingo suivant ajoute le modificateur `#meshDeform` au modèle truc1, puis affiche la liste des sommets de la première maille du modèle truc1.

```
-- Lingo
member("newAlien").model("thing1").addModifier(#meshDeform)
put member("newalien").model("thing1").meshDeform.mesh[1].vertexList

// javascript
member("newAlien").getProp("model",1).addModifier(symbol("meshDeform"));
```

```
put
(member("newalien").getProp("model",1).getPropRef("meshDeform").getPropRef("mesh",1).verte
xList ;
-- [vector(239.0, -1000.5, 27.4), vector(162.5, -1064.7, 29.3), vector(115.3, -1010.8, -
40.6),

vector(239.0, -1000.5, 27.4), vector(115.3, -1010.8, -40.6),
vector(162.5, -1064.7, 29.3), vector(359.0, -828.5, -46.3),
vector(309.9, -914.5, -45.3)]
```

L'instruction suivante affiche le nombre de mailles du modèle Avion.

```
-- Lingo
put member("world").model("Aircraft").meshDeform.mesh.count
-- 4
```

Voir aussi

meshDeform (modificateur), colorList, textureCoordinateList, textureLayer, normalList, vertexList (déformation de maille), face[ ]

meshDeform (modificateur)

Syntaxe

```
member(whichCastmember).model(whichModel).meshDeform.propertyName
```

Description

Modificateur 3D ; permet de contrôler les différents aspects de la structure de maille du modèle référencé. Lorsque vous ajoutez à un modèle le modificateur `#meshDeform` (à l'aide de la commande `addModifier`), vous avez accès aux propriétés suivantes du modificateur `#meshDeform` :

Remarque : *Pour plus d'informations sur ces propriétés, consultez les entrées correspondantes (référencées sous la section Voir aussi de cette entrée).*

- `face.count` renvoie le nombre total de faces du modèle référencé.
- `mesh.count` renvoie le nombre de mailles du modèle référencé.
- `mesh[index]` permet d'accéder aux propriétés de la maille spécifiée.

Paramètres

Aucune.

Exemple

L'instruction suivante affiche le nombre de faces du modèle gbFace.

```
-- Lingo
put member("3D World").model("gbFace").meshDeform.face.count
-- 432

// Javascript
member("3D World").getProp("model", a).getPropRef("meshDeform") ;
// where a refers to the number index of the model "gbFace"
```

L'instruction suivante affiche la liste de sommets des mailles du modèle gbFace :

```
-- Lingo
put member("3D World").model("gbFace").meshDeform.mesh[1].vertexList
```

```
-- 2

// Javscript
member("3D World").getProp("model", a).getPropRef("meshDeform").getPropRef("mesh"
,1).vertexList ;
```

L'instruction suivante affiche le nombre de faces de la deuxième maille du modèle gbFace :

```
put member("3D World").model("gbFace").meshDeform.mesh[2].face.count
-- 204
```

Voir aussi

mesh (propriété), addModifier

min

Syntaxe

```
list.min
min(list)
min (a1, a2, a3...)
```

Description

Fonction (Lingo uniquement) ; spécifie la valeur minimale d'une liste.

Paramètres

a1, a2, a3, ... Facultatif. Liste de valeurs au sein de laquelle la valeur minimale est choisie.

Exemple

Le gestionnaire suivant affecte à la variable `vLowest` la valeur minimale de la liste `bids`, qui est composée de `[#Pierre:600, #Paul:750, #Jean:230]`. Le résultat est alors inséré dans l'acteur champ Désolé :

```
on findLowest bids
    vLowest = bids.min()
    member("Sorry").text = \
    "We're sorry, your bid of $" & vLowest && "is not a winner!"
end
```

Voir aussi

max()

minimize()

Syntaxe

```
-- Lingo syntax
windowObjRef.minimize()

// JavaScript syntax
windowObjRef.minimize();
```

Description

Méthode de fenêtre ; réduit une fenêtre.

Utilisez cette méthode en cas de création de barres de titre personnalisées.

Paramètres

Aucune.

Exemple

Les instructions suivantes réduisent la fenêtre Artistes si elle ne l'est pas déjà.

```
-- Lingo syntax
if (window("Artists").sizeState <> #minimized) then
    window("Artists").minimize()
end if

// JavaScript syntax
if (window("Artists").sizeState.toString() != symbol("minimized").toString())
{
    window("Artists").minimized();
}
```

Voir aussi

`maximize()`, Fenêtre

model

Syntaxe

```
member(whichCastmember).model(whichModel)
member(whichCastmember).model[index]
member(whichCastmember).model.count
member(whichCastmember).model(whichModel).propertyName
member(whichCastmember).model[index].propertyName
```

Description

Commande 3D ; renvoie le modèle trouvé dans l'acteur référencé dont le nom est spécifié par *`whichModel`* ou trouvé à la position d'index spécifiée par *`index`*. Si aucun modèle n'existe pour le paramètre spécifié, la commande renvoie `void`. Tout comme `model.count`, la commande renvoie le nombre de modèles détectés dans l'acteur référencé. Cette commande permet également d'accéder aux propriétés du modèle spécifié.

Les comparaisons de noms de modèle ne sont pas sensibles à la hauteur de casse. La position d'index d'un modèle particulier peut changer lorsque des objets dans des positions inférieures sont supprimés.

Si aucun modèle n'utilise le nom spécifié ou n'est trouvé à la position d'index spécifiée, cette commande renvoie la valeur void.

Paramètres

`whichModel` Facultatif. Chaîne spécifiant le nom du modèle à renvoyer.

Exemple

L'instruction suivante stocke une référence au modèle Lecteur dans la variable `thismodel` :

```
thismodel = member("3DWorld").model("Player Avatar")
```

L'instruction suivante stocke une référence au huitième modèle de l'acteur Univers3D dans la variable `thismodel`.

```
thismodel = member("3DWorld").model[8]
```

L'instruction suivante indique qu'il existe quatre modèles dans l'acteur de l'image-objet 1.

```
put sprite(1).member.model.count
-- 4
```

modelResource

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource)
member(whichCastmember).modelResource[index]
member(whichCastmember).modelResource.count
member(whichCastmember).modelResource(whichModelResource).propertyName
member(whichCastmember).modelResource[index].propertyName
```

Description

Commande 3D ; renvoie la ressource de modèle trouvée dans l'acteur référencé dont le nom est spécifié par *whichModelResource* ou trouvée à la position d'index spécifiée par le paramètre *index*. Si aucune ressource de modèle n'existe pour le paramètre spécifié, la commande renvoie `void`. Tout comme `modelResource.count`, la commande renvoie le nombre de ressources de modèle détectées dans l'acteur référencé. Cette commande permet également d'accéder aux propriétés du modèle spécifié.

Les comparaisons de chaînes de noms de ressources de modèle ne sont pas sensibles à la hauteur de casse. La position d'index d'une ressource de modèle particulière peut changer lorsque des objets dans des positions inférieures sont supprimés.

Paramètres

whichModelResource Facultatif. Chaîne spécifiant le nom de la ressource de modèle à renvoyer.

Exemple

L'instruction suivante stocke une référence à la ressource de modèle MaisonA dans la variable `thismodelResource`.

```
thismodelResource = member("3DWorld").modelResource("HouseA")
```

L'instruction suivante stocke une référence à la quatorzième ressource de modèle de l'acteur Univers3D dans la variable `thismodelResource`.

```
thismodelResource = member("3DWorld").modelResource[14]
```

L'instruction suivante indique qu'il existe dix ressources de modèle dans l'acteur de l'image-objet 1.

```
put sprite(1).member.modelResource.count
--10
```

modelsUnderLoc

Syntaxe

```
member(whichCastmember).camera(whichCamera).modelsUnderLoc(pointWithinSprite, optionsList)
```

Description

Commande 3D ; renvoie une liste des modèles situés sous un point spécifié dans le cadre d'une image-objet utilisant la caméra référencée. La liste des modèles peut également être comparée à un jeu de paramètres facultatifs avant d'être renvoyée.

Dans la liste renvoyée, le premier modèle listé est celui qui est le plus proche du spectateur et le dernier modèle listé est le plus éloigné.

Une seule intersection (la plus proche) est renvoyée par modèle.

La commande renvoie une liste vide s'il n'existe aucun modèle sous le point spécifié.

Paramètres

pointWithinSprite Requis. Point sous lequel une liste de modèles est trouvée. Ce point est calculé en fonction du coin supérieur gauche de l'image-objet en pixels.

optionsList Facultatif. Liste spécifiant le nombre maximal de modèles à renvoyer, le niveau de détail des informations, une liste de modèles sur laquelle repose la distribution, ainsi que la distance maximale à laquelle le rayon doit être dessiné. Toutes ces propriétés sont facultatives.

maxNumberOfModels Facultatif. Nombre entier spécifiant la longueur maximale de la liste renvoyée. Si ce paramètre est omis, la commande renvoie une liste contenant les références de tous les modèles trouvés sous le point spécifié.

levelOfDetail Facultatif. Symbole spécifiant le niveau de détail des informations renvoyées. Les valeurs possibles sont les suivantes :

- `#simple` renvoie une liste contenant les références des modèles situés sous le point. C'est la valeur par défaut.
- `#detailed` renvoie une liste de listes de propriétés représentant chacune un modèle traversé. Chaque liste de propriétés doit contenir les propriétés suivantes :
  - `#model` est une référence à l'objet de modèle traversé.
  - `#distance` est la distance séparant la caméra du point d'intersection avec le modèle.
  - `#isectPosition` est un vecteur représentant la position du point d'intersection dans l'univers.
  - `#isectNormal` est le vecteur de la maille au point d'intersection.
  - `#meshID` est l'identifiant de la maille traversée, qui peut être utilisé comme index dans la liste des mailles du modificateur `meshDeform`.
  - `#faceID` est l'identifiant de la face traversée, qui peut être utilisé comme index dans la liste des faces du modificateur `meshDeform`.
  - `#vertices` est une liste de vecteurs contenant trois éléments qui représentent les positions dans l'univers des sommets de la face traversée.
  - `#uvCoord` est une liste de propriétés contenant les propriétés `#u` et `#v` représentant les coordonnées barycentriques u et v de la face.

modelList Facultatif. Liste des références de modèle incluses éventuellement trouvées sous le rayon spécifié. Les références de modèle non incluses dans cette liste sont ignorées, même si elles figurent sous le rayon spécifié. Utilisez les références de modèle, et non les noms de chaîne des modèles. Spécifiez chacun des modèles que vous souhaitez inclure. L'ajout d'une référence de modèle parent n'inclut pas automatiquement ses références de modèle enfant.

Exemple

L'instruction suivante crée une liste de dix modèles :

```
tModelList = [member("3D").model("foo"),member("3D").model[10]]
```

L'instruction suivante crée une liste d'options demandant le renvoi de dix modèles au maximum, l'inclusion de détails simples et le dessin des résultats à partir de `tModelList` :

```
tOptionsList = [#maxNumberOfModels: 10, #levelOfDetail: #simple, #modelList: tModelList]
```

Une fois cette liste d'options créée, la première ligne du gestionnaire suivant transfère l'emplacement du curseur d'un point de la scène vers un point de l'image-objet 5. La deuxième ligne utilise la commande `modelsUnderLoc` pour obtenir les trois premiers modèles situés sous ce point. La troisième ligne affiche les informations renvoyées concernant les modèles dans la fenêtre Messages.

```
-- Lingo syntax
on mouseUp
    pt = the mouseLoc - point(sprite(5).left, sprite(5).top)
    m = sprite(5).camera.modelsUnderLoc(pt, tOptionsList)
    put m
    end

// JavaScript syntax
function mouseUp() {
    pt = _mouse.mouseLoc - point(sprite(5).left, sprite(5).top);
    m = sprite(5).camera.modelsUnderLoc(pt, tOptionsList);
    put(m);
}
```

Voir aussi

modelsUnderRay, modelUnderLoc

modelsUnderRay

Syntaxe

```
member(whichCastmember).modelsUnderRay(locationVector, directionVector, optionsList)
```

Description

Commande 3D ; renvoie une liste des modèles trouvés sous un rayon tracé à partir d'une position spécifiée et pointant dans une direction spécifique, les deux vecteurs étant spécifiés en coordonnées relatives à l'univers. La liste des modèles peut également être comparée à un jeu de paramètres facultatifs avant d'être renvoyée.

Dans la liste renvoyée, le premier modèle répertorié est le plus proche de la position spécifiée par *locationVector*, tandis que le dernier modèle répertorié est le plus éloigné de cette position.

Une seule intersection (la plus proche) est renvoyée par modèle.

La commande renvoie une liste vide s'il n'existe aucun modèle sous le rayon spécifié.

Paramètres

locationVector Requis. Vecteur depuis lequel un rayon est tracé et sous lequel une liste de modèles est trouvée.

directionVector Requis. Vecteur spécifiant la direction dans laquelle pointe le rayon.

optionsList Facultatif. Liste spécifiant le nombre maximal de modèles à renvoyer, le niveau de détail des informations, une liste de modèles sur laquelle repose la distribution, ainsi que la distance maximale à laquelle le rayon doit être dessiné. Toutes ces propriétés sont facultatives.

maxNumberOfModels Facultatif. Nombre entier spécifiant la longueur maximale de la liste renvoyée. Si ce paramètre est omis, la commande renvoie une liste contenant les références de tous les modèles trouvés sous le rayon spécifié.

levelOfDetail Facultatif. Symbole spécifiant le niveau de détail des informations renvoyées. Les valeurs possibles sont les suivantes :

- `#simple` renvoie une liste contenant les références des modèles situés sous le point. C'est la valeur par défaut.

- `#detailed` renvoie une liste de listes de propriétés représentant chacune un modèle traversé. Chaque liste de propriétés doit contenir les propriétés suivantes :
  - `#model` est une référence à l'objet de modèle traversé.
  - `#distance` est la distance séparant la position d'univers spécifiée par *`locationVector`* du point d'intersection avec le modèle.
  - `#isectPosition` est un vecteur représentant la position du point d'intersection dans l'univers.
  - `#isectNormal` est le vecteur de la maille au point d'intersection.
  - `#meshID` est l'identifiant de la maille traversée, qui peut être utilisé comme index dans la liste des mailles du modificateur `meshDeform`.
  - `#faceID` est l'identifiant de la face traversée, qui peut être utilisé comme index dans la liste des faces du modificateur `meshDeform`.
  - `#vertices` est une liste de vecteurs contenant trois éléments qui représentent les positions dans l'univers des sommets de la face traversée.
  - `#uvCoord` est une liste de propriétés contenant les propriétés `#u` et `#v` représentant les coordonnées barycentriques u et v de la face.

`modelList` Facultatif. Liste des références de modèle incluses éventuellement trouvées sous le rayon spécifié. Les références de modèle non incluses dans cette liste sont ignorées, même si elles figurent sous le rayon spécifié. Utilisez les références de modèle, et non les noms de chaîne des modèles. Spécifiez chacun des modèles que vous souhaitez inclure. L'ajout d'une référence de modèle parent n'inclut pas automatiquement ses références de modèle enfant.

`maxDistance` Facultatif. Distance maximale à partir de la position dans l'univers spécifiée par *`locationVector`*. Si la sphère de délimitation d'un modèle se trouve dans la distance maximale spécifiée, ce modèle est inclus. Si la sphère de délimitation se trouve dans la plage, elle peut contenir des polygones dans la plage et peut donc être traversée.

Exemple

L'instruction suivante crée une liste de dix modèles :

```
tModelList = [member("3D").model("foo"),member("3D").model[10]]
```

L'instruction suivante crée une liste d'options demandant le renvoi de dix modèles au maximum, l'inclusion de détails simples, le dessin des résultats à partir de `tModelList`, ainsi qu'une distance maximale de 50 pour le tracé du rayon :

```
tOptionsList = [#maxNumberOfModels: 10, #levelOfDetail: #simple, #modelList: tModelList,
#maxDistance: 50]
```

Une fois cette liste d'options créée, l'instruction suivante l'inclut sous un rayon tracé à partir de la position vector (0, 0, 300) et pointant vers l'axe des z négatif :

```
put member("3d").modelsUnderRay(vector(0, 0, 300), vector(0, 0, -1), tOptionsList)
```

Voir aussi

`modelsUnderLoc`, `modelUnderLoc`

modelUnderLoc

Syntaxe

```
member(whichCastmember).camera(whichCamera).modelUnderLoc(pointWithinSprite)
```

Description

Commande 3D ; renvoie une référence au premier modèle situé sous un point spécifié dans le cadre d'une image-objet utilisant la caméra référencée.

Cette commande renvoie la valeur `void` si aucun modèle n'est situé sous le point spécifié.

Pour obtenir une liste de tous les modèles situés sous un point spécifié, ainsi que des informations détaillées les concernant, reportez-vous à l'entrée `modelsUnderLoc`.

Paramètres

`pointWithinSprite` Requis. Point sous lequel le premier modèle est trouvé. L'emplacement de *`pointWithinSprite`* est calculé en fonction du coin supérieur gauche de l'image-objet, en pixels.

Exemple

La première ligne du gestionnaire suivant transfère l'emplacement du curseur d'un point de la scène à un point de l'image-objet 5. La deuxième ligne détermine le premier modèle situé sous ce point. La troisième ligne affiche les résultats dans la fenêtre Messages.

```
-- Lingo syntax
on mouseUp
    pt = the mouseLoc - point(sprite(5).left, sprite(5).top)
    m = sprite(5).camera.modelUnderLoc(pt)
    put m
end

// JavaScript syntax
function mouseUp() {
    pt = _mouse.mouseLoc - point(sprite(5).left, sprite(5).top);
    m = sprite(5).camera.modelUnderLoc(pt);
    put(m);
}
```

Voir aussi

`modelsUnderLoc`, `modelsUnderRay`

motion()

Syntaxe

```
member(whichCastmember).motion(whichMotion)
member(whichCastmember).motion[index]
member(whichCastmember).motion.count
```

Description

Commande 3D ; renvoie le mouvement trouvé dans l'acteur référencé dont le nom est spécifié par *`whichMotion`*, ou trouvé à la position d'index spécifiée par *`index`*. Tout comme `motion.count`, cette propriété renvoie le nombre total de mouvements détectés dans l'acteur.

Les comparaisons de chaînes de nom d'objet ne sont pas sensibles à la hauteur de casse. La position d'index d'un mouvement particulier peut changer lorsque des objets dans des positions inférieures sont supprimés.

Si aucun mouvement n'utilise le nom spécifié ou n'est trouvé à la position d'index spécifiée, cette commande renvoie la valeur void.

Exemple

```
thisMotion = member("3D World").motion("Wing Flap")
thisMotion = member("3D World").motion[7]
put member("scene").motion.count
-- 2
```

Voir aussi

duration (3D), map (3D)

move()

Syntaxe

```
-- Lingo syntax
memberObjRef.move({intPosn, castLibName})

// JavaScript syntax
memberObjRef.move({intPosn, castLibName});
```

Description

Méthode d'acteur ; déplace un acteur spécifié vers le premier emplacement vide de la distribution à laquelle il appartient ou vers un emplacement spécifié dans une distribution donnée.

Pour optimiser les résultats, utilisez cette méthode en phase de création, et non pendant l'exécution, car le déplacement est généralement enregistré avec le fichier. En général, l'emplacement réel de l'acteur n'affecte pas la plupart des présentations quand l'utilisateur final les lit. Pour changer le contenu d'une image-objet ou modifier l'affichage pendant l'exécution, définissez l'acteur de l'image-objet.

Paramètres

intPosn Facultatif. Nombre entier spécifiant la position dans la bibliothèque de distribution *castLibName* vers laquelle l'acteur est déplacé.

castLibName Facultatif. Chaîne spécifiant le nom de la bibliothèque de distribution vers laquelle l'acteur est déplacé.

Exemple

L'instruction suivante déplace l'acteur Temple vers la première position vide de la fenêtre Distribution :

```
-- Lingo syntax
member("shrine").move()

// JavaScript syntax
member("shrine").move();
```

L'instruction suivante déplace l'acteur Temple vers l'emplacement 20 de la fenêtre Distribution Bitmaps :

```
-- Lingo syntax
member("shrine").move(member(20, "Bitmaps"))

// JavaScript syntax
member("shrine").move(member(20, "Bitmaps"));
```

Voir aussi

Acteur

moveToBack()

Syntaxe

```
-- Lingo syntax
windowObjRef.moveToBack()

// JavaScript syntax
windowObjRef.moveToBack();
```

Description

Méthode de fenêtre ; place une fenêtre derrière toutes les autres fenêtres.

Paramètres

Aucune.

Exemple

Les instructions suivantes placent la première fenêtre de la liste windowList derrière toutes les autres fenêtres :

```
-- Lingo syntax
myWindow = _player.windowList[1]
myWindow.moveToBack()

// JavaScript syntax
var myWindow = _player.windowList[1];
myWindow.moveToBack();
```

Si vous connaissez le nom de la fenêtre à déplacer, utilisez la syntaxe suivante :

```
-- Lingo syntax
window("Demo Window").moveToBack()

// JavaScript syntax
window("Demo Window").moveToBack();
```

Voir aussi

moveToFront(), Fenêtre

moveToFront()

Syntaxe

```
-- Lingo syntax
windowObjRef.moveToFront()

// JavaScript syntax
windowObjRef.moveToFront();
```

Description

Méthode de fenêtre ; place une fenêtre devant toutes les autres fenêtres.

Paramètres

Aucune.

Exemple

Les instructions suivantes placent la première fenêtre de la liste `windowList` devant toutes les autres fenêtres :

```
-- Lingo syntax
myWindow = _player.windowList[1]
myWindow.moveToFront()

// JavaScript syntax
var myWindow = _player.windowList[1];
myWindow.moveToFront();
```

Si vous connaissez le nom de la fenêtre à déplacer, utilisez la syntaxe suivante :

```
-- Lingo syntax
window("Demo Window").moveToFront()

// JavaScript syntax
window("Demo Window").moveToFront();
```

Voir aussi

moveToBack(), Fenêtre

moveVertex()

Syntaxe

```
-- Lingo syntax
memberObjRef.moveVertex(vertexIndex, xChange, yChange)

// JavaScript syntax
memberObjRef.moveVertex(vertexIndex, xChange, yChange);
```

Description

Fonction ; place le sommet d'un acteur forme vectorielle à un autre emplacement.

Les coordonnées horizontales et verticales du déplacement sont calculées en fonction de la position actuelle du point du sommet. L'emplacement de celui-ci dépend de l'origine de l'acteur forme vectorielle.

La modification de l'emplacement du sommet affecte la forme de la même manière que si vous faites glisser le sommet dans l'éditeur.

Paramètres

vertexIndex Requis. Spécifie la position d'index du sommet à déplacer.

xChange Requis. Spécifie la quantité selon laquelle le sommet doit être déplacé horizontalement.

yChange Requis. Spécifie la quantité selon laquelle le sommet doit être déplacé verticalement.

Exemple

L'instruction suivante déplace le premier sommet de l'acteur forme vectorielle Archie de 25 pixels vers la droite et de 10 pixels vers le bas par rapport à sa position actuelle :

```
-- Lingo syntax
member("Archie").moveVertex(1, 25, 10)

// JavaScript syntax
member("Archie").moveVertex(1, 25, 10);
```

Voir aussi

addVertex(), deleteVertex(), moveVertexHandle(), originMode, vertexList

moveVertexHandle()

Syntaxe

```
-- Lingo syntax
memberObjRef.moveVertexHandle(vertexIndex, handleIndex, xChange, yChange)

// JavaScript syntax
memberObjRef.moveVertexHandle(vertexIndex, handleIndex, xChange, yChange);
```

Description

Fonction ; déplace la poignée du sommet d'un acteur de forme vectorielle vers un autre emplacement.

Les coordonnées horizontales et verticales du déplacement sont calculées par rapport à la position actuelle de la poignée du sommet. L'emplacement de la poignée du sommet dépend du sommet qu'elle contrôle.

La modification de l'emplacement de la poignée de contrôle affecte la forme de la même manière que si vous faites glisser le sommet dans un éditeur.

Paramètres

vertexIndex Requis. Spécifie la position d'index du sommet contenant la poignée à déplacer.

handleIndex Requis. Spécifie la position d'index de la poignée à déplacer.

xChange Requis. Spécifie la quantité selon laquelle la poignée du sommet doit être déplacée horizontalement.

yChange Requis. Spécifie la quantité selon laquelle la poignée du sommet doit être déplacée verticalement.

Exemple

L'instruction suivante déplace la première poignée de contrôle du deuxième sommet de l'acteur forme vectorielle Archie de 15 pixels vers la droite et de 5 pixels vers le haut :

```
-- Lingo syntax
moveVertexHandle(member("Archie"), 2, 1, 15, -5)

// JavaScript syntax
member("Archie").moveVertexHandle(2, 1, 15, -5);
```

Voir aussi

addVertex(), deleteVertex(), originMode, vertexList

multiply()

Syntaxe

```
transform.multiply(transform2)
```

Description

Commande 3D ; applique les effets de position, de rotation et de redimensionnement de *transform2* après la transformation d'origine.

Paramètres

transform2 Requis. Spécifie la transformation contenant les effets à appliquer à une autre transformation.

Exemple

L'instruction suivante applique les effets de position, de rotation et de redimensionnement de la transformation du modèle Mars à la transformation du modèle Pluton. Cette opération produit le même effet que la définition du modèle Mars en tant que parent de Pluton pour une image.

```
-- Lingo
member("scene").model("Pluto").transform.multiply(member("scene").model("Mars").transform)

// Javascript
member("scene").getProp("model" ,i ).transform.multiply(member("scene").getProp("model" ,
j).transform) ;
```

neighbor

Syntaxe

```
member(whichCastmember).model(whichModel).meshdeform.mesh[index].face[index].neighbor[index]
```

Description

Commande 3D ; commande meshDeform renvoyant une liste de listes décrivant les voisins d'une face particulière d'une maille à l'opposé du coin de face spécifié par l'index de voisinage `(1,2,3)`. Si la liste est vide, la face n'a aucun voisin dans cette direction. Si la liste contient plus d'une liste, la maille est non-manifold. La liste contient généralement une liste unique de quatre valeurs entières : `[meshIndex, faceIndex, vertexIndex, flipped]`.

La valeur `meshIndex` est l'index de la maille contenant la face voisine. La valeur `faceIndex` est l'index de la face voisine dans la maille. La valeur `vertexIndex` est l'index des sommets non partagés de la face voisine. La valeur `flipped` indique si l'orientation de la face est identique `(1)` ou opposée `(2)` à celle de la face d'origine.

Paramètres

Aucune.

Voir aussi

meshDeform (modificateur)

netAbort

Syntaxe

```
netAbort(URL)
netAbort(netID)
```

Description

Commande ; annule une opération réseau sans attendre de résultat.

L'utilisation d'un identifiant de réseau constitue la manière la plus efficace d'annuler une opération réseau. Cet identifiant est renvoyé lorsque vous utilisez une fonction réseau telle que `getNetText()` ou `postNetText()`.

Dans certains cas, si l'identifiant de réseau n'est pas disponible, vous pouvez utiliser une URL pour annuler la transmission de données à partir de cette URL. Celle-ci doit être identique à l'URL utilisée au départ de l'opération réseau. Si le transfert des données est terminé, cette commande n'a aucun effet.

Paramètres

URL Requis. Spécifie l'URL à annuler.

netID Facultatif. Spécifie l'identifiant de l'opération réseau à annuler.

Exemple

L'instruction suivante transmet un identifiant de réseau à la méthode netAbort pour annuler une opération réseau spécifique :

```
-- Lingo syntax
on mouseUp
    netAbort(myNetID)
end

// JavaScript syntax
function mouseUp() {
    netAbort(myNetID);
}
```

Voir aussi

getNetText(), postNetText

netDone()

Syntaxe

```
netDone()
netDone(netID)
```

Description

Fonction ; indique si une opération de chargement en tâche de fond (telle que `getNetText`, `preloadNetThing`, `gotoNetMovie`, `gotoNetPage` ou `netTextResult`) est terminée ou a été interrompue par suite d'une erreur du navigateur (`TRUE`, valeur par défaut) ou si elle est encore en cours (`FALSE`).

- Utilisez `netDone()` pour tester la dernière opération réseau.
- Utilisez `netDone(netID)` pour tester l'opération réseau identifiée par `netID`.

La fonction `netDone` renvoie 0 lorsqu'une opération de chargement en tâche de fond est en cours.

Paramètres

netID Facultatif. Spécifie l'identifiant de l'opération réseau à tester.

Exemple

Le gestionnaire suivant utilise la fonction netDone pour vérifier si la dernière opération réseau est terminée. Si l'opération est terminée, le texte renvoyé par `netTextResult` s'affiche dans l'acteur champ Affichage du texte.

```
-- Lingo syntax
on exitFrame
    if netDone() = 1 then
        member("Display Text").text = netTextResult()
    end if
end

// JavaScript syntax
function exitFrame() {
```

```
    if (netDone() == 1) {
        member("Display Text").text = netTextResult();
    }
}
```

Le gestionnaire suivant utilise un identifiant de réseau spécifique comme argument pour que la méthode netDone vérifie l'état d'une opération réseau particulière :

```
-- Lingo syntax
on exitFrame
    -- stay on this frame until the net operation is
    -- completed
    global mynetID
    if netDone(mynetID) = FALSE then
        go to the frame
    end if
end

// JavaScript syntax
function exitFrame() {
    // stay on this frame until the net operation is completed
    global mynetID;
    if (!(netDone(mynetID))) {
        _movie.go(_movie.frame);
    }
}
```

Voir aussi

`getNetText()`, `netTextResult()`, `gotoNetMovie`, `preloadNetThing()`

netError()

Syntaxe

```
netError()
netError(netID)
```

Description

Fonction ; détermine si une erreur s'est produite pendant une opération réseau, et, le cas échéant, renvoie un numéro d'erreur correspondant à un message d'erreur. Si l'opération s'est déroulée sans erreur, cette fonction renvoie un code indiquant que tout va bien. Si aucune opération de chargement en tâche de fond n'a commencé ou si l'opération est en cours, cette fonction renvoie une chaîne vide.

- Utilisez `netError()` pour tester la dernière opération réseau.
- Utilisez `netError(`*`netID`*`)` pour tester l'opération réseau spécifiée par `netID`.

Plusieurs codes d'erreur peuvent être renvoyés :

0	Tout va bien.
1	Le chemin d'accès local pointe vers un répertoire qui n'existe pas.
4	Classe MOA incorrecte. Les Xtras requis, réseau ou non, ne sont pas installés correctement ou ne sont pas installés du tout.
5	Interface MOA incorrecte. Voir 4.

6	URL incorrecte ou classe MOA incorrecte. Les Xtras requis, réseau ou non, ne sont pas installés correctement ou ne sont pas installés du tout.
20	Erreur interne. Renvoyé par `netError()` dans le navigateur Netscape si celui-ci a détecté une erreur de réseau ou une erreur interne.
900	Le fichier dans lequel écrire est en lecture seule.
903	Le disque est plein.
905	Spécification de fichier incorrecte.
2018	Une erreur postNetText se produit généralement lorsque vous utilisez postNetText avec les paramètres dans l'URL. M.Kloss recommande la syntaxe suivante pour éviter l'erreur 2018 : pNetID = postNetText(monURL, maListePropParam)
4144	L'opération réseau a échoué.
4145	L'opération réseau a échoué.
4146	La connexion n'a pas pu être établie avec l'hôte distant.
4149	Les données fournies par le serveur ont un format inattendu.
4150	Fermeture prématurée et inattendue de la connexion.
4154	L'opération n'a pas pu aboutir par suite du dépassement du temps imparti.
4155	Mémoire insuffisante pour terminer la transaction.
4156	La réponse du protocole à la requête indique une erreur de réponse.
4157	La transaction n'a pas pu être authentifiée.
4159	URL non valide.
4164	Impossible de créer de socket.
4165	L'objet demandé est introuvable (l'URL est peut-être incorrecte).
4166	Echec de proxy générique.
4167	Le transfert a été interrompu volontairement par le client.
4242	Téléchargement arrêté par `netAbort(url)`.
4836	Téléchargement annulé ou interrompu pour une raison inconnue, probablement une erreur du réseau.
4153	L'opération réseau a échoué.
4154	L'opération n'a pas pu aboutir par suite du dépassement du temps imparti.
4155	Mémoire insuffisante pour terminer la transaction.
4157	La transaction n'a pas pu être authentifiée.
4159	URL non valide.
4160	L'opération réseau a échoué.
4161	L'opération réseau a échoué.
4162	L'opération réseau a échoué.
4163	L'opération réseau a échoué.
4164	Impossible de créer un socket.

4165	L'objet demandé est introuvable (l'URL est peut-être incorrecte).
4166	Echec de proxy générique.
4167	Le transfert a été interrompu volontairement par le client.
4168	L'opération réseau a échoué.
4240	Les Xtras de réseau n'ont pas été correctement initialisés. (Lewis Francis)
4242	Téléchargement arrêté par netAbort(url)
4836	Téléchargement arrêté pour une raison inconnue. Il peut s'agir d'une erreur de réseau ou d'un abandon du téléchargement.

Paramètres

netID Facultatif. Spécifie l'identifiant de l'opération réseau à tester.

Exemple

L'instruction suivante transmet un identifiant de réseau à `netError` pour vérifier l'état d'erreur d'une opération réseau spécifique :

```
--Lingo syntax
on exitFrame
    global mynetID
    if netError(mynetID)<>"OK" then beep
end

// JavaScript syntax
function exitFrame() {
    global mynetID;
    if (netError(mynetID) != "OK") {
        _sound.beep();
    }
}
```

netLastModDate()

Syntaxe

```
netLastModDate()
```

Description

Fonction ; renvoie la date de la dernière modification de l'en-tête HTTP de l'élément spécifié. Cette chaîne utilise le format de temps universel (GMT) : *Ddd, nn Mmm yyyy hh:mm:ss* `GMT` (par exemple, Thu, 30 Jan 1997 12:00:00 AM GMT). Les jours et les mois peuvent être écrits en entier. Cette chaîne est toujours en anglais.

La fonction `netLastModDate` ne peut être appelée qu'une fois que `netDone` et `netError` ont rapporté que l'opération s'est terminée avec succès. Dès que l'opération suivante démarre, l'animation ou la projection Director efface les résultats de l'opération précédente pour économiser de la mémoire.

La chaîne de date est directement extraite de l'en-tête HTTP dans le format fourni par le serveur. Toutefois, cette chaîne n'est pas toujours fournie et, si tel est le cas, `netLastModDate` renvoie la valeur `EMPTY`.

Paramètres

Aucune.

Exemple

Les instructions suivantes vérifient la date d'un fichier téléchargé depuis Internet :

```
-- Lingo syntax
if netDone() then
    theDate = netLastModDate()
    if theDate.char[6..11] <> "Jan 30" then
        alert "The file is outdated."
    end if
end if

// JavaScript syntax
if (netDone()) {
    theDate = netLastModDate();
    if (theDate.char[6..11] != "Jan 30") {
        alert("The file is outdated");
    }
}
```

Voir aussi

`netDone()`, `netError()`

netMIME()

Syntaxe

```
netMIME()
```

Description

Fonction ; fournit le type MIME du fichier Internet renvoyé par la dernière opération réseau (le fichier HTTP ou FTP téléchargé en dernier).

La fonction `netMIME` ne peut être appelée qu'une fois que `netDone` et `netError` ont rapporté que l'opération s'est terminée avec succès. Dès que l'opération suivante démarre, l'animation ou la projection Director efface les résultats de l'opération précédente pour économiser de la mémoire.

Paramètres

Aucune.

Exemple

Le gestionnaire suivant vérifie le type MIME d'un fichier téléchargé depuis Internet et répond en conséquence :

```
-- Lingo syntax
on checkNetOperation theURL
    if netDone (theURL) then
        set myMimeType = netMIME()
        case myMimeType of
            "image/jpeg": go frame "jpeg info"
            "image/gif": go frame "gif info"
            "application/x-director": goToNetMovie theURL
            "text/html": gotoNetPage theURL
            otherwise: alert "Please choose a different item."
        end case
    else
        go the frame
    end if
```

```
end

// JavaScript syntax
function checkNetOperation(theURL) {
    if (netDone(theURL)) {
        myMimeType = netMIME();
        switch (myMimeType) {
            case "image/jpeg":
                _movie.go("jpeg info");
                break;
            case "image/gif":
                _movie.go("gif info");
                break;
            case "application/x-director":
                goToNetMovie(theURL);
                break;
            case "text/html":
                gotoNetPage(theURL);
                break;
            default:
                alert("Please choose a different item.");
        }
    } else {
        _movie.go(_movie.frame);
    }
}
```

Voir aussi

netDone(), netError(), getNetText(), postNetText, preloadNetThing()

netStatus

Syntaxe

```
netStatus msgString
```

Description

Commande ; affiche la chaîne spécifiée dans la zone d'état de la fenêtre du navigateur.

La commande `netStatus` ne fonctionne pas dans les projections.

Paramètres

`msgString` Requis. Spécifie la chaîne à afficher.

Exemple

Cette instruction place la chaîne Test dans la zone d'état du navigateur web dans lequel l'animation est lue :

```
-- Lingo syntax
on exitFrame
    netStatus "This is a test"
end

// JavaScript syntax
function exitFrame() {
    _movie.netStatus("This is a test");
}
```

netTextResult()

Syntaxe

```
netTextResult(netID)
netTextResult()
```

Description

Fonction ; renvoie le texte obtenu par l'opération réseau spécifiée. Si aucun identifiant réseau n'est spécifié, `netTextResult` renvoie le résultat de la dernière opération réseau.

Si l'opération réseau spécifiée était `getNetText()`, le texte renvoyé correspond à celui du fichier sur le réseau.

Si l'opération réseau spécifiée était `postNetText`, le résultat correspond à la réponse du serveur.

Dès que l'opération suivante démarre, Director efface les résultats de l'opération précédente pour économiser de la mémoire.

Director conserve les résultats des six dernières opérations réseau dans tous les environnements de lecture. Après l'utilisation de la méthode netTextResult() pour récupérer les données, les résultats sont supprimés pour cette requête réseau.

Paramètres

`netID` Facultatif. Spécifie l'identifiant de l'opération réseau contenant le texte à renvoyer.

Exemple

Le gestionnaire suivant utilise les fonctions `netDone` et `netError` pour tester si la dernière opération réseau s'est terminée avec succès. Si l'opération est terminée, le texte renvoyé par `netTextResult` s'affiche dans l'acteur champ Affichage du texte.

```
-- Lingo syntax
global gNetID
on exitFrame
    if (netDone(gNetID) = TRUE) and (netError(gNetID) = "OK") then
        member("Display Text").text = netTextResult()
    end if
end

// JavaScript syntax
global gNetID;
function exitFrame() {
    if (netDone(gNetID) && (netError(gNetID) == "OK")) {
        member("Display Text").text = netTextResult();
    }
}
```

Voir aussi

`netDone()`, `netError()`, `postNetText`

new()

Syntaxe

```
new(type)
new(type, castLib whichCast)
new(type, member whichCastMember of castLib whichCast)
```

```
variableName = new(parentScript arg1, arg2, ...)
new(script parentScriptName, value1, value2, ...)
timeout("name").new(timoutPeriod, #timeoutHandler, {, targetObject})
new(xtra "xtraName")
```

Description

Fonction ; crée un nouvel acteur, objet enfant, objet de temporisation ou instance d'Xtra, et permet d'affecter des valeurs de propriétés individuelles aux objets enfants.

Pour les acteurs, le paramètre *type* définit le type de l'acteur. Les valeurs prédéfinies offertes correspondent aux différents types d'acteurs : `#bitmap`, `#field`, etc. La fonction `new` permet également de créer des types d'acteurs Xtra auxquels vous pouvez attribuer n'importe quel nom.

Vous pouvez également créer un nouvel acteur curseur de couleur en utilisant l'Xtra Custom Cursor. Utilisez `new(#cursor)`, puis définissez les propriétés de l'acteur résultant de manière à pouvoir les utiliser.

Les paramètres facultatifs *whichCastMember* et *whichCast* spécifient la position de l'acteur et la fenêtre Distribution dans laquelle il est créé. Si vous ne spécifiez pas de position, l'acteur est placé dans la première position vide de cette distribution. La fonction `new` renvoie la position de l'acteur.

Lorsque l'argument de la fonction `new` est un script parent, la fonction `new` crée un objet enfant. Le script parent doit contenir un gestionnaire `on new` définissant l'état initial de l'objet enfant ou la valeur de ses propriétés et renvoyant la référence `me` de l'objet enfant.

L'objet enfant possède tous les gestionnaires du script parent. Il utilise les mêmes noms de variables de propriétés que celles qui sont déclarées dans le script parent, mais peut toutefois avoir ses propres valeurs pour ces propriétés.

Puisque l'objet enfant est une valeur, il peut être affecté à des variables, placé dans des listes ou être passé comme paramètre.

Comme avec toute autre variable, vous pouvez utiliser la commande `put` pour afficher des informations sur un objet enfant dans la fenêtre Messages.

Lorsque `new()` est utilisé pour créer un objet de temporisation, la période définit le nombre de millisecondes séparant les événements de temporisation envoyés par l'objet de temporisation. La valeur *#timeoutHandler* est un symbole identifiant le gestionnaire qui est appelé lors de chaque événement de temporisation. La valeur *targetObject* identifie le nom de l'objet enfant contenant la valeur *#gestionnaire*. Si aucune valeur *targetObject* n'est définie, la valeur *#timeoutHandler* est considérée comme figurant dans un script d'animation. La syntaxe de création de temporisation peut varier selon le paramétrage de `scriptExecutionStyle`.

```
-- Lingo syntax when scriptExecutionStyle is set to 9
x = timeout(name).new(period,handler,targetData)

-- Lingo syntax when scriptExecutionStyle is set to 10
x = timeout().new(name, period, handler, targetData)
y = new timeout(name, period, handler, targetData)

// JavaScript syntax
x = new timeout(name, period, function, targetData)
```

Lorsqu'un objet de temporisation est créé, il permet à son *targetObject* de recevoir les événements système `prepareMovie`, `startMovie`, `stopMovie`, `prepareFrame` et `exitFrame`. Pour que cette opération soit possible, l'*targetObject* doit contenir des gestionnaires pour ces événements. Les événements ne doivent pas obligatoirement survenir dans l'ordre pour que le reste de l'animation puisse y accéder.

Remarque : *Un objet de temporisation créé par Lingo peut appeler une fonction de la syntaxe JavaScript, et vice versa.*

Vous pouvez voir un exemple d'utilisation de `new()` dans une animation en consultant les animations Parent Scripts et Read and Write Text du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Exemple

Pour créer un nouvel acteur bitmap dans le premier emplacement disponible de la distribution, utilisez la syntaxe suivante :

```
set newMember = new(#bitmap)
```

Une fois cette ligne exécutée, nouvelActeur contient la référence de l'acteur que vous venez de créer :

```
put newMember
-- (member 1 of castLib 1)
```

Si vous créez des acteurs à l'aide de la syntaxe JavaScript, utilisez la méthode `newMember()` de l'objet Animation. L'instruction suivante crée un nouvel acteur bitmap :

```
var tMem = _movie.newMember(symbol("bitmap"))
```

Le script startMovie suivant crée un acteur Flash à l'aide de la commande new, définit la propriété `linked` de l'acteur nouvellement créé de façon à ce que les éléments de l'acteur soient stockés dans un fichier externe, puis définit la propriété `pathName` de l'acteur sur l'emplacement d'une animation Flash sur le Web :

```
on startMovie
    flashCastMember = new(#flash)
    member(flashCastMember).pathName = "http://www.someURL.com/myFlash.swf"
end
```

Lorsque l'animation démarre, le gestionnaire suivant crée un nouvel acteur curseur animé en couleur et stocke son numéro d'acteur dans la variable `customCursor`. Cette variable sert à définir la propriété `castMemberList` de l'acteur nouvellement créé et à afficher le nouveau curseur.

```
on startmovie
    customCursor = new(#cursor)
    member(customCursor).castMemberList = [member 1, member 2, member 3]
    cursor (member(customCursor))
end
```

Les instructions d'un script parent suivantes contiennent le gestionnaire `on new` servant à créer un objet enfant. Le script parent est un acteur script appelé Oiseau qui contient ces gestionnaires.

```
on new me, nameForBird
    return me
end

on fly me
    put "I am flying"
end
```

La première instruction de l'exemple suivant crée un objet enfant à partir du script de l'exemple précédent et le place dans la variable `myBird`. La seconde instruction fait voler l'oiseau en appelant le gestionnaire `fly` du script parent Oiseau :

```
myBird = script("Bird").new()
myBird.fly()
```

L'instruction suivante utilise un nouveau script parent Oiseau contenant la variable de propriété `speed` :

```
property speed
on new me, initSpeed
    speed = initSpeed
    return me
end
```

```
on fly me
    put "I am flying at " & speed & "mph"
end
```

Les instructions suivantes créent deux objets enfants appelés monOiseau1 et monOiseau2. Les vitesses qui leur sont attribuées initialement sont 15 et 25, respectivement. Lorsque le gestionnaire `fly` est appelé pour chaque objet enfant, la vitesse de ce dernier s'affiche dans la fenêtre Messages.

```
myBird1 = script("Bird").new(15)
myBird2 = script("Bird").new(25)
myBird1.fly()
myBird2.fly()
```

Le message suivant apparaît dans la fenêtre Messages :

```
-- "I am flying at 15 mph"
-- "I am flying at 25 mph"
```

L'instruction suivante crée un objet de temporisation nommé compteurDintervalle qui enverra un événement de temporisation au gestionnaire `on minuteBeep` de l'objet enfant lecteur1 toutes les 60 secondes :

```
timeout("intervalTimer").new(60000, #minuteBeep, playerOne)
```

L'instruction suivante crée un exemple de fonction JavaScript :

```
function sampleTimeout () { trace("hello"); }
```

Vous pouvez utiliser l'instruction suivante partout ailleurs dans votre animation pour créer un objet de temporisation appelant la fonction JavaScript :

```
-- Lingo syntax
gTO = timeout().new("test",50,"sampleTimeout",0)

// JavaScript syntax
_global.gTO = new timeout("test",50,"sampleTimeout",0)
```

Voir aussi

on stepFrame, actorList, ancestor, end, type (acteur), timeout()

newCamera

Syntaxe

```
member(whichCastmember).newCamera(newCameraName)
```

Description

Commande 3D ; crée une caméra dans un acteur.

Paramètres

`newCameraName` Requis. Spécifie le nom de la nouvelle caméra. Le nom de la nouvelle caméra doit être unique dans l'acteur.

Exemple

L'instruction suivante crée une nouvelle caméra appelée Caméra_embarquée.

```
--Lingo
member("3D World").newCamera("in-car camera")

// Javascript
```

```
member("3D World").newCamera("in-car camera") ;
```

newColorRatio

Syntaxe

```
<ColorRatio> newColorRatio(color, ratio, alpha)
```

Description

Fonction globale ; crée un objet ColorRatio permettant de spécifier les proportions de couleur à appliquer à différents niveaux de ratio en cas d'utilisation d'un filtre de rayonnement dégradé ou de biseau dégradé.

Paramètres

Paramètre	Description
couleur	Requis. Objet couleur constitué des composantes rouge, verte et bleue de la couleur.
ratio	Requis. Valeur comprise entre 0 et 255 déterminant le point de départ de la répartition des couleurs par rapport à l'objet ColorRatio en cours dans la liste de couleurs des filtres de dégradé.
alpha	Facultatif. Niveau de transparence (compris entre 0 et 255) de la couleur.

Exemple

L'extrait de code suivant crée un filtre de rayonnement dégradé et lui affecte différents ratios de couleur.

```
--Lingo

on mouseUp me
  fil = filter(#GradientGlowFilter, [#distance:0, #blurX:10, #blurY:10, #strength:1.3,
#inner:0])
  fil.colorList.append(newColorRatio(color(255,255,255),1,0))
  fil.colorList.append(newColorRatio(color(255,0,0),63,255))
  fil.colorList.append(newColorRatio(color(255,255,0),126,25))
  fil.colorList.append(newColorRatio(color(0,204,255),255,255))
  sprite(me.spriteNum).filterList.append(fil)
end

//Java Script

function mouseUp(me)

{
  fil = filter(symbol("GradientGlowFilter"), propList(symbol("distance"),0,
symbol("blurX"), 10, symbol("blurY"), 10, symbol("strength"), 1.3, symbol("inner"), 0));
  fil.colorList.append(newColorRatio(color(255,255,255),1,0));
  fil.colorList.append(newColorRatio(color(255,0,0),63,255));
  fil.colorList.append(newColorRatio(color(255,255,0),126,255));
  fil.colorList.append(newColorRatio(color(0,204,255),255,255));
  sprite(me.spriteNum).filterList.append(fil);
}
```

newCurve()

Syntaxe

```
-- Lingo syntax
```

```
memberObjRef.newCurve(positionInVertexList)

// JavaScript syntax
memberObjRef.newCurve(positionInVertexList);
```

Description

Fonction ; ajoute un symbole `#newCurve` à la liste `vertexList` de *`vectorCastMember`*, qui ajoute une nouvelle forme à la forme vectorielle. Vous pouvez fractionner une forme existante en appelant `newCurve()` avec une position située au milieu d'une série de sommets.

Paramètres

`positionInVertexList` Requis. Spécifie la position dans la liste `vertexList` à laquelle le symbole `#newCurve` est ajouté.

Exemple

Les instructions suivantes ajoutent une nouvelle courbe à l'acteur 2 à la troisième position de la liste des sommets de l'acteur. La seconde ligne de l'exemple remplace le contenu de la courbe 2 par le contenu de la courbe 3.

```
-- Lingo syntax
member(2).newCurve(3)
member(2).curve[2] = member(2).curve[3]

// JavaScript syntax
member(2).newCurve(3);
member(2).curve[2] = member(2).curve[3]
```

Voir aussi

`curve`, `vertexList`

newGroup

Syntaxe

```
member(whichCastmember).newGroup(newGroupName)
```

Description

Commande 3D ; crée un groupe et l'ajoute à la palette des groupes.

Paramètres

`newGroupName` Requis. Spécifie le nom du nouveau groupe. Le nom du nouveau groupe doit être unique dans la palette des groupes.

Exemple

L'instruction suivante crée le groupe gbGroupe2 dans l'acteur Séquence et stocke une référence à ce groupe dans la variable `ng` :

```
-- Lingo
ng = member("Scene").newGroup("gbGroup2")
// Javascript
var ng = member("Scene").newGroup("gbGroup2") ;
```

newLight

Syntaxe

```
member(whichCastmember).newLight(newLightName, #typeIndicator)
```

Description

Commande 3D ; crée une lumière d'un type spécifié et l'ajoute à la palette des lumières.

Paramètres

`newLightName` Requis. Spécifie le nom de la nouvelle lumière. Le nom de la nouvelle lumière doit être unique dans la palette des lumières.

`typeIndicator` Requis. Symbole spécifiant le type de la nouvelle lumière. Les valeurs possibles sont les suivantes :

- `#ambient` est une lumière généralisée dans l'univers 3D.
- `#directional` est une lumière émise à partir d'une direction spécifique.
- `#point` est une source lumineuse telle qu'une ampoule.
- `#spot` est un effet de projecteur.

Exemple

L'instruction suivante crée une lumière dans l'acteur Univers 3D. Il s'agit d'une lumière d'ambiance appelée « lumière ambiante ».

```
-- Lingo
member("3D World").newLight("ambient room light", #ambient)

// Javascript
member("3D World").newLight("ambient room light", symbol("ambient"));
```

newMatrix()

Syntaxe

```
<Matrix> newMatrix(numRows, numColumns, {elementList})
```

Description

Fonction globale ; crée un objet matrice avec le nombre de lignes et de colonnes spécifié. Les index des lignes et des colonnes commencent par la valeur 1.

Remarque : *Pour visualiser les valeurs d'une matrice créée à l'aide de la fonction newmatrix en mode débogage, soumettez une requête portant sur ces valeurs à l'aide de la méthode getval(i,j).*

Paramètres

`numRows` Requis. Spécifie le nombre de lignes de la matrice.

`numColumns` Requis. Spécifie le nombre de colonnes de la matrice.

`elementList` Requis. Liste linéaire de nombres à virgule flottante.

Exemple

L'instruction suivante crée une matrice de 2 lignes et de 3 colonnes, la première ligne contenant [1,2,3] et la seconde [4,5,6].

```
-- Lingo
mat = newMatrix(2, 3, [1,2,3,4,5,6])

// Javascript
mat = newMatrix(2, 3, list(1,2,3,4,5,6));
```

Voir aussi

getVal(), setVal(), numRows(), numColumns(), matrixAddition(), matrixMultiply(), matrixMultiplyScalar(), matrixTranspose()

newMember()

Syntaxe

```
-- Lingo syntax
_movie.newMember(symbol)
_movie.newMember(stringMemberType)

// JavaScript syntax
_movie.newMember(stringMemberType);
```

Description

Méthode d'animation ; crée un acteur et vous permet d'affecter des valeurs de propriétés individuelles aux objets enfants.

Pour les nouveaux acteurs, le paramètre *symbol* ou *stringMemberType* définit le type de l'acteur. Les valeurs prédéfinies offertes correspondent aux différents types d'acteurs : #bitmap, #field, etc. La méthode newMember() permet également de créer des types d'acteurs Xtra auxquels vous pouvez attribuer n'importe quel nom.

Vous pouvez également créer un nouvel acteur curseur de couleur en utilisant l'Xtra Custom Cursor. Utilisez newMember(#cursor), puis définissez les propriétés de l'acteur résultant de manière à pouvoir les utiliser.

Après l'appel de la méthode newMember(), le nouvel acteur est placé à la première position vide de la bibliothèque de distribution.

Vous pouvez voir un exemple d'utilisation de newMember() dans une animation en consultant les animations Parent Scripts et Read and Write Text du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

symbol (Lingo uniquement) Requis. Symbole spécifiant le type du nouvel acteur.

stringMemberType Requis. Chaîne spécifiant le type du nouvel acteur.

Exemple

Les instructions suivantes créent un acteur bitmap et l'affectent à la variable newBitmap.

```
-- Lingo syntax
newBitmap = _movie.newMember(#bitmap) -- using a symbol
newBitmap = _movie.newMember("bitmap") -- using a string

// JavaScript syntax
var newBitmap = _movie.newMember(symbol("bitmap")) ; //using a symbol
var newBitmap = _movie.newMember("bitmap") ; // using a string
```

Voir aussi

Animation, type (acteur)

newMesh

Syntaxe

```
member(whichCastmember).newMesh(name,numFaces, numVertices,
numNormals,numColors,numTextureCoordinates)
```

Description

Commande 3D ; crée une ressource de modèle de maille. Après avoir créé une maille, vous devez au moins définir les valeurs des propriétés `vertexList` et `face[index].vertices` de la nouvelle maille, puis appeler sa commande `build()` pour générer la géométrie.

Paramètres

`meshName` Requis. Spécifie le nom de la nouvelle ressource de modèle de maille.

`numFaces` Requis. Spécifie le nombre total de triangles devant apparaître dans la maille.

`numVertices` Requis. Spécifie le nombre total de sommets utilisés par toutes les faces (triangulaires). Un sommet peut être partagé par plus d'une face.

`numNormals` Facultatif. Spécifie le nombre total de normales. Une normale peut être partagée par plus d'une face. La normale d'un coin de triangle définit la direction extérieure au triangle, ce qui affecte l'éclairage de cet angle. Entrez 0 ou omettez ce paramètre si vous prévoyez d'utiliser la commande `generateNormals()` de la maille pour générer les normales.

`numColors` Facultatif. Spécifie le nombre total de couleurs utilisées par toutes les faces. Une couleur peut être partagée par plus d'une face. Vous pouvez spécifier une couleur pour chaque angle de chaque face. Spécifiez les couleurs pour obtenir un dégradé de couleurs progressif. Entrez 0 ou omettez ce paramètre pour obtenir la couleur blanche par défaut pour chaque coin de face.

`numTextureCoordinates` Facultatif. Spécifie le nombre de coordonnées de texture spécifiées par l'utilisateur utilisées par toutes les faces. Entrez 0 ou omettez ce paramètre pour obtenir les coordonnées de texture par défaut générées par un placage planaire. Pour plus d'informations, consultez la description de `#planar` dans l'entrée `shader.textureWrapMode`. Spécifiez les coordonnées de texture lorsque vous avez besoin de contrôler précisément le placage des textures sur les faces de la maille.

Exemple

L'exemple suivant crée une ressource de modèle de type `#mesh`, en spécifie les propriétés, puis l'utilise pour créer un nouveau modèle. Le processus est décrit dans les explications accompagnant l'exemple suivant :

La **ligne 1** crée une maille contenant 6 faces, composées de 5 sommets et 3 couleurs uniques. Le nombre de normales et de coordonnées de textures n'est pas défini. Les normales sont créées par la commande `generateNormals`.

La **ligne 2** définit les cinq sommets uniques utilisés par les faces de la maille.

La **ligne 3** définit les trois couleurs uniques utilisées par les faces de la maille.

Les **lignes 4 à 9** désignent les sommets à utiliser pour les coins de chaque face de la pyramide. Veuillez noter que l'ordre des sommets suit le sens des aiguilles d'une montre. `GenerateNormals()` repose sur un ordre correspondant au sens des aiguilles d'une montre.

Les **lignes 10 à 15** affectent des couleurs aux coins de chaque face. Les couleurs sont étalées sur les faces en dégradés.

La **ligne 16** crée les normales de Triangle en appelant la commande `generateNormals()`.

La **ligne 17** appelle la commande `build` pour construire la maille.

```
nm = member("Shapes").newMesh("pyramid",6 , 5, 0, 3)
nm.vertexList = [ vector(0,0,0), vector(40,0,0), vector(40,0,40), vector(0,0,40),
vector(20,50,20) ]
nm.colorList = [ rgb(255,0,0), rgb(0,255,0), rgb(0,0,255) ]
nm.face[1].vertices = [ 4,1,2 ]
nm.face[2].vertices = [ 4,2,3 ]
nm.face[3].vertices = [ 5,2,1 ]
nm.face[4].vertices = [ 5,3,2 ]
nm.face[5].vertices = [ 5,4,3 ]
nm.face[6].vertices = [ 5,1,4 ]
nm.face[1].colors = [3,2,3]
nm.face[2].colors = [3,3,2]
nm.face[3].colors = [1,3,2]
nm.face[4].colors = [1,2,3]
nm.face[5].colors = [1,3,2]
nm.face[6].colors = [1,2,3]
nm.generateNormals(#flat)
nm.build()
nm = member("Shapes").newModel("Pyramid1", nm)
```

Voir aussi

newModelResource

newModel

Syntaxe

```
member( whichCastmember ).newModel( newModelName {, whichModelResource } )
```

Description

Commande 3D ; crée un modèle dans l'acteur référencé. La propriété `resource` de tous les nouveaux modèles présente la valeur `VOID` par défaut.

Paramètres

`newModel` Requis. Spécifie le nom du nouveau modèle. Le nom du nouveau modèle doit être unique.

`whichModelResource` Facultatif. Spécifie une ressource de modèle à partir de laquelle le nouveau modèle doit être créé.

Exemple

L'instruction suivante crée le modèle Nouvelle maison dans l'acteur Univers 3D.

```
-- Lingo
member("3D World").newModel("New House")

// Javascript
member("3D World").newModel("New House") ;
```

La ressource de modèle relative au nouveau modèle peut également être définie à l'aide du paramètre facultatif `whichModelResource`.

```
member("3D World").newModel("New House", member("3D World").modelResource("bigBox"))
```

newModelResource

Syntaxe

```
member(whichCastmember).newModelResource(newModelResourceName { ,#type, #facing })
```

Description

Commande 3D ; crée une ressource de modèle, dont le type et la face sont spécifiés en option, puis l'ajoute à la palette des ressources de modèle.

Si vous omettez de spécifier le paramètre *facing* et que vous choisissez la valeur `#box`, `#sphere`, `#particle` ou `#cylinder` pour le paramètre *type*, seules les faces avant sont générées. Si vous spécifiez la valeur `#plane`, les faces avant et arrière sont toutes deux générées. Les ressources de modèle de type `#plane` ont deux mailles générées (une pour chaque côté) et par conséquent, deux matériaux dans la liste `shaderList`.

Le choix de la valeur `#both` pour le paramètre de face crée le double de mailles et donc le double d'entrées de matériau dans la liste `shaderList`. Il y en a 2 pour les plans et les sphères (respectivement, pour l'intérieur et l'extérieur du modèle), 12 pour les cubes (6 à l'extérieur, 6 à l'intérieur) et 6 pour les cylindres (le sommet, la base et les contours extérieur et intérieur).

Paramètres

newModelResourceName Requis. Spécifie le nom de la nouvelle ressource de modèle.

type Facultatif. Spécifie le type de primitive de la nouvelle ressource de modèle. Les valeurs possibles sont les suivantes :

- `#plane`
- `#box`
- `#sphere`
- `#cylinder`
- `#particle`

facing Facultatif. Spécifie la face de la nouvelle ressource de modèle. Les valeurs possibles sont les suivantes :

- `#front`
- `#back`
- `#both`

Exemple

Le gestionnaire suivant crée une boîte. La première ligne du gestionnaire crée une ressource de modèle nommée boîte10. Cette ressource est de type `#box` et est définie pour ne présenter que sa face arrière. Les trois lignes suivantes définissent les dimensions de boîte10 et la dernière ligne crée un nouveau modèle qui utilise boîte10 comme ressource de modèle.

```
on makeBox
    nmr = member("3D").newModelResource("box10", #box, #back)
    nmr.height = 50
    nmr.width = 50
    nmr.length = 50
    aa = member("3D").newModel("gb5", nmr)
end
```

L'instruction suivante crée une ressource de modèle en forme de boîte appelée « boîteAchapeaux4 ».

```
member("Shelf").newModelResource("hatbox4", #box)
```

Voir aussi

primitives

newMotion()

Syntaxe

```
member(whichCastmember).newMotion(name)
```

Description

Commande 3D ; crée un mouvement dans un acteur référencé et renvoie une référence à ce nouveau mouvement. Un nouveau mouvement peut être utilisé pour combiner plusieurs mouvements existants dans la liste des mouvements de l'acteur au moyen de la commande `map()`.

Paramètres

name Requis. Spécifie le nom du nouveau mouvement. Le nom du nouveau mouvement doit être unique dans l'acteur référencé.

Exemple

L'instruction Lingo suivante crée un mouvement dans l'acteur 1 `runWithWave`, qui est utilisé pour combiner les mouvements de course et d'ondulation provenant de la liste des mouvements de l'acteur :

```
runWithWave = member(1).newMotion("runWithWave")
runWithWave.map("run", "pelvisBone")
runWithWave.map("wave", "shoulderBone")
```

newObject()

Syntaxe

```
-- Lingo syntax
spriteObjRef.newObject(objectType {, arg1, arg2 ....})

// JavaScript syntax
spriteObjRef.newObject(objectType {, arg1, arg2 ....});
```

Description

Commande d'image-objet Flash ; crée un objet ActionScript du type spécifié.

La syntaxe suivante crée un objet dans une image-objet Flash :

```
flashSpriteReference.newObject("objectType" {, arg1, arg2 ....})
```

La syntaxe suivante crée un objet global :

```
newObject("objectType" {, arg1, arg2 ....})
```

Remarque : *Si vous n'avez pas importé d'acteur Flash, vous devrez ajouter manuellement l'Xtra Flash Asset à la liste des Xtras de votre animation pour permettre aux commandes Flash globales de fonctionner correctement dans Shockwave Player et dans les projections. Vous ajoutez des Xtras à la liste des Xtras par l'intermédiaire des commandes Modification > Animation > Xtras. Pour plus d'informations sur la gestion des Xtras pour les animations distribuées, reportez-vous aux rubriques Utilisation de Director dans l'Aide de Director.*

Paramètres

objectType Requis. Spécifie le type de l'objet à créer.

arg1, arg2, ... Facultatif. Spécifie les éventuels arguments d'initialisation requis par l'objet. Chaque argument doit être séparé des autres par une virgule.

Exemple

L'instruction Lingo suivante définit la variable `tLocalConObject` sur une référence à un nouvel objet `LocalConnection` dans l'animation Flash au niveau de l'image-objet 3 :

```
-- Lingo syntax
tLocalConObject = sprite(3).newObject("LocalConnection")

// JavaScript syntax
var tLocalConObject = sprite(3).newObject("LocalConnection");
```

L'instruction Lingo suivante définit la variable `tArrayObject` sur une référence à un nouvel objet tableau dans l'animation Flash au niveau de l'image-objet 3. Le tableau contient les 3 nombres entiers 23, 34 et 19.

```
-- Lingo syntax
tArrayObject = sprite(3).newObject("Array",23,34,19)

// JavaScript syntax
var tArrayObject = sprite(3).newObject("Array",23,34,19);
```

Voir aussi

`setCallback()`, `clearAsObjects()`

newShader

Syntaxe

```
member(whichCastmember).newShader(newShaderName, #shaderType)
```

Description

Commande 3D ; crée un matériau d'un type spécifié dans une liste des matériaux de l'acteur référencé et renvoie une référence à ce nouveau matériau.

Chaque type de matériau comporte un groupe de propriétés qui lui sont spécifiques, en complément des propriétés de matériau `#standard`. Toutefois, même si vous pouvez attribuer n'importe quelle propriété de matériau `#standard` à un matériau d'un autre type, il est possible que la propriété n'ait aucun effet visible. Ceci se produit lorsque la propriété `#standard` appliquée remplace la nature du type de matériau. Par exemple, la propriété de matériau standard `diffuseLightMap` est ignorée par les matériaux du type `#engraver`, `#newsprint` et `#painter`.

Paramètres

newShaderName Requis. Spécifie le nom du nouveau matériau. Le nom du nouveau matériau doit être unique dans la liste de matériaux.

shaderType Requis. Symbole déterminant le style dans lequel le matériau est appliqué. Les valeurs possibles sont les suivantes :

- Les matériaux `#standard` sont photoréalistes et comportent les propriétés suivantes : `ambient`, `blend`, `blendConstant`, `blendConstantList`, `blendFunction`, `blendFunctionList`, `blendSource`, `blendSourceList`, `diffuse`, `diffuseLightMap`, `emissive`, `flat`, `glossMap`, `ilk`, `name`, `region`, `renderStyle`, `silhouettes`, `specular`, `specularLightMap`, `texture`, `textureMode`, `textureModeList`, `textureRepeat`, `textureRepeatList`, `textureTransform`, `textureTransformList`, `transparent`, `useDiffuseWithTexture`, `wrapTransform` et `wrapTransformList`.

- Les matériaux `#painter` sont estompés, présentent l'aspect d'une peinture et, en complément de toutes les propriétés `#standard`, possèdent les propriétés suivantes : `colorSteps`, `hilightPercentage`, `hilightStrength`, `name`, `shadowPercentage`, `shadowStrength` et `style`.

- Les matériaux `#engraver` sont striés, présentent l'aspect d'une gravure et, en complément de toutes les propriétés #standard, possèdent les propriétés suivantes : `brightness`, `density`, `name` et `rotation`.

- Les matériaux `#newsprint` sont en pointillés, présentent l'aspect d'une reproduction de journal et, en complément de toutes les propriétés #standard, possèdent les propriétés suivantes : `brightness`, `density` et `name`.

Exemple

L'instruction suivante crée le matériau `#painter` nouveauPeintre :

```
-- Lingo
newPainter = member("3D World").newShader("newPainter",#painter)

// Javascript
var newPainter = member("3D World").newShader("newPainter",symbol("painter")) ;
```

Voir aussi

`shadowPercentage`

newTexture

Syntaxe

```
member(whichCastmember).newTexture(newTextureName {,#typeIndicator,
sourceObjectReference})
```

Description

Commande 3D ; crée une texture dans la palette des textures de l'acteur référencé et renvoie une référence à cette nouvelle texture. Les textures d'acteur ne fonctionnent que lorsque vous spécifiez l'acteur dans le constructeur `newTexture`.

Paramètres

`newTextureName` Requis. Spécifie le nom de la nouvelle texture. Le nom de la nouvelle texture doit être unique dans la palette des textures de l'acteur référencé.

`typeIndicator` Facultatif. Spécifie le type de la nouvelle texture. Si ce paramètre est omis, la nouvelle texture est créée sans type spécifique. Les valeurs possibles sont les suivantes :

- `#fromCastMember` (acteur)
- `#fromImageObject` (objet image Lingo)

sourceObjectReference Facultatif. Spécifie une référence à l'acteur source ou à l'objet image Lingo. Si ce paramètre est omis, la nouvelle texture est créée sans être associée à une source spécifique. Le paramètre *sourceObjectReference* doit faire référence à un acteur si *typeIndicator* présente la valeur `#fromCastMember` et à un objet image Lingo si *typeIndicator* reçoit la valeur `#fromImageObject`.

Exemple

La première ligne de l'instruction suivante crée une texture nommée Gazon 02 à partir de l'acteur 5 de la bibliothèque de distribution 1. La seconde ligne crée une texture vierge appelée Vierge.

```
-- Lingo
member("3D World").newTexture("Grass 02",#fromCastMember,member(5,1))
member("3D World").newTexture("Blank")

// Javascript
member("3D World").newTexture("Grass 02",symbol("fromCastMember"),member(5,1)) ;
member("3D World").newTexture("Blank") ;
```

normalize

Syntaxe

```
normalize(vector)
vector.normalize()
```

Description

Commande 3D ; normalise un vecteur en divisant les composants *x*, *y* et *z* par la magnitude du vecteur. Les vecteurs normalisés ont toujours une magnitude de 1.

Paramètres

Aucune.

Exemple

L'instruction suivante indique la valeur du vecteur monVecteur avant et après la normalisation :

```
-- Lingo
MyVec = vector(-209.9019, 1737.5126, 0.0000)
MyVec.normalize()
put MyVec
-- vector(-0.1199, 0.9928, 0.0000)
put MyVec.magnitude
-- 1.0000

// Javascript
MyVec = vector(-209.9019, 1737.5126, 0.0000);
MyVec.normalize();
trace(MyVec);
// vector(-0.1199, 0.9928, 0.0000)
trace(MyVec.magnitude);
// 1.0000
```

L'instruction suivante indique la valeur du vecteur ceVecteur avant et après la normalisation.

```
-- Lingo
ThisVector = vector(-50.0000, 0.0000, 0.0000)
normalize(ThisVector)
put ThisVector
```

```
-- vector(-1.0000, 0.0000, 0.0000)

// Javascript
ThisVector = vector(-50.0000, 0.0000, 0.0000);
normalize(ThisVector);
trace(ThisVector);
// vector(-1.0000, 0.0000, 0.0000)
```

Voir aussi

getNormalized, randomVector(), magnitude

nothing

Syntaxe

```
nothing
```

Description

Commande ; n'a aucun effet. Cette commande se révèle utile pour clarifier une instruction `if...then`. Une instruction imbriquée `if...then...else` ne contenant aucune commande explicite pour la clause `else` peut nécessiter l'utilisation de `else nothing` afin d'empêcher Lingo d'interpréter la clause `else` comme faisant partie de la clause `if` qui la précède.

Paramètres

Aucune.

Exemple

L'instruction imbriquée `if...then...else` du gestionnaire suivant utilise la commande `nothing` à la suite de la clause `else` de l'instruction :

```
-- Lingo syntax
on mouseDown
    if the clickOn = 1 then
        if sprite(1).moveableSprite = TRUE then member("Notice").text = "Drag the ball"
        else nothing
    else member("Notice").text = "Click again"
    end if
end

// JavaScript syntax
function mouseDown() {
    if (_mouse.clickOn == 1) {
        if (sprite(1).moveableSprite) {
            member("Notice").text = "Drag the ball";
        } else {
            // do nothing
        }
    } else {
        member("Notice").text = "Click again";
    }
}
```

Le gestionnaire suivant demande à l'animation de ne rien faire tant que l'utilisateur appuie sur le bouton de la souris :

```
-- Lingo syntax
on mouseDown
    repeat while the stillDown
```

```
        nothing
    end repeat
end mouseDown

// JavaScript syntax
function mouseDown() {
    do {
        // do nothing
    } while _mouse.stillDown;
}
```

Voir aussi

if

nudge()

Syntaxe

```
-- Lingo syntax
spriteObjRef.nudge(#direction)

// JavaScript syntax
spriteObjRef.nudge(#direction);
```

Description

Commande QuickTime VR ; déplace la perspective de l'image-objet QuickTime VR spécifiée dans une direction spécifique.

Une poussée vers la droite entraîne un déplacement de l'image de l'image-objet vers la gauche. La commande `nudge` ne renvoie aucune valeur.

Paramètres

`direction` Requis. Spécifie la direction dans laquelle la perspective doit être déplacée. Les valeurs possibles sont les suivantes :

- `#down`
- `#downLeft`
- `#downRight`
- `#left`
- `#right`
- `#up`
- `#upLeft`
- `#upRight`

Exemple

Le gestionnaire suivant entraîne le déplacement vers la gauche de la perspective de l'image-objet QuickTime VR pendant que le pointeur de la souris est positionné sur l'image-objet.

```
-- Lingo syntax
on mouseDown me
    repeat while the stillDown
```

```
        sprite(1).nudge(#left)
    end repeat
end

// JavaScript syntax
function mouseDown() {
    do {
        sprite(1).nudge(symbol("left"));
    } while _mouse.stillDown;
}
```

numColumns()

Syntaxe

```
<matrixObject>.numColumns
```

Description

Propriété de matrice ; récupère le nombre de colonnes de la matrice. Vous ne pouvez qu'obtenir le nombre de colonnes ; ce dernier n'est pas modifiable.

Exemple

L'extrait de code suivant crée une matrice et indique son nombre de lignes et de colonnes.

```
--Lingo

mat1 = newMatrix(2, 3, [1,2,3,4,5,6])
put(mat1.numRows)
put(mat1.numColumns)
-- 2
-- 3

//Java Script

mat1 = newMatrix(2, 3, list(1,2,3,4,5,6));
put(mat1.numRows);
put(mat1.numColumns);

// 2

// 3
```

Voir aussi

getVal(), setVal(), numRows(), matrixAddition(), matrixMultiply(), matrixMultiplyScalar(), matrixTranspose(), newMatrix()

numRows()

Syntaxe

```
<matrixObject>.numRows
```

Description

Propriété de matrice ; récupère le nombre de lignes de la matrice. Vous ne pouvez qu'obtenir le nombre de lignes ; ce dernier n'est pas modifiable.

Exemple

L'extrait de code suivant crée une matrice et indique son nombre de lignes et de colonnes.

```
-- Lingo
mat1 = newMatrix(2, 3, [1,2,3,4,5,6])
put(mat1.numRows)
put(mat1.numColumns)

-- 2
-- 3

// Javascript
mat1 = newMatrix(2, 3, list(1,2,3,4,5,6));
put(mat1.numRows);
put(mat1.numColumns);

// 2
// 3
```

Voir aussi

getVal(), setVal(), numColumns(), matrixAddition(), matrixMultiply(), matrixMultiplyScalar(), matrixTranspose(), newMatrix()

numToChar()

Syntaxe

```
numToChar(integerExpression)
```

Description

Fonction ; affiche une chaîne contenant le caractère dont le code ASCII est la valeur d'une expression spécifiée. Elle est utile pour interpréter les données de sources externes qui sont présentées sous forme de nombres plutôt que sous forme de caractères.

Les valeurs ASCII allant jusqu'à 127 sont standard sur tous les ordinateurs. Les valeurs supérieures ou égales à 128 font référence à des caractères différents sur différents ordinateurs.

Paramètres

integerExpression Requis. Spécifie la valeur ASCII auquel correspond le caractère renvoyé.

Exemple

Le gestionnaire suivant supprime tous les caractères non alphabétiques d'une chaîne quelconque et ne renvoie que des majuscules :

```
-- Lingo syntax
on ForceUppercase input
    output = EMPTY
    num = length(input)
    repeat with i = 1 to num
        theASCII = charToNum(input.char[i])
        if theASCII = min(max(96, theASCII), 123) then
            theASCII = theASCII - 32
            if theASCII = min(max(63, theASCII), 91) then
                put numToChar(theASCII) after output
            end if
        end if
    end repeat
```

```
    return output
end

// JavaScript syntax
function ForceUpperCase(input) {
    output = "";
    num = input.length;
    for (i=1;i<=num;i++) {
        theASCII = charToNum (input.getProp("char",i);
        if (theASCII == list(list(96, theASCII).max(), 123).min()) {
            theASCII = theASCII - 32;
            if (theASCII == list(list(63, theASCII).max(), 91).min()) {
                output = output + numToChar (theASCII);
            }
        }
    }
    return output;
}
```

Voir aussi

charToNum()

objectP()

Syntaxe

objectP(expression)

Description

Fonction ; indique si une expression spécifiée est un objet créé par un script parent, un Xtra ou une fenêtre (TRUE) ou non (FALSE).

Le caractère *P* de objectP signifie *prédicat*.

Il est recommandé d'utiliser objectP pour déterminer les éléments en cours d'utilisation lors de la création d'objets par des scripts parents ou des instances d'Xtra.

Vous pouvez voir un exemple d'utilisation de la méthode objectP() dans une animation en consultant l'animation Read and Write Text du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

expression Requis. Spécifie l'expression à tester.

Exemple

L'instruction Lingo suivante vérifie si un objet est affecté à la variable globale gDataBase et, dans la négative, lui en affecte un. Cette vérification s'utilise généralement lorsque vous effectuez des initialisations au début d'une animation ou d'une section que vous ne souhaitez pas répéter.

```
-- Lingo syntax
if objectP(gDataBase) then
 nothing
else
 gDataBase = script("Database Controller").new()
end if

// JavaScript syntax
```

```
if (objectP(gDataBase)) {
 // do nothing
} else {
 gDataBase = script("Database Controller").new();
}
```

Voir aussi

floatP(), ilk(), integerP(), stringP(), symbolP()

offset() (fonction de chaîne)

Syntaxe

```
offset(stringExpression1, stringExpression2)
```

Description

Fonction ; renvoie un nombre entier indiquant la position du premier caractère d'une chaîne dans une autre chaîne. Cette fonction renvoie la valeur 0 si la première chaîne est introuvable dans la seconde chaîne. Lingo considère les espaces comme des caractères dans les deux chaînes.

Sur le Mac, la comparaison des chaînes ne tient pas compte des majuscules ni des accents. Par exemple, Lingo considère les caractères *a* et *Å* comme identiques sur le Mac.

Paramètres

stringExpression1 Requis. Spécifie la sous-chaîne à rechercher dans *stringExpression2*.

stringExpression2 Requis. Spécifie la chaîne contenant la sous-chaîne *stringExpression1*.

Exemple

L'instruction suivante affiche dans la fenêtre Messages la position de départ de la chaîne « media » dans la chaîne « kleptomedia » :

```
put offset("media","kleptomedia")
```

Le résultat est 7.

L'instruction suivante affiche dans la fenêtre Messages la position de départ de la chaîne « Micro » dans la chaîne « Adobe » :

```
put offset("Micro", "Adobe")
```

Le résultat est 0 car la chaîne « Adobe » ne contient pas la chaîne « Micro ».

Le gestionnaire suivant recherche toutes les instances de la chaîne représentée par `stringToFind` dans la chaîne représentée par `input`, puis les remplace par la chaîne représentée par `stringToInsert` :

```
-- Lingo syntax
on SearchAndReplace input, stringToFind, stringToInsert
    output = ""
    findLen = stringToFind.length - 1
    repeat while input contains stringToFind
        currOffset = offset(stringToFind, input)
        output = output & input.char [1..currOffset]
        delete the last char of output
        output = output & stringToInsert
        delete input.char [1.. (currOffset + findLen)]
    end repeat
```

```
    set output = output & input
    return output
end

// JavaScript syntax
function SearchAndReplace(input, stringToFind, stringToInsert) {
    output = "";
    findLen = stringToFind.length - 1;
    do {
        currOffset = offset(stringToFind, input);
        output = output + input.char[0..currOffset];
        output = output.substr(0,output.length-2);
        output = output + stringToInsert;
        input = input.substr(currOffset+findLen,input.length);
    } while (input.indexOf(stringToFind) >= 0);
    output = output + input;
    return output;
}
```

Voir aussi

chars(), length(), contains, starts

offset() (fonction de rectangle)

Syntaxe

```
rectangle.offset(horizontalChange, verticalChange)
offset (rectangle, horizontalChange, verticalChange)
```

Description

Fonction ; produit un rectangle décalé par rapport au rectangle spécifié par *rectangle*.

Paramètres

horizontalChange Requis. Spécifie le décalage horizontal en pixels. Lorsque *changementHorizontal* est supérieur à 0, le décalage se produit vers la droite de la scène ; lorsque *changementHorizontal* est inférieur à 0, le décalage se produit vers la gauche de la scène.

verticalChange Requis. Spécifie le décalage vertical en pixels. Lorsque *verticalChange* est supérieur à 0, le décalage se produit vers le haut de la scène ; lorsque *verticalChange* est inférieur à 0, le décalage se produit vers le bas de la scène.

Exemple

Le gestionnaire suivant déplace l'image-objet 1 de cinq pixels vers la droite et de cinq pixels vers le bas.

```
-- Lingo syntax
on diagonalMove
    newRect=sprite(1).rect.offset(5, 5)
    sprite(1).rect=newRect
end

// JavaScript syntax
function diagonalMove() {
    newRect = sprite(1).rect.offset(5,5);
    sprite(1).rect = newRect;
}
```

open() (lecteur)

Syntaxe

```
-- Lingo syntax
_player.open({stringDocPath,} stringAppPath)

// JavaScript syntax
_player.open({stringDocPath,} stringAppPath);
```

Description

Méthode de lecteur ; ouvre une application spécifiée et, en option, ouvre un fichier spécifique en même temps que l'application.

Si la valeur du paramètre *stringDocPath* ou *stringAppPath* se trouve dans un autre dossier que l'animation en cours, vous devez spécifier le chemin d'accès complet du ou des fichiers.

L'ordinateur doit avoir assez de mémoire pour exécuter simultanément Director et d'autres applications.

Cette méthode constitue une méthode très simple d'ouverture d'une application ou d'un document au sein d'une application. Pour d'autres contrôles, consultez les options disponibles dans les Xtras fournis par d'autres développeurs.

Paramètres

stringDocPath Facultatif. Chaîne spécifiant le document à ouvrir au moment de l'ouverture de l'application spécifiée par *stringDocPath*.

stringDocPath Requis. Chaîne spécifiant le chemin d'accès de l'application à ouvrir.

Exemple

L'instruction suivante ouvre l'application TextEdit qui se trouve dans le dossier Applications sur le disque dur (Mac), ainsi que le document Synopsis :

```
-- Lingo syntax
_player.open("Storyboards", "HD:Applications:TextEdit")

// JavaScript syntax
_player.open("Storyboards", "HD:Applications:TextEdit");
```

Voir aussi

Lecteur

open() (fenêtre)

Syntaxe

```
-- Lingo syntax
windowObjRef.open()

// JavaScript syntax
windowObjRef.open();
```

Description

Méthode de fenêtre ; ouvre une fenêtre et la positionne devant toutes les autres fenêtres.

Si aucune animation n'est affectée à la fenêtre dans laquelle la méthode `open()` est appelée, la boîte de dialogue d'ouverture de fichier s'affiche.

Si la référence à l'objet fenêtre *`windowObjRef`* est remplacée par le nom de fichier d'une animation, la fenêtre utilise le nom de fichier comme nom de fenêtre. Toutefois, une animation doit alors être affectée à la fenêtre à l'aide de la propriété `fileName` de la fenêtre.

Si la référence à l'objet fenêtre *`windowObjRef`* est remplacée par un nom de fenêtre, la fenêtre prend ce nom. Toutefois, une animation doit alors être affectée à la fenêtre à l'aide de la propriété `fileName` de la fenêtre.

Pour ouvrir une fenêtre utilisant une animation à partir d'une URL, commencez par utiliser `downloadNetThing()` pour télécharger le fichier de l'animation sur un disque local, puis utilisez ce fichier depuis ce disque. Cette procédure minimise les problèmes liés à l'attente du téléchargement de l'animation.

En cas d'utilisation d'une animation locale, utilisez `preloadMovie()` pour charger au moins la première image de l'animation avant l'appel de la méthode `open()`. Cette procédure réduit les risques de décalages au niveau du chargement des animations.

L'ouverture d'une animation dans une fenêtre n'est pas encore supportée pour la lecture dans un navigateur web.

Paramètres

Aucune.

Exemple

L'instruction suivante ouvre la fenêtre Tableau de commande et la place au premier plan :

```
-- Lingo syntax
window("Control Panel").open()

// JavaScript syntax
window("Control Panel").open();
```

Voir aussi

`close()`, `downloadNetThing`, `fileName (fenêtre)`, `preLoadMovie()`, `Fenêtre`

openFile()

Syntaxe

```
-- Lingo syntax
fileioObjRef.openFile(stringFileName, intMode)

// JavaScript syntax
fileioObjRef.openFile(stringFileName, intMode)
```

Description

Méthode FileIO ; ouvre un fichier spécifié dans un mode spécifique.

Paramètres

`stringFileName` Requis. Chaîne spécifiant le chemin d'accès complet et le nom du fichier à ouvrir.

`intMode` Requis. Nombre entier spécifiant le mode du fichier. Les valeurs possibles sont :

- 0 : lecture/écriture.
- 1 : lecture seule.

- 2 : possibilité d'écriture.

Exemple

L'instruction suivante ouvre le fichier c:\xtra.txt avec l'autorisation Lecture/écriture.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
```

Voir aussi

`Fileio`

openXlib

Syntaxe

```
openXlib whichFile
```

Description

Commande ; ouvre un fichier Xlibrary spécifié.

Il est recommandé de fermer tout fichier ouvert dès que vous n'en avez plus besoin. La commande `openXlib` n'a aucun effet sur un fichier ouvert.

La commande `openXlib` ne prend pas en charge les URL comme références de fichier.

Les fichiers Xlibrary contiennent des Xtras. A la différence de `openResFile`, `openXlib` met ces Xtras à la disposition de Director.

Lorsque vous ouvrez un Xtra de programmation à l'aide de la méthode `openXlib`, vous devez utiliser `closeXlib` pour le fermer une fois que Director a fini de l'utiliser.

Sous Windows, l'extension .dll est facultative.

Remarque : *Cette commande n'est pas prise en charge dans Shockwave Player.*

Paramètres

`whichFile` Requis. Spécifie le fichier Xlibrary à ouvrir. Si le fichier ne figure pas dans le dossier de l'animation en cours, *`whichFile`* doit inclure le chemin d'accès.

Exemple

L'instruction suivante ouvre le fichier Xlibrary Vidéodisque :

```
openXlib "Video Disc Xlibrary"
```

L'instruction suivante ouvre le fichier Xlibrary Xtras, situé dans un dossier différent de celui de l'animation en cours :

```
openXlib "My Drive:New Stuff:Transporter Xtras"
```

Voir aussi

`closeXlib`, `Interface()`

param()

Syntaxe

```
param(parameterPosition)
```

Description

Fonction ; renvoie la valeur d'un paramètre transmis à un gestionnaire.

Cette fonction peut s'utiliser pour déterminer le type d'un paramètre particulier afin d'éviter les erreurs dans un gestionnaire.

Paramètres

parameterPosition Requis. Spécifie la position du paramètre dans les arguments transmis à un gestionnaire.

Exemple

Le gestionnaire suivant accepte un nombre quelconque d'arguments, fait le total du nombre transmis en paramètres, puis renvoie la somme :

```
--Lingo syntax
on AddNumbers
    sum = 0
    repeat with currentParamNum = 1 to the paramCount
        sum = sum + param(currentParamNum)
    end repeat
    return sum
end

// JavaScript syntax
function AddNumbers() {
    sum = 0;
    for (currentParamNum=0;currentParamNum<arguments.length;currentParamNum++){
        sum = sum + arguments[currentParamNum];
    }
    return sum;
}
```

Vous l'utilisez en transmettant les valeurs que vous souhaitez ajouter :

```
put AddNumbers(3, 4, 5, 6)
-- 18
put AddNumbers(5, 5)
-- 10
```

Voir aussi

getAt, param(), paramCount(), return (mot-clé)

paramCount()

Syntaxe

```
the paramCount
```

Description

Fonction ; indique le nombre de paramètres transmis au gestionnaire actuel.

Paramètres
Aucune.

Exemple
L'instruction suivante définit la variable `counter` sur le nombre de paramètres transmis au gestionnaire en cours :

```
set counter = the paramCount
```

parseString()

Syntaxe
```
parserObject.parseString(stringToParse)
```

Description
Fonction ; utilisée pour analyser un document XML déjà disponible pour une animation Director. Le premier paramètre correspond à la variable contenant l'objet analyseur. La valeur renvoyée est `<VOID>` en cas de succès de l'opération ou un code d'erreur en cas d'échec. L'échec est généralement dû à un problème au niveau de la syntaxe ou de la structure XML. Une fois l'opération terminée, l'objet d'analyse contient les données XML analysées.

L'analyse du code XML d'une URL requiert l'utilisation de la méthode `parseURL()`.

Paramètres
`stringToParse` Requis. Spécifie la chaîne de données XML à analyser.

Exemple
L'instruction suivante analyse les données XML de l'acteur texte texteXML. Une fois l'opération terminée, la variable `gParserObject` contient les données XML analysées.

```
-- Lingo
errorCode = gParserObject.parseString(member("XMLtext").text)

// Javascript
errorCode = gParserObject.parseString(member("XMLtext").text);
```

Voir aussi
`getError() (XML)`, `parseURL()`

parseURL()

Syntaxe
```
parserObject.parseURL(URLstring {,#handlerToCallOnCompletion} {, objectContainingHandler})
```

Description
Fonction ; analyse un document XML résidant à une adresse Internet externe. Le premier paramètre correspond à l'objet analyseur contenant une instance de l'Xtra XML Parser.

Cette fonction renvoie un résultat immédiat et l'adresse URL complète peut donc ne pas être encore analysée. Il est important d'utiliser la fonction `doneParsing()` avec `parseURL()` afin d'être averti de la fin de l'opération d'analyse.

Cette opération étant asynchrone (et pouvant prendre un certain temps), vous pouvez utiliser des paramètres facultatifs permettant d'appeler un gestionnaire spécifique à la fin de l'opération.

La valeur renvoyée est VOID en cas de succès de l'opération ou un code d'erreur en cas d'échec.

L'analyse du code XML local requiert l'utilisation de la méthode `parseString()`.

Paramètres

`URLstring` Requis. Spécifie l'URL à laquelle les données XML résident.

`handlerToCallOnCompletion` Facultatif. Spécifie le nom du gestionnaire à exécuter une fois l'URL complètement analysée.

`objectContainingHandler` Facultatif. Spécifie le nom de l'objet script contenant le gestionnaire *`handlerToCallOnCompletion`*. Si ce paramètre est omis, l'application suppose qu'il s'agit d'un gestionnaire d'animation.

Exemple

L'instruction suivante analyse le fichier exemple.xml qui se trouve sous www.monEntreprise.fr. Utilisez `doneParsing()` pour être averti de la fin de l'opération d'analyse.

```
--Lingo syntax
errorCode = gParserObject.parseURL("http://www.MyCompany.com/sample.xml")

// JavaScript syntax
errorCode = _global.gParserObject.parseURL("http://- www.MyCompany.com/sample.xml");
```

Remarque : *L'exemple suivant suppose qu'une instance de l'Xtra a déjà été créée et qu'une référence à cette dernière a été stockée dans la variable globale appelée gObjetDanalyse.*

L'instruction Lingo suivante analyse le fichier exemple.xml et appelle le gestionnaire `on parseDone`. Aucun objet script n'étant donné avec la fonction `doneParsing()`, le gestionnaire `on parseDone` est supposé se trouver dans un script de l'animation.

```
errorCode = gParserObject.parseURL("http://www.MyCompany.com/sample.xml", #parseDone)
```

Le script de l'animation contient le gestionnaire on parseDone :

```
on parseDone
    global gParserObject
    if voidP(gParserObject.getError()) then
        put "Successful parse"
    else
        put "Parse error:"
        put " " & gParserObject.getError()
    end if
end
```

La syntaxe JavaScript suivante analyse le fichier exemple.xml et appelle la fonction `parseDone`. Aucun objet script n'étant donné avec la fonction `doneParsing()`, la fonction on `parseDone` est supposée se trouver dans un script de l'animation.

```
errorCode = _global.gParserObject.parseURL("http://- www.MyCompany.com/sample.xml",
symbol("parseDone"));
```

Remarque : *L'exemple suivant suppose qu'une instance de l'Xtra a déjà été créée et qu'une référence à cette dernière a été stockée dans la variable globale appelée gObjetDanalyse.*

Le script de l'animation contient le gestionnaire `on parseDone` :

```
// JavaScript syntax
function parseDone () {
    if (_global.gParserObject.getError() == undefined) {
        trace("successful parse");
    } else {
        trace("Parse error:");
        trace(" " + _global.gParserObject.getError());
    }
}
```

L'instruction Lingo suivante analyse le document exemple.xml qui se trouve sur www.monEntreprise.fr et appelle le gestionnaire `on parseDone` dans l'objet script `testObject`, qui est un enfant du script parent scriptTest :

```
parserObject = new(xtra "XMLParser")
testObject = new(script "TestScript", parserObject)
errorCode = gParserObject.parseURL("http://www.MyCompany.com/sample.xml", #parseDone,
testObject)
```

Le script parent scriptTest est le suivant :

```
property myParserObject

on new me, parserObject
    myParserObject = parserObject
end

on parseDone me
    if voidP(myParserObject.getError()) then
        put "Successful parse"
    else
        put "Parse error:"
        put " " & myParserObject.getError()
    end if
end
```

La syntaxe JavaScript suivante analyse le document exemple.xml qui se trouve sur www.monEntreprise.fr et appelle la fonction analyseTerminée dans l'objet objetTest, qui est une instance de la classe scriptTest définie :

```
parserObject = new xtra("XMLParser");
testObject = new TestScript(parserObject);
errorCode = parserObject .parseURL("http://www.MyCompany.com/sam- ple.xml",
symbol("parseDone"), testObject)
```

La définition de la classe scriptTest est la suivante :

```
TestScript = function (aParser) {
    this.myParserObject = aParser;
}
TestScript.prototype.parseDone = function () {
    if (this.myParserObject.getError() == undefined) {
        trace("successful parse");
    } else {
        trace("Parse error:");
        trace(" " + this.myParserObject.getError());
    }
}
```

Voir aussi

`getError() (XML)`, `parseString()`

pass

Syntaxe

```
pass
```

Description

Commande ; transmet un message d'événement à la position suivante dans la hiérarchie des messages et déclenche l'exécution de plusieurs gestionnaires pour un événement donné.

La commande `pass` passe à l'emplacement suivant dès qu'elle est exécutée. Aucun code Lingo suivant la commande `pass` dans le gestionnaire n'est exécuté.

Par défaut, un message d'événement s'arrête à la première position contenant un gestionnaire pour l'événement, généralement au niveau de l'image-objet.

L'inclusion de la commande `pass` dans un gestionnaire entraîne la transmission de l'événement à d'autres objets dans la hiérarchie, même si le gestionnaire pourrait autrement intercepter l'événement.

Paramètres

Aucune.

Exemple

Le gestionnaire suivant détermine les touches sélectionnées et en autorise la transmission à l'image-objet texte modifiable s'il s'agit de caractères valides :

```
-- Lingo syntax
on keyDown me
    legalCharacters = "1234567890"
    if legalCharacters contains the key then
        pass
    else
        beep
    end if
end

// JavaScript syntax
function keyDown() {
    legalCharacters = "1234567890";
    if (legalCharacters.indexOf(_key.key) >= 0) {
        pass();
    } else {
        _sound.beep();
    }
}
```

Voir aussi

`stopEvent()`

pasteClipBoardInto()

Syntaxe

```
-- Lingo syntax
memberObjRef.pasteClipBoardInto()
```

```
// JavaScript syntax
memberObjRef.pasteClipBoardInto();
```

Description

Méthode d'acteur ; colle le contenu du Presse-papiers dans un acteur spécifié et efface l'acteur existant.

Vous pouvez coller tout élément dont le format est utilisable par Director en tant qu'acteur.

Lorsque vous copiez une chaîne à partir d'une autre application, le format de la chaîne n'est pas conservé.

Cette méthode se révèle utile pour copier dans la fenêtre Distribution des objets provenant d'autres animations et applications. Les acteurs copiés devant être conservés en RAM, évitez d'utiliser cette commande pendant la lecture dans les situations où la mémoire arrive à épuisement.

Lorsque vous utilisez cette méthode dans Shockwave Player ou dans l'environnement auteur et les projections dont la propriété `safePlayer` présente la valeur `TRUE`, une boîte de dialogue d'avertissement s'affiche pour permettre à l'utilisateur d'annuler l'opération de collage.

Paramètres

Aucune.

Exemple

L'instruction suivante colle le contenu du Presse-papiers dans l'acteur bitmap Temple :

```
-- Lingo syntax
member("shrine").pasteClipBoardInto()

// JavaScript syntax
member("shrine").pasteClipBoardInto();
```

Voir aussi

Acteur, safePlayer

pause() (DVD)

Syntaxe

```
-- Lingo syntax
dvdObjRef.pause()

// JavaScript syntax
dvdObjRef.pause();
```

Description

Méthode de DVD ; met la lecture en pause.

Paramètres

Aucune.

Exemple

L'instruction suivante met la lecture en pause :

```
-- Lingo syntax
member(1).pause()
```

```
// JavaScript syntax
member(1).pause();
```

Voir aussi

DVD

pause() (piste audio)

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.pause()

// JavaScript syntax
soundChannelObjRef.pause();
```

Description

Méthode de piste audio ; suspend la lecture du son en cours dans une piste audio.

L'exécution ultérieure d'une méthode `play()` entraîne la reprise de la lecture.

Paramètres

Aucune.

Exemple

L'instruction suivante met en pause la lecture de l'acteur son lu dans la piste audio 1 :

```
-- Lingo syntax
sound(1).pause()

// JavaScript syntax
sound(1).pause();
```

Voir aussi

`breakLoop()`, `play() (piste audio)`, `playNext() (piste audio)`, `queue()`, `rewind() (piste audio)`, `Piste audio`, `stop() (piste audio)`

pause() (3D)

Syntaxe

```
member(whichCastmember).model(whichModel).bonesPlayer.pause()
member(whichCastmember).model(whichModel).keyframePlayer.pause()
```

Description

Commande 3D de modificateur `#keyframePlayer` et `#bonesPlayer` ; suspend le mouvement en cours d'exécution par le modèle. Utilisez la commande `play()` pour reprendre le mouvement.

Lorsque le mouvement d'un modèle est suspendu à l'aide de cette commande, la propriété `bonesPlayer.playing` du modèle prend la valeur `FALSE`.

Paramètres

Aucune.

Exemple

L'instruction suivante met l'animation actuelle du modèle fourmi3 en pause.

```
member("PicnicScene").model("Ant3").bonesplayer.pause()
```

Voir aussi

play() (3D), playing (3D), playlist

pause() (RealMedia, SWA, Windows Media)

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.pause()

// JavaScript syntax
memberOrSpriteObjRef.pause();
```

Description

Méthode d'acteur ou d'image-objet RealMedia et Windows Media ; suspend la lecture du flux de médias.

La valeur de `mediaStatus` devient `#paused`.

L'appel de cette méthode pendant la lecture du flux RealMedia ou Windows Media ne modifie pas la propriété `currentTime` et n'efface pas le contenu de la mémoire tampon. En revanche, elle permet aux commandes `play` suivantes de reprendre la lecture sans remettre le flux en mémoire tampon.

Paramètres

Aucune.

Exemple

Les exemples suivants arrêtent la lecture de l'image-objet 2 ou de l'acteur Real.

```
-- Lingo syntax
sprite(2).pause()
member("Real").pause()

// JavaScript syntax
sprite(2).pause();
member("Real").pause();
```

Voir aussi

mediaStatus (RealMedia, Windows Media), play() (RealMedia, SWA, Windows Media), seek(), stop() (RealMedia, SWA, Windows Media)

perlinNoise()

Syntaxe

```
-- Lingo syntax
member(membername).Image.perlinNoise(baseX, baseY, numOctaves, seed, bStitch,
bFractalNoise, [#channelOptions:value, #grayscale:bolean, #offsets:listOfPoints])

// JavaScript syntax
member(membername).Image.perlinNoise(baseX, baseY, numOctaves, seed, bStitch,
bFractalNoise, [#channelOptions:value, #grayscale:bolean, #offsets:listOfPoints]);
```

Description

L'algorithme de génération de bruit de Perlin interpole et combine différentes fonctions de bruit aléatoires (appelées octaves) dans une même fonction générant un bruit aléatoire paraissant plus naturel.

Paramètres

Propriété	Description	Plage de valeurs	Valeur par défaut
baseX : nombre	Détermine la valeur x (taille) des motifs créés.		
baseY : nombre	Détermine la valeur y (taille) des motifs créés.		
numOctaves : nombre	Nombre d'octaves ou de fonctions de bruit individuelles à combiner pour créer ce bruit. Le choix d'un nombre d'octaves élevé permet de créer des images plus détaillées, mais allonge également le temps de traitement requis.		
randomSeed : nombre	Nombre de départ aléatoire		
bStitch : valeur booléenne	Si cette propriété présente la valeur True, cette méthode tente de raccorder (ou de lisser) les bords de transition de l'image afin de créer des textures homogènes pour la juxtaposition sous forme de remplissage de bitmap.	True/false	
bFractalNoise : valeur booléenne	Ce paramètre est associé aux bords des dégradés générés par la méthode. S'il est défini sur True, la méthode génère un bruit fractal qui lisse les bords de l'effet. S'il est défini sur False, la méthode génère des turbulences. Une image comportant des turbulences présente des discontinuités visibles dans le dégradé, permettant ainsi d'obtenir plus efficacement des effets visuels plus marqués, tels que des flammes et des vagues.	True/false	
channelOptions : nombre (facultatif)	Cette propriété spécifie le canal de couleur (du bitmap) auquel le motif de bruit s'applique. Ce nombre peut constituer n'importe quelle combinaison des quatre canaux de couleur RGBA(1, 2, 4 et 8). La valeur par défaut est 7.		0
grayScale : valeur booléenne (facultatif)	Si cette propriété présente la valeur True, elle applique la valeur randomSeed aux pixels de bitmap, supprimant ainsi toute la couleur de l'image. La valeur par défaut est false.	0 ou 1	0
Offsets : liste ou point (facultatif)	Cette propriété peut correspondre à une liste de points ou à un point correspondant aux décalages x et y pour chaque octave. En manipulant les valeurs de décalage, vous pouvez faire défiler en douceur les couches de l'image. Chaque point de la liste de décalage affecte une fonction d'octave spécifique. La valeur par défaut est nulle. Si vous spécifiez un point, les mêmes valeurs sont utilisées pour chaque fonction d'octave.		Point (0,0)

Exemple

Les exemples suivants arrêtent la lecture de l'image-objet 2 ou de l'acteur Real.

```
-- Lingo syntax
myList = [point(0,1), point(5,5)] --List of Points
member("myMember").Image.perlinNoise(300, 300, 2, 2, true, true, [#channelOptions:7,
#grayscale:false, #offsets:myList])
// JavaScript syntax
myList = list(point(0,1), point(5,5)) //List of Points
member("myMember").image.perlinNoise(300, 300, 2, 2, true,
true, propList(symbol("channelOptions"),7, symbol("grayscale"),false,
symbol("offsets"),myList));
```

perpendicularTo

Syntaxe

```
vector1.perpendicularTo(vector2)
```

Description

Commande 3D de vecteur ; renvoie un vecteur perpendiculaire au vecteur d'origine et à un second vecteur. Cette commande équivaut à la commande de vecteur `crossProduct`.

Paramètres

`vector2` Requis. Spécifie le second vecteur.

Exemple

Dans l'exemple suivant, pos1 est un vecteur sur l'axe des x et pos2 est un vecteur sur l'axe des y. La valeur renvoyée par `pos1.perpendicularTo(pos2)` est `vector( 0.0000, 0.0000, 1.00000e4 )`. Les deux dernières lignes de l'exemple indiquent le vecteur perpendiculaire à pos1 et pos2.

```
-- Lingo
pos1 = vector(100, 0, 0)
pos2 = vector(0, 100, 0)
put pos1.perpendicularTo(pos2)
-- vector( 0.0000, 0.0000, 1.00000e4 )

// Javascript
pos1 = vector(100, 0, 0);
pos2 = vector(0, 100, 0);
trace(pos1.perpendicularTo(pos2));
// vector( 0.0000, 0.0000, 1.00000e4 )
```

Voir aussi

`crossProduct()`, `cross`

pictureP()

Syntaxe

```
-- Lingo syntax
pictureP(pictureValue)
```

```
// JavaScript syntax
pictureP(pictureValue);
```

Description

Fonction ; indique si l'état de la propriété `picture` de l'acteur spécifié est `TRUE` (1) ou `FALSE` (0).

Etant donné que `pictureP` ne vérifie pas directement si une image est associée à un acteur, vous devez effectuer cette opération en vérifiant la propriété `picture` de l'acteur.

Paramètres

`pictureValue` Requis. Spécifie une référence à l'image d'un acteur.

Exemple

La première instruction de l'exemple suivant affecte la valeur de la propriété `picture` de l'acteur Temple, qui est un bitmap, à la variable `pictureValue`. La seconde instruction vérifie si Temple est une image en vérifiant la valeur affectée à `pictureValue`.

```
-- Lingo syntax
pictureValue = member("Shrine").picture
put pictureP(pictureValue)

// JavaScript syntax
var pictureValue = member("Shrine").picture;
put(pictureP(pictureValue));
```

Le résultat est 1, équivalent numérique de `TRUE`.

play() (3D)

Syntaxe

```
member(whichCastmember).model(whichModel).bonesPlayer.play()
member(whichCastmember).model(whichModel).keyframePlayer.play()
member(whichCastmember).model(whichModel).bonesPlayer.play(motionName {, looped, startTime,
endTime, scale, offset})
member(whichCastmember).model(whichModel).keyframePlayer.play(motionName {, looped,
startTime, endTime, scale, offset})
```

Description

Commande 3D `#keyframePlayer` et `#bonesPlayer` ; entraîne ou reprend l'exécution d'un mouvement.

Lorsque le mouvement d'un modèle est démarré ou redémarré à l'aide de cette commande, la propriété `bonesPlayer.playing` du modèle prend la valeur `TRUE`.

Utilisez `play()` sans paramètre pour reprendre l'exécution d'un mouvement qui a été arrêté à l'aide de la commande `pause()`.

L'utilisation de la commande `play()` pour démarrer un mouvement insère le mouvement au début de la liste de lecture du modificateur. Si cela interrompt la lecture d'un autre mouvement, le mouvement interrompu reste dans la liste de lecture et est positionné après le mouvement qui vient d'être démarré. Lorsque le mouvement qui vient d'être démarré se termine (s'il n'est pas en boucle) ou que la commande `playNext()` est émise, la lecture du mouvement interrompu reprend là où elle s'était arrêtée.

Paramètres

motionName Requis. Spécifie le nom du mouvement à exécuter. Lorsque *motionName* est le seul paramètre transmis à play(), le mouvement est exécuté une fois par le modèle du début à la fin à la cadence définie par la propriété playRate du modificateur.

looped Facultatif. Spécifie si le mouvement est lu une seule fois (FALSE) ou continuellement (TRUE).

startTime Facultatif. Ce paramètre est mesuré en millisecondes à partir du début du mouvement. Lorsque le paramètre *looped* présente la valeur TRUE, la première itération de la boucle commence au niveau de *offset* et s'achève au niveau de *endTime*, toutes les répétitions ultérieures du mouvement démarrant au niveau *startTime* et se terminant au niveau de *endTime*.

endTime Facultatif. Ce paramètre est mesuré en millisecondes à partir du début du mouvement. Lorsque *looped* présente la valeur FALSE, le mouvement démarre à la position *offset* et se termine au niveau de *endTime*. Lorsque le paramètre *looped* présente la valeur TRUE, la première itération de la boucle commence au niveau de *offset* et s'achève au niveau de *endTime*, toutes les répétitions ultérieures démarrant au niveau *startTime* et se terminant au niveau de *endTime*. Attribuez à *endTime* la valeur -1 si vous souhaitez que le mouvement soit lu jusqu'à la fin.

playRate Facultatif. Spécifie la cadence réelle de la lecture du mouvement. *playRate* est multiplié par la propriété playRate du modificateur #keyframePlayer ou #bonesPlayer du modèle pour déterminer la cadence réelle de la lecture du mouvement.

offset Facultatif. Ce paramètre est mesuré en millisecondes à partir du début du mouvement. Lorsque *looped* présente la valeur FALSE, le mouvement démarre à la position *offset* et se termine au niveau de *endTime*. Lorsque le paramètre *looped* présente la valeur TRUE, la première itération de la boucle commence au niveau de *offset* et s'achève au niveau de *endTime*, toutes les répétitions ultérieures démarrant au niveau *startTime* et se terminant au niveau de *endTime*. Vous pouvez également attribuer au paramètre *offset* la valeur #synchronized pour démarrer le mouvement à la même position par rapport à la durée que l'animation en cours.

Exemple

La commande suivante entraîne le modèle Marcheur à lire le mouvement Chute. Après la lecture de ce mouvement, le modèle reprendra la lecture de tout autre mouvement précédemment interrompu.

```
sprite(1).member.model("Walker").bonesPlayer.play("Fall", 0, 0, -1, 1, 0)
```

La commande suivante entraîne le modèle Marcheur à démarrer la lecture du mouvement coupDenvoi. Si Marcheur est en train d'exécuter un mouvement, il est interrompu par le mouvement coupDenvoi dont une section est jouée en boucle. La première itération de la boucle démarrera 2 000 millisecondes à compter du début du mouvement. Toutes les itérations suivantes de la boucle démarrent à 1 000 millisecondes du début de coupDenvoi et prennent fin à 5 000 millisecondes du début de coupDenvoi. La cadence de lecture est égale à trois fois la propriété playRate du modificateur bonesPlayer du modèle.

```
sprite(1).member.model("Walker").bonesPlayer.play("Kick", 1, 1000, 5000, 3, 2000)
```

Voir aussi

queue() (3D), playNext() (3D), playRate (3D), playlist, pause() (3D), removeLast(), playing (3D)

play() (DVD)

Syntaxe

```
-- Lingo syntax
```

```
dvdObjRef.play()
dvdObjRef.play(beginTitle, beginChapter, endTitle, endChapter)
dvdObjRef.play(beginTimeList, endTimeList)

// JavaScript syntax
dvdObjRef.play();
dvdObjRef.play(beginTitle, beginChapter, endTitle, endChapter);
dvdObjRef.play(beginTimeList, beginTimeList);
```

Description

Méthode de DVD ; démarre ou reprend la lecture.

En l'absence de paramètres, cette méthode reprend la lecture si celle-ci avait été mise en pause ; si la lecture avait été arrêtée, cette méthode recommence la lecture au début d'un disque ou à la position spécifiée par la propriété `startTimeList`. Puis elle poursuit la jusqu'à la position spécifiée par la propriété `stopTimeList` si celle-ci a été définie.

Utilisée avec les paramètres *`beginTitle`*, *`beginChapter`*, *`endTitle`* et *`endChapter`*, cette méthode démarre la lecture au niveau d'un titre et d'un chapitre donnés. Puis elle poursuit la lecture jusqu'aux positions spécifiées par les paramètres *`endTitle`* et *`endChapter`* si ces derniers ont été définis.

Utilisée avec les paramètres *`beginTimeList`* et *`endTimeList`*, cette méthode déclenche la lecture entre la valeur spécifiée par le paramètre *`beginTimeList`* et la valeur du paramètre *`endTimeList`*.

Les formats de liste utilisés pour `beginTimeList` et `endTimeList` sont les suivants :

```
[#title:1, #chapter:1, #hours:0, #minutes:1, #seconds:1]
```

or

```
[#title:1, #hours:0, #minutes:1, #seconds:1]
```

Cette méthode renvoie la valeur 0 si l'opération a réussi.

Paramètres

beginTitle Requis pour faire démarrer la lecture au niveau d'un titre et d'un chapitre donnés. Nombre indiquant le titre contenant le chapitre à lire. Ce paramètre prévaut sur la propriété `startTimeList` de l'acteur.

`beginChapter` Requis pour faire démarrer la lecture au niveau d'un titre et d'un chapitre donnés. Nombre spécifiant le chapitre au niveau duquel la lecture s'arrêtera. Ce paramètre prévaut sur la propriété `startTimeList` de l'acteur.

`endTitle` Requis pour arrêter la lecture au niveau d'un titre et d'un chapitre donnés. Nombre spécifiant le titre au niveau duquel la lecture s'arrêtera. Ce paramètre prévaut sur la propriété `stopTimeList` de l'acteur.

`endChapter` Requis pour arrêter la lecture au niveau d'un titre et d'un chapitre donnés. Nombre spécifiant le chapitre au niveau duquel la lecture s'arrêtera. Ce paramètre prévaut sur la propriété `stopTimeList` de l'acteur.

`beginTimeList` Requis si la lecture doit démarrer à une heure spécifique. Liste de propriétés spécifiant l'heure de début de la lecture. Ce paramètre prévaut sur la propriété `startTimeList` de l'acteur.

`endTimeList` Requis si la lecture doit démarrer à une heure spécifique. Liste de propriétés spécifiant l'heure de fin de la lecture. Ce paramètre prévaut sur la propriété `stopTimeList` de l'acteur.

Exemple

L'instruction suivante reprend la lecture d'une image-objet mise en pause :

```
-- Lingo syntax
member(12).play()
```

```
// JavaScript syntax
member(12).play();
```

Les instructions suivantes entraînent le démarrage de la lecture au chapitre 2 du titre 1 et la fin de la lecture au chapitre 4 :

```
member(15).play([#title:1, #chapter:2], [#title:1, #chapter:4])
```

or

```
member(15).play(1,2,1,4)
```

Les instructions suivantes entraînent le démarrage de la lecture à la 10ème seconde du chapitre 2 et la fin de la lecture à la 17ème seconde :

```
member(15).play([#title:2, #seconds:10], [#title:2, #seconds:17])
```

Voir aussi

DVD, startTimeList, stopTimeList

play() (piste audio)

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.play()
soundChannelObjRef.play(memberObjRef)
soundChannelObjRef.play(propList)

// JavaScript syntax
soundChannelObjRef.play();
soundChannelObjRef.play(memberObjRef);
soundChannelObjRef.play(propList);
```

Description

Méthode de piste audio ; démarre la lecture des sons placés en file d'attente dans une piste audio ou met en file d'attente et commence à lire un acteur donné.

Les acteurs son mettent un certain temps à se charger en mémoire RAM avant de pouvoir commencer à être lus. Il est conseillé de placer les sons en file d'attente à l'aide de la méthode `queue()` avant d'entamer leur lecture, puis d'utiliser la première forme de cette méthode. La seconde forme ne tire pas parti du préchargement accompli à l'aide de la commande `queue()`.

L'utilisation d'une liste de propriétés facultative permet de définir les paramètres de lecture exacts d'un son.

Vous pouvez voir un exemple d'utilisation de `play()` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

`memberObjRef` Requis en cas de lecture d'un acteur spécifique. Référence à l'objet acteur à mettre en file d'attente et à lire.

`propList` Requis en cas de définition de paramètres de lecture pour un son. Liste de propriétés spécifiant les paramètres de lecture exacts du son. La définition de ces propriétés est facultative :

Propriété	Description
#member	Acteur à placer en file d'attente. Cette propriété doit être spécifiée. Toutes les autres sont facultatives.
#startTime	Position temporelle de départ de la lecture du son, en millisecondes. La valeur par défaut est le début du son. Pour plus d'informations, reportez-vous à l'entrée `startTime`.
#endTime	Position temporelle de fin de la lecture du son, en millisecondes. La valeur par défaut est la fin du son. Pour plus d'informations, reportez-vous à l'entrée `endTime`.
#loopCount	Nombre de répétitions d'une boucle défini avec `#loopStartTime` et `#loopEndTime`. La valeur par défaut est 1. Pour plus d'informations, reportez-vous à l'entrée `loopCount`.
#loopStartTime	Position temporelle de départ de la boucle, en millisecondes. Pour plus d'informations, reportez-vous à l'entrée `loopStartTime`.
#loopEndTime	Position temporelle de fin de la boucle, en millisecondes. Pour plus d'informations, reportez-vous à l'entrée `loopEndTime`.
#preloadTime	Quantité de son à placer en mémoire tampon avant la lecture, en millisecondes. Pour plus d'informations, reportez-vous à l'entrée `preloadTime`.

Exemple

L'instruction suivante lit l'acteur son Intro sur la piste audio 1 :

```
-- Lingo syntax
sound(1).play(member("introMusic"))

// JavaScript syntax
sound(1).play(member("introMusic"));
```

L'instruction suivante entraîne la lecture de l'acteur Crédits sur la piste audio 2. La lecture commence à la quatrième seconde du son et se termine à la quinzième seconde. La section comprise entre 10,5 et 14 secondes exécute une lecture en boucle à 6 reprises.

```
-- Lingo syntax
sound(2).play([#member:member("creditsMusic"), #startTime:4000, #endTime:15000,
#loopCount:6, #loopStartTime:10500, #loopEndTime:14000])

// JavaScript syntax
sound(2).play(propList("member",member("creditsMusic"), "startTime",4000,"endTime",15000,
"loopCount",6, "loopStartTime",10500, "loopEndTime",14000));
```

Voir aussi

`endTime`, `loopCount`, `loopEndTime`, `loopStartTime`, `pause() (piste audio)`, `preLoadTime`, `queue()`, `Piste audio`, `startTime`, `stop() (piste audio)`

play() (RealMedia, SWA, Windows Media)

Syntaxe

```
-- Lingo syntax
windowsMediaObjRef.play()
realMediaObjRef.play()

// JavaScript syntax
windowsMediaObjRef.play();
realMediaObjRef.play();
```

Description

Méthode d'image-objet ou d'acteur Windows Media ou RealMedia ; lit l'acteur Windows Media ou RealMedia ou lit l'image-objet sur la scène.

Pour les acteurs, seule la partie audio est rendue si elle existe dans l'animation. Si l'acteur est déjà en cours de lecture, l'appel de cette méthode ne produit aucun effet.

Paramètres

Aucune.

Exemple

Les exemples suivants démarrent le processus de lecture en flux continu de l'image-objet 2 et de l'acteur Real.

```
-- Lingo syntax
sprite(2).play()
member("Real").play()

// JavaScript syntax
sprite(2).play();
member("Real").play();
```

Voir aussi

`RealMedia`, `Windows Media`

playFile()

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.playFile(stringFilePath)

// JavaScript syntax
soundChannelObjRef.playFile(stringFilePath);
```

Description

Méthode de piste audio ; lit le son AIFF, SWA, AU ou WAV dans une piste audio.

Pour que le son soit lu correctement, l'Xtra MIX qui convient doit être accessible à l'animation (il est généralement placé dans le dossier des Xtras de l'application).

Lorsque le fichier audio n'est pas placé dans le même dossier que l'animation, *`stringFilePath`* doit indiquer le chemin d'accès complet de ce fichier.

Pour lire des sons obtenus à partir d'une adresse URL, il est généralement judicieux d'utiliser la commande `downloadNetThing()` ou `preloadNetThing()` pour commencer par télécharger le fichier sur un disque local. Vous éviterez ainsi les problèmes pouvant se produire en attente de téléchargement.

La méthode `playFile()` lit les fichiers en flux continu à partir du disque dur au lieu de les lire depuis la mémoire RAM. Par conséquent, l'utilisation de la commande `playFile()` pendant la lecture d'une vidéo numérique ou le chargement d'acteurs en mémoire peut occasionner des conflits si l'ordinateur tente de lire deux emplacements du disque simultanément.

Paramètres

stringFilePath Requis. Chaîne spécifiant le nom du fichier à lire. Lorsque le fichier audio n'est pas placé dans le même dossier que l'animation en cours de lecture, *stringFilePath* doit également indiquer le chemin d'accès complet de ce fichier.

Exemple

L'instruction suivante lit le fichier Tonnerre dans la piste 1 :

```
-- Lingo syntax
sound(1).playFile("Thunder.wav")

// JavaScript syntax
sound(1).playFile("Thunder.wav");
```

L'instruction suivante lit le fichier Tonnerre dans la piste 3 :

```
-- Lingo syntax
sound(3).playFile(_movie.path & "Thunder.wav")

// JavaScript syntax
sound(3).playFile(_movie.path + "Thunder.wav");
```

Voir aussi

play() (piste audio), Piste audio, stop() (piste audio)

playFromToTime()

Syntaxe

```
-- Lingo syntax
windowsMediaObjRef.playFromToTime(intStartTime, intEndTime)

// JavaScript syntax
windowsMediaObjRef.playFromToTime(intStartTime, intEndTime);
```

Description

Méthode d'image-objet Windows Media. Démarre et achève la lecture aux heures de début et de fin spécifiées.

Paramètres

intStartTime Requis. Nombre entier spécifiant l'heure, en millisecondes, à laquelle la lecture commence.

intEndTime Requis. Nombre entier spécifiant l'heure, en millisecondes, à laquelle la lecture prend fin.

Exemple

L'instruction suivante demande que l'image-objet Vidéo soit lue entre la 30ème seconde et la 40ème seconde.

```
-- Lingo syntax
sprite("Video").playFromToTime(30000, 40000)

// JavaScript syntax
sprite("Video").playFromToTime(30000, 40000);
```

Voir aussi

Windows Media

playNext() (piste audio)

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.playNext()

// JavaScript syntax
soundChannelObjRef.playNext();
```

Description

Méthode de piste audio ; entraîne l'interruption immédiate de la lecture du son en cours sur une piste audio et démarre la lecture du son suivant en file d'attente.

Si la file d'attente ne contient pas d'autres sons, la lecture du son est simplement stoppée.

Paramètres

Aucune.

Exemple

L'instruction suivante exécute la lecture sur la piste audio 2 du son suivant placé en file d'attente.

```
-- Lingo syntax
sound(2).playNext()

// JavaScript syntax
sound(2).playNext();
```

Voir aussi

`pause() (piste audio)`, `play() (piste audio)`, `Piste audio`,`stop() (piste audio)`

playNext() (3D)

Syntaxe

```
member(whichMember).model(whichModel).bonesPlayer.playNext()
member(whichMember).model(whichModel).keyframePlayer.playNext()
```

Description

Commande 3D de modificateur `#keyframePlayer` et `#bonesPlayer` ; démarre la lecture du mouvement suivant dans la liste de lecture du modificateur `#keyframePlayer` ou `#bonesPlayer` du modèle. Le mouvement en cours de lecture, qui est la première entrée de la liste de lecture, est interrompu et retiré de la liste.

Si la fusion des mouvements est activée et qu'au moins deux mouvements sont présents dans la liste de lecture, la fusion du mouvement en cours avec le suivant dans la liste de lecture commence avec l'appel de la méthode `playNext()`.

Exemple

L'instruction suivante interrompt le mouvement actuellement exécuté par le modèle 1 et démarre la lecture du mouvement suivant dans la liste de lecture.

```
-- Lingo
member("scene").model[1].bonesPlayer.playnext()

// Javascript
```

```
member("scene").getProp("model",1).bonesPlayer.playNext();
```

Voir aussi

blend (3D), playlist

playerParentalLevel()

Syntaxe

```
-- Lingo syntax
dvdObjRef.playerParentalLevel()

// JavaScript syntax
dvdObjRef.playerParentalLevel();
```

Description

Méthode de DVD ; renvoie le niveau de contrôle parental du lecteur.

Les niveaux de contrôle parental possibles sont compris entre 1 et 8.

Paramètres

Aucune.

Voir aussi

DVD

point()

Syntaxe

```
-- Lingo syntax
point(intH, intV)

// JavaScript syntax
point(intH, intV);
```

Description

Fonction et type de données de niveau supérieur. Renvoie un point dont les coordonnées horizontale et verticale ont été spécifiées.

Un point comporte une propriété `locH` et une propriété `locV`.

Les coordonnées d'un point sont modifiables par des opérations arithmétiques uniquement dans Lingo. Par exemple, Lingo vous permet d'ajouter les deux-points suivants ensemble, alors que la syntaxe JavaScript renvoie la valeur `NaN` :

```
-- Lingo
pointA = point(10,10)
pointB = point(5,5)
put(pointA + pointB)
-- point(15,15)

// JavaScript syntax
var pointA = point(10,10);
var pointB = point(5,5);
```

```
trace(pointA + pointB);
// NaN
```

Vous pouvez voir un exemple d'utilisation de `point()` dans une animation en consultant les animations Imaging et Vector Shapes du dossier Learning/Lingo, lui-même dans le dossier de Director.

Paramètres

`intH` Requis. Nombre entier spécifiant la coordonnée horizontale du point.

`intV` Requis. Nombre entier spécifiant la coordonnée verticale du point.

Exemple

L'instruction suivante définit la variable `lastLocation` sur le point (250, 400) :

```
-- Lingo syntax
lastLocation = point(250, 400)

// JavaScript syntax
var lastLocation = point(250, 400);
```

L'instruction suivante ajoute 5 pixels à la coordonnée horizontale du point affecté à la variable `myPoint` :

```
-- Lingo syntax
myPoint.locH = myPoint.locH + 5

// JavaScript syntax
myPoint.locH = myPoint.locH + 5;
```

Dans Lingo uniquement, les instructions suivantes définissent les coordonnées sur la scène d'une image-objet en tant que `mouseH` et `mouseV` plus 10 pixels. Ces deux instructions sont équivalentes.

```
-- Lingo syntax
sprite(_mouse.clickOn).loc = point(_mouse.mouseH, _mouse.mouseV) + point(10, 10)
sprite(_mouse.clickOn).loc = _mouse.mouseLoc + 10
```

Voir aussi

`locH`, `locV`

pointAt

Syntaxe

```
member(whichCastmember).model(whichModel).pointAt(vectorPosition{, vectorUp})
member(whichCastmember).camera(whichCamera).pointAt(vectorPosition{, vectorUp})
member(whichCastmember).light(whichLight).pointAt(vectorPosition{, vectorUp})
member(whichCastmember).group(whichGroup).pointAt(vectorPosition{, vectorUp})
```

Description

Commande 3D ; fait pivoter l'objet référencé pour que son vecteur horizontal pointe vers une position relative à l'univers spécifiée, puis fait pivoter l'objet référencé pour que son vecteur vertical pointe dans la direction indiquée par un vecteur relatif spécifié.

Le vecteur vertical et le vecteur horizontal de l'objet sont définis par la propriété `pointAtOrientation` de l'objet.

Paramètres

`vectorPosition` Requis. Spécifie la position relative à l'univers. Cette valeur peut également correspondre à une référence de nœud.

vectorUp Facultatif. Spécifie un vecteur relatif à l'univers indiquant l'orientation du vecteur vertical de l'objet. Si ce paramètre n'est pas spécifié, la méthode `pointAt` utilise par défaut l'axe des y de l'univers comme vecteur vertical recommandé. Si vous essayez de diriger l'objet pour que son vecteur horizontal soit parallèle à l'axe des y de l'univers, l'axe des x de l'univers est utilisé comme vecteur vertical recommandé. La direction horizontale de l'objet et la direction spécifiée par *vectorUp* n'ont pas besoin d'être perpendiculaires car cette commande n'utilise que le paramètre *vectorUp* comme vecteur de recommandation.

Exemple

L'exemple suivant dirige trois objets vers le modèle Mars : la caméra camMars, la lumière Spot et le modèle Pistolet.

```
thisWorldPosn = member("Scene").model("Mars").worldPosition
member("Scene").camera("MarsCam").pointAt(thisWorldPosn)
member("Scene").light("BrightSpot").pointAt(thisWorldPosn)
member("Scene").model("BigGun").pointAt(thisWorldPosn, vector(0,0,45))
```

Si vous utilisez un redimensionnement non uniforme et une propriété `pointAtOrientation` personnalisée sur le même nœud (un modèle, par exemple), l'appel de `pointAt` entraîne probablement un redimensionnement non uniforme inattendu. Cela s'explique par l'ordre dans lequel le redimensionnement non uniforme et la rotation utilisés pour orienter correctement le nœud sont appliqués. Pour remédier à ce problème, effectuez l'une des opérations suivantes :

- Evitez d'utiliser un redimensionnement non uniforme et une valeur pointAtOrientation autre que celle définie par défaut sur le même nœud.
- Supprimez votre propriété d'échelle avant d'utiliser la commande pointAt, puis réappliquez-la par la suite.

  Par exemple :

  ```
  scale = node.transform.scale
  node.scale = vector( 1, 1, 1 )
  node.pointAt(vector(0, 0, 0)) -- non-default pointAtOrientation
  node.transform.scale = scale
  ```

Voir aussi

pointAtOrientation

pointInHyperlink()

Syntaxe

```
-- Lingo syntax
spriteObjRef.pointInHyperlink(point)

// JavaScript syntax
spriteObjRef.pointInHyperlink(point);
```

Description

Fonction d'image-objet texte ; renvoie `TRUE` ou `FALSE` selon que le point spécifié se trouve ou non dans un lien hypertexte de l'image-objet texte. En règle générale, le point est la position du curseur. Cela est utile pour définir des curseurs personnalisés.

Paramètres

point Requis. Spécifie le point à tester.

Exemple

L'instruction suivante vérifie si la souris est positionnée sur un lien hypertexte dans l'image-objet(1).

```
-- Lingo
Put Sprite(1).pointInHyperLink(_mouse.mouseLoc)

// Javascript
trace(sprite(1).pointInHyperLink(_mouse.mouseLoc));
```

Voir aussi

cursor(), mouseLoc

pointToChar()

Syntaxe

```
-- Lingo syntax
spriteObjRef.pointToChar(pointToTranslate)

// JavaScript syntax
spriteObjRef.pointToChar(pointToTranslate);
```

Description

Fonction ; renvoie un nombre entier représentant la position du caractère situé dans l'image-objet texte ou champ à une coordonnée d'écran spécifiée ou renvoie -1 si le point ne se trouve pas dans le texte.

Cette fonction permet de déterminer le caractère sous le curseur.

Paramètres

pointToTranslate Requis. Spécifie la coordonnée d'écran à tester.

Exemple

Les instructions suivantes affichent dans la fenêtre Messages le numéro et la lettre du caractère sur lequel l'utilisateur a cliqué :

```
--Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    charNum = sprite(spriteNum).pointToChar(pointClicked)
    actualChar = currentMember.char[charNum]
    put("Clicked character" && charNum & ", the letter" && actualChar)
end

// JavaScript syntax
function mouseDown() {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var charNum = sprite(this.spriteNum).pointToChar(pointClicked);
    var actualChar = currentMember.getProp("char", charNum);
    put("Clicked character " + charNum +", the letter " + actualChar);
}
```

Voir aussi

mouseLoc, pointToWord(), pointToItem(), pointToLine(), pointToParagraph()

pointToItem()

Syntaxe

```
-- Lingo syntax
spriteObjRef.pointToItem(pointToTranslate)

// JavaScript syntax
spriteObjRef.pointToItem(pointToTranslate);
```

Description

Fonction ; renvoie un nombre entier représentant la position de l'élément situé dans l'image-objet texte ou champ à une coordonnée d'écran spécifiée ou renvoie -1 si le point ne se trouve pas dans le texte. Les éléments sont séparés par la propriété `itemDelimiter`, correspondant à une virgule par défaut.

Cette fonction permet de déterminer l'élément sous le curseur.

Paramètres

`pointToTranslate` Requis. Spécifie la coordonnée d'écran à tester.

Exemple

Les instructions suivantes affichent dans la fenêtre Messages le numéro et le texte de l'élément sur lequel l'utilisateur a cliqué :

```
--Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    itemNum = sprite(spriteNum).pointToItem(pointClicked)
    itemText = currentMember.item[itemNum]
    put("Clicked item" && itemNum & ", the text" && itemText)
end

// JavaScript syntax
function mouseDown() {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var itemNum = sprite(this.spriteNum).pointToItem(pointClicked);
    var itemText = currentMember.getProp("item",itemNum);
    trace( "Clicked item " + itemNum + ", the text " + itemText);
}
```

Voir aussi

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToItem(), pointToLine(), pointToParagraph()

pointToLine()

Syntaxe

```
-- Lingo syntax
spriteObjRef.pointToLine(pointToTranslate)

// JavaScript syntax
spriteObjRef.pointToLine(pointToTranslate);
```

Description

Fonction ; renvoie un nombre entier représentant la position de la ligne située dans l'image-objet texte ou champ à une coordonnée d'écran spécifiée ou renvoie -1 si le point ne se trouve pas dans le texte. Les lignes sont séparées par des retours chariot dans l'acteur texte ou champ.

Cette fonction permet de déterminer la ligne sous le curseur.

Paramètres

pointToTranslate Requis. Spécifie la coordonnée d'écran à tester.

Exemple

Les instructions suivantes affichent dans la fenêtre Messages le numéro et le texte de la ligne sur laquelle l'utilisateur a cliqué :

```
-- Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    lineNum = sprite(spriteNum).pointToLine(pointClicked)
    lineText = currentMember.line[lineNum]
    put("Clicked line" && lineNum & ", the text" && lineText)
end

// JavaScript syntax
functionmouseDown() {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var lineNum = sprite(this.spriteNum).pointToLine(pointClicked);
    var lineText = currentMember.getProp("line", lineNum);
    put("Clicked line " + lineNum + ", the text " + lineText);
}
```

Voir aussi

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToItem(), pointToLine(), pointToParagraph()

pointToParagraph()

Syntaxe

```
-- Lingo syntax
spriteObjRef.pointToParagraph(pointToTranslate)
```

```
// JavaScript syntax
spriteObjRef.pointToParagraph(pointToTranslate);
```

Description

Fonction ; renvoie un nombre entier représentant le numéro du paragraphe situé dans l'image-objet texte ou champ à une coordonnée d'écran spécifiée ou renvoie -1 si le point ne se trouve pas dans le texte. Les paragraphes sont séparés par des retours chariot dans un bloc de texte.

Cette fonction permet de déterminer le paragraphe sous le curseur.

Paramètres

pointToTranslate Requis. Spécifie la coordonnée d'écran à tester.

Exemple

Les instructions suivantes affichent dans la fenêtre Messages le numéro et le texte du paragraphe dans lequel l'utilisateur a cliqué :

```
-- Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    paragraphNum = sprite(spriteNum).pointToParagraph(pointClicked)
    paragraphText = currentMember.paragraph[paragraphNum]
    put("Clicked paragraph" && paragraphNum & ", the text" && paragraphText)
end

// JavaScript syntax
function mouseDown() {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var paragraphNum = sprite(this.spriteNum).pointToParagraph(pointClicked);
    var paragraphText = currentMember.getProp("paragraph", paragraphNum);
    trace("Clicked paragraph " + paragraphNum + ", the text " + paragraphText);
}
```

Voir aussi

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToItem(), pointToLine()

pointToWord()

Syntaxe

```
-- Lingo syntax
spriteObjRef.pointToWord(pointToTranslate)

// JavaScript syntax
spriteObjRef.pointToWord(pointToTranslate);
```

Description

Fonction ; renvoie un nombre entier représentant le numéro d'un mot situé dans l'image-objet texte ou champ à une coordonnée d'écran spécifiée ou renvoie -1 si le point ne se trouve pas dans le texte. Les mots sont séparés par des espaces dans un bloc de texte.

Cette fonction permet de déterminer le mot sous le curseur.

Paramètres

pointToTranslate Requis. Spécifie la coordonnée d'écran à tester.

Exemple

Les instructions suivantes affichent dans la fenêtre Messages le numéro et le texte du mot sur lequel l'utilisateur a cliqué :

```
-- Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    wordNum = sprite(spriteNum).pointToWord(pointClicked)
    wordText = currentMember.word[wordNum]
    put("Clicked word" && wordNum & ", the text" && wordText)
end

// JavaScript syntax
function mouseDown(me) {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var wordNum = sprite(this.spriteNum).pointToWord(pointClicked);
    var wordText = currentMember.getProp("word", wordNum);
    trace("Clicked word " + wordNum + ", the text " + wordText);
}
```

Voir aussi

itemDelimiter, mouseLoc, pointToChar(), pointToItem(), pointToLine(), pointToParagraph()

postNetText

Syntaxe

```
postNetText(url, propertyList {,serverOSString} {,serverCharSetString})
postNetText(url, postText {,serverOSString} {,serverCharSetString})
```

Description

Commande ; envoie une requête POST à une URL, correspondant à une adresse URL HTTP, avec les données spécifiées.

Cette commande est semblable à getNetText(). Comme pour getNetText(), la réponse du serveur est renvoyée par netTextResult(*netID*) une fois que netDone(*netID*) prend la valeur 1 et si netError(*netID*) présente la valeur 0 ou OK.

Les paramètres facultatifs peuvent être omis quelle que soit leur position.

Cette commande offre en outre un avantage supplémentaire par rapport à getNetText() : une requête postNetText() peut être arbitrairement longue, alors que la longueur de la requête getNetText() est limitée à celle d'une adresse URL (1 Ko ou 4 Ko, selon le navigateur).

Remarque : *L'utilisation de postNetText pour publier des données dans un domaine différent de celui depuis lequel l'animation est lue déclenche une alerte de sécurité lors de la lecture dans Shockwave Player.*

Vous pouvez voir un exemple d'utilisation de postNetText dans une animation en consultant l'animation Forms and Post du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

url Requis. Spécifie l'URL à laquelle la requête POST doit être envoyée.

propertyList ou *postText* Requis. Spécifie les données à envoyer avec la requête. Lorsqu'une liste de propriétés est utilisée à la place d'une chaîne, les informations sont envoyées avec METHOD=POST, de la même manière qu'un navigateur publie un formulaire HTML. Cette procédure facilite la construction et l'affichage des données d'un formulaire dans un titre Director. Les noms des propriétés correspondent aux noms des champs du formulaire HTML et leurs valeurs à celles des champs.

La liste de propriétés peut utiliser des chaînes ou des symboles comme noms de propriétés. Si un symbole est utilisé, il est automatiquement converti en chaîne sans le signe # du début. De même, les valeurs numériques sont converties en chaînes si elles sont utilisées comme valeur d'une propriété.

Remarque : *Si la forme secondaire est utilisée (une chaîne remplace la liste de propriétés), la chaîne textePosté est envoyée au serveur sous forme de requête HTTP POST à l'aide du type MIME « text/plain ». Bien que pratique dans certaines applications, cette méthode n'est pas compatible avec l'affichage de formulaires HTML. Par exemple, les scripts PHP doivent toujours utiliser une liste de propriétés.*

serverOSString Facultatif. Ce paramètre présente la valeur UNIX par défaut, mais peut être défini sur Windows ou sur Mac et convertit les retours chariot de l'argument *postText* en ceux utilisés par le serveur afin d'éviter toute confusion. Pour la plupart des applications, ce paramètre n'est pas nécessaire, les sauts de ligne n'étant généralement pas utilisés dans les réponses de formulaires.

serverCharSetString Facultatif. Ne s'applique que si l'utilisateur travaille sur un système Shift-JIS (japonais). Les jeux de caractères possibles sont JIS, EUC, ASCII et AUTO. Les données envoyées sont converties de Shift-JIS dans le jeu de caractères désigné. Les données renvoyées sont traitées de la même façon que par getNetText() (converties du jeu de caractères nommé en Shift-JIS). Si la valeur AUTO est utilisée, les données publiées dans le jeu de caractères local ne sont pas converties ; les résultats renvoyés par le serveur sont convertis comme pour getNetText(). ASCII est la valeur par défaut si *serverCharSetString* est omis. ASCII n'offre aucune conversion pour la publication ou les résultats.

Exemple

L'instruction suivante omet le paramètre serverCharSetString :

```
-- Lingo
netID = postNetText("www.mydomain.com\database.cgi", "Bill Jones", "Win")

// Javascript
netID = postNetText("www.mydomain.com\database.cgi", "Bill Jones", "Win");
```

Notez que serverOSString et serverCharSetString ont été omis :

```
-- LingonetID = postNetText("www.mydomain.com/userbase.cgi", infoList);
lastName = member("Last Name").text
firstName = member("First Name").text
totalScore = member("Current Score").text
infoList = ["FName":firstName, "LName":lastName, "Score":totalScore]
netID = postNetText("www.mydomain.com/userbase.cgi", infoList);

// Javascript
lastName = member("Last Name").text;
firstName = member("First Name").text;
totalScore = member("Current Score").text;
infoList = propList("FName",firstName, "LName",lastName, "Score",totalScore);
netID = postNetText("www.mydomain.com/userbase.cgi", infoList);
```

Voir aussi

`getNetText()`, `netTextResult()`, `netDone()`, `netError()`

power()

Syntaxe

```
power(base, exponent)
```

Description

Fonction mathématique ; calcule la valeur d'un nombre spécifié à une puissance spécifiée.

Paramètres

`base` Requis. Spécifie le nombre de base.

`exponent` Requis. Spécifie la puissance.

Exemple

L'instruction suivante attribue à la variable *`vResult`* la valeur du cube de 4 :

```
-- Lingo
set vResult = power(4,3)

// Javascript
Var vResult = Math.pow(4,3);
```

preLoad() (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.preLoad({toMemberObjRef})

// JavaScript syntax
memberObjRef.preLoad({toMemberObjRef});
```

Description

Méthode d'acteur ; précharge en mémoire un acteur ou une plage d'acteurs et arrête ce préchargement lorsque la mémoire est saturée ou que tous les acteurs spécifiés ont été chargés.

En l'absence du paramètre *`toMemberObjRef`*, la méthode `preLoad()` précharge tous les acteurs utilisés depuis l'image en cours jusqu'à la dernière image d'une animation.

Paramètres

`toMemberObjRef` Facultatif. Référence au dernier acteur d'une plage d'acteurs chargée en mémoire. Le premier acteur de la plage est spécifié par *`memberObjRef`*.

Exemple

L'instruction suivante indique dans la fenêtre Messages si l'animation QuickTime Chaise pivotante peut être préchargée en mémoire :

```
-- Lingo syntax
put(member("Rotating Chair").preload())
```

```
// JavaScript syntax
put(member("Rotating Chair").preload());
```

Le gestionnaire `startMovie` suivant configure un acteur animation Flash pour une lecture en flux continu, puis définit sa propriété `bufferSize` :

```
-- Lingo syntax
on startMovie
    member("Flash Demo").preload = FALSE
    member("Flash Demo").bufferSize = 65536
end

// JavaScript syntax
function startMovie() {
    member("Flash Demo").preload = false;
    member("Flash Demo").bufferSize = 65536;
}
```

Voir aussi

Acteur

preLoad() (animation)

Syntaxe

```
-- Lingo syntax
_movie.preLoad({frameNameOrNum})
_movie.preLoad(fromFrameNameOrNum, toFrameNameOrNum)

// JavaScript syntax
_movie.preLoad({frameNameorNum});
_movie.preLoad(fromFrameNameOrNum, toFrameNameOrNum);
```

Description

Méthode d'animation ; précharge en mémoire des acteurs de l'image ou de la plage d'images spécifiée et s'arrête lorsque la mémoire est saturée ou que tous les acteurs spécifiés ont été préchargés, comme suit :

- En l'absence d'arguments, cette méthode précharge tous les acteurs utilisés depuis l'image en cours jusqu'à la dernière image d'une animation.
- Si un seul argument est spécifié (*`frameNameOrNum`*), cette méthode précharge tous les acteurs utilisés dans la plage d'images depuis l'image en cours jusqu'à l'image *`frameNameOrNum`*, comme spécifié par le numéro ou le libellé de l'image.
- Si deux arguments sont spécifiés (*`fromFrameNameOrNum`* et *`toFrameNameOrNum`*), cette méthode précharge tous les acteurs utilisés dans la plage d'images depuis l'image **`fromFrameNameOrNum`** jusqu'à l'image *`fromFrameNameOrNum`*, comme spécifié par le numéro ou le libellé de l'image.

La méthode `preLoad()` renvoie également le numéro de la dernière image qu'il a été possible de charger. Pour obtenir cette valeur, utilisez la méthode `result()`.

Paramètres

`frameNameOrNum` Facultatif. Chaîne spécifiant l'image à précharger ou nombre entier indiquant le numéro de cette image.

nomOuNumImageDeDébut Requis en cas de préchargement d'une plage d'images. Chaîne spécifiant le nom du libellé de la première image de la plage d'images à précharger ou nombre entier indiquant le numéro de cette première image.

f*romFrameNameOrNum* Requis en cas de préchargement d'une plage d'images. Chaîne spécifiant le nom du libellé de la dernière image de la plage d'images à précharger ou nombre entier indiquant le numéro de cette dernière image.

Exemple

L'instruction suivante précharge les acteurs utilisés depuis l'image actuelle jusqu'à l'image contenant le repère suivant :

```
-- Lingo syntax
_movie.preLoad(_movie.marker(1))

// JavaScript syntax
_movie.preLoad(_movie.marker(1));
```

L'instruction suivante précharge les acteurs utilisés de l'image 10 à l'image 50 :

```
-- Lingo syntax
_movie.preLoad(10, 50)

// JavaScript syntax
_movie.preLoad(10, 50);
```

Voir aussi

Animation, result

preLoadBuffer()

Syntaxe

```
-- Lingo syntax
memberObjRef.preLoadBuffer()

// JavaScript syntax
memberObjRef.preLoadBuffer();
```

Description

Commande ; précharge une partie d'un fichier Shockwave Audio (SWA) spécifié en mémoire. La quantité préchargée est déterminée par la propriété `preLoadTime`. Cette commande ne fonctionne que si l'acteur SWA est arrêté.

Une fois la commande `preLoadBuffer` exécutée avec succès, la propriété d'acteur `state` est égale à 2.

La plupart des propriétés d'acteur SWA ne peuvent être testées qu'une fois la commande `preLoadBuffer` correctement exécutée. Ces propriétés sont : `cuePointNames`, `cuePointTimes`, `currentTime`, `duration`, `percentPlayed`, `percentStreamed`, `bitRate`, `sampleRate` et `numChannels`.

Paramètres

Aucune.

Exemple

L'instruction suivante charge l'acteur Jacques Brel en mémoire :

```
-- Lingo syntax
member("Mel Torme").preLoadBuffer()
```

```
// JavaScript syntax
member("Mel Torme").preLoadBuffer();
```

Voir aussi

preLoadTime

preLoadMember()

Syntaxe

```
-- Lingo syntax
_movie.preLoadMember({memberObjRef})
_movie.preLoadMember(fromMemNameOrNum, toMemNameOrNum)

// JavaScript syntax
_movie.preLoadMember({memberObjRef});
_movie.preLoadMember(fromMemNameOrNum, toMemNameOrNum);
```

Description

Méthode d'animation ; précharge les acteurs et arrête le préchargement lorsque la mémoire est saturée ou que tous les acteurs spécifiés ont été préchargés.

Cette méthode renvoie le numéro du dernier acteur chargé qu'il a été possible de charger. Pour obtenir cette valeur, utilisez la méthode `result()`.

En l'absence d'arguments, `preLoadMember()` précharge tous les acteurs dans l'animation.

Si l'argument *`memberObj`* est spécifié, la méthode `preLoadMember()` ne précharge que cet acteur. Si *`memberObj`* est un nombre entier, il ne fait référence qu'à la première bibliothèque de distribution. Si *`memberObj`* est une chaîne, le premier acteur dont le nom correspond à cette chaîne est utilisé.

Si les arguments *`fromMemNameOrNum`* et *`toMemNameOrNum`* sont spécifiés, la méthode `preLoadMember()` précharge tous les acteurs de la plage spécifiée par les noms ou numéro d'acteur.

Paramètres

`memberObj` Facultatif. Référence à l'acteur à précharger.

`fromMenNameOrNum` Requis en cas de préchargement d'une plage d'acteurs. Chaîne ou nombre entier spécifiant le premier acteur de la plage d'acteurs à précharger.

`toMemNameOrNum` Requis en cas de préchargement d'une plage d'acteurs. Chaîne ou nombre entier spécifiant le premier acteur de la plage d'acteurs à précharger.

Exemple

L'instruction suivante précharge l'acteur SWF dans l'animation.

```
-- Lingo
_movie.preLoadMember(member("SWF"))

// Javascript
_movie.preLoadMember(member("SWF")) ;
```

Voir aussi

Animation, preLoad() (acteur), result

preLoadMovie()

Syntaxe

```
-- Lingo syntax
_movie.preLoadMovie(stringMovieName)

// JavaScript syntax
_movie.preLoadMovie(stringMovieName);
```

Description

Méthode d'animation ; précharge les données et les acteurs associés à la première image de l'animation spécifiée. Le préchargement d'une animation accélère le démarrage de cette dernière à l'aide de la méthode `go()` ou `play()`.

Pour précharger des acteurs depuis une adresse URL, utilisez `preloadNetThing()` pour charger les acteurs directement dans la mémoire cache ou `downloadNetThing()` pour charger une animation sur un disque local depuis lequel vous pouvez ensuite la charger en mémoire de façon à réduire le temps de téléchargement.

Paramètres

`stringMovieName` Requis. Chaîne spécifiant le nom de l'animation à précharger.

Exemple

L'instruction suivante précharge l'animation Introduction, qui est située dans le même dossier que l'animation actuelle :

```
-- Lingo syntax
_movie.preLoadMovie("Introduction")

// JavaScript syntax
_movie.preLoadMovie("Introduction");
```

Voir aussi

`downloadNetThing`, `go()`, `Animation`, `preloadNetThing()`

preloadNetThing()

Syntaxe

```
preloadNetThing (url)
```

Description

Fonction ; précharge un fichier depuis Internet dans la mémoire cache locale afin de le rendre utilisable par la suite sans perte de temps inhérente au téléchargement. La valeur renvoyée est un identifiant réseau utilisable pour suivre le déroulement de l'opération.

La fonction `preloadNetThing()` télécharge le fichier pendant la lecture de l'animation en cours. Utilisez `netDone()` pour vérifier si le téléchargement est terminé.

Un élément téléchargé peut être immédiatement affiché, car il est lu à partir de la mémoire cache locale et non à partir du réseau.

Bien que de nombreuses opérations réseau puissent être actives au même moment, l'exécution de plus de quatre opérations simultanées réduit considérablement les performances.

La taille de la mémoire cache et l'option de vérification des documents dans les préférences d'un navigateur n'affectent pas le comportement de la fonction `preloadNetThing`.

La fonction `preloadNetThing()` n'analyse pas les liens d'un fichier Director. En conséquence, même si un fichier Director est lié à des fichiers de distribution et à des fichiers graphiques, `preloadNetThing()` ne télécharge que le fichier Director. Vous devez toujours précharger séparément les autres objets liés.

Paramètres

`url` Requis. Spécifie un nom de fichier Internet valide, tel qu'une animation Director, un graphique ou l'adresse d'un serveur FTP.

Exemple

L'instruction suivante utilise `preloadNetThing()` et renvoie l'ID réseau pour l'opération :

```
-- Lingo
set mynetid = preloadNetThing("http://www.yourserver.com/menupage/mymovie.dir")

// Javascript
set mynetid = preloadNetThing("http://www.yourserver.com/menupage/mymovie.dir");
```

Une fois le téléchargement terminé, vous pouvez naviguer jusqu'à l'animation avec la même adresse URL. L'animation est lue depuis la mémoire cache, et non depuis l'adresse URL, puisqu'elle a été chargée dans le cache.

Voir aussi

`netDone()`

preMultiply

Syntaxe

```
transform1.preMultiply(transform2)
```

Description

Commande 3D de transformation ; modifie une transformation en lui appliquant au préalable les effets de positionnement, de rotation et de redimensionnement d'une autre transformation.

Si *`transform2`* décrit une rotation de 90 degrés autour de l'axe des x et que *`transform1`* décrit une translation de 100 unités sur l'axe des y, `transformation1.multiply(`*`transform2`*`)` modifie cette transformation pour lui faire décrire une translation suivie d'une rotation. L'instruction `transformation1.preMultiply(`*`transform2`*`)` modifie cette transformation pour lui faire décrire une rotation suivie d'une translation. L'effet obtenu est l'inversement de l'ordre des opérations.

Paramètres

`transform2` Requis. Spécifie la transformation dont les effets sont préappliqués à une autre transformation.

Exemple

L'instruction suivante exécute un calcul qui applique la transformation du modèle Mars à la transformation du modèle Pluton :

```
-- Lingo
member("scene").model("Pluto").transform.preMultiply(member("scene").model("Mars").transfo
rm)

// Javascript
```

```
member("scene").getPropRef("model" ,
i).transform.preMultiply(member("scene").getPropRef("model",j).transform) ;
// where i and j are the number index of the models "Pluto" and "Mars" respectively.
```

preRotate

Syntaxe

```
transformReference.preRotate( xAngle, yAngle, zAngle )
transformReference.preRotate( vector )
transformReference.preRotate( positionVector, directionVector, angle )
member( whichCastmember ).node.transform.preRotate( xAngle, yAngle, zAngle )
member( whichCastmember ).node.transform.preRotate( vector )
member( whichCastmember ).node.transform.preRotate( positionVector, directionVector, angle
)
```

Description

Commande 3D de transformation ; applique une rotation avant les décalages de position, rotation et d'échelle de la transformation. La rotation peut être spécifiée sous la forme d'un ensemble de trois angles, chacun desquels spécifiant l'angle de rotation autour des trois axes correspondants. Ces angles peuvent être spécifiés de façon explicite sous la forme `xAngle`, `yAngle` et `zAngle`ou au moyen d'un vecteur, où le composant x du vecteur correspond à la rotation autour de l'axe des *x*, le composant y à la rotation autour de l'axe des *y* et le composant z à la rotation autour de l'axe des *z*.

La rotation peut également être spécifiée comme une rotation autour d'un axe arbitraire. Cet axe est défini dans l'espace par *`positionVector`* et par *`directionVector`*. Le degré de rotation autour de cet axe est spécifié par *`angle`*.

`Node` peut être une référence à un modèle, un groupe, une lumière ou une caméra.

Paramètres

`xAngle` Requis en cas d'application d'une rotation à l'aide des axes des *x*, des *y* et des *z*. Spécifie l'angle de rotation autour de l'axe des *x*.

`yAngle` Requis en cas d'application d'une rotation à l'aide des axes des *x*, des *y* et des *z*. Spécifie l'angle de rotation autour de l'axe des *y*.

`yAngle` Requis en cas d'application d'une rotation à l'aide des axes des *x*, des *y* et des *z*. Spécifie l'angle de rotation autour de l'axe des *z*.

`vector` Requis en cas d'application d'une rotation à l'aide d'un vecteur. Spécifie le vecteur dont les angles sont utilisés dans la rotation.

*`positionVector`*Requis en cas d'application d'une rotation autour d'un axe arbitraire. Spécifie le décalage de position.

`directionVector` Requis en cas d'application d'une rotation autour d'un axe arbitraire. Spécifie le décalage de direction.

`angle` Requis en cas d'application d'une rotation autour d'un axe arbitraire. Spécifie le degré de rotation autour d'un axe arbitraire.

Exemple

L'instruction suivante effectue une rotation de 20 degrés sur chaque axe. Puisque la propriété `transform` du modèle correspond aux décalages de position, de rotation et de redimensionnement du modèle par rapport au parent de ce dernier et que `preRotate` applique la modification d'orientation avant tout autre effet existant de la propriété `transform` de ce modèle, ce dernier subira une rotation sur place plutôt qu'autour de son parent.

```
-- Lingo
member("scene").model("bip01").transform.preRotate(20, 20, 20)

// Javascript
member("scene").getPropRef("model", i).transform.preRotate(20, 20, 20) ;
// where i is the number index of the model "bip01"
```

L'instruction ci-dessus équivaut à la suivante :

```
member("scene").model("bip01").rotate(20,20,20).
// javascript
member("scene").getPropRef("model", i).rotate(20,20,20) ;
// where i is the number index of the model "bip01"
```

Généralement, `preRotate()` ne se révèle utile qu'avec les variables de transformation. Cette ligne fait orbiter la caméra autour du point (100, 0, 0) dans l'espace, de 180 degrés autour de l'axe des y.

```
-- Lingo
t = transform()
t.position = member("scene").camera[1].transform.position
t.preRotate(vector(100, 0, 0), vector(0, 1, 0), 180)
member("scene").camera[1].transform = t

// javascript
var t = transform() ;
t.position = member("scene").getPropRef("camera", 1).transform.position ;
t.preRotate(vector(100, 0, 0), vector(0, 1, 0), 180) ;
member("scene").getPropRef("camera",1).transform = t ;
```

Voir aussi

`rotate`

preScale()

Syntaxe

```
transformReference.preScale( xScale, yScale, zScale )
transformReference.preScale( vector )
member( whichCastmember ).node.transform.preScale( xScale, yScale, zScale )
member( whichCastmember ).node.transform.preScale( vector )
```

Description

Commande 3D de transformation ; applique une échelle avant d'appliquer les effets de position, de rotation et de redimensionnement existants de la transformation en question.

`Node` peut être une référence à un modèle, un groupe, une lumière ou une caméra.

Paramètres

`xScale` Requis en cas d'application d'une échelle à l'aide des axes des *x*, des *y* et des *z*. Spécifie l'échelle autour de l'axe des *x*.

yScale Requis en cas d'application d'une échelle à l'aide des axes des *x*, des *y* et des *z*. Spécifie l'échelle autour de l'axe des *y*.

zScale Requis en cas d'application d'une échelle à l'aide des axes des *x*, des *y* et des *z*. Spécifie l'échelle autour de l'axe des *z*.

vector Requis en cas d'application d'une échelle à l'aide d'un vecteur. Spécifie le vecteur contenant l'échelle à appliquer.

Exemple

La **ligne 1** du code Lingo suivant crée un double de la transformation de Lune1. N'oubliez pas que l'accès à la propriété de transformation d'un modèle s'effectue par référence.

La **ligne 2** applique une échelle à cette transformation avant tout autre effet existant de position ou de rotation. Supposez que la transformation représente le décalage de position et l'orbite de rotation de Lune1 par rapport à sa planète parent. Supposez également que le parent de Lune2 est le même que celui de Lune1. Si nous utilisions ici `scale()` au lieu de `preScale()`, Lune2 serait placé deux fois plus loin et sa rotation autour de la planète se produirait deux fois plus souvent que Lune1. Ceci s'explique par le fait que le redimensionnement serait appliqué aux décalages de position et de rotation existants de la transformation. L'utilisation de `preScale()` applique la modification de taille sans affecter les décalages de position et de rotation existants.

La **ligne 3** applique une rotation supplémentaire de 180 degrés autour de l'axe des x de la planète. Cette opération place Lune2 du côté opposé à l'orbite de Lune1. Si la méthode `preRotate()` avait été utilisée, Lune2 serait resté à la même place que Lune1 et aurait été pivoté de 180 degrés autour de son propre axe des x.

La **ligne 4** affecte cette nouvelle transformation à Lune2.

```
-- Lingo
t = member("scene").model("Moon1").transform.duplicate()
t.preScale(2,2,2)
t.rotate(180,0,0)
member("scene").model("Moon2").transform = t

// Javascript
var t = member("scene").getPropRef("model", i).transform.duplicate() ;
t.preScale(2,2,2) ;
t.rotate(180,0,0) ;
member("scene").getPropRef("model", i).transform = t ;
// where i the number index of model " Moon2".
```

preTranslate()

Syntaxe

```
transformReference.preTranslate( xIncrement, yIncrement, zIncrement )
transformReference.preTranslate( vector )
member( whichCastmember ).node.transform.preTranslate(xIncrement, yIncrement, zIncrement)
member( whichCastmember ).node.transform.preTranslate( vector )

// Javascript
member( whichCastmember ).getProp("model",a).transform.preTranslate(xIncrement, yIncrement,
zIncrement) ;
```

Description

Commande 3D de transformation ; applique une translation avant les décalages de position, rotation et d'échelle de la transformation. La translation peut être spécifiée sous la forme d'un jeu de trois incréments le long des trois axes correspondants. Ces incréments peuvent être spécifiés explicitement sous la forme *xIncrement*, *yIncrement* et *zIncrement* ou au moyen d'un vecteur, où le composant x correspond à la translation autour de l'axe des x, le composant y à la translation autour de l'axe des y et le composant z à la translation autour de l'axe des z.

A la suite d'une série de transformations, effectuées dans l'ordre suivant, l'origine locale du modèle se situe à la position (0, 0, -100) si le parent du modèle est l'univers :

```
model.transform.identity()
model.transform.rotate(0, 90, 0)
model.transform.preTranslate(100, 0, 0)
```

Si la méthode `translate()` avait été utilisée à la place de `preTranslate()`, l'origine locale du modèle se serait située à la position (100, 0, 0) et le modèle aurait été pivoté de 90 degrés autour de son propre axe des y. L'instruction `model.transform.pretranslate(x, y, z)` est équivalente à `model.translate(x, y, z)`. La méthode `preTranslate()` n'est généralement utile qu'avec les variables de transformation plutôt qu'avec les références `model.transform`.

Paramètres

xIncrement Requis en cas d'application d'une translation à l'aide des axes des *x*, des *y* et des *z*. Spécifie la translation autour de l'axe des *x*.

yIncrement Requis en cas d'application d'une translation à l'aide des axes des *x*, des *y* et des *z*. Spécifie la translation autour de l'axe des *y*.

zIncrement Requis en cas d'application d'une translation à l'aide des axes des *x*, des *y* et des *z*. Spécifie la translation autour de l'axe des *z*.

vector Requis en cas d'application d'une translation à l'aide d'un vecteur. Spécifie le vecteur à utiliser dans la translation.

Exemple

```
-- Lingo
 t = transform()
t.transform.identity()
t.transform.rotate(0, 90, 0)
t.transform.preTranslate(100, 0, 0)
gbModel = member("scene").model("mars")
gbModel.transform = t
put gbModel.transform.position
-- vector(0.0000, 0.0000, -100.0000)

// Javascript

gbModel = member("scene").getProp("model" , a) ;
// where a is the number index for the mars model.
gbModel.transform.preTranslate(xIncrement , yIncrement, zIncrement) ;
member("scene").getProp("model" , a).transform.preTranslate(xIncrement , yIncrement,
zIncrement) ;
```

print()

Syntaxe

```
-- Lingo syntax
spriteObjRef.print({targetName, #printingBounds})

// JavaScript syntax
spriteObjRef.print({targetName, #printingBounds});
```

Description

Commande ; appelle la commande `print` ActionScript correspondante qui est apparue dans Flash 5. Toutes les images de l'animation Flash libellées `#p` sont imprimées. Si aucune image individuelle n'a été libellée, toute l'animation est imprimée.

L'impression d'animations Flash étant une opération relativement complexe, il est recommandé de lire la section consacrée à l'impression dans la documentation de Flash avant d'utiliser cette fonction d'image-objet.

Paramètres

`targetName` Facultatif. Spécifie le nom de l'animation ou du clip d'animation cible à imprimer. Si ce paramètre est omis (si la cible est égale à 0), l'animation Flash principale est imprimée.

`printingBounds` Facultatif. Spécifie les options relatives aux limites d'impression. Si ce paramètre est omis, les limites de l'animation cible sont utilisées. Si le paramètre *`printingBounds`* est spécifié, il doit prendre l'une des valeurs suivantes :

- `#bframe`. Si cette valeur est spécifiée, les limites d'impression de chaque page sont modifiées en fonction de chacune des images à imprimer.
- `#bmax`. Si cette valeur est spécifiée, les limites d'impression forment un cadre virtuel suffisamment grand pour contenir toutes les images à imprimer.

Exemple

L'instruction suivante imprime l'animation Flash présente en tant qu'acteur SWF.

```
-- Lingo
member("SWF").print()
// javascript
member("SWF").print() ;
```

printAsBitmap()

Syntaxe

```
-- Lingo syntax
spriteObjRef.printAsBitmap({targetName, #printingBounds})

// JavaScript syntax
spriteObjRef.printAsBitmap({targetName, #printingBounds});
```

Description

Commande d'image-objet Flash ; fonctionne de façon semblable à la commande `print`, mais n'est utilisable qu'avec les images-objets Flash. Toutefois, la commande `printAsBitmap` peut être utilisée pour imprimer des objets contenant des informations de couche alpha.

printFrom()

Syntaxe

```
-- Lingo syntax
_movie.printFrom(startFrameNameOrNum {, endFrameNameOrNum, redux})

// JavaScript syntax
_movie.printFrom(startFrameNameOrNum {, endFrameNameOrNum, redux});
```

Description

Méthode d'animation ; imprime tout le contenu de chaque image de la scène, que l'image soit sélectionnée ou non, à partir de l'image spécifiée par *startFrame*. Si vous le souhaitez, vous pouvez indiquer l'*endFrame* ainsi qu'une valeur de réduction (*redux*) (100, 50 ou 25 %).

L'image en cours d'impression n'a pas besoin d'être affichée. Cette commande imprime toujours en orientation portrait (verticale) et à une résolution de 72 points par pouce, en transformant en bitmaps tout le contenu de l'écran (ce qui peut parfois réduire la qualité du texte) ; de plus, elle ignore les paramètres de format d'impression. Pour augmenter la souplesse de l'impression depuis Director, consultez l'Xtra PrintOMatic Lite, qui se trouve sur le disque d'installation.

Paramètres

startFrameNameOrNum Requis. Chaîne ou nombre entier spécifiant le nom ou le numéro de la première image à imprimer.

endFrameNameOrNum Facultatif. Chaîne ou nombre entier spécifiant le nom ou le numéro de la dernière image à imprimer.

redux Facultatif. Nombre entier spécifiant la valeur de réduction. Les valeurs possibles sont 100, 50 et 25.

Exemple

L'instruction suivante imprime le contenu de la scène à l'image 1 :

```
-- Lingo syntax
_movie.printFrom(1)

// JavaScript syntax
_movie.printFrom(1);
```

L'instruction suivante imprime le contenu de chaque image de la scène entre l'image 10 et l'image 25. La réduction est de 50 %.

```
-- Lingo syntax
_movie.printFrom(10, 25, 50)

// JavaScript syntax
_movie.printFrom(10, 25, 50);
```

Voir aussi

Animation

propList()

Syntaxe

```
-- Lingo syntax
propList()
[:]
propList(string1, value1, string2, value2, ...)
propList(#symbol1, value1, #symbol2, value2, ...)
[#symbol1:value1, #symbol2:value2, ...]

// JavaScript syntax
propList();
propList(string1, value1, string2, value2, ...);
```

Description

Fonction de niveau supérieur ; crée une liste de propriétés dont chaque élément constitue une paire nom/valeur.

Lors de la création d'une liste de propriétés à l'aide de la syntaxe `propList()` ou `[:]` (Lingo uniquement), avec ou sans paramètres, l'index des valeurs de la liste commence par 1.

La longueur maximale d'une ligne de script exécutable est de 256 caractères. La fonction `propList()` ne permet pas de créer des listes de propriétés volumineuses. Pour créer une liste de propriétés incluant un important volume de données, indiquez ces données entre crochets (`[]`), insérez les données dans un champ, puis affectez le champ à une variable. Le contenu de cette variable correspondra à une liste de ces données.

Paramètres

`string1, string2, ...` Facultatif. Chaînes spécifiant la partie nom des éléments de la liste.

`value1, value2, ...` Facultatif. Valeurs spécifiant la partie valeur des éléments de la liste.

`#symbol1, #symbol2, ...` (Lingo uniquement) Facultatif. Symboles représentant la partie nom des éléments de la liste.

Exemple

L'instruction suivante crée une liste de propriétés avec différentes propriétés et valeurs, puis affiche les diverses valeurs de propriété dans la fenêtre Messages :

```
-- Lingo syntax
-- using propList()
colorList = propList(#top,"red", #sides,"blue", #bottom,"green")
-- using brackets
colorList = [#top:"red", #sides:"blue", #bottom:"green"]
put(colorList.top) -- "red"
put(colorList.sides) -- "blue"
put(colorList.bottom) -- "green"

// JavaScript syntax
var colorList = propList("top","red", "sides","blue", "bottom","green");
put(colorList.top); // red
put(colorList.sides); // blue
put(colorList.bottom); // green
```

Voir aussi

`list()`

proxyServer

Syntaxe

```
proxyServer serverType, "ipAddress", portNum
proxyServer()
```

Description

Commande ; définit les valeurs d'un serveur proxy FTP ou HTTP.

En l'absence de paramètres, la méthode `proxyServer()` renvoie les paramètres d'un serveur proxy FTP ou HTTP.

Paramètres

`serverType` Facultatif. Symbole spécifiant le type du serveur proxy. Ce paramètre peut recevoir la valeur `#ftp` ou `#http`.

`ipAddress` Facultatif. Chaîne spécifiant l'adresse IP.

`portNum` Facultatif. Nombre entier spécifiant le numéro de port.

Exemple

L'instruction suivante configure un serveur proxy HTTP à l'adresse IP 197.65.208.157 avec le port 5 :

```
-- Lingo
proxyServer #http,"197.65.208.157",5
// Javascript
proxyServer (symbol("http"),"197.65.208.157",5) ;
```

L'instruction suivante renvoie le numéro de port d'un serveur proxy HTTP :

```
-- Lingo
put proxyServer(#http,#port)

// Javascript
put (proxyServer(symbol("http"),symbol("port"))) ;
```

Si aucun type de serveur n'est spécifié, la fonction renvoie la valeur 1.

L'instruction suivante renvoie la chaîne de l'adresse IP d'un serveur proxy HTTP :

```
-- Lingo
put proxyServer(#http)

// Javascript
put (proxyServer(symbol("http")) ;
```

L'instruction suivante désactive un serveur proxy FTP :

```
proxyServer #ftp,#stop
// Javascript
proxyServer(symbol("ftp"),symbol("stop")) ;
```

ptToHotSpotID()

Syntaxe

```
-- Lingo syntax
spriteObjRef.ptToHotSpotID(point)
```

```
// JavaScript syntax
spriteObjRef.ptToHotSpotID(point);
```

Description

Fonction QuickTime VR ; renvoie l'identifiant de la zone référencée (si elle existe) présent au point spécifié. S'il n'existe pas de zone référencée, la fonction renvoie 0.

Paramètres

point Requis. Spécifie le point à tester.

puppetPalette()

Syntaxe

```
-- Lingo syntax
_movie.puppetPalette(palette {, speed} {, frames})

// JavaScript syntax
_movie.puppetPalette(palette {, speed} {, frames});
```

Description

Méthode d'animation ; asservit la piste des palettes et permet à un script d'outrepasser les paramètres de la piste des palettes du scénario et d'affecter des palettes à l'animation.

La méthode `puppetPalette()` définit comme palette en cours l'acteur palette spécifié par *palette*. Si le paramètre *palette* correspond à une chaîne, il spécifie le nom de bibliothèque de distribution de la palette. Si le paramètre *palette* est un nombre entier, il spécifie le numéro d'acteur de la palette.

Pour optimiser les résultats, utilisez la méthode `puppetPalette()` avant de passer à l'image sur laquelle l'effet se produira, afin que Director puisse effectuer une conversion dans la palette souhaitée avant de dessiner l'image suivante.

Vous pouvez faire apparaître progressivement la palette en remplaçant le paramètre *speed* par un nombre entier compris entre 1 (vitesse la plus lente) et 60 (vitesse la plus rapide). Vous pouvez également faire apparaître progressivement la palette sur plusieurs images en remplaçant *frames* par un nombre entier correspondant au nombre d'images.

Une palette asservie reste active jusqu'au moment de sa désactivation par le biais de la syntaxe `_movie.puppetPalette(0)`. Aucune autre modification de palette n'est apportée dans le scénario lorsque la palette asservie est active.

Remarque : *Le navigateur web contrôle la palette pour toute la page web. Par conséquent, Shockwave Player utilise toujours la palette du navigateur.*

Paramètres

palette Requis. Chaîne ou nombre entier spécifiant le nom ou le numéro de la nouvelle palette.

speed Facultatif. Nombre entier spécifiant la vitesse de l'apparition progressive de la palette. Les valeurs possibles sont comprises entre 1 et 60.

frames Facultatif. Nombre entier spécifiant le nombre d'images sur lesquelles la palette apparaît progressivement.

Exemple

L'instruction suivante définit Arc-en-ciel en tant que palette de l'animation :

```
-- Lingo syntax
_movie.puppetPalette("Rainbow")

// JavaScript syntax
_movie.puppetPalette("Rainbow");
```

L'instruction suivante définit Arc-en-ciel en tant que palette de l'animation. La transition vers la palette Arc-en-ciel s'effectue sur un laps de temps de 15 et sur 20 images.

```
-- Lingo syntax
_movie.puppetPalette("Rainbow", 15, 20)

// JavaScript syntax
_movie.puppetPalette("Rainbow", 15, 20);
```

Voir aussi

Animation

puppetSprite()

Syntaxe

```
-- Lingo syntax
_movie.puppetSprite(intSpriteNum, bool)

// JavaScript syntax
_movie.puppetSprite(intSpriteNum, bool);
```

Description

Méthode d'animation ; détermine si une piste d'image-objet est asservie et sous le contrôle d'un script (`TRUE`) ou si elle n'est pas asservie et est contrôlée par le scénario (`FALSE`).

Tant que la tête de lecture reste dans la même image-objet, la désactivation de l'asservissement de la piste d'image-objet à l'aide de la syntaxe `puppetSprite(`*`intSpriteNum`*`, FALSE)` rend à l'image-objet les propriétés définies dans le scénario.

Les propriétés initiales de la piste d'image-objet sont celles de la piste au moment de l'exécution de la méthode `puppetSprite()`. Vous pouvez utiliser un script pour modifier les propriétés d'image-objet comme suit :

- Si une piste d'image-objet est asservie, toute modification apportée par un script aux propriétés d'image-objet de la piste reste en vigueur une fois que la tête de lecture est sortie de l'image-objet.
- Si une piste d'image-objet n'est pas asservie, toute modification apportée par un script à une image-objet ne dure que jusqu'à la fin de l'image-objet en cours.

La piste doit contenir une image-objet lorsque vous utilisez la méthode `puppetSprite()`.

L'asservissement d'une piste d'image-objet vous permet de contrôler à partir d'un script de nombreuses propriétés d'image-objet, telles que `member`, `locH` et `width`, une fois que la tête de lecture est sortie de l'image-objet.

Utilisez la syntaxe `puppetSprite(`*`intSpriteNum`*`, FALSE)` pour rendre le contrôle au scénario lorsque vous avez fini de contrôler une piste d'image-objet à partir d'un script et pour éviter des résultats imprévisibles qui risquent de se produire lorsque la tête de lecture se trouve dans des images non destinées à être asservies.

Remarque : *La version 6 de Director a introduit l'asservissement automatique qui, dans la plupart des cas, évite d'avoir à asservir explicitement une image-objet. Le contrôle explicite se révèle toujours utile pour conserver le contrôle complet du contenu d'une piste, même après la fin de la lecture de la plage d'une image-objet.*

Paramètres

intSpriteNum Requis. Nombre entier spécifiant la piste d'image-objet à tester.

bool Requis. Valeur booléenne spécifiant si une piste d'image-objet est contrôlée par un script (TRUE) ou par le scénario (FALSE).

Exemple

L'instruction suivante asservit l'image-objet de la piste 15 :

```
-- Lingo syntax
_movie.puppetSprite(15, TRUE)

// JavaScript syntax
_movie.puppetSprite(15, true);
```

This statement removes the puppet condition from the sprite in the channel numbered i + 1:

```
-- Lingo syntax
_movie.puppetSprite(i + 1, FALSE)

// JavaScript syntax
_movie.puppetSprite(i + 1, false);
```

Voir aussi

makeScriptedSprite(), Animation, Piste d'image-objet

puppetTempo()

Syntaxe

```
-- Lingo syntax
_movie.puppetTempo(intTempo)

// JavaScript syntax
_movie.puppetTempo(intTempo);
```

Description

Méthode d'animation ; asservit la piste des cadences et règle la cadence sur un nombre d'images spécifié.

Lorsque la piste des cadences est asservie, un script peut outrepasser le paramètre de cadence du scénario et modifier la cadence de l'animation.

Il n'est pas nécessaire de désactiver la cadence asservie pour que les modifications ultérieurement apportées à la cadence dans le scénario prennent effet.

Remarque : *Bien que la méthode* `puppetTempo()` *permette théoriquement d'atteindre des cadences d'image de 30 000 ips (images par seconde), ce résultat ne serait possible qu'avec très peu d'animation et une machine extrêmement puissante.*

Paramètres

intTempo Requis. Nombre entier spécifiant la cadence.

Exemple

L'instruction suivante fixe la cadence de l'animation sur 30 images par seconde :

```
-- Lingo syntax
_movie.puppetTempo(30)

// JavaScript syntax
_movie.puppetTempo(30);
```

L'instruction suivante augmente l'ancienne cadence de l'animation de 10 images par seconde :

```
-- Lingo syntax
_movie.puppetTempo(oldTempo + 10)

// JavaScript syntax
_movie.puppetTempo(oldTempo + 10);
```

Voir aussi

Animation

puppetTransition()

Syntaxe

```
-- Lingo syntax
_movie.puppetTransition(memberObjRef)
_movie.puppetTransition(int {, time} {, size} {, area})

// JavaScript syntax
_movie.puppetTransition(memberObjRef);
_movie.puppetTransition(int {, time} {, size} {, area});
```

Description

Méthode d'animation ; exécute la transition spécifiée entre l'image en cours et l'image suivante.

Pour utiliser un acteur Xtra de transition, utilisez la syntaxe `puppetTransition(`*`memberObjRef`*`)`.

Pour utiliser une transition Director intégrée, remplacez *`int`* par l'une des valeurs du tableau suivant. Remplacez *`time`* par le nombre de quarts de seconde utilisés pour effectuer la transition. La valeur minimum est 0 ; la valeur maximum est 120 (30 secondes). Remplacez *`size`* par le nombre de pixels dans chaque bloc de la transition. La valeur minimum est 1 ; la valeur maximum est 128. Plus les blocs sont petits, plus la transition se fait en douceur, mais plus elle est lente.

Code	Transition	Code	Transition
01	Balayage vers la droite	27	Rangées aléatoires
02	Balayage vers la gauche	28	Colonnes aléatoires
03	Balayage vers le bas	29	Recouvrir vers le bas
04	Balayage vers le haut	30	Recouvrir vers le bas à gauche
05	Centre vers les bords, horizontal	31	Recouvrir vers le bas à droite
06	Bords vers centre, horizontal	32	Recouvrir vers la gauche
07	Centre vers les bords, vertical	33	Recouvrir vers la droite
08	Bords vers centre, vertical	34	Recouvrir vers le haut
09	Centre vers les bords, carré	35	Recouvrir vers le haut à gauche

Code (Suite)	Transition (Suite)	Code (Suite)	Transition (Suite)
10	Bords vers centre, carré	36	Recouvrir vers le haut à droite
11	Pousser vers la gauche	37	Stores vénitiens
12	Pousser vers la droite	38	Damier
13	Pousser vers le bas	39	Bandes vers la gauche en bas
14	Pousser vers le haut	40	Bandes vers la droite en bas
15	Révéler vers le haut	41	Bandes vers le bas à gauche
16	Révéler vers le haut à droite	42	Bandes vers le haut à gauche
17	Révéler vers la droite	43	Bandes vers le bas à droite
18	Révéler vers le bas à droite	44	Bandes vers le haut à droite
19	Révéler vers le bas	45	Bandes vers la gauche sur le dessus
20	Révéler vers le bas à gauche	46	Bandes vers la droite sur le dessus
21	Révéler vers la gauche	47	Zoom ouvert
22	Révéler vers le haut à gauche	48	Zoom fermé
23	Fondu, pixels rapides*	49	Stores verticaux
24	Fondu, rectangles carrés	50	Fondu, bits rapides*
25	Fondu, carrés d'encadrement	51	Fondu, pixels*
26	Fondu, motifs	52	Fondu, bits*

Les transitions identifiées par un astérisque (*) ne fonctionnent pas sur les moniteurs 32 bits.

Il n'y a pas de lien direct entre une faible valeur de durée et une transition rapide. La vitesse réelle de la transition dépend de la relation entre *size* et *time*. Par exemple, si le paramètre *size* présente une valeur d'un pixel, la transition prend plus de temps, quelle que soit la valeur de durée, car cette opération représente un travail intense pour l'ordinateur. Pour accélérer les transitions, augmentez la taille des blocs au lieu de réduire la durée.

Remplacez *area* par une valeur qui détermine si la transition se produit uniquement dans la zone modifiée (TRUE) ou sur toute la scène (FALSE, valeur par défaut). La variable *area* est une zone dans laquelle les images-objets ont changé.

Paramètres

memberObjRef Requis en cas d'utilisation d'un acteur Xtra de transition. Référence à l'acteur Xtra à utiliser comme transition.

int Requis en cas d'utilisation d'une transition Director intégrée. Nombre entier spécifiant le numéro de la transition à utiliser.

time Facultatif. Nombre entier spécifiant le nombre de quarts de seconde utilisés pour effectuer la transition. Les valeurs possibles sont comprises entre 0 et 120.

size Facultatif. Nombre entier spécifiant le nombre de pixels dans chaque bloc de la transition. Les valeurs possibles sont comprises entre 1 et 128.

area Facultatif. Valeur booléenne spécifiant si la transition se produit uniquement dans la zone modifiée (TRUE) ou sur toute la scène (FALSE).

Exemple

L'instruction suivante effectue une transition de type balayage vers la droite. Aucune valeur n'étant spécifiée pour *zone*, la transition se produit sur toute la scène, ce qui correspond à la valeur par défaut.

```
-- Lingo syntax
_movie.puppetTransition(1)

// JavaScript syntax
_movie.puppetTransition(1);
```

L'instruction suivante effectue une transition de type balayage vers la gauche sur toute la scène pendant 1 seconde avec une taille de bloc de 20 pixels :

```
-- Lingo syntax
_movie.puppetTransition(2, 4, 20, FALSE)

// JavaScript syntax
_movie.puppetTransition(2, 4, 20, false);
```

Voir aussi

Animation

put()

Syntaxe

```
-- Lingo syntax
put(value)

// JavaScript syntax
put(value);
```

Description

Fonction de niveau supérieur ; évalue une expression et affiche le résultat dans la fenêtre Messages.

Cette méthode remplit la même fonction que la méthode de niveau supérieur `trace()` qui est disponible à la fois en syntaxe Lingo et JavaScript.

Cette méthode peut servir d'outil de débogage pour suivre la valeur des variables au cours de la lecture d'une animation.

Paramètres

valeur Requis. Expression à évaluer.

Exemple

L'instruction suivante affiche l'heure dans la fenêtre Messages :

```
-- Lingo syntax
put(_system.time())

// JavaScript syntax
put(_system.time());
```

L'instruction suivante affiche la valeur affectée à la variable `bid` dans la fenêtre Messages :

```
-- Lingo syntax
bid = "Johnson"
put(bid) -- "Johnson"

// JavaScript syntax
var bid = "Johnson";
put(bid); // Johnson
```

Voir aussi

trace()

qtRegisterAccessKey()

Syntaxe

```
-- Lingo syntax
qtRegisterAccessKey(categoryString, keyString)

// JavaScript syntax
qtRegisterAccessKey(categoryString, keyString);
```

Description

Commande ; permet l'enregistrement d'une clé pour un média QuickTime chiffré.

La clé agit au niveau applicatif et non au niveau système. Une fois que l'application annule l'enregistrement de la clé ou se ferme, le média n'est plus accessible.

Remarque : *Pour raisons de sécurité, il est impossible d'afficher la liste de toutes les clés enregistrées.*

Voir aussi

qtUnRegisterAccessKey()

qtUnRegisterAccessKey()

Syntaxe

```
-- Lingo syntax
qtUnRegisterAccessKey(categoryString, keyString)

// JavaScript syntax
qtUnRegisterAccessKey(categoryString, keyString);
```

Description

Commande ; permet d'annuler l'enregistrement d'une clé pour un média QuickTime chiffré.

La clé agit au niveau applicatif et non au niveau système. Une fois l'enregistrement de la clé annulé par l'application, seules les animations chiffrées avec cette clé peuvent toujours être lues. Les autres médias ne sont plus accessibles.

Voir aussi

qtRegisterAccessKey()

queue()

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.queue(memberObjRef)
soundChannelObjRef.queue(propList)

// JavaScript syntax
soundChannelObjRef.queue(memberObjRef);
soundChannelObjRef.queue(propList);
```

Description

Méthode de piste audio ; ajoute un acteur son à la file d'attente d'une piste audio.

Dès qu'un son est placé en file d'attente, il peut être lu immédiatement à l'aide de la méthode `play()`. Ceci s'explique par le fait que Director précharge une certaine partie de chaque son placé en file d'attente afin d'éviter les délais d'attente entre la méthode `play()` et le début de la lecture. Cette proportion du son préchargée s'élève par défaut à 1 500 millisecondes. Vous pouvez modifier ce paramètre en transmettant une liste de propriétés contenant un ou plusieurs paramètres par l'intermédiaire de la méthode `queue()`. Ces paramètres peuvent également être transmis avec la méthode `setPlayList()`.

Vous pouvez voir un exemple d'utilisation de `queue()` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

`réfObjActeur` Requis en cas de spécification d'un acteur son. Référence à l'acteur son à placer en file d'attente.

`listeDesPropriétés` Requis en cas de transmission d'une liste de propriétés en tant que paramètres. Liste de propriétés applicable à l'acteur son à placer en file d'attente. Ces propriétés sont :

Propriété	Description
#member	Acteur à placer en file d'attente. Cette propriété doit être spécifiée. Toutes les autres sont facultatives.
#startTime	Position temporelle de départ de la lecture du son, en millisecondes. La valeur par défaut est le début du son. Pour plus d'informations, reportez-vous à l'entrée `startTime`.
#endTime	Position temporelle de fin de la lecture du son, en millisecondes. La valeur par défaut est la fin du son. Pour plus d'informations, reportez-vous à l'entrée `endTime`.
#loopCount	Nombre de répétitions d'une boucle défini avec `#loopStartTime` et `#loopEndTime`. La valeur par défaut est 1. Pour plus d'informations, reportez-vous à l'entrée `loopCount`.
#loopStartTime	Position temporelle de départ de la boucle, en millisecondes. Pour plus d'informations, reportez-vous à l'entrée `loopStartTime`.
#loopEndTime	Position temporelle de fin de la boucle, en millisecondes. Pour plus d'informations, reportez-vous à l'entrée `loopEndTime`.
#preloadTime	Quantité de son à placer en mémoire tampon avant la lecture, en millisecondes. Pour plus d'informations, reportez-vous à l'entrée `preloadTime`.

Exemple

Le gestionnaire suivant place en file d'attente et lit deux sons. Le premier son, l'acteur Carillons, est lu dans son intégralité. Le second son, l'acteur Intro, est lu à partir de ses 3 secondes et exécute cinq lectures en boucle successives, de 8 secondes à 8,9 secondes, et s'arrête au point 10 secondes.

```
-- Lingo syntax
on playMusic
    sound(2).queue(member("Chimes"))
    sound(2).queue([#member:member("introMusic"), #startTime:3000, #endTime:10000,
#loopCount:5, #loopStartTime:8000, #loopEndTime:8900])
    sound(2).play()
end playMusic

// JavaScript syntax
function playMusic() {
    sound(2).queue(member("Chimes"))
    sound(2).queue(propList("member",member("introMusic"), "startTime",3000,
"endTime",10000, "loopCount",5, "loopStartTime",8000, "loopEndTime",8900));
    sound(2).play();
}
```

Voir aussi

endTime, loopCount, loopEndTime, loopStartTime, pause() (piste audio), play() (piste audio), preLoadTime, setPlayList(), Piste audio, startTime, stop() (piste audio)

queue() (3D)

Syntaxe

```
member(whichCastmember).model(whichModel).bonesPlayer.queue(motionName {, looped,
startTime, endTime, scale, offset})
member(whichCastmember).model(whichModel).keyframePlayer.queue(motionName {, looped,
startTime, endTime, scale, offset})
```

Description

Commande 3D de modificateur `keyframePlayer` et `bonesPlayer` ; ajoute un mouvement spécifié à la fin de la propriété `playList` du modificateur. Le mouvement est exécuté par le modèle lorsque tous les mouvements qui le précèdent dans la liste de lecture ont été lus.

Paramètres

`nomDeMouvement` Requis. Spécifie le nom du mouvement à ajouter.

`looped` Facultatif. Spécifie si le mouvement est lu une seule fois (`FALSE`) ou continuellement (`TRUE`).

`startTime` Facultatif. Ce paramètre est mesuré en millisecondes à partir du début du mouvement. Lorsque *`looped`* présente la valeur `FALSE`, le mouvement démarre à la position *`offset`* et se termine au niveau de *`endTime`*. Lorsque *`looped`* présente la valeur `TRUE`, la première itération de la boucle démarre au niveau *`offset`* et se termine au niveau *`endTime`*. Toutes les répétitions suivantes démarrent au niveau *`startTime`* et se terminent au niveau *`endTime`*.

`endTime` Facultatif. Ce paramètre est mesuré en millisecondes à partir du début du mouvement. Lorsque *`looped`* présente la valeur `FALSE`, le mouvement démarre à la position *`offset`* et se termine au niveau de *`endTime`*. Lorsque *`looped`* présente la valeur `TRUE`, la première itération de la boucle démarre au niveau *`offset`* et se termine au niveau *`endTime`*. Toutes les répétitions suivantes démarrent au niveau *`startTime`* et se terminent au niveau *`endTime`*. Attribuez à *`endTime`* la valeur -1 si vous souhaitez que le mouvement soit lu jusqu'à la fin.

`échelle` Facultatif. Spécifie la cadence réelle de la lecture du mouvement. La valeur *`échelle`* est multipliée par la propriété `playRate` du modificateur `#keyframePlayer` ou `#bonesPlayer` du modèle pour déterminer la cadence réelle de la lecture du mouvement.

offset Facultatif. Ce paramètre est mesuré en millisecondes à partir du début du mouvement. Lorsque le paramètre *looped* présente la valeur FALSE, le mouvement démarre à la position *offset* et se termine au niveau *endTime*. Lorsque *looped* présente la valeur TRUE, la première itération de la boucle démarre au niveau *offset* et se termine au niveau *endTime*. Toutes les répétitions suivantes démarrent au niveau *startTime* et se terminent au niveau *endTime*.

Exemple

Le code Lingo suivant ajoute le mouvement Chute à la fin de la liste de lecture bonesPlayer du modèle Marcheur. Lorsque tous les mouvements précédant Chute dans la liste de lecture ont été exécutés, Chute est lu une fois du début à la fin.

```
sprite(1).member.model("Walker").bonesPlayer.queue("Fall", 0, 0, -1, 1, 0)
```

Le code Lingo suivant ajoute le mouvement coupDenvoi à la fin de la liste de lecture bonesPlayer du modèle Marcheur. Lorsque tous les mouvements précédant coupDenvoi dans la liste de lecture ont été exécutés, une section de coupDenvoi est lue en boucle. La première itération de la boucle démarre 2 000 millisecondes à compter du début du mouvement. Toutes les itérations suivantes de la boucle démarrent à 1 000 millisecondes du début de coupDenvoi et prennent fin à 5 000 millisecondes du début de coupDenvoi. La cadence de lecture est égale à trois fois la propriété playRate du modificateur bonesPlayer du modèle.

```
sprite(1).member.model("Walker").bonesPlayer.queue("Kick", 1, 1000, 5000, 3, 2000)
```

Voir aussi

play() (3D), playNext() (3D), playRate (3D)

QuickTimeVersion()

Syntaxe

```
-- Lingo syntax
QuickTimeVersion()

// JavaScript syntax
QuickTimeVersion();
```

Description

Fonction ; renvoie une valeur à virgule flottante identifiant la version de QuickTime installée et remplace la fonction QuickTimePresent en vigueur.

Si plusieurs versions de QuickTime 3.0 (ou ultérieur) sont installées sous Windows, quickTimeVersion() renvoie le numéro de version le plus récent. Si une version antérieure à QuickTime 3.0 est installée, QuickTimeVersion() renvoie le numéro de version 2.1.2, quelle que soit la version installée.

Paramètres

Aucune.

Exemple

L'instruction suivante utilise quickTimeVersion() pour afficher dans la fenêtre Messages la version de QuickTime actuellement installée :

```
-- Lingo syntax
put(QuickTimeVersion())

// JavaScript syntax
```

```
put(QuickTimeVersion());
```

quit()

Syntaxe

```
-- Lingo syntax
_player.quit()

// JavaScript syntax
_player.quit();
```

Description

Méthode de lecteur ; permet de quitter Director ou une projection et d'accéder au Bureau Windows ou au Finder Mac.

Paramètres

Aucune.

Exemple

L'instruction suivante indique à l'ordinateur de quitter l'application et d'accéder au Bureau Windows ou au Finder Mac lorsque l'utilisateur appuie sur Ctrl+Q (Windows) ou sur Cmd+Q (Mac) :

```
-- Lingo syntax
if (_key.key = "q" and _key.commandDown) then
    _player.quit()
end if

// JavaScript syntax
if (_key.key == "q" && _key.commandDown) {
    _player.quit();
}
```

Voir aussi

Lecteur

ramNeeded()

Syntaxe

```
-- Lingo syntax
_movie.ramNeeded(intFromFrame, intToFrame)

// JavaScript syntax
_movie.ramNeeded(intFromFrame, intToFrame);
```

Description

Méthode d'animation ; détermine la mémoire requise, en octets, pour afficher une plage d'images. Par exemple, vous pouvez tester la taille d'images contenant des illustrations en mode 32 bits : si la valeur de `ramNeeded()` est supérieure à celle de `freeBytes()`, utilisez des images contenant des illustrations en mode 8 bits et divisez par 1 024 pour convertir les octets en kilo-octets.

Paramètres

entImageInitiale Requis. Nombre entier spécifiant le numéro de la première image de la plage.

entImageFinale Requis. Nombre entier spécifiant le numéro de la dernière image de la plage.

Exemple

L'instruction suivante affecte à la variable `diceRoll` le nombre d'octets requis pour afficher les images 100 à 125 de l'animation :

```
-- Lingo syntax
frameSize = _movie.ramNeeded(100, 125)

// JavaScript syntax
var frameSize = _movie.ramNeeded(100, 125);
```

L'instruction suivante détermine si la mémoire requise pour afficher les images 100 à 125 excède la mémoire disponible et, le cas échéant, passe à la section utilisant des acteurs comportant un nombre de couleurs inférieur :

```
-- Lingo syntax
if (_movie.ramNeeded(100, 125) > _system.freeBytes) then
    _movie.go("8-bit")
end if

// JavaScript syntax
if (_movie.ramNeeded(100, 125) > _system.freeBytes) {
    _movie.go("8-bit");
}
```

Voir aussi

freeBytes(), Animation

random()

Syntaxe

```
-- Lingo syntax
random(integerExpression)

// JavaScript syntax
random(integerExpression);
```

Description

Fonction de niveau supérieur ; renvoie un nombre entier aléatoire compris entre 1 et une valeur spécifiée. Cette fonction peut s'utiliser pour modifier les valeurs d'une animation et peut servir notamment à faire varier les trajectoires dans les jeux, à affecter des nombres aléatoires ou à changer la couleur ou la position des images-objets.

Pour faire commencer un ensemble de nombres aléatoires par un nombre différent de 1, soustrayez la quantité appropriée de la fonction `random()`. Par exemple, l'expression `random(n + 1) - 1` utilise une plage comprise entre 0 et le nombre `n`.

Paramètres

expressionEntière Requis. Spécifie la valeur maximale du nombre aléatoire.

Exemple

L'instruction suivante affecte des valeurs aléatoires à la variable `diceRoll` :

```
-- Lingo syntax
diceRoll = (random(6) + random(6))

// JavaScript syntax
var diceRoll = (random(6) + random(6));
```

L'instruction suivante modifie de façon aléatoire la couleur du premier plan de l'image-objet 10 :

```
-- Lingo syntax
sprite(10).foreColor = (random(256) - 1)

// JavaScript syntax
sprite(10).foreColor = (random(256) - 1);
```

Le gestionnaire suivant choisit de manière aléatoire celui des deux segments de l'animation qui est lu :

```
-- Lingo syntax
on SelectScene
    if (random(2) = 2) then
        _movie.go("11a")
    else
        _movie.go("11b")
    end if
end

// JavaScript syntax
function SelectScene() {
    if (random(2) == 1) {
        _movie.go("11a");
    } else {
        _movie.go("11b");
    }
}
```

L'instruction suivante produit un multiple de 5 aléatoire compris entre 5 et 100 :

```
-- Lingo syntax
theScore = (5 * random(20))

// JavaScript syntax
var theScore = (5 * random(20));
```

randomVector()

Syntaxe

```
-- Lingo syntax
randomVector()

// JavaScript syntax
randomVector();
```

Description

Fonction de niveau supérieur ; renvoie un vecteur unitaire décrivant un point choisi de manière aléatoire à la surface d'une sphère unitaire.

Cette fonction diffère de `vector(random(10)/10.0, random(10)/10.0, random(10)/10.0)` dans la mesure où le vecteur résultant de l'utilisation de `randomVector()` constitue systématiquement un vecteur unitaire.

Un vecteur unitaire présente toujours une longueur de 1.

Paramètres

Aucune.

Exemple

Les instructions suivantes créent et affichent deux vecteurs unitaires définis de manière aléatoire dans la fenêtre Messages :

```
-- Lingo syntax
vec1 = randomVector()
vec2 = randomVector()
put(vec1 & RETURN & vec2)

// JavaScript syntax
var vec1 = randomVector();
var vec2 = randomVector();
put(vec1 + "\n" + vec2);
```

Voir aussi

vector()

randomVector

Syntaxe

```
randomVector()
```

Description

Commande 3D ; renvoie un vecteur unitaire qui décrit un point choisi de manière aléatoire à la surface d'une sphère unitaire. Cette méthode diffère de `vector( random(10)/10.0, random(10)/10.0, random(10)/10.0)` dans la mesure où le vecteur résultant constitue systématiquement un vecteur unitaire.

Paramètres

Aucune.

Exemple

Les instructions suivantes créent et affichent deux vecteurs unitaires définis de manière aléatoire dans la fenêtre Messages :

```
vec = randomVector()
put vec
-- vector(-0.1155, 0.9833, -0.1408)
vec2 = randomVector()
put vec2
-- vector(0.0042, 0.8767, 0.4810)
```

Voir aussi

getNormalized, generateNormals(), normalize

rawNew()

Syntaxe

```
parentScript.rawNew()
```

```
rawNew(parentScript)
```

Description

Fonction ; crée un objet enfant à partir d'un script parent sans appeler son gestionnaire `on new`. Cela permet la création d'objets enfants sans initialiser leurs propriétés. Cette fonction s'avère particulièrement pratique lors de la création d'un grand nombre d'objets enfants pour une utilisation ultérieure. Pour initialiser les propriétés de l'un de ces objets enfants bruts, appelez son gestionnaire `on new`.

Paramètres

Aucune.

Exemple

L'instruction suivante crée un objet enfant appelé voitureRouge à partir du script parent ScriptParentDeVoiture, sans initialiser ses propriétés :

```
RedCar = script("CarParentScript").rawNew()
```

L'instruction suivante initialise les propriétés de l'objet enfant voitureRouge :

```
RedCar.new()
```

Voir aussi

`new()`, `script()`

readChar()

Syntaxe

```
-- Lingo syntax
fileioObjRef.readChar()

// JavaScript syntax
fileioObjRef.readChar();
```

Description

Méthode FileIO ; lit et renvoie le caractère suivant d'un fichier.

Vous devez avoir ouvert un fichier à l'aide de la méthode `openFile()` avant d'utiliser la méthode `readChar()` pour lire un caractère. En cas de lecture de fichiers unicode, `readchar()` ne renvoie que l'octet suivant, et non le caractère unicode. Pour lire des caractères unicode, utilisez la méthode `readFile()`.

Paramètres

Aucune.

Exemple

L'instruction suivante ouvre le fichier c:\xtra.txt avec l'autorisation Lecture/écriture et lit tous les caractères de ce fichier jusqu'à ce qu'elle rencontre le caractère « e ».

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
repeat while(numToChar(objFileio.readChar()) <> 'e')
    put numToChar(objFileio.readChar())
end repeat
```

```
// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
while(numToChar(objFileio.readChar()) != 'e')
{
    trace(numToChar(objFileio.readChar()));
}
```

Voir aussi

Fileio, openFile()

readFile()

Syntaxe

```
-- Lingo syntax
fileioObjRef.readFile()

// JavaScript syntax
fileioObjRef.readFile();
```

Description

Méthode FileIO ; lit un fichier spécifié entre la position en cours et la fin de ce fichier, puis renvoie le résultat sous forme de chaîne.

Vous devez avoir ouvert un fichier à l'aide de la méthode `openFile()` avant d'utiliser la méthode `readFile()` pour lire un fichier.

Paramètres

Aucune.

Exemple

L'instruction suivante ouvre le fichier c:\xtra.txt avec l'autorisation Lecture/écriture et en lit le contenu.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
contents =objFileio.readFile()
put contents

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
var contents = objFileio.readFile();
trace(contents);
```

Voir aussi

Fileio, openFile()

readLine()

Syntaxe

```
-- Lingo syntax
```

```
fileioObjRef.readLine()

// JavaScript syntax
fileioObjRef.readLine();
```

Description

Méthode FileIO ; lit la ligne suivante d'un fichier, y compris le retour chariot suivant, puis renvoie le résultat sous forme de chaîne.

Vous devez avoir ouvert un fichier à l'aide de la méthode `openFile()` avant d'utiliser la méthode `readLine()` pour lire une ligne.

Paramètres

Aucune.

Exemple

L'instruction suivante ouvre le fichier c:\xtra.txt avec l'autorisation Lecture/écriture et en lit la première ligne.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
contents =objFileio.readLine()
put contents

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
contents = objFileio.readLine();
trace(contents);
```

Voir aussi

`Fileio`, `openFile()`

readToken()

Syntaxe

```
-- Lingo syntax
fileioObjRef.readToken(stringSkip, stringBreak)

// JavaScript syntax
fileioObjRef.readToken(stringSkip, stringBreak);
```

Description

Méthode FileIO ; lit le jeton suivant et le renvoie sous forme de chaîne.

Vous devez avoir ouvert un fichier à l'aide de la méthode `openFile()` avant d'utiliser la méthode `readToken()` pour lire un jeton.

Paramètres

`chaîneDébut` Requis. Chaîne spécifiant le jeu de caractères après lequel le jeton démarre. La chaîne *`chaîneDébut`* n'est pas incluse dans la chaîne renvoyée.

`chaîneFin` Requis. Chaîne spécifiant le jeu de caractères avant lequel le jeton se termine. La chaîne *`chaîneFin`* n'est pas incluse dans la chaîne renvoyée.

Exemple

L'instruction suivante ouvre le fichier c:\xtra.txt avec l'autorisation Lecture/écriture et récupère le jeton commençant juste après la chaîne « D » et se terminant juste avant la chaîne « g » dans la chaîne « Director est un grand produit ».

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
contents =objFileio.readToken("D","g")
put contents

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
contents = objFileio.readToken("D","g");
trace(contents);
```

Le résultat renvoyé est « irector est un ».

Voir aussi

`Fileio`, `openFile()`

readWord()

Syntaxe

```
-- Lingo syntax
fileioObjRef.readWord()

// JavaScript syntax
fileioObjRef.readWord();
```

Description

Méthode FileIO ; lit le mot suivant d'un fichier et le renvoie sous forme de chaîne.

Vous devez avoir ouvert un fichier à l'aide de la méthode `openFile()` avant d'utiliser la méthode `readWord()` pour lire un mot.

Paramètres

Aucune.

Exemple

L'instruction suivante ouvre le fichier c:\xtra.txt avec l'autorisation Lecture/écriture et en lit le premier mot.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
contents =objFileio.readWord()
put contents

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
contents = objFileio.readWord();
trace(contents);
```

Voir aussi

Fileio, openFile()

realPlayerNativeAudio()

Syntaxe

```
-- Lingo syntax
realPlayerNativeAudio()

// JavaScript syntax
realPlayerNativeAudio();
```

Description

Fonction RealMedia ; permet d'obtenir ou de définir un indicateur global déterminant si la partie audio de l'acteur RealMedia est traitée par RealPlayer® (TRUE) ou par Director (FALSE). Cette fonction renvoie la valeur précédente de l'indicateur.

Pour être efficace, cet indicateur doit être défini avant le premier chargement de RealPlayer (lors de la rencontre du premier acteur RealMedia dans le scénario ou de la première référence Lingo à un acteur RealMedia). Toute modification apportée à cet indicateur après le chargement de RealPlayer est ignorée. Cet indicateur devrait être exécuté dans un gestionnaire d'événement prepareMovie d'un script d'animation. Cet indicateur est défini pour la session entière (depuis le moment où Shockwave Player est chargé jusqu'à sa fermeture et son redémarrage) et non uniquement pour la durée de l'animation en cours.

Par défaut, cet indicateur est défini sur FALSE et le traitement audio est lancé par Director, ce qui permet de définir la propriété soundChannel et d'utiliser les méthodes et propriétés audio standard de Lingo en vue du traitement du flux audio d'une image-objet RealMedia, par exemple le mixage d'un élément RealAudio® avec d'autres composants audio de Director. Si cet indicateur est défini sur TRUE, le contrôle de la piste audio par Lingo n'est pas effectué et le son est traité par RealPlayer.

Paramètres

Aucune.

Exemple

Le code suivant indique que la fonction realPlayerNativeAudio() présente la valeur FALSE, ce qui signifie que la partie audio de l'acteur RealMedia est traitée par Director :

```
-- Lingo syntax
put(realPlayerNativeAudio())
-- 0

// JavaScript syntax
trace(realPlayerNativeAudio());
// 0
```

Le code suivant attribue à la fonction realPlayerNativeAudio() la valeur TRUE, ce qui signifie que la partie audio du flux RealMedia est traitée par RealPlayer et que le contrôle de la piste audio par Lingo n'est pas pris en compte :

```
-- Lingo syntax
realPlayerNativeAudio(TRUE)

// JavaScript syntax
realPlayerNativeAudio(1);
```

Voir aussi

soundChannel (RealMedia)

realPlayerPromptToInstall()

Syntaxe

```
-- Lingo syntax
realPlayerPromptToInstall()

// JavaScript syntax
realPlayerPromptToInstall();
```

Description

Fonction RealMedia ; permet d'obtenir ou de définir un indicateur global déterminant si la détection automatique et les alertes de RealPlayer 8 sont activées (TRUE) ou non (FALSE).

Par défaut, cette fonction est définie sur TRUE, ce qui signifie que si les utilisateurs ne disposent pas de la version 8 de RealPlayer et tentent de charger une animation contenant RealMedia, un message s'affiche automatiquement pour leur demander s'ils souhaitent accéder au site Web de RealNetworks® et installer RealPlayer. Vous pouvez définir cet indicateur sur FALSE si vous souhaitez créer votre propre système de détection et d'alerte à l'aide de la fonction « realPlayerVersion() », page 516 et de code personnalisé. Si cet indicateur est défini sur FALSE et qu'un autre système de détection et d'alerte RealPlayer 8 n'est pas mis en place, les utilisateurs non équipés de RealPlayer peuvent charger des animations contenant des acteurs RealMedia, mais les images-objets RealMedia n'apparaissent pas.

Cette fonction détecte le numéro de la version de RealPlayer installée sur le système pour déterminer si RealPlayer 8 est installé. Sous Windows, les numéros de version 6.0.8.132 ou supérieurs indiquent que RealPlayer 8 est installé. Sur les systèmes Mac, les composants de base RealPlayer portant le numéro de version 6.0.7.1001 ou ultérieur indiquent que RealPlayer 8 est installé.

Cet indicateur devrait être exécuté dans un gestionnaire d'événement prepareMovie d'un script d'animation.

Cette fonction renvoie la valeur précédente de l'indicateur.

Paramètres

Aucune.

Exemple

Le code suivant indique que la fonction realPlayerPromptToInstall() présente la valeur TRUE, ce qui signifie que les utilisateurs non équipés de RealPlayer sont invités à l'installer :

```
-- Lingo syntax
put(realPlayerPromptToInstall()) -- 1

// JavaScript syntax
trace(realPlayerPromptToInstall()); // 1
```

Le code suivant attribue à la fonction realPlayerPromptToInstall() la valeur FALSE, ce qui signifie que les utilisateurs ne sont pas invités à installer RealPlayer à moins que vous n'ayez créé un système de détection et d'alerte :

```
-- Lingo syntax
realPlayerPromptToInstall(FALSE)


// JavaScript syntax
realPlayerPromptToInstall(0);
```

realPlayerVersion()

Syntaxe

```
-- Lingo syntax
realPlayerVersion()

// JavaScript syntax
realPlayerVersion();
```

Description

Fonction RealMedia ; renvoie une chaîne identifiant le numéro de version du logiciel RealPlayer installé sur le système de l'utilisateur ou une chaîne vide si RealPlayer n'est pas installé. RealPlayer 8 ou une version ultérieure doit être installé sur votre ordinateur pour visualiser des animations Director intégrant du contenu RealMedia. Sous Windows, les numéros de version 6.0.8.132 ou supérieurs indiquent que RealPlayer 8 est installé. Sur les systèmes Mac, les composants de base RealPlayer portant le numéro de version 6.0.7.1001 ou ultérieur indiquent que RealPlayer 8 est installé.

Le rôle de cette fonction consiste à vous permettre de créer votre propre système de détection et d'alerte RealPlayer si vous ne souhaitez pas utiliser celui qui est fourni par la fonction « realPlayerPromptToInstall() », page 515.

Si vous choisissez de créer votre propre système de détection et d'alerte à l'aide de la fonction `realPlayerVersion()`, effectuez les opérations suivantes :

- Appelez `realPlayerPromptToInstall(FALSE)` (par défaut, cette fonction est définie sur `TRUE`) avant que des acteurs RealMedia soient référencés dans Lingo ou apparaissent dans le scénario. Cette fonction devrait être définie dans un gestionnaire d'événement `prepareMovie` d'un script d'animation.
- Utilisez la propriété système `xtraList` pour vérifier si l'Xtra pour RealMedia (RealMedia Asset.x32) est répertorié dans la boîte de dialogue Xtras de l'animation. La fonction `realPlayerVersion()` ne fonctionne pas si l'Xtra pour RealMedia est absent.

Le numéro de version renvoyé par cette fonction est identique au numéro de version que vous pouvez afficher dans RealPlayer.

Pour afficher le numéro de version de RealPlayer sous Windows :

1. Démarrez RealPlayer.
2. Choisissez A propos de RealPlayer dans le menu Aide.

   Dans la fenêtre qui s'affiche, le numéro de version apparaît dans la partie supérieure de l'écran, au niveau de la seconde ligne.

Pour afficher le numéro de version de RealPlayer sur le Mac :

1. Démarrez RealPlayer.
2. Choisissez A propos de RealPlayer dans le menu Apple®.

   La boîte de dialogue A propos de RealPlayer apparaît. Ignorez le numéro de version indiqué dans la seconde ligne, dans la partie supérieure de l'écran ; il est incorrect.
3. Cliquez sur le bouton d'infos sur la version.

   La boîte de dialogue d'informations de version de RealPlayer s'affiche.
4. Sélectionnez Composants de base de RealPlayer dans la liste des composants installés.

Le numéro de version affiché pour le composant de base RealPlayer (par exemple, 6.0.8.1649) est identique à celui qui est renvoyé par `realPlayerVersion()`.

Paramètres

Aucune.

Exemple

Le code suivant indique que le numéro de version du logiciel RealPlayer® installé sur le système est 6.0.9.357 :

```
-- Lingo syntax
put(realPlayerVersion())

// JavaScript syntax
put(realPlayerVersion());
```

recordFont

Syntaxe

```
recordFont(whichCastMember, font {[,face]} {,[bitmapSizes]} {,characterSubset} {,
userFontName})
```

Description

Commande ; inclut une police TrueType ou Type 1 comme acteur. Une fois incluses, ces polices sont disponibles à l'auteur tout comme les autres polices installées sur le système.

Vous devez créer un acteur police vide à l'aide de la commande `new()` avant d'utiliser `recordFont`.

La commande crée une police shockée dans *`quelActeur`* en utilisant la police nommée dans le paramètre *`police`*. La valeur renvoyée par la commande indique si l'opération a réussi. La valeur zéro indique que l'opération a réussi.

Paramètres

`police` Requis. Spécifie le nom de la police initiale à enregistrer.

`style` Facultatif. Spécifie une liste de symboles indiquant le style de la police initiale. Les valeurs possibles sont `#plain`, `#bold` et `#italic`. Si vous ne définissez aucune valeur pour ce paramètre, la valeur `#plain` est utilisée.

`tailleDesBitmaps` Facultatif. Spécifie une liste d'entiers indiquant les tailles pour lesquelles les bitmaps doivent être enregistrés. Ce paramètre peut être vide. Si vous omettez ce paramètre, aucun bitmap n'est généré. Ces bitmaps donnent généralement de meilleurs résultats pour les petites tailles (inférieures à 14 points), mais occupent davantage de mémoire.

`sousEnsembleDeCaractères` Facultatif. Spécifie une chaîne de caractères à coder. Seuls les caractères spécifiés sont disponibles dans la police. Si ce paramètre est omis, tous les caractères sont codés. Si seuls certains caractères sont codés mais qu'un caractère non codé est utilisé, ce caractère apparaît comme une case vide.

`nouveauNom` Facultatif. Spécifie une chaîne à utiliser en tant que nom de l'acteur police nouvellement enregistré.

Exemple

L'instruction suivante crée une police shockée simple n'utilisant que les deux arguments pour l'acteur et la police à enregistrer :

```
-- Lingo
myNewFontMember = new(#font)
recordFont(myNewFontMember, "Lunar Lander")
```

```
// Javascript
var myNewFontMember = new(symbol("font")) ;
myNewFontMember.recordFont( "Lunar Lander") ;
```

L'instruction suivante spécifie les tailles de bitmaps à générer et les caractères pour lesquels les données de police doivent être créées :

```
-- Lingo
myNewFontMember = new(#font)
recordfont(mynewmember,"lunar lander",[],[14, 18, 45], "Lunar Lander Game High Score First
Last Name")

// Javascript
var myNewFontMember = new(symbol("font")) ;
recordfont(mynewmember,"lunar lander",[],[14, 18, 45], "Lunar Lander Game High Score First
Last Name") ;
```

Remarque : *La méthode* `recordFont` *resynthétisant les données de la police au lieu de les utiliser directement, la distribution des polices shockées n'est soumise à aucune restriction légale.*

Voir aussi

`newMember()`

rect()

Syntaxe

```
-- Lingo syntax
rect(intLeft, intTop, intRight, intBottom)

// JavaScript syntax
rect(intLeft, intTop, intRight, intBottom);
```

Description

Fonction de niveau supérieur ; définit un rectangle.

Vous pouvez effectuer des opérations arithmétiques sur les rectangles aussi bien en syntaxe Lingo qu'en syntaxe JavaScript. Si vous ajoutez une seule valeur à un rectangle, la syntaxe Lingo ou JavaScript l'ajoute à chaque élément du rectangle.

Vous pouvez faire référence aux composants de rectangles avec les syntaxes de liste ou de propriétés. Par exemple, les affectations suivantes définissent `monRectangleLargeur1` et `monRectangleLargeur2` sur 50 :

```
// JavaScript syntax
var myRect = rect(40,30,90,70);
var myRectWidth1 = myRect.right - myRect.left; // 50
var myRectWidth2 = myRect[3] - myRect[1]; // 50
```

Vous pouvez voir un exemple d'utilisation de la fonction `rect()` dans une animation en consultant l'animation Imaging du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

`entGauche` Requis. Nombre entier spécifiant le nombre de pixels entre le côté gauche du rectangle et le bord gauche de la scène.

entHaut Requis. Nombre entier spécifiant le nombre de pixels entre le côté supérieur du rectangle et le bord supérieur de la scène.

entDroit Requis. Nombre entier spécifiant le nombre de pixels entre le côté droit du rectangle et le bord gauche de la scène.

entBas Requis. Nombre entier spécifiant le nombre de pixels entre le côté inférieur du rectangle et le bord supérieur de la scène.

Exemple

L'instruction suivante définit la variable `newArea` sur un rectangle dont le côté gauche est placé à 100 pixels, le côté supérieur à 150 pixels, le côté droit à 300 pixels et le côté inférieur à 400 pixels :

```
-- Lingo syntax
newArea = rect(100, 150, 300, 400)

// JavaScript syntax
var newArea = rect(100, 150, 300, 400);
```

Dans Lingo uniquement, les instructions suivantes définissent la variable `newArea` sur le rectangle défini par les points `firstPoint` et `secondPoint` :

```
-- Lingo syntax
firstPoint = point(100, 150)
secondPoint = point(300, 400)
newArea = rect(firstPoint, secondPoint)
```

Dans Lingo uniquement, les instructions suivantes additionnent et soustraient des valeurs pour les rectangles :

```
-- Lingo syntax
put(rect(0, 0, 100, 100) + rect(30, 55, 120, 95)) -- rect(30, 55, 220, 195)
put(rect(0, 0, 100, 100) -rect(30, 55, 120, 95)) -- rect(-30, -55, -20, 5)
```

Dans Lingo uniquement, l'instruction suivante ajoute 80 à chaque coordonnée d'un rectangle :

```
-- Lingo syntax
put(rect(60, 40, 120, 200) + 80) -- rect(140, 120, 200, 280)
```

Dans Lingo uniquement, l'instruction suivante divise chaque coordonnée d'un rectangle par 3 :

```
-- Lingo syntax
put(rect(60, 40, 120, 200) / 3) -- rect(20, 13, 40, 66)
```

Voir aussi

`point()`, `quad`

registerForEvent()

Syntaxe

```
member(whichCastmember).registerForEvent(eventName, handlerName, scriptObject {, begin,
period, repetitions})
```

Description

Commande 3D ; déclare le gestionnaire spécifié comme étant celui à appeler lorsque l'événement spécifié se produit à l'intérieur de l'acteur spécifié.

Les descriptions de paramètre suivantes s'appliquent aux deux commandes `registerForEvent()` et `registerScript()`.

Remarque : *Vous pouvez associer l'enregistrement d'un script à un nœud spécifique plutôt qu'à un acteur à l'aide de la commande* `registerScript()`.

Paramètres

`nomDévénement` Requis. Spécifie le nom de l'événement. Il peut s'agir de l'un quelconque des événements prédéfinis suivants ou de tout événement personnalisé :

- `#collideAny` est un événement de collision.
- `#collideWith` est un événement de collision impliquant ce modèle. La commande `setCollisionCallback()` est un raccourci de la commande `registerScript()` pour l'événement `#collideWith`.
- `#animationStarted` et `#animationEnded` sont des événements de notification utilisés au démarrage ou à l'arrêt de la lecture d'une animation de segments ou d'images-clés. Le gestionnaire reçoit trois arguments : *`nomDévénement`*, *`mouvement`* et *`position`*. L'argument *`nomDévénement`* a pour valeur `#animationStarted` ou `#animationEnded`. L'argument *`mouvement`* est le nom du mouvement dont la lecture a démarré ou a pris fin, *`position`* étant la position en cours du mouvement.
- Pour les animations en boucle, l'événement `#animationStarted` n'est émis que pour la première boucle, et non pour les suivantes. Cet événement est envoyé au début de la fusion entre deux animations.
- Lorsqu'une série d'animations est placée en file d'attente pour le modèle et que la propriété `autoBlend` de l'animation présente la valeur `TRUE`, l'événement `#animationEnded` peut se produire avant la fin apparente d'un mouvement donné. En effet, la propriété `autoBlend` peut encore donner une impression de mouvement alors que l'animation s'est terminée comme prévu.
- `#timeMS` est un événement horaire. Le premier événement `#timeMS` se produit une fois que le nombre de millisecondes spécifié dans le paramètre *`début`* s'est écoulé après l'appel de `registerForEvent`. Le paramètre *`période`* détermine le nombre de millisecondes entre les événements `#timeMS` lorsque la valeur de *`répétitions`* est supérieure à 0. Si la valeur de *`répétitions`* est égale à 0, l'événement `#timeMS` se produit indéfiniment.

`nomDeGestionnaire` Requis. Spécifie le nom du gestionnaire qui est appelé lorsque l'événement *`nomDévénement`* se produira ; ce gestionnaire se trouve dans l'objet script indiqué par *`objetScript`*. Le gestionnaire reçoit les arguments suivants :

- `type` est toujours égal à 0.
- `delta` est le temps (en millisecondes) écoulé depuis le dernier événement `#timeMS`.
- `time` est le nombre de millisecondes écoulées depuis le premier événement `#timeMS`. Par exemple, s'il existe trois itérations d'une période de 500 ms, la première itération est de 0, la deuxième de 500 et la troisième de 1 000.
- `duration` est le nombre total de millisecondes écoulées entre l'appel `registerForEvent` et le dernier événement `#timeMS`. Par exemple, avec cinq itérations d'une période de 500 ms, la durée est 2 500 ms. Pour les tâches avec des itérations illimitées, la durée est 0.
- `systemTime` est la durée absolue, en millisecondes, depuis le début de l'animation Director.

`objetScript` Requis. Spécifie l'objet script contenant le gestionnaire *`nomDeGestionnaire`*. Si 0 est spécifié pour *`objetScript`*, le premier gestionnaire d'événement portant le nom donné dans un script d'animation est appelé.

`début` Facultatif. Spécifie le nombre de millisecondes au bout duquel le premier événement `#timeMS` survient après l'appel de la fonction `registerForEvent()`.

période Facultatif. Spécifie le nombre de millisecondes entre les événements `#timeMS` lorsque la valeur de *répétitions* est supérieure à 0.

répétitions Facultatif. Spécifie le nombre de répétitions pour l'événement `#timeMS`. Si la valeur *répétitions* est égale à 0, l'événement `#timeMS` se produit indéfiniment.

Exemple

L'instruction suivante enregistre le gestionnaire d'événement `promptUser` détecté dans un script d'animation pour qu'il soit appelé à deux reprises à 5 secondes d'intervalle :

```
member("Scene").registerForEvent(#timeMS, #promptUser, 0, 5000, 5000, 2)
```

L'instruction suivante enregistre le gestionnaire d'événement `promptUser` détecté dans un script d'animation pour qu'il soit appelé chaque fois qu'une collision se produit au sein de l'acteur Séquence :

```
member("Scene").registerForEvent(#collideAny, #promptUser, 0)
```

L'instruction suivante déclare que le gestionnaire `promptUser` du même script que celui contenant la commande `registerForEvent` doit être appelé lorsqu'un objet entre en collision avec le modèle Pluton dans l'acteur Séquence :

```
member("Scene").registerForEvent(#collideWith, #promptUser, me,
member("Scene").model("Pluto"))
```

Voir aussi

`setCollisionCallback()`, `registerScript()`, `play() (3D)`, `playNext() (3D)`, `autoblend`, `blendTime`, `sendEvent`, `unregisterAllEvents`

registerScript()

Syntaxe

```
member(whichCastmember).model(whichModel).registerScript(eventName, handlerName,
scriptObject {, begin, period, repetitions})
member(whichCastmember).camera(whichCamera).registerScript(eventName, handlerName,
scriptObject {, begin, period, repetitions})
member(whichCastmember).light(whichLight).registerScript(eventName, handlerName,
scriptObject {, begin, period, repetitions})
member(whichCastmember).group(whichGroup).registerScript(eventName, handlerName,
scriptObject {, begin, period, repetitions})
```

Description

Commande 3D ; enregistre le gestionnaire spécifié comme devant être appelé lorsque l'événement spécifié se produit pour le nœud référencé.

Les descriptions de paramètre suivantes s'appliquent aux deux commandes `registerForEvent()` et `registerScript()`.

Paramètres

nomDévénement Requis. Spécifie le nom de l'événement. Il peut s'agir de l'un quelconque des événements prédéfinis suivants ou de tout événement personnalisé :

- `#collideAny` est un événement de collision.
- `#collideWith` est un événement de collision impliquant ce modèle. La commande `setCollisionCallback()` est un raccourci de la commande `registerScript()` pour l'événement `#collideWith`.

• `#animationStarted` et `#animationEnded` sont des événements de notification utilisés au démarrage ou à l'arrêt de la lecture d'une animation de segments ou d'images-clés. Le gestionnaire reçoit trois arguments : *`nomDévénement`*, *`mouvement`* et *`position`*. L'argument *`nomDévénement`* a pour valeur `#animationStarted` ou `#animationEnded`. L'argument *`mouvement`* est le nom du mouvement dont la lecture a démarré ou a pris fin, *`position`* étant la position en cours du mouvement.

Pour les animations en boucle, l'événement `#animationStarted` n'est émis que pour la première boucle, et non pour les suivantes. Cet événement est envoyé au début de la fusion entre deux animations.

Lorsqu'une série d'animations est placée en file d'attente pour le modèle et que la propriété `autoBlend` de l'animation présente la valeur `TRUE`, l'événement `#animationEnded` peut se produire avant la fin apparente d'un mouvement donné. En effet, la propriété `autoBlend` peut encore donner une impression de mouvement alors que l'animation s'est terminée comme prévu.

• `#timeMS` est un événement horaire. Le premier événement `#timeMS` se produit une fois que le nombre de millisecondes spécifié dans le paramètre *`début`* s'est écoulé après l'appel de `registerForEvent`. Le paramètre *`période`* détermine le nombre de millisecondes entre les événements `#timeMS` lorsque la valeur de *`répétitions`* est supérieure à 0. Si la valeur de *`répétitions`* est égale à 0, l'événement `#timeMS` se produit indéfiniment.

`nomDeGestionnaire` Requis. Spécifie le nom du gestionnaire qui est appelé lorsque l'événement *`nomDévénement`* se produira ; ce gestionnaire se trouve dans l'objet script indiqué par *`objetScript`*. Le gestionnaire reçoit les arguments suivants :

• `type` est toujours égal à 0.

• `delta` est le temps (en millisecondes) écoulé depuis le dernier événement `#timeMS`.

• `time` est le nombre de millisecondes écoulées depuis le premier événement `#timeMS`. Par exemple, s'il existe trois itérations d'une période de 500 ms, la première itération est de 0, la deuxième de 500 et la troisième de 1 000.

• `duration` est le nombre total de millisecondes écoulées entre l'appel `registerForEvent` et le dernier événement `#timeMS`. Par exemple, avec cinq itérations d'une période de 500 ms, la durée est 2 500 ms. Pour les tâches avec des itérations illimitées, la durée est 0.

• `systemTime` est la durée absolue, en millisecondes, depuis le début de l'animation Director.

`objetScript` Requis. Spécifie l'objet script contenant le gestionnaire *`nomDeGestionnaire`*. Si 0 est spécifié pour *`objetScript`*, le premier gestionnaire d'événement portant le nom donné dans un script d'animation est appelé.

`début` Facultatif. Spécifie le nombre de millisecondes au bout duquel le premier événement `#timeMS` survient après l'appel de la fonction `registerForEvent()`.

`période` Facultatif. Spécifie le nombre de millisecondes entre les événements `#timeMS` lorsque la valeur de *`répétitions`* est supérieure à 0.

`répétitions` Facultatif. Spécifie le nombre de répétitions pour l'événement `#timeMS`. Si la valeur *`répétitions`* est égale à 0, l'événement `#timeMS` se produit indéfiniment.

Exemple

L'instruction suivante enregistre le gestionnaire d'événement `messageReceived`, situé dans un script d'animation, pour qu'il soit appelé lorsque le modèle Lecteur reçoit l'événement personnalisé `#message` défini par l'utilisateur :

```
member("Scene").model("Player").registerScript(#message, #messageReceived, 0)
```

L'instruction suivante enregistre le gestionnaire d'événement `collisionResponder`, situé dans le même script que la commande `collisionResponder`, pour qu'il soit appelé chaque fois qu'une collision se produit entre le modèle Lecteur et tout autre modèle utilisant le modificateur `#collision` :

```
member("Scene").model("Player").registerScript(#collideWith, #collisionResponder, me)
```

Voir aussi

registerForEvent(), sendEvent, setCollisionCallback()

removeBackdrop

Syntaxe

```
member(whichCastmember).camera(whichCamera).removeBackdrop(index)
```

Description

Commande 3D ; supprime le fond trouvé à une position spécifiée de la liste des fonds de la caméra.

Paramètres

index Requis. Spécifie la position d'index du fond dans la liste des fonds.

Exemple

L'instruction suivante retire le troisième fond de la liste des fonds de la caméra 1 de l'acteur Séquence. Le fond disparaîtra de la scène si des images-objets utilisent actuellement cette caméra.

```
-- Lingo
member("Scene").camera[1].removeBackdrop(3)

// Javascript
member("Scene").getProp("camera", 1).removeBackdrop(3) ;
```

removeFromWorld

Syntaxe

```
member(whichCastmember).model(whichModel).removeFromWorld()
member(whichCastmember).light(whichLight).removeFromWorld()
member(whichCastmember).camera(whichCamera).removeFromWorld()
member(whichCastmember).group(whichGroup).removeFromWorld()
```

Description

Commande 3D ; pour les modèles, les lumières, les caméras ou les groupes dont la hiérarchie amont se termine dans l'univers, cette commande donne une valeur nulle aux parents et les retire de l'univers.

Pour les objets dont la hiérarchie amont ne se termine pas dans l'univers, cette commande n'a aucun effet.

Paramètres

Aucune.

Exemple

La commande suivante supprime le modèle gbCyl de l'univers 3D de l'acteur Séquence :

```
-- Lingo
member("Scene").model("gbCyl").removeFromWorld()

// Javascript
member("Scene").getPropRef("model" , a).removeFromWorld() ;
```

```
// where a is the number index for gbCyl model.
```

removeLast()

Syntaxe

```
member(whichCastmember).model(whichModel).bonesPlayer.removeLast()
member(whichCastmember).model(whichModel).keyframePlayer.removeLast()
```

Description

Commande 3D de modificateur `keyframePlayer` et `bonesPlayer` ; supprime le dernier mouvement de la liste de lecture du modificateur.

Paramètres

Aucune.

Exemple

L'instruction suivante supprime le dernier mouvement de la liste de lecture du modificateur `bonesPlayer` pour le modèle Marcheur :

```
member("MyWorld").model("Walker").bonesPlayer.removelast()
```

removeModifier

Syntaxe

```
member(whichCastmember).model(whichModel).removeModifier.(#whichModifier)
```

Description

Commande 3D ; supprime le modificateur spécifié du modèle spécifié.

Cette commande renvoie `TRUE` si elle est exécutée avec succès et `FALSE` si *`#quelModificateur`* n'est pas un modificateur valide ou si le modificateur n'est pas associé au modèle.

Paramètres

`quelModificateur` Requis. Spécifie le modificateur à supprimer.

Exemple

L'instruction suivante supprime le modificateur `#toon` du modèle Boîte :

```
-- Lingo
member("shapes").model("Box").removeModifier(#toon)

// JavaScript syntax
member("shapes").getPropRef("model" , i).removeModifier(symbol("toon"));
// where "i" is the number index.
```

Voir aussi

`addModifier`, `modifier`, `modifier[]`, `modifiers`

removeOverlay

Syntaxe

```
member(whichCastmember).camera(whichCamera).removeOverlay(index)
```

Description

Commande 3D ; supprime le recouvrement trouvé à une position spécifiée de la liste des recouvrements de la caméra.

Paramètres

index Requis. Spécifie la position d'index du recouvrement dans la liste des recouvrements.

Exemple

L'instruction suivante supprime le premier recouvrement de la liste de recouvrements de la caméra utilisée par l'image-objet 5. Le recouvrement disparaîtra de la scène.

```
-- Lingo
sprite(5).camera.removeOverlay(1)

// Javascript
sprite(5).camera.removeOverlay(1) ;
```

Voir aussi

overlay

removeScriptedSprite()

Syntaxe

```
-- Lingo syntax
spriteChannelObjRef.removeScriptedSprite()

// JavaScript syntax
spriteChannelObjRef.removeScriptedSprite();
```

Description

Méthode de piste d'image-objet ; demande qu'une piste d'image-objet contrôlée par un script soit de nouveau contrôlée par le scénario.

Paramètres

Aucune.

Exemple

L'instruction suivante supprime l'image-objet contrôlée par un script de la piste d'image-objet 5 :

```
-- Lingo syntax
channel(5).removeScriptedSprite()

// JavaScript syntax
channel(5).removeScriptedSprite();
```

Voir aussi

makeScriptedSprite(), Piste d'image-objet

resetWorld

Syntaxe

```
member(whichCastmember).resetWorld()
member(whichTextCastmember).resetWorld()
```

Description

Commande 3D ; réattribue aux propriétés de l'acteur 3D référencé les valeurs enregistrées lorsque l'acteur a été mis en mémoire pour la première fois. La propriété `state` de l'acteur doit présenter la valeur 0 (déchargé), 4 (chargé) ou -1 (erreur) pour que cette commande puisse être utilisée ; dans le cas contraire, une erreur de script survient.

Cette commande diffère de `revertToWorldDefaults` dans la mesure où les valeurs utilisées reflètent l'état de l'acteur au moment où ce dernier a été mis en mémoire pour la première fois, plutôt qu'au moment où il a été créé.

Paramètres

Aucune.

Exemple

L'instruction suivante redonne aux propriétés de l'acteur Séquence les valeurs utilisées lorsque l'acteur a été mis en mémoire pour la première fois.

```
-- Lingo
member("Scene").resetWorld()

// Javascript
member("Scene").resetWorld() ;
```

Voir aussi

`revertToWorldDefaults`

resolveA

Syntaxe

```
collisionData.resolveA(bResolve)
```

Description

Méthode 3D de collision ; annule le comportement de collision défini par la propriété `collision.resolve` pour `collisionData.modelA`. N'appelez cette fonction que si vous souhaitez remplacer le comportement défini pour modelA à l'aide de `collision.resolve`.

Paramètres

`bRésolution` Requis. Spécifie si la collision est résolue pour modelA. Si *`bRésolution`* présente la valeur `TRUE`, la collision est résolue pour modelA ; si *`bRésolution`* reçoit la valeur `FALSE`, la collision n'est pas résolue pour modelA.

Voir aussi

`collisionData`, `registerScript()`, `resolve`, `modelA`, `setCollisionCallback()`

resolveB

Syntaxe

```
collisionData.resolveB(bResolve)
```

Description

Méthode 3D de collision ; annule le comportement de collision défini par la propriété `collision.resolve` pour `collisionData.modelB`. N'appelez cette fonction que si vous souhaitez remplacer le comportement défini pour modelB à l'aide de `collision.resolve`.

Paramètres

`bRésolution` Requis. Spécifie si la collision est résolue pour modelB. Si *`bRésolution`* présente la valeur `TRUE`, la collision est résolue pour modelB ; si *`bRésolution`* reçoit la valeur `FALSE`, la collision n'est pas résolue pour modelB.

Voir aussi

`collisionData`, `resolve`, `registerScript()`, `modelB`, `setCollisionCallback()`

restart()

Syntaxe

```
-- Lingo syntax
_system.restart()

// JavaScript syntax
_system.restart();
```

Description

Méthode système ; ferme toutes les applications ouvertes et redémarre l'ordinateur.

Paramètres

Aucune.

Exemple

L'instruction suivante redémarre l'ordinateur lorsque l'utilisateur appuie sur la combinaison de touches Cmd+R (Mac) ou Ctrl+R (Windows) :

```
-- Lingo syntax
if (_key.key = "r" and _key.commandDown) then
 _system.restart()
end if

// JavaScript syntax
if (_key.key = "r" && _key.commandDown) {
 _system.restart();
}
```

Voir aussi

`Système`

restore()

Syntaxe

```
-- Lingo syntax
windowObjRef.restore()

// JavaScript syntax
windowObjRef.restore();
```

Description

Méthode de fenêtre ; restaure la taille initiale d'une fenêtre ayant été agrandie.

Utilisez cette méthode en cas de création de barres de titre personnalisées pour des animations dans une fenêtre.

Paramètres

Aucune.

Exemple

L'instruction suivante restaure la taille initiale de la fenêtre agrandie Tableau de commande :

```
-- Lingo syntax
window("Control Panel").restore()

// JavaScript syntax
window("Control Panel").restore();
```

Voir aussi

`maximize()`, `Fenêtre`

result

Syntaxe

```
the result
```

Description

Fonction ; affiche la valeur de l'expression renvoyée par le dernier gestionnaire exécuté.

La fonction `result` se révèle utile pour obtenir des valeurs provenant d'animations lues dans des fenêtres et pour suivre l'évolution de Lingo en affichant les résultats des gestionnaires dans la fenêtre Messages pendant la lecture de l'animation.

Pour renvoyer le résultat d'un gestionnaire, affectez ce résultat à une variable, puis vérifiez la valeur de cette dernière. Utilisez une instruction telle que `set maVariable = fonction()`, où *`fonction()`* est le nom d'une fonction spécifique.

Paramètres

Aucune.

Exemple

Le gestionnaire suivant renvoie un résultat aléatoire de deux dés :

```
on diceRoll
    return random(6) + random(6)
```

```
end
```

Dans l'exemple suivant, les deux instructions

```
diceRoll
roll = the result
```

sont équivalentes à l'instruction suivante :

```
set roll = diceRoll()
```

L'instruction `set roll = diceRoll` n'appelle pas le gestionnaire car, en l'absence de parenthèses après `diceRoll` ; `diceRoll` est considéré comme une référence de variable.

Voir aussi

return (mot-clé)

resume()

Syntaxe

```
-- Lingo syntax
animGifSpriteRef.resume()

// JavaScript syntax
animGifSpriteRef.resume();
```

Description

Méthode de GIF animé ; reprend la lecture de l'image-objet à partir de l'image située immédiatement après celle sur laquelle la lecture s'est arrêtée. Cette commande n'a aucun effet si l'image-objet GIF animé n'est pas en pause.

Paramètres

Aucune.

Voir aussi

rewind() (GIF animé, Flash)

returnToTitle()

Syntaxe

```
-- Lingo syntax
dvdObjRef.returnToTitle()

// JavaScript syntax
dvdObjRef.returnToTitle();
```

Description

Méthode de DVD ; reprend la lecture après l'affichage d'un menu.

Paramètres

Aucune.

Exemple

L'instruction suivante reprend la lecture après l'affichage d'un menu :

```
-- Lingo syntax
member(1).returnToTitle()

// JavaScript syntax
member(1).returnToTitle()
```

Voir aussi

DVD

revertToWorldDefaults

Syntaxe

```
member(whichCastmember).revertToWorldDefaults()
```

Description

Commande 3D ; redonne aux propriétés de l'acteur 3D spécifié les valeurs enregistrées lorsque l'acteur a été créé. La propriété `state` de l'acteur doit présenter la valeur 4 (chargé) ou -1 (erreur) pour que cette commande puisse être utilisée ; dans le cas contraire, une erreur de script survient.

Cette commande diffère de `resetWorld` dans la mesure où les valeurs utilisées reflètent l'état de l'acteur lorsque ce dernier a été créé plutôt que lorsqu'il a été chargé en mémoire pour la première fois.

Paramètres

Aucune.

Exemple

L'instruction suivante redonne aux propriétés de l'acteur Séquence les valeurs enregistrées lorsque l'acteur a été créé.

```
-- Lingo
member("Scene").revertToWorldDefaults()

// Javascript
member("Scene").revertToWorldDefaults() ;
```

Voir aussi

resetWorld

rewind() (piste audio)

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.rewind()

// JavaScript syntax
soundChannelObjRef.rewind();
```

Description

Méthode de piste audio ; interrompt la lecture du son en cours sur une piste audio et la redémarre à sa position de départ `startTime`.

Si le son est mis en pause, il reste en pause, avec la propriété `currentTime` définie sur la position de départ `startTime`.

Paramètres

Aucune.

Exemple

L'instruction suivante recommence la lecture de l'acteur son de la piste audio 1 depuis le début :

```
-- Lingo syntax
sound(1).rewind()

// JavaScript syntax
sound(1).rewind();
```

Voir aussi

Piste audio, startTime

rewind() (Windows Media)

Syntaxe

```
-- Lingo syntax
windowsMediaObjRef.rewind()

// JavaScript syntax
windowsMediaObjRef.rewind();
```

Description

Méthode d'image-objet ou d'acteur Windows Media. Repasse à la première image d'une image-objet ou d'un acteur Windows Media.

L'appel de cette méthode n'a aucun effet sur la propriété `mediaStatus`.

Paramètres

Aucune.

Voir aussi

mediaStatus (RealMedia, Windows Media), Windows Media

rewind() (GIF animé, Flash)

Syntaxe

```
-- Lingo syntax
animGifSpriteRef.rewind()

// JavaScript syntax
animGifSpriteRef.rewind();
```

Description

Commande ; renvoie une image-objet animation Flash ou GIF animé à l'image 1 lorsque l'image-objet est arrêtée ou en cours de lecture.

Paramètres

Aucune.

Exemple

Le script d'image suivant détermine si l'image-objet animation Flash dans laquelle le comportement était placé est en cours de lecture et, le cas échéant, continue la boucle dans la même image. Lorsque l'animation est terminée, l'image-objet la rembobine (si bien que la première image de l'animation apparaît sur la scène) et permet à la tête de lecture de passer à l'image suivante.

```
-- Lingo syntax
property spriteNum

    on exitFrame
    if sprite(spriteNum).playing then
        _movie.go(_movie.frame)
    else
        sprite(spriteNum).rewind()
        _movie.updatestage()
    end if
end

// JavaScript syntax
function exitFrame(me) {
    var plg = sprite(this.spriteNum).playing;
    if (plg == 1) {
        _movie.go(_movie.frame);
    } else {
        sprite(this.spriteNum).rewind();
        _movie.updateStage();
    }
}
```

rollOver()

Syntaxe

```
-- Lingo syntax
_movie.rollOver({intSpriteNum})

// JavaScript syntax
_movie.rollOver({intSpriteNum});
```

Description

Méthode d'animation ; indique si le pointeur (curseur) se trouve sur le rectangle de délimitation d'une image-objet spécifiée (`TRUE` ou 1) ou non (`FALSE` ou 0).

La méthode `rollOver()` est généralement utilisée dans les scripts d'image et se révèle utile pour créer des gestionnaires qui exécutent une action lorsque l'utilisateur place le pointeur sur une image-objet spécifique.

Si l'utilisateur continue à déplacer la souris, la valeur de `rollOver()` peut changer pendant qu'un script exécute un gestionnaire et risque donc de produire des résultats inattendus. Vous pouvez vous assurer qu'un gestionnaire utilise une valeur de survol constante en affectant `rollOver()` à une variable au moment du démarrage du gestionnaire.

Lorsque le pointeur se trouve sur une zone de la scène dans laquelle une image-objet est apparue précédemment, la méthode `rollOver()` se produit malgré tout et signale l'image-objet comme si elle se trouvait encore dans cette zone. Pour éviter ce problème, évitez d'effectuer des survols sur ces emplacements ou placez l'image-objet au-dessus de la barre de menus avant de la supprimer.

Paramètres

`numDImageObjet` Facultatif. Nombre entier spécifiant le numéro d'image-objet.

Exemple

L'instruction suivante change le contenu de l'acteur champ Message en « C'est bien là » lorsque le curseur se trouve sur l'image-objet 6 :

```
-- Lingo syntax
if (_movie.rollOver(6)) then
    member("Message").text = "This is the place."
end if

// JavaScript syntax
if (_movie.rollOver(6)) {
    member("Message").text = "This is the place.";
}
```

Le gestionnaire suivant positionne la tête de lecture sur d'autres images lorsque le curseur se trouve sur certaines images-objets de la scène. Il affecte d'abord la valeur `rollOver` à une variable. Ceci permet au gestionnaire d'utiliser la valeur rollOver en vigueur au démarrage du survol, que l'utilisateur continue ou non à déplacer la souris.

```
-- Lingo syntax
on exitFrame
    currentSprite = _movie.rollOver()
    case currentSprite of
        1: _movie.go("Left")
        2: _movie.go("Middle")
        3: _movie.go("Right")
    end case
end exitFrame

// JavaScript syntax
function exitFrame() {
    var currentSprite = _movie.rollOver();
    switch (currentSprite) {
        case 1: _movie.go("Left");
            break;
        case 2: _movie.go("Middle");
            break;
        case 3: _movie.go("Right");
            break;
    }
}
```

Voir aussi

Animation

rootMenu()

Syntaxe

```
-- Lingo syntax
dvdObjRef.rootMenu()

// JavaScript syntax
dvdObjRef.rootMenu();
```

Description

Méthode de DVD ; affiche le menu racine.

Paramètres

Aucune.

Exemple

L'instruction suivante affiche le menu racine :

```
-- Lingo syntax
member(1).rootMenu()

// JavaScript syntax
member(1).rootMenu();
```

Voir aussi

DVD

rotate

Syntaxe

```
member(whichCastmember).node(whichNode).rotate(xAngle, yAngle, zAngle {, relativeTo})
member(whichCastmember).node(whichNode).rotate(rotationVector {, relativeTo})
member(whichCastmember).node(whichNode).rotate(position, axis, angle {, relativeTo})
transform.rotate(xAngle, yAngle, zAngle {, relativeTo})
transform.rotate(rotationVector {, relativeTo})
transform.rotate(position, axis, angle {, relativeTo})
```

Description

Commande 3D ; applique une rotation après les décalages de position, de rotation et d'échelle d'un objet de transformation d'un nœud référencé ou d'un objet de transformation directement référencé. La rotation doit être spécifiée sous la forme d'un ensemble de trois angles, chacun desquels spécifiant l'angle de rotation autour des trois axes correspondants. Ces angles peuvent être spécifiés explicitement sous la forme *angleX*, *angleY* et *angleZ*ou au moyen d'un *vecteurDeRotation*, où le composant *x* du vecteur correspond à la rotation autour de l'axe des x, le composant *y* à la rotation autour de l'axe des y et le composant *z* à la rotation autour de l'axe des z. La rotation peut également être spécifiée autour d'un axe arbitraire passant par un point de l'espace.

Paramètres

angleX Requis en cas d'application d'une rotation à l'aide des axes des *x*, des *y* et des *z*. Spécifie l'angle de rotation autour de l'axe des *x*.

angleY Requis en cas d'application d'une rotation à l'aide des axes des *x*, des *y* et des *z*. Spécifie l'angle de rotation autour de l'axe des *y*.

angleZ Requis en cas d'application d'une rotation à l'aide des axes des *x*, des *y* et des *z*. Spécifie l'angle de rotation autour de l'axe des *z*.

vecteurDeRotation Requis en cas d'application d'une rotation à l'aide d'un vecteur. Spécifie le vecteur contenant les angles à appliquer.

position Requis en cas d'application d'une rotation autour d'un axe arbitraire passant par un point de l'espace. Spécifie la position dans l'espace.

axe Requis en cas d'application d'une rotation autour d'un axe arbitraire passant par un point de l'espace. Spécifie l'axe passant par la position spécifiée par le paramètre *position*.

angle Requis en cas d'application d'une rotation autour d'un axe arbitraire passant par un point de l'espace. Spécifie le degré de rotation autour de l'axe spécifié par le paramètre *axe*.

parRapportA Facultatif. Spécifie les axes du système de coordonnées utilisés pour appliquer les modifications de rotation souhaitées. Le paramètre *parRapportA* peut prendre l'une des valeurs suivantes :

- `#self` applique les incréments en fonction du système de coordonnées local du nœud (axes des x, des y et des z spécifiés pour le modèle en phase de création). Cette valeur est utilisée comme valeur par défaut si vous utilisez la commande `rotate` avec une référence de nœud et que le paramètre *parRapportA* n'est pas spécifié.
- `#parent` applique les incréments par rapport au système de coordonnées du parent du nœud. Cette valeur est utilisée comme valeur par défaut si vous utilisez la commande `rotate` avec une référence de transformation et que le paramètre *parRapportA* n'est pas spécifié.
- `#world` applique les incréments par rapport au système de coordonnées de l'univers. Lorsque le parent d'un modèle est l'univers, ceci équivaut à utiliser `#parent`.
- `nodeReference` vous permet de spécifier un nœud servant de base à la rotation, la commande appliquant les incréments en fonction du système de coordonnées du nœud spécifié.

Exemple

L'exemple suivant fait d'abord pivoter le modèle Lune autour de son propre axe des z (en le faisant pivoter sur place), puis le fait pivoter autour de son nœud parent, le modèle Terre (ce qui entraîne le déplacement du modèle Lune en orbite autour du modèle Terre).

```
member("Scene").model("Moon").rotate(0,0,15)
member("Scene").model("Moon").rotate(vector(0, 0, 5), member("Scene").model("Moon"))
```

L'exemple suivant fait pivoter le modèle Balle autour d'une position de l'espace occupée par le modèle Bâton. L'effet obtenu est le déplacement du modèle Balle en orbite autour du modèle Bâton dans le plan xy.

```
polePos = member("3d Scene").model("Pole").worldPosition
member("3d Scene").model("Ball").rotate(polePos, vector(0,0,1), 5, #world)
```

Voir aussi

pointAt, preRotate, rotation (transformation), rotation (matériau de gravure), rotation (fond et recouvrement), preScale(), transform (propriété)

run

Syntaxe

```
run(MUIObject)
```

Description

Cette commande affiche une boîte de dialogue modale générale créée à partir d'une instance de l'Xtra MUI.

Avant l'ouverture de la boîte de dialogue par Director, utilisez la commande Initialize pour définir la boîte de dialogue.

Utilisez la commande WindowOperation avec l'option #show pour ouvrir une boîte de dialogue non modale. Cette commande permet l'exécution d'autres instructions Lingo dans l'animation lorsqu'une boîte de dialogue non modale est ouverte.

Exemple

Ce gestionnaire vérifie l'existence de l'objet MUIObject et affiche une boîte de dialogue générale à partir de l'objet UIObject si l'instruction est la suivante :

```
--Lingo syntax
on runDialog
   global MUIObject
   if objectP(MUIObject) then
      run(MUIObject)
   end if
end
```

runMode

Syntaxe

```
the runMode
```

Description

Fonction ; renvoie une chaîne indiquant le mode de lecture de l'animation. Les valeurs possibles sont :

- `Author` : l'animation est lue dans Director.
- `Projector` : l'animation est lue en tant que projection.
- `BrowserPlugin` : l'animation est lue en tant que module externe Shockwave Player ou un autre environnement de programmation tel que LiveConnect ou ActiveX.

Le moyen le plus sûr pour tester des valeurs spécifiques de cette propriété consiste à utiliser l'opérateur `contains`. Cela évite les erreurs et permet les correspondances partielles.

Paramètres

Aucune.

Exemple

L'instruction suivante détermine si des paramètres externes sont disponibles et, le cas échéant, les obtient :

```
--Lingo syntax
if the runMode contains "Plugin" then
    -- decode the embed parameter
    if externalParamName(swURL) = swURL then
        put externalParamValue(swURL) into myVariable
    end if
end if

// JavaScript syntax
```

```
if (_system.environmentPropList.runMode.indexOf("Plugin") >=0) {
    // decode the embed parameter
    if (_player.externalParamName(swURL) == swURL) {
        myVariable = _player.externalParamValue(swURL);
    }
}
```

Voir aussi

environmentPropList, platform

save castLib

Syntaxe

```
castLib(whichCast).save()
save castLib whichCast {,pathName&newFileName}
```

Description

Commande ; enregistre les modifications apportées à la distribution dans son fichier d'origine ou dans un nouveau fichier. Les opérations ou les références ultérieures à la distribution utilisent l'acteur enregistré.

Cette commande ne fonctionne pas avec les fichiers compressés.

La commande `save CastLib` ne prend pas en charge les adresses URL en tant que références de fichier.

Paramètres

nomDuChemin&nomDeNouveauFichier Facultatif. Spécifie le chemin d'accès et le nom du fichier dans lequel enregistrer les modifications. Si ce paramètre est omis, la distribution initiale doit être liée.

Exemple

L'instruction suivante demande à Director d'enregistrer la version révisée de la distribution Boutons dans le nouveau fichier BoutonsActualisés au sein du même dossier :

```
-- Lingo
castLib("Buttons").save(the moviePath & "UpdatedButtons.cst")

// Javascript
castLib("Buttons").save(_movie.path & "UpdatedButtons.cst") ;
```

Voir aussi

@ (chemin d'accès)

saveMovie()

Syntaxe

```
-- Lingo syntax
_movie.saveMovie({stringFilePath})

// JavaScript syntax
_movie.saveMovie({stringFilePath});
```

Description

Méthode d'animation ; enregistre l'animation en cours.

L'inclusion du paramètre facultatif *chaîneCheminDeFichier* enregistre l'animation dans le fichier spécifié. Cette méthode ne fonctionne pas avec les fichiers compressés. Le nom du fichier spécifié doit comporter l'extension .dir.

La méthode `saveMovie()` ne prend pas en charge les adresses URL en tant que références de fichier.

Paramètres

chaîneCheminDeFichier Facultatif. Chaîne spécifiant le chemin d'accès et le nom du fichier dans lequel l'animation est enregistrée.

Exemple

L'instruction suivante enregistre l'animation actuelle dans le fichier MiseAjour :

```
-- Lingo syntax
_movie.saveMovie(_movie.path & "Update.dir")

// JavaScript syntax
_movie.saveMovie(_movie.path + "Update.dir");
```

Voir aussi

Animation

scale (commande)

Syntaxe

```
member(whichCastmember).node(whichNode).scale(xScale, yScale, zScale)
member(whichCastmember).node(whichNode).scale(uniformScale)
transform.scale(xScale, yScale, zScale)
transform.scale(uniformScale)
```

Description

Commande 3D de transformation ; applique un redimensionnement après les décalages de position, de rotation et d'échelle d'une transformation d'un nœud référencé ou d'une transformation directement référencée. Le redimensionnement doit être spécifié soit comme un groupe de trois redimensionnements individuels des axes correspondants, soit comme un redimensionnement unique à appliquer à tous les axes. Vous pouvez spécifier un redimensionnement individuel à l'aide des paramètres *échelleX*, *échelleY* et *échelleZ*ou spécifier une valeur de redimensionnement uniforme à l'aide du paramètre *échelleUniforme*.

Un nœud peut être un objet de caméra, groupe, lumière ou modèle. L'utilisation de la commande `scale` ajuste la propriété `transform.scale` du nœud référencé, mais ne produit aucun effet visuel sur les lumières ou sur les caméras car elles ne contiennent pas de géométrie.

Les valeurs du redimensionnement doivent être supérieures à zéro.

Paramètres

échelleX Requis en cas de spécification de trois redimensionnements. Spécifie l'échelle le long de l'axe des *x*.

échelleY Requis en cas de spécification de trois redimensionnements. Spécifie l'échelle autour de l'axe des *y*.

échelleZ Requis en cas de spécification de trois redimensionnements. Spécifie l'échelle autour de l'axe des *z*.

échelleUniforme Requis en cas de spécification d'un seul redimensionnement uniforme. Spécifie le redimensionnement uniforme.

Exemple

L'exemple suivant commence par afficher la propriété `transform.scale` du modèle Lune, redimensionne ensuite le modèle à l'aide de la commande `scale`, puis affiche la valeur `transform.scale` résultante.

```
-- Lingo
put member("Scene").model("Moon").transform.scale

// Javascript
put member("Scene").getProp("model", i).transform.scale ;
-- vector( 1.0000, 1.0000, 1.0000)
```

L'instruction suivante redimensionne le modèle Pluton de façon uniforme le long des trois axes selon la valeur 0,5, ce qui réduit de moitié la taille du modèle affiché.

```
-- Lingo
member("Scene").model("Pluto").scale(0.5)

// Javascript
member("Scene").getPropRef("model", a).scale(0.5);
// where a is the number index of the model "Pluto"
```

L'instruction suivante redimensionne le modèle Ovale de façon non uniforme en le modifiant le long de son axe des *z*, mais non de ses axes des *x* ou des *y*.

```
-- Lingo
member("Scene").model("Pluto").scale(0.0, 0.0, 0.5)

// Javascript
member("Scene").getPropRef("model", a).scale(0.0, 0.0, 0.5);
// where a is the number index of the model "Pluto"
```

Voir aussi

transform (propriété), preScale(), scale (transformation)

script()

Syntaxe

```
-- Lingo syntax
script(memberNameOrNum {, castNameOrNum})

// JavaScript syntax
script(memberNameOrNum {, castNameOrNum});
```

Description

Fonction de niveau supérieur ; crée une référence à un acteur donné contenant un script et, en option, spécifie la bibliothèque de distribution contenant cet acteur.

Une erreur est renvoyée si l'acteur donné ne contient aucun script ou s'il n'existe pas.

Paramètres

nomOuNumActeur Requis. Chaîne spécifiant le nom de l'acteur contenant un script ou nombre entier spécifiant la position d'index de cet acteur.

nomOuNumDistribution Facultatif. Chaîne spécifiant le nom de la bibliothèque de distribution contenant l'acteur *nomOuNumActeur* ou nombre entier spécifiant la position d'index de cette bibliothèque de distribution. Si ce paramètre est omis, la méthode `script()` recherche dans la première bibliothèque de distribution.

Exemple

Dans Lingo uniquement, les instructions suivantes vérifient si un objet enfant est une instance du script parent Fourmi :

```
-- Lingo syntax
if (bugObject.script = script("Warrior Ant")) then
    bugObject.attack()
end if
```

L'instruction suivante définit la variable `actionMember` sur l'acteur script Actions :

```
-- Lingo syntax
actionMember = script("Actions")

// JavaScript syntax
var actionMember = script("Actions");
```

scrollByLine()

Syntaxe

```
-- Lingo syntax
memberObjRef.scrollByLine(amount)

// JavaScript syntax
memberObjRef.scrollByLine(amount);
```

Description

Commande ; fait défiler l'acteur champ ou texte spécifié vers le haut ou vers le bas d'un nombre de lignes spécifié. Les lignes sont définies comme des lignes séparées par des retours chariot ou produites par un retour automatique.

Paramètres

`quantité` Requis. Spécifie le nombre de lignes à faire défiler. Lorsque *`quantité`* présente une valeur positive, le champ défile vers le bas. Lorsque *`quantité`* présente une valeur négative, le champ défile vers le haut.

Exemple

L'instruction suivante fait défiler l'acteur champ Nouvelles du jour de cinq lignes vers le bas :

```
--Lingo syntax
member("Today's News").scrollbyline(5)

// JavaScript syntax
member("Today's News").scrollbyline(5);
```

L'instruction suivante fait défiler l'acteur champ Nouvelles du jour de cinq lignes vers le haut :

```
--Lingo syntax
member("Today's News").scrollByLine(-5)

// JavaScript syntax
member("Today's News").scrollByLine(-5);
```

scrollByPage()

Syntaxe

```
-- Lingo syntax
memberObjRef.scrollByPage(amount)

// JavaScript syntax
memberObjRef.scrollByPage(amount);
```

Description

Commande ; fait défiler l'acteur champ ou texte spécifié vers le haut ou vers le bas d'un nombre de pages spécifié. Une page correspond au nombre de lignes de texte visibles à l'écran.

Paramètres

quantité Requis. Spécifie le nombre de pages à faire défiler. Lorsque *quantité* présente une valeur positive, le champ défile vers le bas. Lorsque *quantité* présente une valeur négative, le champ défile vers le haut.

Exemple

L'instruction suivante fait défiler l'acteur champ Nouvelles du jour d'une page vers le bas :

```
--Lingo syntax
member("Today's News").scrollbypage(1)

// JavaScript syntax
member("Today's News").scrollbypage(1);
```

L'instruction suivante fait défiler l'acteur champ Nouvelles du jour d'une page vers le haut :

```
--Lingo syntax
member("Today's News").scrollbypage(-1)

// JavaScript syntax
member("Today's News").scrollbypage(-1);
```

Voir aussi

scrollTop

seek()

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.seek(milliseconds)

// JavaScript syntax
memberOrSpriteObjRef.seek(milliseconds);
```

Description

Méthode d'acteur ou d'image-objet RealMedia ; modifie l'emplacement de lecture du flux multimédia vers l'emplacement spécifié par le nombre de millisecondes *écoulées depuis le début du flux*. La valeur `mediaStatus` devient généralement `#seeking`, puis `#buffering`.

Vous pouvez utiliser cette méthode pour initialiser la lecture à des points autres que le début du flux RealMedia ou pour avancer ou reculer dans le flux. Le nombre entier spécifié par le paramètre *millisecondes* correspond au nombre de millisecondes écoulées depuis le début du flux. Ainsi, pour reculer, vous devez spécifier un nombre de millisecondes inférieur, et non un nombre négatif.

Si la commande `seek` est appelée lorsque `mediaStatus` présente la valeur `#paused`, le flux repasse en mémoire tampon et reprend la valeur `#paused` au nouvel emplacement spécifié par `seek`. Si la méthode `seek` est appelée lorsque `mediaStatus` présente la valeur `#playing`, le flux repasse en mémoire tampon et sa lecture démarre automatiquement au nouvel emplacement du flux. Si la méthode `seek` est appelée lorsque `mediaStatus` présente la valeur `#closed`, rien ne se passe.

Si vous tentez de lancer une recherche au-delà de la valeur `duration` du flux, l'argument entier spécifié est ajouté à la plage à partir de 0 pour la durée du flux. Vous ne pouvez pas accéder directement à une image-objet RealMedia qui est en cours de lecture en flux continu.

L'instruction `x.seek(n)` est identique à `x.currentTime = n` et l'un ou l'autre de ces appels entraîne la remise en mémoire tampon du flux.

Paramètres

millisecondes Requis. Nombre entier spécifiant le nombre de millisecondes à partir du début du flux.

Exemple

Les exemples suivants définissent la position de lecture actuelle du flux sur 10 000 millisecondes (10 secondes) :

```
-- Lingo syntax
sprite(2).seek(10000)
member("Real").seek(10000)

// JavaScript syntax
sprite(2).seek(10000);
member("Real").seek(10000);
```

Voir aussi

duration (RealMedia, SWA), currentTime (RealMedia), play() (RealMedia, SWA, Windows Media), pause() (RealMedia, SWA, Windows Media), stop() (RealMedia, SWA, Windows Media), mediaStatus (RealMedia, Windows Media)

selectAtLoc()

Syntaxe

```
-- Lingo syntax
dvdObjRef.selectAtLoc(point(x, y))

// JavaScript syntax
dvdObjRef.selectAtLoc(point(x, y));
```

Description

Méthode de DVD ; sélectionne le bouton situé sous un point spécifié.

Cette méthode équivaut à pointer sur un bouton à l'aide de la souris.

Paramètres

point(x, y) Requis. Point de coordonnées de la scène spécifiant l'emplacement auquel un bouton est sélectionné.

Exemple

L'instruction suivante sélectionne le bouton situé sous un point spécifié :

```
-- Lingo syntax
member(10).selectAtLoc(point(50, 75))

// JavaScript syntax
member(10).selectAtLoc(point(50, 75));
```

Voir aussi

DVD

selectButton()

Syntaxe

```
-- Lingo syntax
dvdObjRef.selectButton(intButton)

// JavaScript syntax
dvdObjRef.selectButton(intButton);
```

Description

Méthode de DVD ; sélectionne un bouton spécifié.

Cette méthode renvoie la valeur `0` si l'opération a réussi.

Remarque : *Cette méthode n'est pas prise en charge dans Mac®-Intel®.*

Paramètres

`entButton` Requis. Nombre entier spécifiant le bouton sélectionné.

Exemple

L'instruction suivante sélectionne le bouton 5 :

```
-- Lingo syntax
sprite(11).selectButton(5)

// JavaScript syntax
sprite(11).selectButton(5);
```

Voir aussi

DVD

selectButtonRelative()

Syntaxe

```
-- Lingo syntax
dvdObjRef.selectButtonRelative(direction)

// JavaScript syntax
dvdObjRef.selectButtonRelative(direction);
```

Description
Méthode de DVD ; sélectionne un bouton par rapport à la position du bouton en cours dans le menu.

Paramètres
direction Requis. Symbole (Lingo) ou chaîne (syntaxe JavaScript) spécifiant la direction dans laquelle le déplacement doit être effectué à partir de la position du bouton en cours. Les valeurs possibles sont `left` ou `right`.

Remarque : *Cette méthode n'est pas prise en charge dans Mac®-Intel®.*

Exemple
L'instruction suivante entraîne la sélection du bouton situé à gauche du bouton en cours :

```
-- Lingo syntax
member(12).member.selectButtonRelative(#left)

// JavaScript syntax
member(12).member.selectButtonRelative("left");
```

Voir aussi
DVD

selection() (fonction)

Syntaxe
```
the selection
```

Description
Fonction ; renvoie une chaîne de caractères contenant la partie sélectionnée du champ modifiable actuel. Elle permet notamment de tester la sélection faite par l'utilisateur dans un champ.

La fonction `selection` indique uniquement la chaîne de caractères sélectionnée ; vous ne pouvez pas utiliser `selection` pour sélectionner une chaîne de caractères.

Paramètres
Aucune.

Exemple
L'instruction suivante vérifie si des caractères sont sélectionnés et, à défaut, affiche le message d'alerte « Veuillez sélectionner un mot » :

```
if the selection = EMPTY then alert "Please select a word."
```

Voir aussi
selStart, selEnd

sendAllSprites()

Syntaxe
```
-- Lingo syntax
_movie.sendAllSprites(stringEventMessage {, args})
```

```
// JavaScript syntax
_movie.sendAllSprites(stringEventMessage {, args});
```

Description

Méthode d'animation ; envoie un message désigné à toutes les images-objets, et non uniquement à l'image-objet impliquée dans l'événement. Comme tout autre message, celui-ci est envoyé à chaque script associé à l'image-objet, à moins que la méthode `stopEvent()` ne soit utilisée.

Pour de meilleurs résultats, n'envoyez le message qu'aux images-objets qui sont en mesure de le gérer correctement par le biais de la méthode `sendSprite()`. Aucune erreur ne se produira si le message est envoyé à toutes les images-objets, mais cela nuira néanmoins à la performance. La présence d'un même gestionnaire dans un comportement donné pour différentes images-objets risquant également de poser des problèmes, il est important d'utiliser des noms uniques pour les messages qui sont diffusés afin d'éviter tout conflit éventuel.

Une fois le message transmis à tous les comportements, l'événement suit la hiérarchie de message classique : script d'acteur, script d'image, puis script d'animation.

Lorsque vous utilisez la méthode `sendAllSprites()`, effectuez les opérations suivantes :

- Remplacez *`chaîneMessageDévénement`* par le message.
- Remplacez *`arguments`* par tout argument à envoyer avec le message.

Si aucune image-objet ne comporte de comportement associé contenant le gestionnaire donné, `sendAllSprites()` renvoie la valeur `FALSE`.

Paramètres

`chaîneMessageDévénement` Requis. Chaîne spécifiant le message à envoyer à toutes les images-objets.

`arguments` Facultatif. Argument(s) à envoyer avec le message.

Exemple

Le gestionnaire suivant envoie le message personnalisé `allSpritesShouldBumpCounter` et l'argument 2 à toutes les images-objets lorsque l'utilisateur clique avec la souris :

```
-- Lingo syntax
on mouseDown me
    _movie.sendAllSprites(#allspritesShouldBumpCounter, 2)
end

// JavaScript syntax
function mouseDown() {
    _movie.sendAllSprites("allspritesShouldBumpCounter", 2);
}
```

Voir aussi

`Animation`, `sendSprite()`, `stopEvent()`

sendEvent

Syntaxe

```
member(whichCastmember).sendEvent(#eventName, arg1, arg2,...)
```

Description

Commande 3D ; envoie un événement et un nombre arbitraire d'arguments à tous les scripts enregistrés pour recevoir l'événement. Utilisez `registerForEvent()` ou `setCollisionCallback()` pour enregistrer les scripts pour les événements.

Paramètres

`nomDévénement` Requis. Spécifie le nom de l'événement à envoyer.

`arg1, arg2, ...` Requis. Un ou plusieurs arguments envoyés avec l'événement *`nomDévénement`*.

Exemple

La première ligne de l'exemple suivant crée une instance d'un script parent appelé `tester`. La deuxième ligne définit le gestionnaire de l'instance de script, sautPluton, en tant que gestionnaire à appeler lorsque l'événement `#jump` est envoyé. La troisième ligne enregistre le gestionnaire de script d'animation `jumpMars` comme un autre gestionnaire à appeler lorsque l'événement `#saut` est envoyé. La quatrième ligne envoie l'événement `#jump`. Le gestionnaire `#jumpMars` d'un script d'animation et le gestionnaire `#jumpPluton` sont appelés, ainsi que tout autre gestionnaire enregistré pour l'événement `#jump`. Une valeur d'instance de script de 0 indique que vous enregistrez le gestionnaire d'un script d'animation, lequel diffère du gestionnaire d'une instance de comportement ou de l'enfant d'un script parent.

```
t = new (script "tester")
member("scene").registerForEvent(#jump, #jumpPluto, t)
member("scene").registerForEvent(#jump, #jumpMars, 0)
member("scene").sendEvent(#jump)
```

Voir aussi

`registerScript()`, `registerForEvent()`, `setCollisionCallback()`

sendSprite()

Syntaxe

```
-- Lingo syntax
_movie.sendSprite(spriteNameOrNum, event {, args})

// JavaScript syntax
_movie.sendSprite(spriteNameOrNum, event {, args});
```

Description

Méthode d'animation ; envoie un message à tous les scripts associés à une image-objet spécifiée.

Les messages envoyés à l'aide de `sendSprite()` sont envoyés à chacun des scripts associés à l'image-objet. Ces messages suivent ensuite la hiérarchie de messages classique : script d'acteur, script d'image, puis script d'animation.

Si l'image-objet donnée ne comporte pas de comportement associé contenant le gestionnaire donné, `sendSprite()` renvoie la valeur `FALSE`.

Paramètres

`nomOuNumImageObjet` Requis. Chaîne ou nombre entier spécifiant le nom ou le numéro de l'image-objet qui recevra l'événement.

`événement` Requis. Symbole ou chaîne spécifiant l'événement à envoyer à l'image-objet spécifiée.

`arguments` Facultatif. Argument(s) à envoyer avec le message.

Exemple

Le gestionnaire suivant envoie le message personnalisé `bumpCounter` et l'argument 2 à l'image-objet 1 lorsque l'utilisateur clique avec la souris :

```
-- Lingo syntax
on mouseDown me
    _movie.sendSprite(1, #bumpCounter, 2)
end

// JavaScript syntax
function mouseDown() {
    _movie.sendSprite(1, "bumpCounter", 2);
}
```

Voir aussi

Animation

setAlpha()

Syntaxe

```
imageObject.setAlpha(alphaLevel)
imageObject.setAlpha(alphaImageObject)
```

Description

Fonction ; attribue à la couche alpha d'un objet image un niveauAlpha plat ou un *objetImageAlpha* existant. Le *niveauAlpha* doit être un nombre compris entre 0 et 255. Plus la valeur est faible, plus l'image semble transparente. Des valeurs supérieures font apparaître l'image plus opaque. La valeur 255 a le même effet que la valeur 0. Pour que le *niveauAlpha* puisse prendre effet, la propriété `useAlpha()` de l'objet image doit être définie sur `TRUE`.

L'objet image doit être de 32 bits. Si vous spécifiez un objet image alpha, il doit être de 8 bits. Les deux images doivent avoir les mêmes dimensions. Si ces conditions ne sont pas remplies, `setAlpha()` n'a aucun effet et renvoie la valeur FALSE. La fonction renvoie `TRUE` si elle réussit.

Exemple

L'instruction Lingo suivante rend l'image de l'acteur bitmap Premier plan opaque et désactive simultanément la couche alpha. Cette méthode est efficace pour supprimer la couche alpha d'une image :

```
member("Foreground").image.setAlpha(255)
member("Foreground").image.useAlpha = FALSE
```

L'instruction Lingo suivante récupère la couche alpha de l'acteur Lever de soleil et la place dans la couche alpha de l'acteur Coucher de soleil :

```
tempAlpha = member("Sunrise").image.extractAlpha()
member("Sunset").image.setAlpha(tempAlpha)
```

Voir aussi

useAlpha, extractAlpha()

setaProp

Syntaxe

```
setaProp list, listProperty, newValue
setaProp (childObject, listProperty, newValue)
list.listProperty = newValue
list[listProperty] = newValue
childObject.listProperty = newValue
```

Description

Commande ; remplace la valeur affectée à *propriétéDeListe* par la valeur spécifiée par *nouvelleValeur*. La commande `setaProp` fonctionne uniquement avec des listes de propriétés et des objets enfants. L'utilisation de `setaProp` avec une liste linéaire produit une erreur de script.

- Pour les listes de propriétés, `setaProp` remplace une propriété dans la liste spécifiée par *liste*. Lorsque la propriété ne figure pas déjà dans la liste, Lingo ajoute la nouvelle propriété ainsi que sa valeur.
- Pour les objets enfants, `setaProp` remplace une propriété de l'objet enfant. Lorsque la propriété ne figure pas déjà dans l'objet, Lingo ajoute la nouvelle propriété ainsi que sa valeur.
- La commande `setaProp` peut également définir des propriétés `ancestor`.

Paramètres

propriétéDeListe Requis. Symbole (Lingo uniquement) ou chaîne spécifiant le nom de la propriété dont la valeur est modifiée.

nouvelleValeur Requis. Nouvelle valeur de la propriété *propriétéDeListe*.

Exemple

Les instructions suivantes créent une liste de propriétés, puis ajoutent l'élément `#c:10` à la liste :

```
newList = [#a:1, #b:5]
put newList
-- [#a:1, #b:5]
setaProp newList, #c, 10
put newList
```

L'opérateur point permet de modifier la valeur de propriété d'une propriété figurant déjà dans une liste sans utiliser `setaProp` :

```
newList = [#a:1, #b:5]
put newList
-- [#a:1, #b:5]
newList.b = 99
put newList
-- [#a:1, #b:99]
```

Remarque : *Pour que l'opérateur point soit utilisable pour manipuler une propriété, cette dernière doit déjà exister dans la liste, dans l'objet enfant ou dans le comportement.*

Voir aussi

`ancestor`, `property`, `. (opérateur point)`

setAt

Syntaxe

```
setAt list, orderNumber, value
list[orderNumber] = value
```

Description

Commande ; remplace l'élément spécifié par *numéroDordre* par la valeur spécifiée par *valeur* dans la liste spécifiée par *liste*. Lorsque *numéroDordre* est supérieur au nombre d'éléments d'une liste de propriétés, la commande `setAt` renvoie une erreur de script. Lorsque *numéroDordre* est supérieur au nombre d'éléments d'une liste linéaire, Director insère des entrées vierges dans la liste pour ajouter le nombre d'emplacements spécifiés par *numéroDordre*.

Exemple

Le gestionnaire suivant affecte un nom à la liste [12, 34, 6, 7, 45], remplace le quatrième élément de la liste par la valeur 10, puis affiche le résultat dans la fenêtre Messages :

```
--Lingo
on enterFrame
    set vNumbers = [12, 34, 6, 7, 45]
    setAt vnumbers, 4, 10
    put vNumbers
end enterFrame

// Javascript
function enterFrame
{
    vNumbers =list(12, 34, 6, 7, 45) ;
    vNumbers.setAt(4, 10) ;
    put (vNumbers);
}
```

Lorsque le gestionnaire est exécuté, la fenêtre Messages affiche le message suivant :

```
[12, 34, 6, 10, 45]
```

Vous pouvez effectuer la même opération en utilisant des crochets d'accès pour la liste de la manière suivante :

```
--Lingo
on enterFrame
    set vNumbers = [12, 34, 6, 7, 45]
    vnumbers[4] = 10
    put vNumbers
end enterFrame

// Javascript
function enterFrame
{
    vNumbers = list(12, 34, 6, 7, 45);
    vnumbers[4] = 10 ;
    put (vNumbers);
}
```

Lorsque le gestionnaire est exécuté, la fenêtre Messages affiche le message suivant :

```
[12, 34, 6, 10, 45]
```

Voir aussi

[ ] (crochets d'accès)

setCallback()

Syntaxe

```
-- Lingo syntax
spriteObjRef.setCallback(actionScriptObject, ASEventName, #LingoHandlerName,
lingoScriptObject)

// JavaScript syntax
spriteObjRef.setCallback(actionScriptObject, ASEventName, #LingoHandlerName,
lingoScriptObject);
```

Description

Commande Flash ; cette commande peut être utilisée en tant qu'image-objet ou comme méthode globale pour définir un appel de gestionnaire Lingo pour un événement particulier généré par l'objet spécifié. Lorsque ActionScript déclenche l'événement dans l'objet, cet événement est redirigé vers le gestionnaire Lingo spécifié, avec tous les arguments transmis avec l'événement.

Si l'objet ActionScript a été initialement créé dans une image-objet Flash, utilisez la syntaxe *`réfDimageObjetFlash`*. Si l'objet a été créé globalement, utilisez la syntaxe globale.

Remarque : *Si vous n'avez pas importé d'acteur Flash, vous devrez ajouter manuellement l'Xtra Flash Asset à la liste des Xtras de votre animation pour permettre aux commandes Flash globales de fonctionner correctement. Vous ajoutez des Xtras à la liste des Xtras par l'intermédiaire des commandes Modification > Animation > Xtras. Pour plus d'informations sur la gestion des Xtras pour les animations distribuées, reportez-vous aux rubriques Utilisation de Director dans l'Aide de Director.*

Paramètres

`objetAS` Requis. Spécifie l'objet ActionScript contenant l'événement *`nomDévénAS`*.

`nomDévénAS` Requis. Spécifie l'événement ActionScript qui se produit.

`gestLingo` Requis. Spécifie le gestionnaire Lingo gérant l'événement *`nomDévénAS`*.

`objetLingo` Requis. Spécifie l'objet script Lingo contenant le gestionnaire *`gestLingo`*.

Exemple

L'instruction suivante définit l'appel du gestionnaire Lingo appelé `myOnStatus` dans l'objet script Lingo me lorsqu'un événement `onStatus` est généré par l'objet ActionScript `tLocalConObject` dans l'animation Flash au niveau de l'image-objet 3 :

```
Lingo syntax
sprite(3).setCallback(tLocalConObject, "onStatus", #myOnStatus, me)

// JavaScript syntax
sprite(3).setCallback(tLocalConObject, "onStatus", symbol("myOnStatus"), me);
```

Les instructions suivantes créent un nouvel objet XML global, ainsi qu'un gestionnaire `callback` qui analyse les données XML à leur arrivée. La troisième ligne entraîne le chargement d'un fichier XML. Le gestionnaire `callback` est également inclus.

```
-- Lingo syntax
gXMLCB = newObject("XML")
setCallback( gXMLCB, "onData", #dataFound, 0 )
gXMLCB.load( "myfile.xml")

-- Callback handler invoked when xml data arrives
```

```
on dataFound me, obj, source
 obj.parseXML(source)
 obj.loaded = 1
 obj.onload(TRUE)
end dataFound

// JavaScript syntax
gXMLCB = newObject("XML");
setCallback( gXMLCB, "onData", symbol("dataFound"), 0 );
gXMLCB.load( "myfile.xml" );

// Callback handler invoked when xml data arrives
function dataFound(me, obj, source) {
 obj.parseXML(source);
 obj.loaded = 1;
 obj.onload(1);
}
```

Voir aussi

newObject(), clearAsObjects()

setCollisionCallback()

Syntaxe

```
member(whichCastmember).model(whichModel).collision.setCollisionCallback (#handlerName,
scriptInstance)
```

Description

Commande 3D de collision ; enregistre un gestionnaire spécifié, dans une instance de script donnée, en tant que gestionnaire à appeler lorsque *quelModèle* est impliqué dans une collision.

Cette commande ne fonctionne que si la propriété `collision.enabled` du modèle présente la valeur `TRUE`. Le comportement par défaut est déterminé par la valeur de `collision.resolve`, que vous pouvez remplacer à l'aide des commandes `collision.resolveA` et/ou `collision.resolveB`. N'utilisez pas la commande `updateStage` dans le gestionnaire spécifié.

Cette commande constitue une alternative plus courte à la commande `registerScript` pour les collisions, mais produit le même résultat global. L'utilisation de cette commande peut être envisagée pour exécuter une petite partie de la fonctionnalité de la commande `registerScript`.

Paramètres

nomDeGestionnaire Requis. Spécifie le gestionnaire appelé lorsqu'un modèle est impliqué dans une collision.

instanceDeScript Requis. Spécifie l'instance de script contenant le gestionnaire spécifié par *nomDeGestionnaire*.

Exemple

L'instruction suivante entraîne l'appel du gestionnaire `#rebond` de l'acteur scriptDeCollision lorsque le modèle Sphère entre en collision avec un autre modèle :

```
member("3d world").model("Sphere").collision.setCollisionCallback(#bounce,
member("colScript"))
```

Voir aussi

collisionData, collision (modificateur), resolve, resolveA, resolveB, registerForEvent(), registerScript(), sendEvent

setFilterMask()

Syntaxe

```
-- Lingo syntax
fileioObjRef.setFilterMask(stringMask)

// JavaScript syntax
fileioObjRef.setFilterMask(stringMask);
```

Description

Méthode FileIO ; définit le masque de filtre pour le champ *Types de fichiers* d'une boîte de dialogue afin de spécifier les types de fichiers qui s'affichent à l'ouverture de cette boîte de dialogue.

Paramètres

chaîneMasque Requis. Chaîne spécifiant le masque de filtre.

Exemple

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.setFilterMask(stringMask)

 // JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.setFilterMask(stringMask) ;
```

Voir aussi

Fileio

setFinderInfo()

Syntaxe

```
-- Lingo syntax
fileioObjRef.setFinderInfo(stringAttrs)

// JavaScript syntax
fileioObjRef.setFinderInfo(stringAttrs)
```

Description

Méthode FileIO (Mac uniquement) ; définit les informations du Finder relatives à un fichier ouvert.

Paramètres

chaîneInfos Requis. Chaîne spécifiant les informations du Finder.

Exemple

```
-- Lingo syntax
```

```
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.setFinderInfo(stringAttrs)

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.setFinderInfo(stringAttrs) ;
```

Voir aussi

Fileio

setFlashProperty()

Syntaxe

```
-- Lingo syntax
spriteObjRef.setFlashProperty(targetName, #property, newValue)

// JavaScript syntax
spriteObjRef.setFlashProperty(targetName, #property, newValue);
```

Description

Fonction ; permet à Lingo d'invoquer la fonction script d'action Flash `setProperty()` dans l'image-objet Flash donnée. Utilisez la fonction `setFlashProperty()` pour définir les propriétés des clips ou des niveaux dans une animation Flash. Cette fonction est similaire à la définition des propriétés d'images-objets dans Director.

Pour définir une propriété globale de l'image-objet Flash, spécifiez une chaîne vide dans le paramètre *`nomDeCible`*. Vous pouvez définir les propriétés Flash globales suivantes : `#focusRect` et `#spriteSoundBufferTime`.

Consultez la documentation de Flash pour plus d'informations sur ces propriétés.

Paramètres

`nomDeCible` Requis. Spécifie le nom du clip ou du niveau de l'animation dont vous souhaitez définir la propriété dans l'image-objet Flash donnée.

`propriété` Requis. Spécifie le nom de la propriété à définir. Vous pouvez définir les propriétés suivantes : `#posX`, `#posY`, `#scaleX`, `#scaleY`, `#visible`, `#rotate`, `#alpha` et `#name`.

`nouvelleValeur` Requis. Spécifie la nouvelle valeur.

Exemple

L'instruction suivante définit la valeur de la propriété `#rotate` du clip Etoile dans l'acteur Flash de l'image-objet 3 sur 180 :

```
-- Lingo syntax
sprite(3).setFlashProperty("Star", #rotate, 180)

// JavaScript syntax
sprite(3).setFlashProperty("Star", symbol("rotate"), 180);
```

Voir aussi

getFlashProperty()

setNewLineConversion()

Syntaxe

```
-- Lingo syntax
fileioObjRef.setNewLineConversion(intOnOff)

// JavaScript syntax
fileioObjRef.setNewLineConversion(intOnOff)
```

Description

Méthode FileIO (Mac uniquement) ; spécifie si la conversion automatique des nouveaux caractères de ligne est activée ou désactivée.

Paramètres

entActivéDésactivé Requis. Nombre entier spécifiant si la conversion automatique est activée ou désactivée. Les valeurs possibles sont 0 (désactivé) ou 1 (activé).

Exemple

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.setNewLineConversion(intOnOff)

 // JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.setNewLineConversion(intOnOff) ;
```

Voir aussi

Fileio

setPixel()

Syntaxe

```
-- Lingo syntax
imageObjRef.setPixel(x, y, colorObjOrIntValue)
imageObjRef.setPixel(point(x, y), colorObjOrIntValue)

// JavaScript syntax
imageObjRef.setPixel(x, y, colorObjOrIntValue);
imageObjRef.setPixel(point(x, y), colorObjOrIntValue);
```

Description

Méthode d'image. Définit la valeur de couleur du pixel à un point spécifié d'une image donnée.

Si vous définissez un grand nombre de pixels sur la couleur d'un autre pixel à l'aide de `getPixel()`, il est plus rapide de les définir sous forme de nombres entiers.

Pour obtenir des performances optimales avec les objets couleur, utilisez un objet couleur indexé pour les images 8 bits (ou inférieur) et un objet couleur RVB pour les images 16 bits (ou supérieur).

Cette méthode renvoie la valeur FALSE si le pixel spécifié est situé à l'extérieur de l'image donnée.

Vous pouvez voir un exemple d'utilisation de cette méthode dans une animation en consultant l'animation Imaging du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

x Requis en cas de spécification d'un pixel à l'aide de coordonnées x et y. Nombre entier spécifiant la coordonnée x du pixel.

y Requis en cas de spécification d'un pixel à l'aide de coordonnées x et y. Nombre entier spécifiant la coordonnée y du pixel.

`point(x, y)` Requis en cas de spécification d'un pixel à l'aide d'un point. Point spécifiant le pixel.

objOuValeurCouleur Requis en cas de définition de la couleur sur un objet couleur ou sur une valeur entière. Référence à un objet couleur spécifiant la couleur du pixel ou nombre entier indiquant la valeur de couleur de ce pixel.

Exemple

Les instructions suivantes définissent la couleur du pixel au point (20, 20) dans l'acteur Image de la scène sur la couleur rouge.

```
-- Lingo
objImage = _movie.stage.image
objImage.setPixel(20, 20 , rgb(255,0,0))
put (objImage)

-- Javascript
var objImage = _movie.stage.image ;
objImage.setPixel(20, 20 , color(255,0,0)) ;
put (objImage) ;
```

Voir aussi

color(), draw(), fill(), getPixel(), image()

setPlayList()

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.setPlayList(linearListOfPropLists)

// JavaScript syntax
soundChannelObjRef.setPlayList(linearListOfPropLists);
```

Description

Méthode de piste audio ; définit ou réinitialise la liste de lecture d'une piste audio.

Cette méthode se révèle utile pour mettre plusieurs sons en file d'attente en une seule opération.

Vous pouvez voir un exemple d'utilisation de `setPlaylist()` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

listeLinéaireListesDePropriétés Requis. Liste linéaire de listes de propriétés spécifiant les paramètres d'une liste de lecture. Vous pouvez définir ces paramètres pour chaque son à placer en file d'attente :

Propriété	Description
#member	Acteur à placer en file d'attente. Cette propriété doit être spécifiée. Toutes les autres sont facultatives.
#startTime	Position temporelle de départ de la lecture du son, en millisecondes. La valeur par défaut est le début du son. Pour plus d'informations, reportez-vous à l'entrée `startTime`.
#endTime	Position temporelle de fin de la lecture du son, en millisecondes. La valeur par défaut est la fin du son. Pour plus d'informations, reportez-vous à l'entrée `endTime`.
#loopCount	Nombre de répétitions d'une boucle défini avec `#loopStartTime` et `#loopEndTime`. La valeur par défaut est 1. Pour plus d'informations, reportez-vous à l'entrée `loopCount`.
#loopStartTime	Position temporelle de départ de la boucle, en millisecondes. Pour plus d'informations, reportez-vous à l'entrée `loopStartTime`.
#loopEndTime	Position temporelle de fin de la boucle, en millisecondes. Pour plus d'informations, reportez-vous à l'entrée `loopEndTime`.
#preloadTime	Quantité de son à placer en mémoire tampon avant la lecture, en millisecondes. Pour plus d'informations, reportez-vous à l'entrée `preloadTime`.

Exemple

Le gestionnaire suivant place en file d'attente et lit l'acteur son Intro, à partir de ses 3 secondes et exécute cinq lectures en boucle successives, de 8 secondes à 8,9 secondes, et s'arrête au point 10 secondes.

```
-- Lingo syntax
on playMusic
    sound(2).queue([#member:member("introMusic"), #startTime:3000, #endTime:10000,
#loopCount:5, #loopStartTime:8000, #loopEndTime:8900])
    sound(2).play()
end playMusic

// JavaScript syntax
function playMusic() {
    sound(2).queue(propList("member",member("introMusic"),"startTime",3000,
"endTime",10000, "loopCount",5, "loopStartTime",8000,"loopEndTime",8900));
    sound(2).play();
}
```

Voir aussi

endTime, getPlayList(), loopCount, loopEndTime, loopStartTime, Acteur, member, preLoadTime, queue(), Piste audio, startTime

setPosition()

Syntaxe

```
-- Lingo syntax
fileioObjRef.setPosition(intPosition)

// JavaScript syntax
fileioObjRef.setPosition(intPosition);
```

Description

Méthode FileIO ; définit la position d'un fichier.

Paramètres

entPosition Requis. Nombre entier spécifiant la nouvelle position du fichier.

Exemple

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.setPosition(intPosition)

 // JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.setPosition(intPosition) ;
```

Voir aussi

Fileio

setPref()

Syntaxe

```
-- Lingo syntax
_player.setPref(stringPrefName, prefString)

// JavaScript syntax
_player.setPref(stringPrefName, prefString);
```

Description

Méthode de lecteur ; écrit une chaîne spécifiée dans un fichier donné sur le disque local de l'ordinateur. Le fichier est un fichier texte standard.

Après l'exécution de la méthode `setPref()`, si l'animation est lue dans un navigateur, un dossier nommé Prefs est créé à l'intérieur du dossier Plug-In Support. La méthode `setPref()` ne peut écrire que dans ce dossier.

Le fichier est créé aux emplacements suivants.

Projection Windows/Director ..\Documents and Settings\<nomDutilisateur>\Application Data\Adobe\Director<numéroDeVersion>\Prefs\

Shockwave Windows ..\Documents and Settings\<nomDutilisateur>\Application Data\Adobe\Shockwave Player <numéroDeVersion>\Prefs\

Projection MAC ~/Library/Application Support/Adobe/Director <numéroDeVersion>/Prefs

Shockwave MAC ~/Library/Application Support/Adobe/Shockwave Player<numéroDeVersion>/Prefs

N'utilisez pas cette méthode pour écrire sur un média en lecture seule. Selon la plate-forme et la version du système d'exploitation utilisé, vous pourriez rencontrer des erreurs ou autres problèmes.

Dans un navigateur, les données écrites par `setPref()` ne sont pas confidentielles. Toute animation comportant du contenu Shockwave est en mesure de lire ces informations et de les charger sur un serveur. Les informations confidentielles ne doivent donc pas être stockées à l'aide de la méthode `setPref()`.

Sous Windows, la méthode `setPref()` échoue si l'utilisateur dispose de droits d'accès restreints.

Vous pouvez voir un exemple d'utilisation de `setPref()` dans une animation en consultant l'animation Read and Write Text du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

`nomDePréf` Requis. Chaîne spécifiant le fichier dans lequel le texte doit être écrit. Le paramètre *`nomDePréf`* doit correspondre à un nom de fichier valide. Pour vous assurer de sa validité sur toutes les plates-formes, n'utilisez pas plus de huit caractères alphanumériques dans ce nom de fichier.

`valeurDePréf` Requis. Chaîne spécifiant le texte à écrire dans le fichier *`nomDePréf`*.

Exemple

Le gestionnaire suivant enregistre le contenu de l'acteur champ Saisie dans un fichier nommé PréfsActuelles :

```
-- Lingo syntax
on mouseUp me
    _player.setPref("DayWare", member("Text Entry").text)
end

// JavaScript syntax
function mouseUp() {
    _player.setPref("DawWare", member("Text Entry").text);
}
```

Voir aussi

`getPref()`, `Lecteur`

setProp

Syntaxe

```
setProp list, property, newValue
list.listProperty = newValue
list[listProperty] = newValue
```

Description

Commande ; dans une liste, remplace la valeur affectée à une propriété spécifiée par une nouvelle valeur. Si la liste ne contient pas la propriété spécifiée, `setProp` renvoie une erreur de script.

La commande `setProp` fonctionne uniquement avec les listes de propriétés. L'utilisation de `setProp` avec une liste linéaire produit une erreur de script.

Cette commande est semblable à la commande `setaProp`, à l'exception du fait que `setProp` renvoie une erreur lorsque la propriété ne figure pas dans la liste.

Paramètres

`propriété` Requis. Symbole (Lingo uniquement) ou chaîne spécifiant la propriété dont la valeur est remplacée par *`nouvelleValeur`*.

`nouvelleValeur` Requis. Nouvelle valeur pour la propriété spécifiée par *`propriété`*.

Exemple

L'instruction suivante remplace la valeur affectée à la propriété `age` de la liste de propriétés x par la valeur 11 :

```
--Lingo
setProp x, #age, 11
```

```
// Javascript
x[age] = 11
```

L'opérateur point permet de modifier la valeur d'une propriété figurant déjà dans une liste, exactement comme ci-dessus :

```
x.age = 11
```

Voir aussi

setaProp

setScriptList()

Syntaxe

```
spriteReference.setScriptList(scriptList)
sprite(whichSprite).setScriptList(scriptList)
```

Description

Cette commande définit la liste `listeDeScript` de l'image-objet donnée. La liste `listeDeScript` indique les scripts associés à l'image-objet ainsi que les paramètres de chaque propriété de script. La définition de cette liste permet de modifier les paramètres attachés à une image-objet ou de modifier les propriétés de comportement.

La liste prend la forme suivante :

```
[ [ (whichBehaviorMember), " [ #property1: value, #property2: value, . . . ] ",
[(whichBehaviorMember), " [ #property1: value, #property2: value, . . . ] " ] ]
```

Cette commande ne peut pas être utilisée lors d'une session d'enregistrement du scénario. Utilisez `setScriptList()` pour les images-objets ajoutées pendant l'enregistrement du scénario après la session d'enregistrement du scénario.

Paramètres

`listeDeScripts` Requis. Spécifie la liste de scripts pour une image-objet donnée.

Voir aussi

scriptList, value(), string()

settingsPanel()

Syntaxe

```
-- Lingo syntax
spriteObjRef.settingsPanel({integerPanelIndex})

// JavaScript syntax
spriteObjRef.settingsPanel({integerPanelIndex});
```

Description

Commande d'image-objet Flash ; invoque la boîte de dialogue des paramètres de Flash, à l'index de panneau indiqué. Il s'agit de la boîte de dialogue que vous pouvez ouvrir en cliquant avec le bouton droit de la souris (Windows) ou en cliquant tout en appuyant sur la touche Ctrl (Mac) sur une animation Flash lue dans un navigateur.

La boîte de dialogue des paramètres ne s'affiche pas si les dimensions du rectangle de l'image-objet Flash ne le permettent pas.

Si vous souhaitez émuler Flash Player en appelant la boîte de dialogue des paramètres lorsque l'utilisateur clique avec le bouton droit de la souris (Windows) ou clique tout en appuyant sur la touche Ctrl (Mac), vous pouvez utiliser cette commande dans un gestionnaire `mouseDown` qui teste la propriété `rightMouseDown` ou `controlDown`.

Pour émuler Flash Player en activant la boîte de dialogue des paramètres dans une animation Director exécutée dans un navigateur, vous devez d'abord désactiver le menu contextuel Shockwave Player, accessible en cliquant avec le bouton droit de la souris (Windows) ou en cliquant tout en appuyant sur la touche Ctrl (Mac) dans une animation avec du contenu Shockwave lue dans un navigateur. Pour plus d'informations sur la désactivation de ce menu, reportez-vous aux rubriques Utilisation de Director dans l'Aide de Director.

Paramètres

`entierIndexPanneau` Facultatif. Spécifie le panneau à activer lorsque la boîte de dialogue est ouverte. Les valeurs possibles sont 0, 1, 2 ou 3. La valeur 0 ouvre la boîte de dialogue présentant l'onglet Contrôle de l'accès, la valeur 1, l'onglet Enregistrement local, la valeur 2, l'onglet Microphone et la valeur 3, l'onglet Caméra. L'index par défaut est 0.

Exemple

L'instruction suivante ouvre le panneau des paramètres de Flash, qui présente l'onglet Enregistrement local :

```
-- Lingo syntax
sprite(3).settingsPanel(1)

// JavaScript syntax
sprite(3).settingsPanel(1);
```

Voir aussi

on mouseDown (gestionnaire d'événement), rightMouseDown, controlDown

setPref()

Syntaxe

```
-- Lingo syntax
_player.setPref(stringPrefName, prefString)

// JavaScript syntax
_player.setPref(stringPrefName, prefString);
```

Description

Méthode de lecteur ; écrit la chaîne spécifiée par *`chaîneDePréf`* dans le fichier spécifié par *`chaîneNomPréf`* sur le disque local de l'ordinateur.

L'argument *`chaîneNomPréf`* doit correspondre à un nom de fichier valide. Pour vous assurer de sa validité sur toutes les plates-formes, n'utilisez pas plus de huit caractères alphanumériques dans ce nom de fichier.

Après l'exécution de la méthode `setPref()`, si l'animation est lue dans un navigateur, un dossier nommé Prefs est créé à l'intérieur du dossier Plug-In Support. La méthode `setPref()` ne peut écrire que dans ce dossier.

Si l'animation est lue dans une projection ou dans Director, un dossier est créé dans le même dossier que l'application. Le dossier est appelé *Prefs*.

N'utilisez pas cette méthode pour écrire sur un média en lecture seule. Selon la plate-forme et la version du système d'exploitation utilisé, vous pourriez rencontrer des erreurs ou autres problèmes.

Cette méthode ne procède à aucune manipulation complexe des données de la chaîne ou de son formatage. Toute opération de formatage ou autre manipulation doit être effectuée parallèlement à l'utilisation de la méthode `getPref()` ; vous pouvez manipuler les données en mémoire, puis les inscrire dans l'ancien fichier par le biais de `setPref()`.

Dans un navigateur, les données écrites par `setPref()` ne sont pas confidentielles. Toute animation comportant du contenu Shockwave est en mesure de lire ces informations et de les charger sur un serveur. Les informations confidentielles ne doivent donc pas être stockées à l'aide de la méthode `setPref()`.

Sous Windows, la méthode `setPref()` échoue si l'utilisateur dispose de droits d'accès restreints.

Vous pouvez voir un exemple d'utilisation de `setPref()` dans une animation en consultant l'animation Read and Write Text du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

`chaîneNomPréf` Requis. Chaîne spécifiant le nom du fichier dans lequel la chaîne *`chaîneDePréf`* est écrite. Le fichier est un fichier texte standard.

`chaîneDePréf` Requis. Chaîne à écrire dans le fichier spécifié par *`chaîneNomPréf`*.

Exemple

Le gestionnaire suivant enregistre le contenu de l'acteur champ Saisie dans un fichier nommé PréfsActuelles :

```
-- Lingo syntax
on mouseUp me
    _player.setPref("CurPrefs", member("Text Entry").text)
end

// JavaScript syntax
function mouseUp() {
    _player.setPref("CurPrefs", member("Text Entry").text);
}
```

Voir aussi

`getPref()`, `Lecteur`

setTrackEnabled()

Syntaxe

```
-- Lingo syntax
spriteObjRef.setTrackEnabled(whichTrack, trueOrFalse)

// JavaScript syntax
spriteObjRef.setTrackEnabled(whichTrack, trueOrFalse);
```

Description

Commande ; définit si la lecture de la piste spécifiée d'une vidéo numérique est activée.

- Si `setTrackEnabled` présente la valeur `TRUE`, la piste indiquée est activée et lue.
- Si `setTrackEnabled` présente la valeur `FALSE`, la piste indiquée est désactivée et n'est pas lue. Pour les vidéos numériques, cela signifie qu'elles ne sont plus mises à jour à l'écran.

Pour tester si une piste est déjà activée, testez la propriété d'image-objet `trackEnabled`.

Paramètres

`quellePiste` Requis. Spécifie la piste à tester.

`trueOuFalse` Requis. Spécifie si la piste de la vidéo numérique est activée (`TRUE`) ou non (`FALSE`).

Exemple

L'instruction suivante active la piste 3 de la vidéo numérique affectée à la piste d'image-objet 8 :

```
-- Lingo syntax
sprite(8).setTrackEnabled(3, TRUE)

// JavaScript syntax
sprite(8).setTrackEnabled(3, 1);
```

Voir aussi

`trackEnabled`

setVal()

Syntaxe

```
<Void> Matrix.setVal(whichRow, whichColumn, valueToSet)
```

Description

Méthode de matrice ; définit la valeur de l'élément spécifié dans la matrice indiquée.

Paramètres

`quelleLigne` Requis. Numéro de ligne de l'élément dont la valeur est définie.

`quelleColonne` Requis. Numéro de colonne de l'élément dont la valeur est définie.

`valeurAdéfinir` Requis. Valeur à virgule flottante à définir pour les numéros de ligne et de colonne indiqués dans la matrice.

Exemple

La fonction suivante crée une matrice de 4 lignes et de 5 colonnes et définit la valeur de chaque élément de cette matrice sur des valeurs aléatoires.

```
--Lingo
on randomMatrix()
  rows = 4
  cols = 5
  mat = newMatrix(rows,cols)
  repeat with i = 1 to rows
    repeat with j = 1 to cols
      mat.setVal(i,j,random(255))
    end repeat
  end repeat
  return mat
end

//Java Script
function randomMatrix()
{
```

```
  rows = 4;
  cols = 5;
  mat = newMatrix(rows,cols);
  for(i = 1 ; i <= rows;i++)
  for(j = 1 ; j <= rows;j++)
      mat.setVal(i,j,random(255));
  return mat;
}
```

Voir aussi

getVal(), numRows(), numColumns(), matrixAddition(), matrixMultiply(), matrixMultiplyScalar(), matrixTranspose(), newMatrix()

setVariable()

Syntaxe

```
-- Lingo syntax
spriteObjRef.setVariable(variableName, newValue)

// JavaScript syntax
spriteObjRef.setVariable(variableName, newValue);
```

Description

Fonction ; définit la valeur de la variable spécifiée dans l'image-objet donnée. Les variables Flash ont été introduites dans la version 4 de Flash.

Paramètres

nomDeVariable Requis. Spécifie le nom de la variable.

nouvelleValeur Requis. Spécifie la nouvelle valeur de la variable.

Exemple

L'instruction suivante définit la valeur de la variable `currentURL` dans l'acteur Flash au niveau de l'image-objet 3. La nouvelle valeur de la variable URLactuelle est « http://www.adobe.com/software/flash/ ».

```
-- Lingo syntax
sprite(3).setVariable("currentURL", "http://www.adobe.com/software/flash/")

// JavaScript syntax
sprite(3).setVariable("currentURL", "http://www.adobe.com/software/flash/");
```

Voir aussi

hitTest(), getVariable()

shader()

Syntaxe

```
member(whichCastmember).shader(whichShader)
member(whichCastmember).shader[index]
member(whichCastmember).model(whichModel).shader
member(whichCastmember).modelResource(whichModelResource).face[index].shader
```

Description

Propriété 3D d'élément, de modèle et de face ; objet utilisé pour définir l'apparence de la surface du modèle. Le matériau est la « peau » qui entoure la ressource de modèle utilisée par le modèle.

Le matériau même n'est pas une image. Le composant visible d'un matériau est créé avec un maximum de huit couches de `texture`. Ces huit couches de texture ont été créées à partir d'acteurs bitmap ou d'objets images dans Director ou importées avec des modèles de programmes de modélisation 3D. Pour plus d'informations, reportez-vous à l'entrée `texture`.

Tout modèle dispose d'une liste linéaire de matériaux appelée `shaderList`. Le nombre d'entrées de la liste est égal au nombre de mailles de la ressource utilisée par le modèle. Un seul matériau peut être appliqué à chaque maille.

L'acteur 3D possède un matériau par défaut nommé `DefaultShader`, qui ne peut pas être supprimé. Ce matériau est utilisé lorsqu'aucun matériau n'est affecté à un modèle et lorsqu'un matériau utilisé par un modèle est supprimé.

La syntaxe `member(`*`quelActeur`*`).model(`*`quelModèle`*`).shader` donne accès au premier matériau de la liste de matériaux `shaderlist` du modèle et équivaut à `member(`*`quelActeur`*`).model(`*`quelModèle`*`).shaderList[1]`.

Les matériaux sont créés et supprimés à l'aide des commandes `newShader()` et `deleteShader()`.

Les matériaux sont enregistrés dans la palette des matériaux de l'acteur 3D. Ils peuvent être référencés par nom (*`quelMatériau`*) ou par index de palette (*`indexDeMatériau`*). Un matériau peut être utilisé par n'importe quel nombre de modèles. Les modifications apportées à un matériau apparaissent dans tous les modèles qui l'utilisent.

Il existe quatre types de matériaux :

Les matériaux `#standard` présentent leurs textures de façon réaliste.

Les matériaux `#painter`, `#engraver` et `#newsprint` stylisent leurs textures pour qu'elles aient l'apparence d'une peinture, d'une gravure ou d'un journal. Ils comportent des propriétés spéciales en plus des propriétés de matériau `#standard`.

Les matériaux utilisés par les différentes faces des primitives `#mesh` peuvent être définis à l'aide de la syntaxe `member(`*`quelActeur`*`).modelResource(`*`quelleRessDeMod`*`).face[`*`index`*`].shader`. Pour modifier cette propriété, il est nécessaire d'appeler la commande `build()`.

Exemple

L'instruction suivante affecte le matériau surfaceDuMur à la propriété `shader` du modèle Mur :

```
member("Room").model("Wall").shader = member("Room").shader("WallSurface")
```

Voir aussi

`shaderList`, `newShader`, `deleteShader`, `face[ ]`, `texture()`

showLocals()

Syntaxe

```
-- Lingo syntax
showLocals()
```

Description

Fonction de niveau supérieur (Lingo uniquement) ; affiche toutes les variables locales dans la fenêtre Messages. Elle n'est utile que dans les gestionnaires ou les scripts parents contenant des variables locales à afficher. Toutes les variables utilisées dans la fenêtre Messages sont automatiquement globales.

Les variables locales des gestionnaires sont supprimées après l'exécution de ces derniers. L'insertion de l'instruction `showLocals()` dans un gestionnaire a pour effet d'afficher toutes les variables locales de ce gestionnaire dans la fenêtre Messages.

Cette commande est pratique pour déboguer les scripts.

Paramètres

Aucune.

Voir aussi

`clearGlobals()`, `global`, `showGlobals()`

showProps()

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.showProps()

// JavaScript syntax
memberOrSpriteObjRef.showProps();
```

Description

Commande ; affiche une liste des paramètres de propriétés actuels d'une animation Flash, d'un acteur vectoriel ou d'un son en cours de lecture dans la fenêtre Messages. Cette commande n'est utile qu'en phase de création ; elle ne fonctionne pas dans les projections ni dans les animations comportant du contenu Shockwave.

Paramètres

Aucune.

Exemple

Le gestionnaire suivant accepte le nom d'une distribution comme paramètre, recherche les acteurs animation Flash de cette distribution et affiche le nom, le numéro et les propriétés de l'acteur dans la fenêtre Messages :

```
-- Lingo syntax
on ShowCastProperties(whichCast)
    repeat with i = 1 to castLib(whichCast).member.count
        castType = member(i, whichCast).type
        if (castType = #flash) OR (castType = #vectorShape) then
            put castType&&"cast member" && i & ":" && member(i, whichCast).name
            put RETURN
            member(i ,whichCast).showProps()
        end if
    end repeat
end

// JavaScript syntax
function ShowCastProperties(whichCast) {
    i = 1;
    while( i < (castLib(whichCast).member.count) +1 ) {
        castType = member(i, whichCast).type;
        if ((castType = "flash") || (castType = "vectorShape")) {
            trace (castType + " cast member " + i + ": " + member(i, whichCast).name) + \n;
            member(i ,whichCast).showProps();
i++;
```

```
        }
    }
}
```

Voir aussi

queue(), setPlayList()

showGlobals()

Syntaxe

```
-- Lingo syntax
_global.showGlobals()

// JavaScript syntax
_global.showGlobals();
```

Paramètres

Aucune.

Description

Méthode globale ; affiche toutes les variables globales dans la fenêtre Messages.

Cette méthode se révèle utile pour déboguer les scripts.

Exemple

L'instruction suivante affiche toutes les variables globales dans la fenêtre Messages :

```
-- Lingo syntax
on mouseDown
 _global.showGlobals()
end

// JavaScript syntax
function mouseDown() {
 _global.showGlobals();
}
```

Voir aussi

Global

shutDown()

Syntaxe

```
-- Lingo syntax
_system.shutDown()

// JavaScript syntax
_system.shutDown();
```

Description

Méthode système ; ferme toutes les applications ouvertes et éteint l'ordinateur.

Paramètres
Aucune.

Exemple
L'instruction suivante vérifie si l'utilisateur a utilisé le raccourci clavier Ctrl-S (Windows) ou Cmde-S (Mac) et, le cas échéant, éteint l'ordinateur :

Voir aussi
Système

sin()

Syntaxe
```
sin(angle)
```

Description
Fonction mathématique (Lingo uniquement) ; calcule le sinus de l'angle spécifié. Celui-ci doit être exprimé en radians sous forme de nombre à virgule flottante.

En syntaxe JavaScript, utilisez la fonction `sin()` d'un objet mathématique.

Paramètres
`angle` Requis. Spécifie l'angle.

Exemple
L'instruction suivante calcule le sinus de `pi`/2 :

```
put sin (PI/2.0)
-- 1
```

Voir aussi
PI

sort

Syntaxe
```
list.sort()
sort list
```

Description
Commande ; trie les éléments d'une liste par ordre alphabétique.

- S'il s'agit d'une liste linéaire, elle est triée par valeurs.
- S'il s'agit d'une liste de propriétés, elle est triée alphabétiquement par propriété.

Une fois que la liste est triée, elle le reste, même si vous ajoutez des variables avec la commande `add`.

Paramètres
Aucune.

Exemple

L'instruction suivante trie la liste Valeurs, composée de [#a: 1, #d: 2, #c: 3], dans l'ordre alphanumérique. Le résultat apparaît sous l'instruction.

```
put values
-- [#a: 1, #d: 2, #c: 3]
values.sort()
put values
--[#a: 1, #c: 3, #d: 2]
```

sound()

Syntaxe

```
-- Lingo syntax
sound(intSoundChannel)

// JavaScript syntax
sound(intSoundChannel);
```

Description

Fonction de niveau supérieur ; renvoie une référence à une piste audio spécifiée.

Cette méthode remplit la même fonction que la méthode `channel()` de l'objet son.

Paramètres

`entPisteAudio` Requis. Nombre entier spécifiant la piste audio à référencer.

Exemple

L'exemple suivant affecte la piste audio 1 à la variable `music` et lit un son.

```
-- Lingo syntax
music = sound(1)
music.play(member("waltz1"))

// JavaScript syntax
var music = sound(1);
music.play(member("waltz1"));
```

Voir aussi

`channel() (son)`, `Piste audio`

sprite()

Syntaxe

```
-- Lingo syntax
sprite(nameOrNum)

// JavaScript syntax
sprite(nameOrNum);
```

Description

Fonction de niveau supérieur ; renvoie une référence à une image-objet donnée du scénario.

Si la propriété `scriptExecutionStyle` de l'animation est définie sur la valeur 9, l'appel de la méthode `sprite("abc")` lorsqu'aucune image-objet ne porte ce nom renvoie une référence à l'image-objet 1. Si la propriété `scriptExecutionStyle` de l'animation reçoit la valeur 10, l'appel de `sprite("abc")` lorsqu'aucune image-objet ne porte ce nom renvoie la valeur `VOID` si la méthode est appelée depuis Lingo ou la valeur `undefined` si elle est appelée depuis JavaScript.

Paramètres

`nomOuNum` Requis. Chaîne ou nombre entier spécifiant le nom ou la position d'index de l'image-objet.

Exemple

L'instruction suivante définit la variable `thisSprite` sur l'image-objet Grotte :

```
-- Lingo syntax
thisSprite = sprite("Cave")

// JavaScript syntax
var thisSprite = sprite("Cave");
```

Voir aussi

`Piste d'image-objet`

spriteSpaceToWorldSpace

Syntaxe

```
sprite(whichSprite).camera.spriteSpaceToWorldSpace(loc)
sprite(whichSprite).camera(index).spriteSpaceToWorldSpace(loc)
```

Description

Commande 3D ; renvoie une position d'univers située sur le plan de projection de la caméra spécifiée, qui correspond à un emplacement dans l'image-objet référencée.

Le plan de projection est défini par les axes des x et y de la caméra, et sa distance devant la caméra est telle qu'un pixel représente une unité de mesure d'univers. C'est ce plan de projection qui est utilisé pour l'affichage de l'image-objet sur la scène.

La forme `camera.spriteSpaceToWorldSpace()` de cette commande est un raccourci de `camera(1).spriteSpaceToWorldSpace()`.

Toutes les caméras utilisées par l'image-objet référencée répondent à la commande `spriteSpaceToWorldSpace` comme si leur cadre d'affichage était de la même taille que l'image-objet.

Paramètres

`emplacement` Requis. Spécifie l'emplacement de l'image-objet référencée. Cet emplacement doit être un point relatif au coin supérieur gauche de l'image-objet.

Exemple

L'instruction suivante indique que le point (50, 50) de l'image-objet 5 équivaut à vector(-1993,6699, 52,0773, 2263,7446) sur le plan de projection de la caméra de l'image-objet 5 :

```
-- Lingo
put sprite(5).camera.spriteSpaceToWorldSpace(point(50, 50))
-- vector(-1993.6699, 52.0773, 2263.7446)
```

```
// Javascript
put (sprite(5).camera.spriteSpaceToWorldSpace(point(50, 50)) );
//<vector(-1993.6699, 52.0773, 2263.7446)>
```

Voir aussi

worldSpaceToSpriteSpace, rect (caméra), camera

sqrt()

Syntaxe

```
sqrt(number)
the sqrt of number
```

Description

Fonction mathématique (Lingo uniquement) ; renvoie la racine carrée d'un nombre spécifié.

La valeur doit être un nombre décimal supérieur à 0. Les valeurs négatives renvoient la valeur 0.

En syntaxe JavaScript, utilisez la fonction `sqrt()` d'un objet mathématique.

Paramètres

nombre Requis. Spécifie le nombre dont la racine carrée doit être calculée. Ce nombre est un nombre à virgule flottante ou un nombre entier arrondi à l'entier le plus proche.

Exemple

L'instruction suivante affiche la racine carrée de 3,0 dans la fenêtre Messages :

```
put sqrt(3.0)
-- 1.7321
```

L'instruction suivante affiche la racine carrée de 3 dans la fenêtre Messages :

```
put sqrt(3)
-- 2
```

Voir aussi

floatPrecision

stageBottom

Syntaxe

```
the stageBottom
```

Description

Fonction ; utilisée conjointement avec `stageLeft`, `stageRight` et `stageTop`, indique la position de la scène sur le bureau. Elle renvoie la coordonnée verticale du bord inférieur de la scène, par rapport au coin supérieur gauche de l'écran principal. La hauteur de la scène en pixels est déterminée par la formule `the stageBottom - the stageTop`.

Lorsque l'animation est lue en tant qu'applet, la propriété `stageBottom` correspond à la hauteur de l'applet en pixels.

Cette fonction peut être testée, mais pas définie.

Paramètres

Aucune.

Exemple

Les instructions suivantes placent l'image-objet 3 à une distance de 50 pixels à partir du bord inférieur de la scène :

```
stageHeight = the stageBottom - the stageTop
sprite(3).locV = stageHeight - 50
```

Les coordonnées des images-objets sont exprimées par rapport au coin supérieur gauche de la scène. Pour plus d'informations, reportez-vous aux rubriques Utilisation de Director dans l'Aide de Director.

```
// Javascript
var stageHeight =_movie.stage.rect.Bottom -_movie.stage.rect.Top ;
sprite(3).locV = stageHeight - 50 ;
```

Voir aussi

stageLeft, stageRight, stageTop, locH, locV

stageLeft

Syntaxe

```
the stageLeft
```

Description

Fonction ; utilisée conjointement avec `stageRight`, `stageTop` et `stageBottom`, indique la position de la scène sur le bureau. Elle renvoie la coordonnée horizontale gauche de la scène, par rapport au coin supérieur gauche de l'écran principal. Lorsque le bord de la scène coïncide avec le côté gauche de l'écran principal, cette coordonnée est 0.

Lorsque l'animation est lue sous forme d'applet, la propriété `stageLeft` présente la valeur 0, qui correspond à l'emplacement du côté gauche de l'applet.

Cette propriété peut être testée, mais pas définie.

Les coordonnées des images-objets sont exprimées par rapport au coin supérieur gauche de la scène.

Paramètres

Aucune.

Exemple

L'instruction suivante vérifie si le bord gauche de la scène dépasse le bord gauche de l'écran et, le cas échéant, appelle le gestionnaire `leftMonitorProcedure` :

```
if the stageLeft < 0 then leftMonitorProcedure
```

Voir aussi

stageBottom, stageRight, stageTop, locH, locV

stageRight

Syntaxe

```
the stageRight
```

Description

Fonction ; utilisée conjointement avec `stageLeft`, `stageTop` et `stageBottom`, indique la position de la scène sur le bureau. Cette fonction renvoie la coordonnée horizontale droite de la scène, par rapport au coin supérieur gauche de l'écran principal. La largeur de la scène en pixels est déterminée par la formule `the stageRight - the stageLeft`.

Lorsque l'animation est lue sous forme d'applet, la propriété `stageRight` correspond à la largeur de l'applet en pixels.

Cette fonction peut être testée, mais pas définie.

Les coordonnées des images-objets sont exprimées par rapport au coin supérieur gauche de la scène.

Paramètres

Aucune.

Exemple

Les deux instructions suivantes placent l'image-objet 3 à une distance de 50 pixels du bord droit de la scène :

```
stageWidth = the stageRight - the stageLeft
    sprite(3).locH = stageWidth - 50
```

Voir aussi

`stageLeft`, `stageBottom`, `stageTop`, `locH`, `locV`

stageToFlash()

Syntaxe

```
-- Lingo syntax
spriteObjRef.stageToFlash(pointOnDirectorStage)

// JavaScript syntax
spriteObjRef.stageToFlash(pointOnDirectorStage);
```

Description

Fonction ; renvoie la coordonnée d'une animation Flash correspondant à une coordonnée spécifiée sur la scène de Director. Cette fonction accepte la coordonnée de la scène Director et renvoie la coordonnée de l'animation Flash sous la forme de valeurs de point Director : par exemple, point (300, 300).

Les coordonnées de l'animation Flash sont mesurées en pixels d'animation Flash, déterminés par la taille d'origine de l'animation lorsqu'elle a été créée dans Flash. Le point (0, 0) d'une animation Flash est toujours placé dans son coin supérieur gauche. La propriété `originPoint` de l'acteur n'intervient pas dans le calcul des coordonnées d'une animation ; elle est uniquement utilisée pour la rotation et la mise à l'échelle.

La fonction `stageToFlash()` et la fonction `flashToStage()` correspondante se révèlent utiles pour déterminer la coordonnée d'une animation Flash directement située sur une coordonnée spécifique de la scène Director. Pour Flash et Director, le point (0, 0) est le coin supérieur gauche de la scène Flash ou Director. Ces coordonnées peuvent ne pas coïncider sur la scène Director si une image-objet Flash est étirée, mise à l'échelle ou a pivoté.

Paramètres

pointDeLaScèneDeDirector Requis. Spécifie le point sur la scène Director.

Exemple

Le gestionnaire suivant vérifie si la souris (dont l'emplacement est suivi dans les coordonnées de la scène Director) est placée sur une coordonnée spécifique (130, 10) dans une image-objet animation Flash placée dans la piste 5. Si la souris est placée sur cette coordonnée, le script interrompt l'animation Flash.

```
-- Lingo syntax
on checkFlashRollover
    if sprite(5).stageToFlash(point(_mouse.mouseH,_mouse.mouseV)) = point(130,10) then
        sprite(5).stop()
    end if
end

// JavaScript syntax
function checkFlashRollover() {
    var stf = sprite(5).stageToFlash(point(_mouse.mouseH,_mouse.mouseV));
    if (stf = point(130,10)) {
        sprite(5).stop();
    }
}
```

Voir aussi

flashToStage()

stageTop

Syntaxe

```
the stageTop
```

Description

Fonction ; utilisée conjointement avec `stageBottom`, `stageLeft` et `stageRight`, indique la position de la scène sur le bureau. Cette fonction renvoie la coordonnée verticale supérieure de la scène par rapport au coin supérieur gauche de l'écran principal. Si la scène est dans le coin supérieur gauche de l'écran principal, cette coordonnée est 0.

Lorsque l'animation est lue en tant qu'applet, la valeur de la propriété `stageTop` est toujours égale à 0, qui correspond à l'emplacement du côté gauche de l'applet.

Cette fonction peut être testée, mais pas définie.

Les coordonnées des images-objets sont exprimées par rapport au coin supérieur gauche de la scène.

Paramètres

Aucune.

Exemple

L'instruction suivante vérifie si le bord supérieur de la scène dépasse le bord supérieur de l'écran et, le cas échéant, appelle le gestionnaire `upperMonitorProcedure` :

```
if the stageTop < 0 then upperMonitorProcedure
```

Voir aussi

stageLeft, stageRight, stageBottom, locH, locV

status()

Syntaxe

```
-- Lingo syntax
fileioObjRef.status()

// JavaScript syntax
fileioObjRef.status();
```

Description

Méthode FileIO ; renvoie le code d'erreur de la dernière méthode appelée.

Paramètres

Aucune.

Exemple

L'instruction suivante affiche l'état actuel de la piste d'image-objet 2 dans la fenêtre Messages :

```
-- Lingo syntax
put(sound(2).status)

// JavaScript syntax
put(sound(2).status);
```

Voir aussi

Fileio

stop() (DVD)

Syntaxe

```
-- Lingo syntax
dvdObjRef.stop()

// JavaScript syntax
dvdObjRef.stop();
```

Description

Méthode de DVD ; arrête la lecture.

Cette méthode renvoie la valeur TRUE (1) si l'opération a réussi.

Paramètres

Aucune.

Exemple

L'instruction suivante arrête la lecture :

```
-- Lingo syntax
member(1).stop()

// JavaScript syntax
member(1).stop();
```

Voir aussi

DVD

stop() (piste audio)

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.stop()

// JavaScript syntax
soundChannelObjRef.stop();
```

Description

Méthode de piste audio ; arrête le son en cours de lecture dans une piste audio.

L'appel d'une méthode `play()` démarre la lecture du premier des sons de la file d'attente de la piste audio donnée.

Vous pouvez voir un exemple d'utilisation de `stop()` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

Aucune.

Exemple

L'instruction suivante arrête la lecture de l'acteur son lu dans la piste audio 1 :

```
-- Lingo syntax
sound(1).stop()

// JavaScript syntax
sound(1).stop();
```

Voir aussi

`getPlayList()`, `pause() (piste audio)`, `play() (piste audio)`, `playNext() (piste audio)`, `rewind() (piste audio)`, `Piste audio`

stop() (Flash)

Syntaxe

```
-- Lingo syntax
spriteObjRef.stop()

// JavaScript syntax
spriteObjRef.stop();
```

Description

Commande Flash ; interrompt une image-objet animation Flash lue dans l'image actuelle.

Paramètres

Aucune.

Exemple

Le script d'image suivant interrompt les images de l'animation Flash lues dans les pistes 5 à 10 :

```
-- Lingo syntax
on enterFrame
    repeat with i = 5 to 10
        sprite(i).stop()
    end repeat
end

// JavaScript syntax
function enterFrame() {
    var i = 5;
    while (i < 11) {
        sprite(i).stop();
        i++;
    }
}
```

Voir aussi

hold()

stop() (RealMedia, SWA, Windows Media)

Syntaxe

```
-- Lingo syntax
windowsMediaObjRef.stop()
realMediaObjRef.stop()

// JavaScript syntax
windowsMediaObjRef.stop();
realMediaObjRef.stop();
```

Description

Méthode d'image-objet ou d'acteur Windows Media ou RealMedia. Arrête la lecture d'une image-objet ou d'un acteur Windows Media ou RealMedia.

Paramètres

Aucune.

Exemple

Les exemples suivants arrêtent la lecture de l'image-objet 2 ou l'acteur Real.

```
-- Lingo syntax
sprite(2).stop()
member("Real").stop()

// JavaScript syntax
sprite(2).stop();
member("Real").stop();
```

Voir aussi

RealMedia, Windows Media

stop

Syntaxe

```
stop(MUIObject, stopItem)
```

Description

Cette fonction interrompt la boîte de dialogue générale créée à partir d'une instance de l'Xtra MUI. Après l'appel de cette fonction, l'instruction Lingo renvoie les résultats en tant que numéro du paramètre stopItem.

Le paramètre stopItem est renvoyé depuis l'appel run(MUIOject). Utilisez ce paramètre pour transmettre un paramètre indiquant la manière dont la boîte de dialogue a été interrompue. Par exemple, si l'utilisateur a cliqué sur OK, la valeur transmise pourrait être 1 ou, si l'utilisateur a cliqué sur Annuler, la valeur pourrait être 0.

Remarque : *Utilisez la commande WindowOperation avec l'option #hide pour fermer une boîte de dialogue non modale.*

Exemple

Ce gestionnaire interrompt la boîte de dialogue générale créée à partir d'un objet MUIObject. Le deuxième paramètre de la commande d'interruption est défini sur zéro. Cette valeur permet de satisfaire à l'exigence de saisie d'une valeur, mais n'a pas d'autre but :

```
--Lingo syntax
on stopDialog
   global MUIObject
   if ( objectP (MUIObject)) then
      stop(MUIObject, 0)
   end if
end stopDialog
```

stopEvent()

Syntaxe

```
-- Lingo syntax
_movie.stopEvent()

// JavaScript syntax
_movie.stopEvent();
```

Description

Méthode d'animation ; empêche les scripts de transmettre un message d'événement au reste de la hiérarchie des messages.

Cette méthode s'applique également aux scripts d'image-objet.

Utilisez la méthode `stopEvent()` pour arrêter le message dans un gestionnaire d'événement principal ou dans un script d'image-objet, rendant ainsi le message non disponible pour les scripts d'image-objet suivants.

Par défaut, les messages sont d'abord mis à la disposition d'un gestionnaire d'événement principal (s'il existe), puis à celle de tous les scripts associés à une image-objet impliquée dans l'événement. Si plusieurs scripts sont associés à l'image-objet, le message est mis à la disposition de tous les scripts de l'image-objet. Si aucun script d'image-objet ne répond au message, celui-ci passe à un script d'acteur, puis à un script d'image et enfin à un script d'animation.

La commande `stopEvent()` ne s'applique qu'à l'événement en cours. Elle n'affecte pas les événements futurs. La méthode `stopEvent()` ne s'applique qu'aux gestionnaires d'événements principaux, aux gestionnaires appelés par des gestionnaires d'événements principaux ou à plusieurs scripts d'image-objet. Elle n'a aucun effet ailleurs.

Paramètres

Aucune.

Exemple

L'instruction suivante affiche l'événement `mouseUp` interrompu dans un comportement lorsque la variable globale `grandTotal` est égale à 500 :

```
-- Lingo syntax
global grandTotal
on mouseUp me
    if (grandTotal = 500) then
        _movie.stopEvent()
    end if
end

// JavaScript syntax
_global.grandTotal;
function mouseUp() {
    if (_global.grandTotal == 500) {
        _movie.stopEvent();
    }
}
```

Ni les scripts ultérieurs, ni les autres comportements sur l'image-objet ne reçoivent l'événement si ce dernier est interrompu de cette façon.

Voir aussi

Animation

stream()

Syntaxe

```
-- Lingo syntax
memberObjRef.stream(numberOfBytes)

// JavaScript syntax
memberObjRef.stream(numberOfBytes);
```

Description

Commande ; permet de faire passer manuellement en mémoire une partie d'un acteur animation Flash spécifié.

La commande `stream` renvoie le nombre d'octets réellement chargés. En fonction d'un ensemble de conditions variées (telles que la vitesse du réseau ou la disponibilité des données requises), le nombre d'octets réellement chargés peut être inférieur au nombre d'octets requis.

Vous pouvez toujours utiliser la commande `stream` pour un acteur, quelle que soit la propriété `streamMode` de cet acteur.

Paramètres

nombreDoctets Facultatif. Nombre entier spécifiant le nombre d'octets à charger en mémoire. Si vous omettez le paramètre *nombreDoctets*, Director tente de lire le nombre d'octets défini par la propriété `bufferSize` de l'acteur.

Exemple

Le script d'image suivant vérifie si un acteur animation Flash liée a été complètement chargé en mémoire en contrôlant sa propriété `percentStreamed`. Si l'acteur n'est pas chargé en mémoire à cent pour cent, le script tente de charger 32 000 octets de l'animation en mémoire.

Le script enregistre également le nombre réel d'octets chargé dans une variable intitulée `bytesReceived`. Si le nombre d'octets chargés ne correspond pas au nombre d'octets requis, le script met à jour un acteur texte pour indiquer le nombre d'octets réellement reçus. Le script contraint la tête de lecture à passer en boucle dans l'image actuelle jusqu'à ce que l'acteur soit complètement chargé en mémoire.

```
-- Lingo syntax
on exitFrame
    if member(10).percentStreamed < 100 then
        bytesReceived = member(10).stream(32000)
        if bytesReceived < 32000 then
            member("Message Line").text = "Received only" && bytesReceived && "of 32,000
bytes requested."
            _movie.updateStage()
        else
            member("Message Line").text = "Received all 32,000 bytes."
        end if
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var pctStm = member(10).percentStreamed;
    if (pctStm < 100) {
        var bytesReceived = member(10).stream(32000);
        if (bytesReceived < 32000) {
            member("Message Line").text = "Received only " + bytesReceived + " of32,000 bytes
requested.";
            _movie.updateStage();
        } else {
            member("Message Line").text = "Received all 32,000 bytes.";
        }
        _movie.go(_movie.frame);
    }
}
```

string()

Syntaxe

```
string(expression)
```

Description

Fonction ; convertit en chaîne un nombre entier, un nombre à virgule flottante, une référence d'objet, une liste, un symbole ou toute autre expression n'existant pas sous la forme d'une chaîne.

Paramètres

expression Requis. Expression à convertir en chaîne.

Exemple

L'instruction suivante additionne 2,0 + 2,5 et insère le résultat dans l'acteur champ Total :

```
--Lingo
member("total").text = string(2.0 + 2.5)

// Javascript
member("total").text = (2.0 + 2.5).toString() ;
```

L'instruction suivante convertit le symbole `#red` en une chaîne et l'insère dans l'acteur champ Couleur :

```
--Lingo
member("Color").text = string(#red)

// Javascript
member("Color").text = symbol("red").toString();
```

Voir aussi

`value()`, `stringP()`, `float()`, `integer()`, `symbol()`

stringP()

Syntaxe

```
stringP(expression)
```

Description

Fonction ; détermine si une expression est une chaîne (`TRUE`) ou non (`FALSE`).

Le *P* de `stringP` signifie *prédicat*.

Paramètres

expression Requis. Expression à tester.

Exemple

L'instruction suivante vérifie si *3* est une chaîne :

```
put stringP("3")
```

Le résultat est 1, équivalent numérique de `TRUE`.

L'instruction suivante vérifie si le nombre à virgule flottante 3,0 est une chaîne :

```
put stringP(3.0)
```

La valeur 3,0 étant un nombre à virgule flottante et non une chaîne, le résultat est 0, équivalent numérique de `FALSE`.

Voir aussi

`floatP()`, `ilk()`, `integerP()`, `objectP()`, `symbolP()`

subPictureType()

Syntaxe

```
-- Lingo syntax
dvdObjRef.subPictureType(intStream)

// JavaScript syntax
dvdObjRef.subPictureType(intStream);
```

Description

Méthode de DVD ; spécifie le type de sous-image d'un flux spécifié.

Cette méthode peut renvoyer les valeurs suivantes :

Symbole	Description
#unknown	Le type de sous-image est inconnu.
#Language	La sous-image intègre du contenu relatif à la langue, tel que des sous-titres d'animation ou d'autres textes.
#Other	La sous-image contient du contenu non relatif à la langue, tel qu'une balle rebondissant sur les paroles dans les titres de karaoké.

Paramètres

entFlux Requis. Nombre entier spécifiant le flux à tester.

Exemple

L'instruction suivante renvoie le type de sous-image du flux 2 :

```
-- Lingo syntax
member(12).member.subPictureType(2)

// JavaScript syntax
member(12).member.subPictureType(2);
```

Voir aussi

DVD

substituteFont

Syntaxe

```
TextMemberRef.substituteFont(originalFont, newFont)
substituteFont(textMemberRef, originalFont, newFont)
```

Description

Commande d'acteur texte ; remplace toutes les instances d'une police par une autre police dans un acteur texte.

Paramètres

policeDorigine Requis. Police à remplacer.

nouvellePolice Requis. Nouvelle police remplaçant la police spécifiée par *policeDorigine*.

Exemple

Le script suivant vérifie si la police Bonneville est disponible dans un acteur texte et, dans la négative, la remplace par Arial :

```
-- Lingo syntax
property spriteNum

on beginSprite me
    currMember = sprite(spriteNum).member
    if currMember.missingFonts contains "Bonneville" then
        currMember.substituteFont("Bonneville", "Arial")
    end if
end

// JavaScript syntax
function beginSprite() {
    currMember = sprite(spriteNum).member;
    if (currMember.missingFonts contains "Bonneville") { //check syntax
        currMember.substituteFont("Bonneville", "Arial");
    }
}
```

Voir aussi

missingFonts

swing()

Syntaxe

```
-- Lingo syntax
spriteObjRef.swing(pan, tilt, fieldOfView, speedToSwing)

// JavaScript syntax
spriteObjRef.swing(pan, tilt, fieldOfView, speedToSwing);
```

Description

Fonction d'image-objet QuickTime VR ; déplace une image-objet QuickTime 3 contenant un panoramique autour de nouveaux réglages de vue. Cette fonction produit un effet de travelling régulier.

quelleImageObjetQTVR : numéro de l'image-objet contenant l'acteur QuickTime VR.

Cette fonction renvoie immédiatement une valeur, mais l'image-objet continue de changer de vue jusqu'à ce que la vue finale soit atteinte. La durée requise pour arriver aux paramètres finaux varie en fonction du type d'ordinateur, de la taille du rectangle de l'image-objet, du codage des couleurs et d'autres paramètres affectant la performance.

Pour vérifier si le mouvement est terminé, vérifiez si la propriété `pan` de l'image-objet a atteint la valeur finale.

Paramètres

pan Requis. Spécifie la nouvelle position panoramique en degrés.

inclin Requis. Spécifie la nouvelle inclinaison en degrés.

champDeVue Requis. Spécifie le nouveau champ de vue en degrés.

vitesseDeMouv Requis. Spécifie la vitesse du mouvement. Les valeurs possibles sont comprises entre 1 (lente) et 10 (rapide).

Exemple

L'instruction suivante ajuste très progressivement l'affichage de l'image-objet QuickTime VR 1 à une position panoramique de 300 degrés, une inclinaison de -15 degrés et un champ de vue de 40 degrés :

```
-- Lingo syntax
sprite(1).swing(300, -15, 40, 1)

// JavaScript syntax
sprite(1).swing(300, -15, 40, 1);
```

Voir aussi

pan (propriété QTVR)

symbol()

Syntaxe

```
-- Lingo syntax
symbol(stringValue)

// JavaScript syntax
symbol(stringValue);
```

Description

Fonction de niveau supérieur ; considère une chaîne et renvoie le symbole correspondant.

Paramètres

valeurDeChaîne Requis. Chaîne à convertir en symbole.

Exemple

L'instruction suivante affiche le symbole #hello :

```
--Lingo syntax
put(symbol("hello"))

// JavaScript syntax
put(symbol("hello"));
```

L'instruction suivante affiche le symbole #auRevoir :

```
--Lingo syntax
x = "goodbye"
put(symbol(x))

// JavaScript syntax
var x = "goodbye";
put(symbol(x));
```

Voir aussi

value(), string()

symbolP()

Syntaxe

```
Expression.symbolP
symbolP(expression)
```

Description

Fonction ; détermine si une expression spécifiée est un symbole (`TRUE`) ou non (`FALSE`).

Le *P* de `symbolP` signifie prédicat.

Paramètres

`expression` Requis. Spécifie l'expression à tester.

Exemple

L'instruction suivante vérifie si la variable `myVariable` est un symbole :

```
put myVariable.symbolP
```

Voir aussi

`ilk()`

tan()

Syntaxe

```
tan(angle)
```

Description

Fonction mathématique ; renvoie la tangente de l'angle spécifié exprimée en radians sous forme de nombre à virgule flottante.

En syntaxe JavaScript, utilisez la fonction `tan()` d'un objet mathématique.

Paramètres

`angle` Requis. Spécifie l'angle dont la tangente est renvoyée.

Exemple

La fonction suivante renvoie la tangente de pi/4 :

```
tan (PI/4.0) = 1
```

Le symbole p n'est pas utilisable dans une expression Lingo.

Voir aussi

`PI`

tellStreamStatus()

Syntaxe

```
tellStreamStatus(onOrOffBoolean)
```

Description

Fonction ; active (TRUE) ou désactive (FALSE) le gestionnaire d'état de la lecture en flux continu.

La forme `tellStreamStatus()` détermine l'état du gestionnaire.

Lorsque la fonction `streamStatusHandler` présente la valeur TRUE, l'activité de lecture en flux continu sur Internet entraîne des appels périodiques du script de l'animation, déclenchant ainsi le gestionnaire streamStatusHandler. Le gestionnaire est alors exécuté, Director remplissant automatiquement les paramètres avec des informations concernant l'évolution des téléchargements.

Paramètres

booléenActivéOuDésactivé Facultatif. Spécifie l'état du gestionnaire.

Exemple

Le gestionnaire `on prepareMovie` suivant active le gestionnaire `on streamStatus` au démarrage de l'animation :

```
-- Lingo syntax
on prepareMovie
    tellStreamStatus(TRUE)
end

// JavaScript syntax
function prepareMovie() {
    tellStreamStatus(TRUE);
}
```

L'instruction suivante détermine l'état du gestionnaire d'état `stream status` :

```
-- Lingo syntax
on mouseDown
    put tellStreamStatus()
end

// JavaScript syntax
function mouseDown() {
    put(tellStreamStatus());
}
```

Voir aussi

on streamStatus

tellTarget()

Syntaxe

```
-- Lingo syntax
spriteObjRef.tellTarget(targetName)

// JavaScript syntax
spriteObjRef.tellTarget(targetName);
```

Description

Commande ; équivalente aux méthodes Flash `beginTellTarget` et `endTellTarget`. La commande `tellTarget()` permet à l'utilisateur de configurer un scénario principal sur lequel agissent les commandes d'image-objet ultérieures. Une fois qu'un clip d'animation ou un niveau contenant une animation Flash chargée a été configuré en tant que cible, certaines commandes agissent sur le composant cible, plutôt que sur le scénario principal. Pour les contraindre à agir de nouveau sur le scénario principal, appelez la commande `endTellTarget()`.

Le seul argument valide de `tellTarget` est le nom de la cible. Il n'existe aucun argument valide pour `endTellTarget`.

Les fonctions d'image-objet Flash affectées par `tellTarget` sont `stop`, `play`, `getProperty`, `setProperty`, `gotoFrame`, `call(image)` et `find(libellé)`. En outre, la propriété d'image-objet `frame` (qui renvoie l'image en cours) est affectée par `tellTarget`.

Paramètres

`nomDeCible` Requis. Spécifie le nom cible.

Exemple

La commande suivante définit le clip comme cible :

```
-- Lingo syntax
sprite(1).tellTarget("myMovieClip")

// JavaScript syntax
sprite(1).tellTarget("myMovieClip");
```

La commande suivante arrête le clip :

```
-- Lingo syntax
sprite(1).stop()

// JavaScript syntax
sprite(1).stop();
```

La commande suivante entraîne la lecture du clip :

```
-- Lingo syntax
sprite(1).play()

// JavaScript syntax
sprite(1).play();
```

La commande suivante entraîne le retour au scénario principal :

```
-- Lingo syntax
sprite(1).endTellTarget()

// JavaScript syntax
sprite(1).endTellTarget();
```

La commande suivante arrête l'animation principale :

```
-- Lingo syntax
sprite(1).stop()

// JavaScript syntax
sprite(1).stop();
```

texture()

Syntaxe

```
member(whichCastmember).texture(whichTexture)
member(whichCastmember).texture[index]
member(whichCastmember).shader(whichShader).texture
member(whichCastmember).model(whichModel).shader.texture
member(whichCastmember).model(whichModel).shaderList.texture
member(whichCastmember).model(whichModel).shaderList[index].texture
member(whichCastmember).modelResource(whichParticleSystemModelResource).texture
```

Description

Propriété 3D d'élément et de matériau ; objet image utilisé par un matériau pour définir l'apparence de la surface d'un modèle. L'image est enrobée sur la géométrie du modèle par le matériau.

Le composant visible d'un matériau est créé avec un maximum de huit couches de textures. Ces huit couches de texture ont été créées à partir d'acteurs bitmap ou d'objets images dans Director ou importées avec des modèles de programmes de modélisation 3D.

Créez et supprimez des textures avec les commandes `newTexture()` et `deleteTexture()`.

Les textures sont enregistrées dans la palette des textures de l'acteur 3D. Elles peuvent être référencées par nom (*`quelleTexture`*) ou par index de palette (*`indexDeTexture`*). Une texture peut être utilisée par n'importe quel nombre de matériaux. Les modifications apportées à une texture apparaissent dans tous les matériaux qui l'utilisent.

Il existe trois types de textures :

`#fromCastmember` ; la texture est créée à partir d'un acteur bitmap avec la commande `newTexture()`.

`#fromImageObject` ; la texture est créée à partir d'un objet image Lingo avec la commande `newTexture()`.

`#importedFromFile` ; la texture est importée avec un modèle à partir d'un programme de modélisation 3D.

La texture d'un système de particules est une propriété de la ressource de modèle, dont le type est `#particle`.

Exemple

L'instruction suivante attribue à la propriété `texture` du matériau surfaceDuMur la texture peintureBleue :

```
member("Room").shader("WallSurface").texture = member("Room").texture("BluePaint")
```

Voir aussi

`newTexture`, `deleteTexture`

time() (système)

Syntaxe

```
-- Lingo syntax
_system.time()

// JavaScript syntax
_system.time();
```

Description

Méthode système ; renvoie l'heure en cours dans l'horloge système sous la forme d'une chaîne. Le format de la chaîne d'heure dépend des paramètres d'heure de l'ordinateur.

Paramètres

Aucune.

Exemple

Le gestionnaire suivant renvoie l'heure en cours dans un champ de texte.

```
-- Lingo syntax
on exitFrame
    member("clock").text = _system.time()
end

// JavaScript syntax
function exitFrame() {
    member("clock").text = _system.time();
}
```

Voir aussi

`date() (système)`, `Système`

timeout()

Syntaxe

```
-- Lingo syntax
timeout(timeoutObjName)

// JavaScript syntax
timeout(timeoutObjName);
```

Description

Fonction de niveau supérieur ; renvoie un objet de temporisation donné.

Pour créer un objet de temporisation et l'ajouter à la liste `timeoutList`, utilisez la méthode `new()`.

Paramètres

`nomObjetTemporisation` Requis. Chaîne spécifiant le nom de l'objet de temporisation à renvoyer.

Exemple

Le gestionnaire suivant supprime l'objet de temporisation appelé Foudre aléatoire :

```
-- Lingo syntax
on exitFrame
    timeout("Random Lightning").forget()
end

// JavaScript syntax
function exitFrame() {
    timeout("Random Lightning").forget();
}
```

Voir aussi

new(), timeoutList, timeoutHandler, time (objet de temporisation), name (temporisation), period, persistent, target

titleMenu()

Syntaxe

```
-- Lingo syntax
dvdObjRef.titleMenu()

// JavaScript syntax
dvdObjRef.titleMenu();
```

Description

Méthode de DVD ; affiche le menu titre.

Paramètres

Aucune.

Exemple

L'instruction suivante affiche le menu titre :

```
-- Lingo syntax
member(1).titleMenu()

// JavaScript syntax
member(1).titleMenu();
```

Voir aussi

DVD

top (3D)

Syntaxe

```
modelResourceObjectReference.top
```

Description

Commande 3D ; utilisée avec une ressource de modèle de type `#box`, cette commande vous permet d'obtenir et de définir la propriété `top` de la ressource de modèle.

La propriété `top` détermine si le haut de la boîte est fermé (`TRUE`) ou ouvert (`FALSE`). La valeur par défaut est `TRUE`.

Paramètres

Aucune.

Exemple

L'instruction suivante vérifie si la valeur de la propriété `top` de l'image-objet 3 dépasse le haut de la scène et, le cas échéant, appelle le gestionnaire `offTopEdge` :

```
-- Lingo syntax
if (sprite(3).top < 0) then
```

```
    offTopEdge()
end if

// JavaScript syntax
if (sprite(3).top < 0) {
    offTopEdge();
}
```

Voir aussi

back, bottom (3D), front

topCap

Syntaxe

```
modelResourceObjectReference.topCap
```

Description

Commande 3D ; utilisée avec une ressource de modèle de type `#cylinder`, cette commande vous permet d'obtenir et de définir la propriété `topCap` de la ressource de modèle.

La propriété `topCap` détermine si l'extrémité supérieure du cylindre est fermée (`TRUE`) ou ouverte (`FALSE`). La valeur par défaut de cette propriété est `FALSE`.

Paramètres

Aucune.

Exemple

L'instruction suivante attribue à la propriété `topCap` de la ressource de modèle Tube la valeur `FALSE`, ce qui signifie que l'extrémité supérieure de ce cylindre est ouverte :

```
-- Lingo syntax
member("3D World").modelResource("Tube").topCap = FALSE

// JavaScript syntax
member("3D World").getPropRef("modelResource", 10).topCap = false;
```

topRadius

Syntaxe

```
modelResourceObjectReference.topRadius
```

Description

Commande 3D ; utilisée avec une ressource de modèle de type `#cylinder`, cette commande vous permet d'obtenir et de définir la propriété `topRadius` de la ressource de modèle sous la forme d'une valeur à virgule flottante.

La propriété `topRadius` détermine le rayon de l'extrémité supérieure du cylindre. Cette propriété doit toujours être supérieure ou égale à 0,0. La valeur par défaut est de 25,0. La définition de la propriété `topRadius` sur la valeur 0,0 produit un cône.

Paramètres

Aucune.

Exemple

L'instruction suivante attribue à la propriété `topRadius` de la ressource de modèle Tube la valeur 0,0. Si le rayon inférieur a une valeur supérieure à 0, les modèles utilisant Tube sont coniques.

```
-- Lingo syntax
member("3D World").modelResource("Tube").topRadius = 0.0

// JavaScript syntax
member("3D World").getPropRef("modelResource", 10).topRadius = 0.0;
```

trace()

Syntaxe

```
-- Lingo syntax
trace(value)

// JavaScript syntax
trace(value);
```

Description

Fonction de niveau supérieur ; évalue une expression et affiche le résultat dans la fenêtre Messages.

Cette méthode remplit la même fonction que la méthode de niveau supérieur `put()` qui est également disponible à la fois en syntaxe Lingo et JavaScript.

Cette méthode peut servir d'outil de débogage pour suivre la valeur des variables au cours de la lecture d'une animation.

Paramètres

`valeur` Requis. Expression à évaluer.

Exemple

L'instruction suivante renvoie la valeur de la variable `counter` dans la fenêtre Messages.

```
-- Lingo syntax
counter = (_system.milliseconds / 1000)
trace(counter)

// JavaScript syntax
var counter = (_system.milliseconds / 1000);
trace(counter);
```

Voir aussi

`put()`

transform (commande)

Syntaxe

```
transform()
```

Description

Commande 3D ; cette commande crée un objet de transformation égal à la transformation d'identité. La transformation d'identité possède des composants de position et de rotation de `vector(0,0,0)` et un composant d'échelle de `vector(1,1,1)`.

Si vous avez besoin de stocker et de recréer des informations de transformation, stockez les propriétés de transformation (position, rotation et échelle), puis recréez la transformation en produisant une transformation d'identité et en définissant la position, la rotation et l'échelle à l'aide des données stockées.

Paramètres

Aucune.

Exemple

L'instruction suivante crée une transformation d'identité et la stocke dans la variable `tTransform` :

```
-- Lingo syntax
tTransform = transform()

// JavaScript syntax
tTransform = transform();
```

Voir aussi

transform (propriété), preRotate, preTranslate(), preScale(), rotate, translate, scale (commande)

translate

Syntaxe

```
member(whichCastmember).node(whichNode).translate(xIncrement, yIncrement, zIncrement {,
relativeTo})
member(whichCastmember).node(whichNode).translate(translateVector {, relativeTo})
transform.translate(xIncrement, yIncrement, zIncrement {, relativeTo})
transform.translate(translateVector {, relativeTo})
```

Description

Commande 3D ; applique une translation après les décalages de position, de rotation et d'échelle d'un objet de transformation d'un nœud référencé ou d'un objet de transformation directement référencé. La translation doit être spécifiée sous la forme d'un jeu de trois incréments le long des trois axes correspondants. Ces incréments peuvent être spécifiés explicitement sous la forme *incrémentX*, *incrémentY* et *incrémentZ*ou par un *vecteurDeTranslation*, dont le composant x correspond à la translation le long de l'axe des x, le composant y à la translation le long de l'axe des y et le composant z à la translation le long de l'axe des z.

Un nœud peut être une caméra, un modèle, une lumière ou un groupe.

Paramètres

incrémentX Requis en cas de spécification d'un jeu de trois incréments. Spécifie l'incrément le long de l'axe des *x*.

incrémentY Requis en cas de spécification d'un jeu de trois incréments. Spécifie l'incrément le long de l'axe des *y*.

incrémentZ Requis en cas de spécification d'un jeu de trois incréments. Spécifie l'incrément le long de l'axe des *z*.

vecteurDeTranslation Requis en cas de spécification d'un vecteur. Spécifie le vecteur contenant les composants *x*, *y* et *z*.

parRapportA Facultatif. Détermine les axes du système de coordonnées utilisés pour appliquer les modifications de translation souhaitées. Le paramètre *parRapportA* peut prendre l'une des valeurs suivantes :

- `#self` applique les incréments en fonction du système de coordonnées local du nœud (axes des x, des y et des z spécifiés pour le modèle en phase de création). Cette valeur est utilisée comme valeur par défaut si vous utilisez la commande `translate` avec une référence de nœud et que le paramètre *parRapportA* n'est pas spécifié.
- `#parent` applique les incréments par rapport au système de coordonnées du parent du nœud. Cette valeur est utilisée comme valeur par défaut si vous utilisez la commande `translate` avec une référence de transformation et que le paramètre *parRapportA* n'est pas spécifié.
- `#world` applique les incréments par rapport au système de coordonnées de l'univers. Lorsque le parent d'un modèle est l'univers, ceci équivaut à utiliser `#parent`.
- `nodeReference` permet de spécifier un nœud servant de base à la translation, la commande appliquant les translations en fonction du système de coordonnées du nœud spécifié.

Exemple

L'exemple suivant construit une transformation à l'aide de la commande transform, puis initialise la position et l'orientation de la transformation dans l'espace avant d'affecter la transformation au modèle Mars. Cet exemple indique ensuite la position résultante du modèle.

```
t =transform()
t.transform.identity()
t.transform.rotate(0, 90, 0)
t.transform.translate(100, 0, 0)
gbModel = member("scene").model("mars")
gbModel.transform = t
put gbModel.transform.position
-- vector(100.0000, 0.0000, 0.0000)
```

Le code Lingo suivant déplace le modèle Bip de 20 unités le long de l'axe des x de son nœud parent :

```
put member("Scene").model("Bip").position
-- vector( -38.5000, 21.2500, 2.0000)
member("Scene").model("Bip").translate(20, 10, -0.5)
put member("Scene").model("Bip").position
-- vector( -18.5000, 31.2500, 1.5000)
```

Voir aussi

`transform (propriété)`, `preTranslate()`, `scale (commande)`, `rotate`

union()

Syntaxe

```
rect(1).union(rect(2))
union (rect1, rect2)
```

Description

Fonction ; renvoie le plus petit rectangle renfermant deux rectangles.

Paramètres

rectangle2 Requis. Spécifie le second rectangle.

Exemple

L'instruction suivante renvoie le rectangle qui renferme les rectangles spécifiés :

```
-- Lingo syntax
put union (rect (0, 0, 10, 10), rect (15, 15, 20, 20))
-- rect (0, 0, 20, 20)
```

or

```
put rect(0, 0, 10, 10).union(rect(15, 15, 20, 20))
--rect (0, 0, 20, 20)

// JavaScript syntax
put ( rect (0, 0, 10, 10).union( rect (15, 15, 20, 20) ) );
// <rect(0, 0, 20, 20)>
```

Voir aussi

`map()`, `rect()`

unLoad() (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.unLoad({toMemberObjRef})

// JavaScript syntax
memberObjRef.unLoad({toMemberObjRef});
```

Description

Méthode d'acteur ; demande à Director de purger les acteurs spécifiés de la mémoire.

Director purge automatiquement les acteurs les moins récemment utilisés pour permettre l'utilisation des méthodes `preLoad()` ou le chargement normal des bibliothèques de distribution.

- Lorsque la méthode `unLoad()` est utilisée sans paramètre, elle purge de la mémoire l'acteur en cours.
- Lorsqu'elle est utilisée avec le paramètre *`réfObjActeurFinal`*, la méthode `unLoad()` purge de la mémoire tous les acteurs compris dans la plage spécifiée.

Si elle est utilisée dans une nouvelle animation sans acteurs chargés, cette méthode renvoie une erreur.

Les acteurs ayant été modifiés en phase de création ou par la définition de `picture`, `pasteClipBoadInto()` et ainsi de suite, ne peuvent pas être purgés de la mémoire.

Paramètres

`réfObjActeurFinal` Facultatif. Référence au dernier acteur de la plage à purger de la mémoire.

Exemple

L'instruction suivante purge de la mémoire l'acteur Vaisseaux :

```
-- Lingo syntax
member("Ships").unLoad()

// JavaScript syntax
member("Ships").unLoad();
```

L'instruction suivante purge de la mémoire les acteurs 10 à 15 :

```
-- Lingo syntax
member(10).unLoad(15)

// JavaScript syntax
member(10).unLoad(15);
```

Voir aussi

Acteur unLoad() (animation) unLoadMember() unLoadMovie()

unLoad() (animation)

Syntaxe

```
-- Lingo syntax
_movie.unLoad({intFromFrameNum} {, intToFrameNum})

// JavaScript syntax
_movie.unLoad({intFromFrameNum} {, intToFrameNum});
```

Description

Méthode d'animation ; purge de la mémoire la plage d'images spécifiée de l'animation.

Cette commande est utile pour forcer la purge des animations lorsque la mémoire diminue.

Vous pouvez utiliser une adresse URL comme référence de fichier.

Si l'animation ne se trouve pas encore dans la RAM, le résultat est -1.

Paramètres

entNumImageInitiale Facultatif. Nombre entier spécifiant le numéro de la première image d'une plage à purger de la mémoire.

entNumImageFinale Facultatif. Nombre entier spécifiant le numéro de la dernière image d'une plage à purger de la mémoire.

Exemple

Les instructions suivantes purgent de la mémoire les images 10 à 25.

```
-- Lingo syntax
_movie.unLoad(10, 25)

// JavaScript syntax
_movie.unLoad(10, 25);
```

Voir aussi

Animation unLoad() (acteur) unLoadMember() unLoadMovie()

unLoadMember()

Syntaxe

```
-- Lingo syntax
_movie.unLoadMember({memberObjRef})
_movie.unLoadMember(fromMemberNameOrNum, toMemberNameOrNum)
```

```
// JavaScript syntax
_movie.unLoadMember({memberObjRef});
_movie.unLoadMember(fromMemberNameOrNum, toMemberNameOrNum);
```

Description

Méthode d'animation ; demande à Director de purger de la mémoire un acteur spécifique ou une plage d'acteurs. Director purge automatiquement les acteurs les moins récemment utilisés pour permettre l'utilisation des méthodes `preLoad()` ou le chargement normal des bibliothèques de distribution.

- Lorsqu'elle est utilisée sans argument, la méthode `unLoadMember()` purge de la mémoire les acteurs figurant dans toutes les images d'une animation.
- Lorsqu'elle n'est utilisée qu'avec l'argument *`réfObjActeur`*, la méthode `unLoadMember()` purge de la mémoire l'acteur spécifié.
- Lorsqu'elle est utilisée avec deux arguments, *`nomOuNumActeurInitial`* et *`nomOuNumActeurFinal`*, la méthode `unLoadMember()` purge de la mémoire tous les acteurs contenus dans la plage spécifiée. Vous pouvez spécifier une plage d'acteurs par numéro d'acteur ou par nom d'acteur.

Paramètres

`réfObjActeur` Facultatif. Référence à l'acteur à purger de la mémoire.

`nomOuNumActeurInitial` Requis en cas de purge d'une plage d'acteurs. Chaîne ou nombre entier spécifiant le nom ou le numéro du premier acteur d'une plage à purger de la mémoire.

`nomOuNumActeurFinal` Requis en cas de purge d'une plage d'acteurs. Chaîne ou nombre entier spécifiant le nom ou le numéro du dernier acteur d'une plage à purger de la mémoire.

Exemple

L'instruction suivante purge de la mémoire l'acteur Ecran1 :

```
-- Lingo syntax
_movie.unLoadMember(member("Screen1"))

// JavaScript syntax
_movie.unLoadMember(member("Screen1"));
```

L'instruction suivante purge de la mémoire tous les acteurs compris entre l'acteur 1 et l'acteur Cinémascope :

```
-- Lingo syntax
_movie.unLoadMember(member(1), member("Big Movie"))

// JavaScript syntax
_movie.unLoadMember(member(1), member("Big Movie"));
```

Voir aussi

`Animation unLoad() (acteur) unLoad() (animation) unLoadMovie()`

unLoadMovie()

Syntaxe

```
-- Lingo syntax
_movie.unLoadMovie(stringMovieName)

// JavaScript syntax
_movie.unLoadMovie(stringMovieName);
```

Description

Méthode d'animation ; purge de la mémoire l'animation préchargée spécifiée.

Cette commande est utile pour forcer la purge des animations lorsque la mémoire diminue.

Vous pouvez utiliser une adresse URL comme référence de fichier.

Si l'animation ne se trouve pas encore dans la RAM, le résultat est -1.

Paramètres

chaîneNomAnimation Requis. Chaîne spécifiant le nom de l'animation à purger de la mémoire.

Exemple

L'instruction suivante vérifie si le plus grand bloc contigu de mémoire disponible est inférieur à 100 Ko et, le cas échéant, purge l'animation Parsifal :

```
-- Lingo syntax
if (_system.freeBlock < (100*1024)) then
    _movie.unLoadMovie("Parsifal")
end if

// JavaScript syntax
if (_system.freeBlock < (100*1024)) {
    _movie.unLoadMovie("Parsifal");
}
```

L'instruction suivante purge l'animation à l'adresse http://www.cbDemille.com/SunsetBlvd.dir :

```
-- Lingo syntax
_movie.unLoadMovie("http://www.cbDemille.com/SunsetBlvd.dir")

// JavaScript syntax
_movie.unLoadMovie("http://www.cbDemille.com/SunsetBlvd.dir");
```

Voir aussi

Animation unLoad() (acteur) unLoad() (animation) unLoadMember()

unregisterAllEvents

Syntaxe

```
-- Lingo syntax
member(whichMember).unregisterAllEvents()

// JavaScript syntax
member(whichMember).unregisterAllEvents();
```

Description

Commande 3D ; annule l'enregistrement de l'acteur référencé pour toutes les notifications d'événements. Tous les gestionnaires précédemment enregistrés pour répondre aux événements utilisant la commande `registerForEvent` ne sont donc plus déclenchés lors de ces événements.

Paramètres

Aucune.

Exemple

L'instruction suivante annule l'enregistrement de l'acteur Séquence pour toutes les notifications d'événements.

```
-- Lingo syntax
member("Scene").unregisterAllEvents()

// JavaScript syntax
member("Scene").unregisterAllEvents();
```

Voir aussi

registerForEvent()

update

Syntaxe

```
-- Lingo syntax
member(whichCastmember).model(whichModel).update

// JavaScript syntax
member(whichCastMember).model(whichModel).update();
```

Description

Commande 3D ; entraîne la mise à jour des animations du modèle sans rendu. Utilisez cette commande pour déterminer la position exacte d'un modèle animé avec Lingo.

Paramètres

Aucune.

Exemple

```
-- Lingo syntax
member(whichCastmember).model(whichModel).update

// JavaScript syntax
member(whichCastMember).getPropRef("model",1).update();
```

updateFrame()

Syntaxe

```
-- Lingo syntax
_movie.updateFrame()

// JavaScript syntax
_movie.updateFrame();
```

Description

Méthode d'animation ; en phase de création du scénario uniquement, cette méthode entre les changements apportés à l'image en cours pendant l'enregistrement du scénario, puis passe à l'image suivante. N'importe quel objet qui était déjà dans l'image quand la session de mise à jour a commencé reste dans l'image. Vous devez appeler une méthode `updateFrame()` pour chaque image que vous mettez à jour.

Paramètres

Aucune.

Exemple

Lorsqu'elle est utilisée dans le gestionnaire suivant, la commande `updateFrame` entre les changements apportés à l'image en cours et passe à l'image suivante chaque fois que Lingo atteint la fin de la boucle de répétition. Le nombre d'images est déterminé par l'argument `numberOfFrames`.

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    horizontal = 0
    vertical = 100
    repeat with i = 1 to numberOfFrames
        _movie.go(i)
        sprite(20).member = member("Ball").number
        sprite(20).locH = horizontal
        sprite(20).locV = vertical
        sprite(20).foreColor = 255
        horizontal = horizontal + 3
        vertical = vertical + 2
        _movie.updateFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    var horizontal = 0;
    var vertical = 100;
    for (var i = 1; i <= numberOfFrames; i++) {
        _movie.go(1);
        sprite(20).member = member("Ball");
        sprite(20).locH = horizontal;
        sprite(20).locV = vertical;
        sprite(20).foreColor = 255;
        horizontal = horizontal + 3;
        vertical = vertical + 2;
        _movie.updateFrame();
    }
    _movie.endRecording();
}
```

Voir aussi

`beginRecording()`, `endRecording()`, `Animation`, `scriptNum`, `tweened`

updateStage()

Syntaxe

```
-- Lingo syntax
_movie.updateStage()

// JavaScript syntax
_movie.updateStage();
```

Description

Méthode d'animation ; rafraîchit la scène immédiatement au lieu de ne la rafraîchir qu'entre les images.

La méthode `updateStage()` rafraîchit les images-objets, effectue les transitions, lit des sons et envoie un message `prepareFrame` (affectant les scripts d'animation et de comportement) ainsi qu'un message `stepFrame` (affectant la liste `actorList`).

Paramètres

Aucune.

Exemple

Le gestionnaire suivant modifie la position horizontale et verticale de l'image-objet et rafraîchit la scène de façon à ce que l'image-objet apparaisse à son nouvel emplacement sans devoir attendre le déplacement de la tête de lecture :

```
-- Lingo syntax
on moveRight(whichSprite, howFar)
    sprite(whichSprite).locH = sprite(whichSprite).locH + howFar
    _movie.updateStage()
end moveRight

// JavaScript syntax
function moveRight(whichSprite, howFar) {
    sprite(whichSprite).locH = sprite(whichSprite).locH + howFar;
    _movie.updateStage();
}
```

Voir aussi

`actorList`, `Animation`, `on prepareFrame`, `on stepFrame`

URLEncode

Syntaxe

```
URLEncode(proplist_or_string {, serverOSString} {, characterSet})
```

Description

Fonction ; renvoie la chaîne en codage URL pour son premier argument. Permet l'utilisation de paramètres CGI dans d'autres commandes. Cette fonction effectue la même conversion que pour les commandes `postNetText` et `getNetText()` lorsqu'elles reçoivent une liste de propriétés.

Paramètres

`propertyListOuChaîne` Requis. Spécifie la liste de propriétés ou la chaîne à coder en URL.

`chaîneOSduServeur` Facultatif. Code les caractères de retour dans *`listeDePropriétésOuChaîne`*. Ce paramètre présente la valeur `UNIX` par défaut, mais peut être défini sur `Win` ou sur `Mac` et convertit les retours chariot du paramètre *`listeDePropriétésOuChaîne`* en ceux utilisés par le serveur. Pour la plupart des applications, ce paramètre n'est pas nécessaire, les sauts de ligne n'étant généralement pas utilisés dans les réponses de formulaires.

jeuDeCaractères Facultatif. Ne s'applique que si l'utilisateur travaille sur un système Shift-JIS (japonais). Les jeux de caractères possibles sont JIS, EUC, ASCII et AUTO. Les données récupérées sont converties de Shift-JIS dans le jeu de caractères désigné. Les données renvoyées sont traitées de la même façon que par getNetText() (converties du jeu de caractères nommé en Shift-JIS). Si la valeur AUTO est utilisée, les données publiées dans le jeu de caractères local ne sont pas converties ; les résultats renvoyés par le serveur sont convertis comme pour getNetText(). ASCII est la valeur par défaut si *jeuDeCaractères* est omis. ASCII n'offre aucune conversion pour la publication ou les résultats.

Exemple

Dans l'exemple suivant, la fonction URLEncode fournit la chaîne codée en URL à une requête CGI à l'emplacement spécifié.

```
URL = "http://aserver/cgi-bin/echoquery.cgi"
gotonetpage URL & "?" & URLEncode( [#name: "Ken", #hobby: "What?"] )
```

Voir aussi

getNetText(), postNetText

value()

Syntaxe

```
value(stringExpression)
```

Description

Fonction ; renvoie la valeur d'une chaîne. Lorsque la méthode value() est appelée, Lingo analyse l'*expressionChaîne* fournie et renvoie sa valeur logique.

Toute expression Lingo pouvant être affichée (put) dans la fenêtre Messages ou définie comme valeur d'une variable est également utilisable avec value().

Les deux instructions Lingo suivantes sont équivalentes :

```
put sprite(2).member.duration * 5
put value("sprite(2).member.duration * 5")
```

Les deux instructions Lingo suivantes sont également équivalentes :

```
x = (the mouseH - 10) / (the mouseV + 10)
x = value("(the mouseH - 10) / (the mouseV + 10)")
```

Les expressions que Lingo ne peut pas analyser produisent des résultats inattendus, mais ne produisent pas d'erreurs Lingo. Le résultat est la valeur de la portion initiale de l'expression jusqu'à la première erreur de syntaxe détectée dans la chaîne.

La fonction value() se révèle utile pour l'analyse d'expressions placées dans des champs de texte par les utilisateurs, les expressions de chaîne transmises à Lingo par des Xtras ou toute autre expression requise pour la conversion d'une chaîne en valeur Lingo.

Gardez à l'esprit que dans certaines situations, l'utilisation de value() impliquant l'intervention des utilisateurs peut se révéler risquée, par exemple lorsqu'un utilisateur saisit le nom d'un gestionnaire personnalisé dans le champ. Cette opération entraînerait l'exécution du gestionnaire lors de sa transmission à la méthode value().

Ne confondez pas les actions de la fonction value avec celles des fonctions integer() et float().

Paramètres

expressionChaîne Requis. Spécifie la chaîne à partir de laquelle une valeur est renvoyée. Cette chaîne peut être constituée de n'importe quelle expression reconnue par Lingo.

Exemple

L'instruction suivante affiche la valeur numérique de la chaîne `"the sqrt of" && "2.0"` :

```
put value("the sqrt of" && "2.0")
```

Le résultat est 1,4142.

L'instruction suivante affiche la valeur numérique de la chaîne Centime :

```
put value("penny")
```

Le résultat affiché dans la fenêtre Messages est la valeur `VOID`, le mot *Centime* n'ayant pas de valeur numérique.

Vous pouvez utiliser la syntaxe suivante pour convertir une chaîne présentée sous forme de liste en liste véritable :

```
myString = "[" & QUOTE & "cat" & QUOTE & ", " & QUOTE & "dog" & QUOTE & "]"
myList = value(myString)
put myList
-- ["cat", "dog"]
```

Cette opération permet de placer une liste dans un champ ou un acteur texte, puis de l'extraire et de la reformater aisément sous forme de liste.

L'instruction suivante analyse la chaîne `"3 5"` et renvoie la valeur de la portion de la chaîne reconnue par Lingo :

```
put value("3 5")
-- 3
```

Voir aussi

`string()`, `integer()`, `float()`

vector()

Syntaxe

```
-- Lingo syntax
vector()
vector(intX, intY, intZ)

// JavaScript syntax
vector();
vector(intX, intY, intZ);
```

Description

Fonction et type de données de niveau supérieur. Décrit un point dans l'espace 3D en fonction de trois paramètres qui correspondent aux distances spécifiques à partir du point de référence le long des axes des *x*, des *y* et des *z*.

Si le vecteur se trouve dans l'espace de l'univers, le point de référence est l'origine de l'univers, `vector(0, 0, 0)`. Si le vecteur se trouve dans l'espace de l'objet, le point de référence constitue la position et l'orientation de l'objet.

Cette méthode renvoie un objet vecteur.

Les valeurs des vecteurs peuvent être manipulées à l'aide des opérateurs `+`, `-`, `*` et `/`. Consultez la définition des différents opérateurs pour plus d'informations.

Paramètres

entX Facultatif. Nombre entier spécifiant le point de l'axe des *x*.

entY Facultatif. Nombre entier spécifiant le point de l'axe des *x*.

entZ Facultatif. Nombre entier spécifiant le point de l'axe des *z*.

Exemple

L'instruction suivante crée un vecteur et l'affecte à la variable `myVector` :

```
-- Lingo syntax
myVector = vector(10.0, -5.0, 0.0)

// JavaScript syntax
var myVector = vector(10.0, -5.0, 0.0);
```

Dans Lingo uniquement, l'instruction suivante additionne deux vecteurs et affecte la valeur résultante à la variable `thisVector` :

```
-- Lingo syntax
thisVector = vector(1.0, 0.0, 0.0) + vector(0.0, -12.5, 2.0)
```

version()

Syntaxe

```
-- Lingo syntax
fileioObjRef.version()

// JavaScript syntax
fileioObjRef.version();
```

Description

Méthode FileIO ; affiche les informations de version et de build FileIO dans la fenêtre Messages.

Paramètres

Aucune.

Exemple

L'instruction suivante affiche la version de Director dans la fenêtre Messages :

```
-- Lingo
put(_player.productVersion)

// Javascript
trace(_player.productVersion)
```

Voir aussi

`Fileio`

voiceCount()

Syntaxe

```
voiceCount()
```

Description

Fonction : renvoie le nombre de voix installées disponibles pour la fonction de conversion de texte en voix. La valeur renvoyée est un nombre entier. Ce nombre de voix peut être utilisé avec `voiceSet()` et `voiceGet()` pour activer une voix spécifique.

Paramètres

Aucune.

Exemple

L'instruction suivante définit la variable `numVoices` sur le nombre de voix de la fonction de conversion de texte en voix :

```
-- Lingo
numVoices = voiceCount()

// Javascript
Var numVoices = voiceCount();
```

Voir aussi

`voiceInitialize()`, `voiceSet()`, `voiceGet()`

voiceGet()

Syntaxe

```
voiceGet()
```

Description

Fonction ; renvoie une liste de propriétés décrivant la voix actuelle utilisée pour la conversion de texte en voix. La liste contient les propriétés suivantes :

- `#name` indique le nom de la voix installée.
- `#age` indique l'âge de la voix. Cette valeur est une chaîne. Les valeurs possibles sont « Teen », « Adult », « Toddler » et « Senior », ainsi que des valeurs numériques telles que « 35 ». Les valeurs réelles dépendent du système d'exploitation, de la version du logiciel de conversion de texte en voix et des voix installées.
- `#gender` indique si la voix est celle d'une femme ou d'un homme. Cette valeur est une chaîne.
- `#index` indique la position de la voix dans la liste des voix installées. Vous pouvez faire référence à une voix par son index lors de l'utilisation de la commande `voiceSet()`.

Utilisez `voiceCount()` pour déterminer le nombre de voix disponibles.

Paramètres

Aucune.

Exemple

L'instruction suivante définit la variable `oldVoice` sur la liste de propriétés décrivant la voix en cours dans la fonction de conversion de texte en voix :

```
-- Lingo
oldVoice = voiceGet()

// Javascript
```

```
Var oldVoice = voiceGet();
```

L'instruction suivante affiche la liste des propriétés de la voix en cours dans la fonction de conversion de texte en voix :

```
-- Lingo
put voiceGet()
-- [#name: "Mary", #age: "teen", #gender: "female", #index: 5]

// Javascript
trace(voiceget())
// <[#name: "Mary", #age: "teen", #gender: "female", #index: 5]>
```

Voir aussi

`voiceInitialize()`, `voiceCount()`, `voiceSet()`, `voiceGet()`

voiceGetAll()

Syntaxe

```
voiceGetAll()
```

Description

Fonction ; renvoie la liste des voix disponibles installées sur l'ordinateur. Cette liste est composée de listes de propriétés, une pour chaque voix disponible.

Chaque liste de propriétés contient les propriétés suivantes :

- `#name` indique le nom de la voix installée.
- `#age` indique l'âge de la voix. Cette valeur est une chaîne. Les valeurs possibles sont « Teen », « Adult », « Toddler » et « Senior », ainsi que des valeurs numériques telles que « 35 ». Les valeurs réelles dépendent du système d'exploitation, de la version du logiciel de conversion de texte en voix et des voix installées.
- `#gender` indique si la voix est celle d'une femme ou d'un homme.
- `#index` indique la position de la voix dans la liste des voix installées. Vous pouvez faire référence à une voix par son index lors de l'utilisation de la commande `voiceSet()`.

Vous pouvez également utiliser `voiceCount()` pour déterminer le nombre de voix disponibles.

Paramètres

Aucune.

Exemple

L'instruction suivante définit la variable `currentVoices` sur la liste des voix installées sur l'ordinateur :

```
-- Lingo
currentVoices = voiceGetAll()

// Javascript
Var currentVoices = voiceGetAll();
```

L'instruction suivante affiche la liste des propriétés décrivant chacune des voix installées de la fonction de conversion de texte en voix :

```
-- Lingo
put voiceGetAll()
```

```
-- [[#name: "Mary", #age: "teen", #gender: "female", #index: 1], [#name: "Joe", #age:
"adult", #gender: "male", #index: 2]]

// Javascript
trace(voiceGetAll());
// <[[#name: "Mary", #age: "teen", #gender: "female", #index: 1], [#name: "Joe", #age:
"adult", #gender: "male", #index: 2]]>
```

Voir aussi

voiceInitialize(), voiceCount(), voiceSet(), voiceGet()

voiceGetPitch()

Syntaxe

```
voiceGetPitch()
```

Description

Fonction ; renvoie la tonalité actuelle de la voix actuelle, sous forme de nombre entier. La plage des valeurs valides dépend du système d'exploitation et du logiciel vocal.

Paramètres

Aucune.

Exemple

Les instructions suivantes contrôlent si la tonalité de la voix actuelle est supérieure à 10 et la redéfinissent sur 10 si c'est le cas :

```
-- Lingo syntax
if voiceGetPitch() > 10 then
    voiceSetPitch(10)
end if

// JavaScript syntax
if (voiceGetPitch() > 10) {
    voiceSetPitch(10);
}
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceGetRate()

Syntaxe

```
voiceGetRate()
```

Description

Fonction ; renvoie la cadence de lecture actuelle de la fonction de conversion de texte en voix. La valeur renvoyée est un nombre entier. La plage des valeurs valides dépend du système d'exploitation et du logiciel vocal. Il s'agit généralement de valeurs comprises entre -10 et 10.

Paramètres

Aucune.

Exemple

Les instructions suivantes vérifient que la cadence de synthèse vocale est inférieure à 50 et la définit à 50 le cas échéant :

```
-- Lingo syntax
if voiceGetRate() < 50 then
    voiceSetRate(50)
end if

// JavaScript syntax
if (voiceGetRate() < 50) {
    voiceSetRate(50);
}
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceGetVolume()

Syntaxe

```
voiceGetVolume()
```

Description

Fonction : renvoie le volume actuel de la fonction de conversion de texte en voix. La valeur renvoyée est un nombre entier. La plage des valeurs valides dépend du système d'exploitation.

Paramètres

Aucune.

Exemple

Les instructions suivantes vérifient si le volume de la conversion de texte en voix s'élève au moins à 55 et le définissent sur 55 s'il est inférieur :

```
-- Lingo syntax
if voiceGetVolume() < 55 then
    voiceSetVolume(55)
end if

// JavaScript syntax
if (voiceGetVolume() < 55) {
    voiceSetVolume(55);
}
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceInitialize()

Syntaxe

```
voiceInitialize()
```

Description

Commande ; charge le logiciel de conversion de texte en voix. Si la commande `voiceInitialize()` renvoie la valeur 0, cela signifie que le logiciel de conversion de texte en voix n'est pas présent ou que son chargement a échoué.

Cette commande renvoie la valeur 1 s'il le détecte, et 0 dans le cas contraire.

Paramètres

Aucune.

Exemple

Les instructions suivantes chargent le logiciel de conversion de texte en voix, puis vérifient si ce chargement est terminé avant d'utiliser la commande `voiceSpeak()` qui prononce la phrase « Bienvenue dans Shockwave. » :

```
-- Lingo syntax
err = voiceInitialize()
if err = 1 then
    voiceSpeak("Welcome to Shockwave")
else
    alert "Text-to-speech software failed to load."
end if

// JavaScript syntax
err = voiceInitialize();
if (err == 1) {
    voiceSpeak("Welcome to Shockwave");
} else {
    alert("Text-to-speech software failed to load.");
}
```

Voir aussi

`voiceCount()`, `voiceSet()`, `voiceGet()`

voicePause()

Syntaxe

```
voicePause()
```

Description

Commande ; met la sortie vocale en pause. La commande renvoie la valeur 1 en cas de réussite, et 0 dans le cas contraire.

Paramètres

Aucune.

Exemple

Les instructions suivantes provoquent l'interruption de la fonction de conversion de texte en voix lorsque l'utilisateur clique avec la souris :

```
-- Lingo syntax
on mouseUp
    voicePause()
end mouseUp

// JavaScript syntax
function mouseUp() {
    voicePause();
}
```

Voir aussi

voiceSpeak(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceResume()

Syntaxe

```
voiceResume()
```

Description

Commande ; reprend la sortie vocale. La commande renvoie la valeur 1 en cas de réussite, et 0 dans le cas contraire.

Paramètres

Aucune.

Exemple

Les instructions suivantes réactivent la fonction vocale lorsque la tête de lecture se déplace sur l'image suivante dans le scénario :

```
-- Lingo syntax
on exitFrame
    voiceResume()
end exitFrame

// JavaScript syntax
function exitFrame() {
    voiceResume();
}
```

Voir aussi

voiceSpeak(), voicePause(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceSet()

Syntaxe

```
voiceSet(integer)
```

Description

Commande : définit la voix actuelle de la fonction de conversion de texte en voix. Dans ce cas, la commande renvoie la nouvelle valeur définie. Utilisez `voiceCount()` pour déterminer le nombre de voix disponibles.

Paramètres

entier Requis. Nombre entier spécifiant le numéro de voix de la fonction de conversion de texte en voix à utiliser. La plage des valeurs valides dépend du nombre de voix installées sur l'ordinateur de l'utilisateur. Si une valeur non comprise dans cette plage est spécifiée, la voix est définie sur la valeur autorisée la plus proche.

Exemple

L'instruction suivante définit la voix en cours de la conversion de texte en voix sur la troisième voix installée sur l'ordinateur :

```
voiceSet(3)
```

Voir aussi

voiceInitialize(), voiceCount(), voiceGet()

voiceSetPitch()

Syntaxe

```
voiceSetPitch(integer)
```

Description

Commande ; définit la tonalité de la voix actuelle de la fonction de conversion de texte en voix sur la valeur spécifiée. La valeur renvoyée est la nouvelle valeur de tonalité définie.

Paramètres

entier Requis. Nombre entier spécifiant la tonalité de la voix de la fonction de conversion de texte en voix. La plage des valeurs valides dépend du système d'exploitation et du logiciel vocal.

Exemple

L'instruction suivante définit la tonalité de la voix actuelle sur 75 :

```
-- Lingo
voiceSetPitch(75)

// Javascript
voiceSetPitch(75);
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceSetRate()

Syntaxe

```
voiceSetRate(integer)
```

Description

Commande ; définit la cadence de la voix actuelle de la fonction de conversion de texte en voix sur la valeur entière spécifiée. Cette commande renvoie la nouvelle valeur définie.

Paramètres

entier Requis. Nombre entier spécifiant la cadence de lecture utilisée par le logiciel de conversion de texte en voix. La plage des valeurs valides dépend du système d'exploitation. En règle générale, les valeurs comprises entre -10 et 10 conviennent à la plupart des logiciels de conversion de texte en voix. Si une autre valeur est spécifiée, la cadence est définie sur la valeur admise la plus proche.

Exemple

L'instruction suivante définit la cadence de lecture de la fonction de conversion de texte en voix sur 7.

```
-- Lingo
voiceSetRate(7)

// Javascript
voiceSetRate(7);
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceSetVolume()

Syntaxe

```
voiceSetVolume(integer)
```

Description

Commande ; définit le volume actuel de la fonction de conversion de texte en voix.

Paramètres

entier Requis. Nombre entier spécifiant le volume de la fonction de conversion de texte en voix. La plage des valeurs valides dépend du système d'exploitation. Dans ce cas, la commande renvoie la nouvelle valeur définie. Si une valeur non admise est spécifiée, le volume est défini sur la valeur admise la plus proche.

Exemple

L'instruction suivante définit le volume de la fonction de conversion de texte en voix sur 55.

```
-- Lingo
voiceSetVolume(55)

// Javascript
voiceSetVolume(55);
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceState(), voiceWordPos()

voiceSpeak()

Syntaxe

```
-- Lingo syntax
voiceSpeak("string")
```

```
// JavaScript syntax
voiceSpeak("string"); // documentation n/a
```

Description

Commande ; active la lecture de la chaîne spécifiée par le logiciel de conversion de texte en voix. Si cette commande est utilisée, toute lecture en cours est automatiquement interrompue par la nouvelle chaîne.

Paramètres

chaîne Requis. Chaîne prononcée par le logiciel de conversion de texte en voix.

Exemple

L'instruction suivante entraîne la lecture de la chaîne « Bienvenue dans Shockwave » par le logiciel de conversion de texte en voix :

```
voiceSpeak("Welcome to Shockwave")
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceState()

Syntaxe

```
-- Lingo syntax
voiceState()

// JavaScript syntax
voiceState(); // documentation n/a
```

Description

Fonction ; renvoie l'état actuel de la voix sous forme d'un symbole. Les valeurs possibles sont `#playing`, `#paused` et `#stopped`.

Paramètres

Aucune.

Exemple

Les instructions suivantes vérifient si le logiciel de conversion de texte en voix est actuellement actif et définissent la voix sur 1 dans le cas contraire :

```
--Lingo syntax
if voiceState() <> #playing then
    voiceSet(1)
end if

// JavaScript syntax
if (voiceState() != symbol("playing")) {
    voiceSet(1);
}
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceWordPos(), voiceSpeak()

voiceStop()

Syntaxe

```
-- Lingo syntax
voiceStop()

// JavaScript syntax
voiceStop(); // documentation n/a
```

Description

Commande ; arrête la sortie vocale de la conversion de texte en voix et vide la mémoire tampon de cette fonction. La commande renvoie la valeur 1 en cas de réussite, et 0 dans le cas contraire.

Paramètres

Aucune.

Exemple

Les instructions suivantes arrêtent la fonction vocale lorsque la tête de lecture se déplace sur l'image suivante dans le scénario :

```
-- Lingo syntax
on exitFrame
    voiceStop()
end exitFrame

// JavaScript syntax
function exitFrame() {
    voiceStop();
}
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos(), voiceSpeak()

voiceWordPos()

Syntaxe

```
-- Lingo syntax
voiceWordPos()

// JavaScript syntax
voiceWordPos(); // documentation n/a
```

Description

Fonction ; renvoie un nombre entier indiquant la position du mot actuellement en cours de lecture au sein de la chaîne qui le contient. Par exemple, si un acteur contenant 15 mots est lu et que c'est le cinquième mot de l'acteur qui est en cours de lecture lorsque cette fonction est activée, la valeur renvoyée est 5.

Paramètres

Aucune.

Exemple

Les instructions suivantes entraînent la lecture de la phrase « Bonjour, comment allez-vous ? » et affichent la position du mot en cours dans la fenêtre Messages. La fonction `voiceWordPos()` étant appelée immédiatement après l'utilisation de la commande `voiceSpeak()`, la valeur renvoyée est 1.

```
-- Lingo syntax
voiceSpeak("Hello, how are you?")
put voiceWordPos()
-- 1

// JavaScript syntax
voiceSpeak("Hello, how are you?");
put(voiceWordPos());
// 1
```

Voir aussi

`voiceSpeak()`, `voicePause()`, `voiceResume()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceSpeak()`

voidP()

Syntaxe

```
-- Lingo syntax
voidP(variableName)

// JavaScript syntax
variableName == null
```

Description

Fonction ; détermine si une variable spécifiée comporte une valeur. Si la variable n'a aucune valeur ou est égale à `VOID`, cette fonction renvoie `TRUE`. Si la variable comporte une valeur différente de `VOID`, cette fonction renvoie `FALSE`.

Paramètres

`nomDeVariable` Requis. Spécifie la variable à tester.

Exemple

L'instruction suivante vérifie si la variable `answer` comporte une valeur initiale :

```
-- Lingo syntax
put voidP(answer)

// JavaScript syntax
put(answer == null));
```

Voir aussi

`ilk()`, `VOID`

window()

Syntaxe

```
-- Lingo syntax
window(stringWindowName)

// JavaScript syntax
window(stringWindowName);
```

Description

Fonction de niveau supérieur ; renvoie une référence à une fenêtre spécifiée.

La fenêtre spécifiée doit contenir une animation Director.

Les fenêtres contenant les animations sont pratiques pour créer des palettes flottantes, des tableaux de commande indépendants et des fenêtres de formes différentes. L'utilisation de fenêtres contenant les animations vous permet d'ouvrir plusieurs animations simultanément et de les faire dialoguer.

Paramètres

`chaîneNomDeFenêtre` Requis. Chaîne spécifiant le nom de la fenêtre à référencer.

Exemple

L'instruction suivante définit la variable `myWindow` sur la fenêtre Collections :

```
-- Lingo syntax
myWindow = window("Collections")

// JavaScript syntax
var myWindow = window("Collections");
```

Voir aussi

`Fenêtre`

WindowOperation

Syntaxe

```
WindowOperation(MUIObject, operation)
```

Description

Cette commande permet de contrôler la fenêtre d'une boîte de dialogue générale.

Remplacer ce paramètre par une valeur déterminant l'action que doit effectuer la fenêtre. Les valeurs possibles et leurs résultats sont indiqués dans le tableau ci-dessous.

Valeurs possibles	Résultat
#show	Affiche une boîte de dialogue non modale uniquement (pour ouvrir une boîte de dialogue modale, utilisez la commande Run).
#hide	Masque une boîte de dialogue non modale (pour fermer une boîte de dialogue modale, utilisez la commande Stop).
#center	Positionne la fenêtre au centre de l'écran.
#zoom	Envoie un message lorsque l'utilisateur clique sur la case de zoom de la fenêtre. Le gestionnaire de rappel doit redimensionner la boîte de dialogue, si vous souhaitez le redimensionnement de la fenêtre lorsque l'utilisateur clique sur la case de zoom.
#tipsOn	Active les info-bulles (propriété réservée aux versions ultérieures de l'Xtra MUI).
#tipsOff	Désactive les info-bulles (propriété réservée aux versions ultérieures de l'Xtra MUI).

Exemple

Ce gestionnaire vérifie l'existence de l'objet MUIObject et affiche la boîte de dialogue si l'instruction est la suivante :

```
--Lingo syntax
on showDialog
   global MUIObject
   if objectP( MUIObject ) then
      WindowOperation( MUIObject, #show )
   end if
end showDialog
```

L'instruction suivante masque la boîte de dialogue créée à partir de l'objet MUIObject :

```
WindowOperation(MUIObject, #hide)
```

windowPresent()

Syntaxe

```
-- Lingo syntax
_player.windowPresent(stringWindowName)

// JavaScript syntax
_player.windowPresent(stringWindowName);
```

Description

Méthode de lecteur ; indique si l'objet spécifié par *chaîneNomDeFenêtre* est exécuté sous la forme d'une animation dans une fenêtre (`TRUE`) ou non (`FALSE`).

Si une fenêtre a été ouverte, la valeur de `windowPresent()` reste `TRUE` pour cette fenêtre, jusqu'à ce que cette dernière soit supprimée de la propriété `windowList`.

L'argument *chaîneNomDeFenêtre* doit correspondre au nom de la fenêtre tel qu'il apparaît dans la propriété `windowList`.

Paramètres

chaîneNomDeFenêtre Requis. Chaîne spécifiant le nom de la fenêtre à tester.

Exemple

L'instruction suivante vérifie si l'objet maFenêtre est une animation dans une fenêtre et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(_player.windowPresent(myWindow))

// JavaScript syntax
put(_player.windowPresent(myWindow));
```

Voir aussi

Lecteur, windowList

worldSpaceToSpriteSpace

Syntaxe

```
-- Lingo syntax
member(whichCastmember).camera(whichCamera).worldSpaceToSpriteSpace(vector)

// JavaScript syntax
member(whichCastmember).camera(whichCamera).worldSpaceToSpriteSpace(vector);
```

Description

Commande 3D ; renvoie le point du cadre de la caméra au niveau duquel une position spécifiée relative à l'univers apparaîtrait. La position renvoyée par cette commande est relative au coin supérieur gauche du cadre de la caméra.

Si la position spécifiée est en dehors du champ de la caméra, cette commande renvoie `void`.

Paramètres

vecteur Requis. Spécifie la position par rapport à l'univers qui apparaîtrait.

Exemple

L'instruction suivante indique que l'origine de l'univers, spécifiée par vector (0, 0, 0), apparaît au point (250, 281) du cadre de la caméra :

```
-- Lingo syntax
put sprite(5).camera.worldSpaceToSpriteSpace(vector(0, 0, 0))
-- point(250, 281)

// JavaScript syntax
put(sprite(5).camera.worldSpaceToSpriteSpace(vector(0,0,0)));
```

Voir aussi

spriteSpaceToWorldSpace, rect (caméra)

writeChar()

Syntaxe

```
-- Lingo syntax
fileioObjRef.writeChar(stringChar)

// JavaScript syntax
```

```
fileioObjRef.writeChar(stringChar)
```

Description

Méthode FileIO ; écrit un caractère ASCII spécifié dans un fichier.

Vous devez avoir ouvert un fichier à l'aide de la méthode `openFile()` avant d'utiliser la méthode `writeChar()` pour écrire un caractère.

Paramètres

`chaîneCaractère` Requis. Spécifie le caractère ASCII à écrire dans le fichier.

Exemple

L'instruction suivante ouvre le fichier c:\xtra.txt avec l'autorisation Lecture/écriture, écrit le caractère « d » dans le fichier, puis ferme ce dernier.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
objFileio.writeChar("d")
objFileio.closeFile()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
objFileio.writeChar("d");
objFileio.closeFile();
```

Voir aussi

Fileio

writeReturn()

Syntaxe

```
-- Lingo syntax
fileioObjRef.writeReturn(symbol(""))

// JavaScript syntax
fileioObjRef.writeReturn(symbol(""));
```

Description

Méthode FileIO ; insère un saut de ligne dans un fichier.

Paramètres

Aucune.

Exemple

L'instruction suivante ouvre le fichier c:\xtra.txt avec l'autorisation Lecture/écriture, écrit la ligne « Première ligne » dans le fichier, insère un saut de ligne, écrit la seconde ligne « Seconde ligne », puis ferme le fichier.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
objFileio.writeString("First Line")
objFileio.writeReturn(#windows)
```

```
objFileio.writeString("Second Line")
objFileio.closeFile()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
objFileio.writeString("First Line");
objFileio.writeReturn(Symbol("windows"));
objFileio.writeString("Second Line");
objFileio.closeFile();
```

Voir aussi

Fileio

writeString()

Syntaxe

```
-- Lingo syntax
fileioObjRef.writeString(string)

// JavaScript syntax
fileioObjRef.writeString(string)
```

Description

Méthode FileIO ; écrit une chaîne terminée par un caractère nul dans un fichier.

Paramètres

chaîne Requis. Chaîne à écrire dans un fichier.

Exemple

L'instruction suivante ouvre le fichier c:\xtra.txt avec l'autorisation Lecture/écriture, écrit la ligne « Première ligne » et la seconde ligne « Seconde ligne », puis ferme le fichier.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
objFileio.writeString("First Line")
objFileio.writeString("Second Line")
objFileio.closeFile()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
objFileio.writeString("First Line");
objFileio.writeString("Second Line");
objFileio.closeFile();
```

Voir aussi

Fileio

xtra()

Syntaxe

```
-- Lingo syntax
xtra(xtraNameOrNum)

// JavaScript syntax
xtra(xtraNameOrNum);
```

Description

Fonction de niveau supérieur ; renvoie une instance d'un Xtra spécifié.

Si le Xtra spécifié est introuvable, cette fonction renvoie une référence à un objet vide.

Vous pouvez voir un exemple d'utilisation de `xtra` dans une animation en consultant l'animation Read and Write Text du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Paramètres

`nomOuNumXtra` Requis. Chaîne spécifiant le nom du Xtra à renvoyer ou nombre entier indiquant la position d'index de cet Xtra. Les noms de chaîne ne font pas l'objet d'une distinction majuscules-minuscules.

Exemple

L'instruction suivante définit la variable `myNetLingo` sur l'Xtra NetLingo :

```
-- Lingo syntax
myNetLingo = xtra("netlingo")

// JavaScript syntax
var myNetLingo = xtra("netlingo");
```

zoomBox

Syntaxe

```
-- Lingo syntax
zoomBox startSprite, endSprite {,delayTicks}

// JavaScript syntax
zoomBox(startSprite, endSprite {,delayTicks}); // not yet documented
```

Description

Commande ; crée un effet de zoom, semblable à l'agrandissement de fenêtres dans le Finder Mac. L'effet de zoom démarre au niveau du rectangle de délimitation d'une image-objet initiale spécifiée et s'achève au niveau du rectangle de délimitation d'une image-objet finale spécifiée. La commande `zoomBox` utilise la logique suivante lors de l'exécution :

1. Rechercher *`imageObjetDestination`* dans l'image en cours : sinon,
2. Rechercher *`imageObjetDestination`* dans l'image suivante.

Notez que la commande `zoomBox` ne fonctionne pas si l'*`imageObjetDestination`* se trouve dans la même piste que l'*`imageObjetSource`*.

Paramètres

imageObjetSource Requis. Spécifie l'image-objet initiale.

imageObjetDestination Requis. Spécifie l'image-objet finale.

délaiEnBattements Facultatif. Spécifie le délai en battements entre chaque mouvement des rectangles de zoom. Si *délaiEnBattements* n'est pas spécifié, le délai est de 1.

Exemple

L'instruction suivante crée un effet de zoom entre les images-objets 7 et 3 :

```
-- Lingo syntax
zoomBox 7, 3

// JavaScript syntax
zoomBox(7, 3); // not yet documented
```

Chapitre 13 : Opérateurs

Ce chapitre répertorie dans l'ordre alphabétique tous les opérateurs disponibles dans Director®.

La plupart de ces opérateurs ne s'appliquent qu'à Lingo. Toutefois, la syntaxe JavaScript comporte certains opérateurs semblables ou identiques aux opérateurs Lingo répertoriés ici ; lorsque tel est le cas, ce chapitre indique la syntaxe JavaScript et fournit des exemples d'utilisation de cette dernière pour vous aider à mettre en correspondance les opérateurs Lingo avec leurs équivalents JavaScript les plus proches. Pour plus d'informations concernant les opérateurs de la syntaxe JavaScript, consultez « Principes de base de la programmation dans Director », page 4.

(symbole)

Syntaxe

```
--Lingo syntax
#symbolName

// JavaScript syntax
symbol("symbolName");
```

Description

Opérateur de symbole ; définit un symbole, unité autonome pouvant représenter une condition ou un indicateur. La valeur *`nomDeSymbole`* commence par un caractère alphabétique et peut être suivie de plusieurs caractères alphabétiques ou numériques.

Un symbole vous permet d'effectuer les opérations suivantes :

- Affecter une valeur à une variable.
- Comparer des chaînes, des entiers, des rectangles et des points.
- Passer un paramètre à un gestionnaire ou à une méthode.
- Renvoyer une valeur à partir d'un gestionnaire ou d'une méthode.

Les symboles prennent moins de place que les chaînes et sont plus facilement manipulables, mais ne sont pas formés de caractères individuels de la même manière qu'une chaîne. Vous pouvez convertir un symbole en chaîne pour l'afficher, à l'aide de la fonction `string`.

Les points suivants concernant la syntaxe des symboles sont très importants :

- La différence entre les majuscules et les minuscules n'a pas d'importance dans les symboles.
- Les symboles ne peuvent pas commencer pas un chiffre.
- Vous ne pouvez pas utiliser d'espaces, mais pouvez employer des caractères de soulignement pour les simuler.
- Les symboles utilisent les 128 caractères ASCII. Les lettres portant des marques diacritiques ou d'accent sont traitées en fonction de leur lettre de base.
- Vous ne pouvez pas utiliser de points dans les symboles.

Tous les symboles, variables globales et noms de paramètres transmis aux variables globales sont conservés dans une table commune.

Exemple

L'instruction suivante affecte le symbole `#Playing` à la variable nommée état :

```
-- Lingo syntax
state = #Playing

// JavaScript syntax
var state = symbol("Playing");
```

Voir aussi

ilk(), string(), symbol(), symbolP()

. (opérateur point)

Syntaxe

```
-- Lingo syntax
objectReference.objectProperty
textExpression.objectProperty
object.commandOrFunction()

// JavaScript syntax
objectReference.objectProperty;
textExpression.objectProperty;
object.commandOrFunction();
```

Description

Opérateur ; utilisé pour tester ou définir les propriétés des objets, émettre une commande ou exécuter une fonction de l'objet. L'objet peut être un acteur, une image-objet, une liste de propriétés, un objet enfant d'un script parent ou un comportement.

Exemple

L'instruction suivante indique l'acteur actuel contenu dans l'image-objet de la piste 10 :

```
-- Lingo syntax
put(sprite(10).member)

// JavaScript syntax
put(sprite(10).member);
```

Pour utiliser une autre syntaxe et appeler une fonction, utilisez la forme :

```
-- Lingo syntax
myColorObject = color(124, 22, 233)
put(myColorObject.ilk())
-- #color

// JavaScript syntax
var myColorObject = color(124, 22, 233);
put(myColorObject.ilk());
// #color
```

- (signe moins)

Syntaxe

```
-- Lingo syntax
(Negation): -expression
(Subtraction): expression1 - expression2

// JavaScript syntax
(Negation): -expression
(Subtraction): expression1 - expression2
```

Description

Opérateur mathématique ; lorsqu'il est utilisé pour une négation, l'opérateur - (moins) inverse le signe de la valeur de l'*expression* ; lorsqu'il est utilisé pour une soustraction, l'opérateur - (moins) effectue la soustraction arithmétique de deux expressions numériques, en soustrayant *expression2* de *expression1*.

Lorsqu'il est utilisé pour une négation, l'opérateur - (moins) est un opérateur arithmétique avec un niveau de priorité de 5.

Lorsqu'il est utilisé pour une soustraction et que les deux expressions correspondent à des nombres entiers, la différence est également un nombre entier. Si l'une ou les deux expressions sont des nombres à virgule flottante, la différence est un nombre à virgule flottante. L'opérateur - (moins) est un opérateur arithmétique avec un niveau de priorité de 3.

Exemple

(Négation) : L'instruction suivante inverse le signe de l'expression 2 + 3 :

```
-- Lingo syntax
put(-(2 + 3))

// JavaScript syntax
put(-(2 + 3));
```

Le résultat est -5.

(Soustraction) : L'instruction suivante soustrait le nombre entier 2 de 5 et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(5 - 2)

// JavaScript syntax
put(5 - 2);
```

Le résultat est 3, qui est un nombre entier.

(Soustraction) : Cette instruction soustrait le nombre à virgule flottante 1,5 de 3,25 et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(3.25 - 1.5)

// JavaScript syntax
put(3.25 - 1.5);
```

Le résultat est 1,75, qui est un nombre à virgule flottante.

-- (séparateur de commentaires)

Syntaxe

```
-- Lingo syntax
-- comment

// JavaScript syntax
// comment
```

Description

Séparateur de commentaires ; indique le début d'un commentaire dans un script. Dans n'importe quelle ligne de code, ce qui se trouve entre le symbole de commentaire (double trait d'union) et le caractère de retour chariot est interprété comme un commentaire, et non comme une instruction Lingo.

Exemple

Le gestionnaire suivant utilise un double trait d'union pour transformer les deuxième, quatrième et sixième lignes en commentaire :

```
-- Lingo syntax
on resetColors
    -- This handler resets the sprite's colors.
    sprite(1).forecolor = 35
    -- bright red
    sprite(1).backcolor = 36
    -- light blue
end

// JavaScript syntax
function resetColors() {
    // this handler resets the sprite's colors
    sprite(1).forecolor = 35;
    // bright red
    sprite(1).backcolor = 36;
    // light blue
}
```

&, + (opérateur de concaténation)

Syntaxe

```
-- Lingo syntax
expression1 & expression2

// JavaScript syntax
expression1 + expression2
```

Description

Opérateur de chaîne ; concatène les chaînes de deux expressions. Si *expression1* ou *expression2* est un nombre, elle est d'abord convertie en chaîne. Le résultat est une chaîne.

Cet opérateur de chaîne a un niveau de priorité de 2.

Lingo vous permet d'utiliser certaines commandes et fonctions ne comportant qu'un seul argument sans que ce dernier soit encadré de parenthèses. Lorsque l'argument comprend un opérateur, Lingo n'interprète que le premier argument comme partie de la fonction, ce qui risque de provoquer une confusion.

Evitez ce problème en encadrant de parenthèses l'expression comprenant un opérateur. Les parenthèses suppriment toute confusion en modifiant la priorité avec laquelle Lingo traite l'opérateur, l'entraînant à traiter les deux parties de l'argument comme un argument complet.

Exemple

L'instruction suivante concatène les chaînes « abra » et « cadabra » et affiche la chaîne résultante dans la fenêtre Messages :

```
-- Lingo syntax
put("abra" & "cadabra")

// JavaScript syntax
put("abra" + "cadabra");
```

Le résultat correspond à la chaîne « abracadabra ».

L'instruction suivante concatène le contenu de la variable `prix` et la chaîne « $ ». La chaîne concaténée est alors affectée à l'acteur champ Prix :

```
-- Lingo syntax
member("Price").text = "$" & price

// JavaScript syntax
member("Price").text = "$" + price;
```

&&, + (opérateur de concaténation)

Syntaxe

```
-- Lingo syntax
expression1 && expression2

// JavaScript syntax
expression1 + expression2
```

Description

Opérateur de chaîne ; concatène deux expressions et insère un espace entre les expressions chaînes d'origine. Si *`expression1`* ou *`expression2`* est un nombre, elle est d'abord convertie en chaîne. Le résultat est une chaîne.

Cet opérateur de chaîne a un niveau de priorité de 2.

Exemple

L'instruction suivante concatène les chaînes « abra » et « cadabra » et insère un espace entre ces deux chaînes :

```
-- Lingo syntax
put("abra" && "cadabra")

// JavaScript syntax
put("abra " + "cadabra");
```

Le résultat correspond à la chaîne « abra cadabra ».

L'instruction suivante concatène les chaînes « Date : » et la date du jour au format long et insère un espace entre ces deux chaînes :

```
-- Lingo syntax
put("Today is" && date())
```

```
// JavaScript syntax
put("Today is " + Date());
```

() (parenthèses)

Syntaxe

```
-- Lingo syntax
(expression)

// JavaScript syntax
(expression)
```

Description

Opérateur de regroupement ; effectue une opération de regroupement pour contrôler l'ordre d'exécution des opérateurs d'une expression. Cet opérateur permet de contrôler l'ordre dans lequel les opérateurs sont exécutés dans cette expression et supplante l'ordre de priorité automatique, de sorte que l'expression entre parenthèses soit évaluée en premier. Lorsque des parenthèses sont imbriquées, le contenu des parenthèses intérieures est évalué avant celui des parenthèses extérieures.

Cet opérateur de regroupement a un niveau de priorité de 5.

Sachez que Lingo vous permet d'utiliser certaines commandes et fonctions ne prenant qu'un seul argument sans que ce dernier soit encadré de parenthèses. Lorsque l'argument comprend un opérateur, Lingo n'interprète que le premier argument comme partie de la fonction, ce qui risque de provoquer une confusion.

Par exemple, la commande `open window` permet à un argument de spécifier une fenêtre à ouvrir. Si vous utilisez l'opérateur `&` pour définir un nom de fichier et un chemin, Director n'interprète que la chaîne précédant l'opérateur `&` comme nom de fichier. Par exemple, Lingo interprète l'instruction `open window the applicationPath & "Animation"` comme `(open window the applicationPath) & ("Animation")`. Evitez ce problème en encadrant de parenthèses l'expression comprenant un opérateur, comme suit :

```
-- Lingo syntax
open window (the applicationPath & "theMovie")

// JavaScript syntax
window(the applicationPath + "theMovie").open();
```

Exemple

Les instructions suivantes utilisent l'opérateur de regroupement pour modifier l'ordre dans lequel les différentes opérations se produisent (le résultat apparaît en dessous de chaque instruction) :

```
-- Lingo syntax
put((2 + 3) * (4 + 5))
-- 45
put(2 + (3 * (4 + 5)))
-- 29
put(2 + 3 * 4 + 5)
-- 19

// JavaScript syntax
put((2 + 3) * (4 + 5));
// 45
put(2 + (3 * (4 + 5)));
// 29
put(2 + 3 * 4 + 5);
// 19
```

* (multiplication)

Syntaxe

```
-- Lingo syntax
expression1 * expression2

// JavaScript syntax
expression1 * expression2
```

Description

Opérateur mathématique ; effectue une multiplication arithmétique de deux expressions numériques. Si les deux expressions sont des nombres entiers, le produit est également un nombre entier. Si l'une ou les deux expressions sont des nombres à virgule flottante, le produit est un nombre à virgule flottante.

Cet opérateur arithmétique a un niveau de priorité de 4.

Exemple

L'instruction suivante multiplie le nombre entier 2 par 3 et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(2 * 3)

// JavaScript syntax
put(2 * 3);
```

Le résultat est 6, qui est un nombre entier.

L'instruction suivante multiplie le nombre à virgule flottante 2,0 par 3,1414 et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(2.0 * 3.1416)

// JavaScript syntax
put(2.0 * 3.1416);
```

Le résultat est 6,2832, qui est un nombre à virgule flottante.

+ (addition)

Syntaxe

```
-- Lingo syntax
expression1 + expression2

// JavaScript syntax
expression1 + expression2
```

Description

Opérateur mathématique ; effectue une somme arithmétique de deux expressions numériques. Si ces deux expressions sont des nombres entiers, la somme est également un nombre entier. Si l'une ou les deux expressions sont des nombres à virgule flottante, la somme est un nombre à virgule flottante.

Cet opérateur arithmétique a un niveau de priorité de 4.

Exemple

L'instruction suivante additionne les nombres entiers 2 et 3, puis affiche le résultat (le nombre entier 5) dans la fenêtre Messages :

```
-- Lingo syntax
put(2 + 3)

// JavaScript syntax
put(2 + 3);
```

L'instruction suivante additionne les nombres à virgule flottante 2,5 et 3,25, puis affiche le résultat (le nombre à virgule flottante 5,7500) dans la fenêtre Messages :

```
-- Lingo syntax
put(2.5 + 3.25)

// JavaScript syntax
put(2.5 + 3.25);
```

+ (addition) (3D)

Syntaxe

```
-- Lingo syntax
vector1 + vector2
vector + scalar

// JavaScript syntax
vector1 + vector2
vector + scalar
```

Description

Opérateur 3D de vecteur ; ajoute les composants de deux vecteurs ou la valeur scalaire à chaque composant du vecteur, et renvoie un nouveau vecteur.

vector1 + *vector2* ajoute les composants de *vector1* au composant correspondant de *vector2* et renvoie un nouveau vecteur.

vecteur + *scalaire* ajoute la valeur scalaire à chaque composant du vecteur et renvoie un nouveau vecteur.

- (soustraction)

Syntaxe

```
-- Lingo syntax
vector1 - vector2
vector - scalar

// JavaScript syntax
vector1 - vector2
vector - scalar
```

Description

Opérateur 3D de vecteur ; soustrait les composants de *vector2* des composants correspondants de *vector1*ou soustrait la valeur scalaire de chacun des composants et renvoie un nouveau vecteur.

vector1 - *vector2* soustrait les valeurs de *vector2* des composants correspondants de *vector1* et renvoie un nouveau vecteur.

vecteur - *scalaire* soustrait la valeur scalaire de chaque composant du vecteur et renvoie un nouveau vecteur.

* (multiplication)

Syntaxe

```
-- Lingo syntax
vector1 * vector2
vector * scalar
transform * vector

// JavaScript syntax
vector1 * vector2
vector * scalar
transform * vector
```

Description

Opérateur 3D de vecteur ; multiplie les composants de *vector1* par les composants correspondants de *vector2* et renvoie le produit ou multiplie chaque composant du vecteur par la valeur scalaire et renvoie un nouveau vecteur.

vector1 * *vector2* renvoie le produit de deux vecteurs, qui n'est pas un nouveau vecteur. Cette opération est équivalente à *vector1*.dotproduct.*vector2*.

vecteur * *scalaire* multiplie chaque composant du vecteur par la valeur scalaire et renvoie un nouveau vecteur.

transformation * *vecteur* multiplie la *transformation* par le *vecteur* et renvoie un nouveau vecteur. Le nouveau vecteur est le résultat de l'application des modifications de position et de rotation définies par *transformation* à *vecteur*. Vous remarquerez que *vecteur* * *transformation* n'est pas pris en charge.

Voir aussi

dotProduct()

/ (division)

Syntaxe

```
-- Lingo syntax
expression1 / expression2

// JavaScript syntax
expression1 / expression2
```

Description

Opérateur mathématique ; effectue une division arithmétique de deux expressions numériques, divisant *expression1* par *expression2*. Si ces deux expressions sont des nombres entiers, le quotient est également un nombre entier. Si l'une ou les deux expressions sont des nombres à virgule flottante, le quotient est un nombre à virgule flottante.

Cet opérateur arithmétique a un niveau de priorité de 4.

Exemple

L'instruction suivante divise le nombre entier 22 par 7, puis affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(22 / 7)

// JavaScript syntax
put(22 / 7);
```

Le résultat est 3. Puisque les deux nombres de la division sont des nombres entiers, Lingo arrondit le résultat au nombre entier le plus proche.

L'instruction suivante divise le nombre à virgule flottante 22,0 par 7,0, puis affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(22.0 / 7.0)

// JavaScript syntax
put(22.0 / 7.0);
```

Le résultat est 3,1429, qui est un nombre à virgule flottante.

/ (division) (3D)

Syntaxe

```
-- Lingo syntax
vector / scalar

// JavaScript syntax
vector / scalar
```

Description

Opérateur 3D de vecteur ; divise chaque composant du vecteur par la valeur scalaire et renvoie un nouveau vecteur.

< (inférieur à)

Syntaxe

```
-- Lingo syntax
expression1 < expression2

// JavaScript syntax
expression1 < expression2
```

Description

Opérateur de comparaison ; compare deux expressions et détermine si *`expression1`* est inférieure à *`expression2`* (`TRUE`) ou si *`expression1`* est supérieure ou égale à *`expression2`* (`FALSE`).

Cet opérateur peut comparer des chaînes, des nombres entiers, des nombres à virgule flottante, des rects et des points. Sachez que les comparaisons effectuées sur les rects ou les points sont traitées comme s'il s'agissait de listes, avec chaque élément de la première liste comparé à l'élément correspondant de la seconde liste.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 1.

<= (inférieur ou égal à)

Syntaxe

```
-- Lingo syntax
expression1 <= expression2

// JavaScript syntax
expression1 <= expression2
```

Description

Opérateur de comparaison ; compare deux expressions et détermine si *expression1* est inférieure ou égale à *expression2* (TRUE) ou si *expression1* est supérieure à *expression2* (FALSE).

Cet opérateur peut comparer des chaînes, des nombres entiers, des nombres à virgule flottante, des rects et des points. Sachez que les comparaisons effectuées sur les rects ou les points sont traitées comme s'il s'agissait de listes, avec chaque élément de la première liste comparé à l'élément correspondant de la seconde liste.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 1.

<> (différent de)

Syntaxe

```
-- Lingo syntax
expression1 <> expression2

// JavaScript syntax
expression1 != expression2
```

Description

Opérateur de comparaison ; compare deux expressions, symboles ou opérateurs et détermine si *expression1* n'est pas égale à *expression2* (TRUE) ou si *expression1* est égale à *expression2* (FALSE).

Cet opérateur peut comparer des chaînes, des nombres entiers, des nombres à virgule flottante, des rects et des points. Sachez que les comparaisons effectuées sur les rects ou les points sont traitées comme s'il s'agissait de listes, avec chaque élément de la première liste comparé à l'élément correspondant de la seconde liste.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 1.

= (signal égal à)

Syntaxe

```
-- Lingo syntax
expression1 = expression2

// JavaScript syntax
expression1 = expression2
```

Description

Opérateur de comparaison ; compare deux expressions, symboles ou objets et détermine si *expression1* est égale à *expression2* (TRUE) ou si *expression1* n'est pas égale à *expression2* (FALSE).

Cet opérateur peut comparer des chaînes, des nombres entiers, des nombres à virgule flottante, des rects, des listes et des points.

Les listes sont comparées sur la base du nombre d'éléments qu'elles contiennent. La liste contenant le plus d'éléments est considérée comme plus longue que la liste contenant moins d'éléments.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 1.

> (supérieur à)

Syntaxe

```
-- Lingo syntax
expression1 > expression2

// JavaScript syntax
expression1 > expression2
```

Description

Opérateur de comparaison ; compare deux expressions et détermine si *expression1* est supérieure à *expression2* (TRUE) ou si *expression1* est inférieure ou égale à *expression2* (FALSE).

Cet opérateur peut comparer des chaînes, des nombres entiers, des nombres à virgule flottante, des rects et des points. Sachez que les comparaisons effectuées sur les rects ou les points sont traitées comme s'il s'agissait de listes, avec chaque élément de la première liste comparé à l'élément correspondant de la seconde liste.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 1.

> = (supérieur ou égal à)

Syntaxe

```
-- Lingo syntax
expression1 >= expression2

// JavaScript syntax
expression1 >= expression2
```

Description

Opérateur de comparaison ; compare deux expressions et détermine si *expression1* est supérieure ou égale à *expression2* (TRUE) ou si *expression1* est inférieure à *expression2* (FALSE).

Cet opérateur peut comparer des chaînes, des nombres entiers, des nombres à virgule flottante, des rects et des points. Sachez que les comparaisons effectuées sur les rects ou les points sont traitées comme s'il s'agissait de listes, avec chaque élément de la première liste comparé à l'élément correspondant de la seconde liste.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 1.

[ ] (crochets d'accès)

Syntaxe

```
-- Lingo syntax
textExpression[chunkNumberBeingAddressed]
textExpression[firstChunk..lastChunk]
```

Description

Opérateur ; permet de désigner une expression de sous-chaîne par un nombre. Cette fonction est utile pour trouver la *n*ième sous-chaîne de l'expression. Cette expression peut être un mot, une ligne, un caractère, un paragraphe ou une autre sous-chaîne d'un acteur texte.

Exemple

L'instruction suivante renvoie le premier mot de la troisième ligne de l'acteur texte Prénoms :

```
-- Lingo syntax
put(member("First Names").text.line[3].word[1])

// JavaScript syntax
put(member("First Names").getPropRef("line", 1).getProp("word", 1));
```

[ ] (liste)

Syntaxe

```
[entry1, entry2, entry3, ...]
```

Description

Opérateur de liste ; spécifie que les entrées contenues dans ces crochets appartiennent à l'un des quatre types de listes suivants :

- Listes linéaires non triées
- Listes linéaires triées
- Listes de propriétés non triées
- Listes de propriétés triées

Chaque entrée d'une liste linéaire est une valeur unique à laquelle aucune propriété n'est associée. Chaque entrée d'une liste de propriétés est constituée d'une propriété et d'une valeur. La propriété précède la valeur et est séparée de celle-ci par le signe deux-points. Vous ne pouvez pas stocker une propriété dans une liste linéaire. Lors de l'utilisation de chaînes comme entrées d'une liste, il convient de placer les chaînes entre guillemets.

Par exemple, [6, 3, 8] est une liste linéaire. Les nombres qu'elle contient ne sont associés à aucune propriété. Par contre, [#engrenages:6, #billes:3, #rampes:8] est une liste de propriétés. Chaque nombre de cette liste est associé à une propriété, une pièce de machine dans cet exemple. Cette liste de propriétés peut servir à contrôler le nombre de pièces de chaque sorte se trouvant sur la scène dans une simulation mécanique. Les propriétés peuvent apparaître plusieurs fois dans une liste de propriétés.

Les listes peuvent être triées en ordre alphanumérique. Une liste linéaire triée est classée selon les valeurs qu'elle contient. Une liste de propriétés triée est classée selon les propriétés qu'elle contient. Le tri d'une liste linéaire ou de propriétés s'opère avec la commande de tri appropriée.

- Dans les listes linéaires, les symboles et les chaînes sont sensibles à l'emploi des minuscules ou des majuscules.
- Dans les listes de propriétés, les symboles ne différencient pas les majuscules des minuscules, contrairement aux chaînes.

Une liste linéaire ou une liste de propriétés peut ne contenir aucune valeur. Une liste vide consiste en deux crochets ([ ]). Pour créer ou supprimer une liste linéaire, affectez-lui la valeur `[ ]`. Pour créer ou supprimer une liste de propriétés, affectez-lui la valeur [:].

Vous pouvez modifier, tester ou lire les éléments contenus dans les listes.

Lingo traite toute instance de liste comme une référence à cette liste. Autrement dit, chaque instance représente les mêmes éléments de données et sa modification modifie l'original. Utilisez la commande `duplicate` pour créer des copies de listes.

Les listes sont automatiquement effacées lorsqu'aucune variable n'y fait plus référence. Si une liste est référencée dans une variable globale, elle existe d'animation en animation.

Vous pouvez initialiser une liste dans le gestionnaire `on prepareMovie` ou rédiger la liste comme acteur champ, l'affecter à une variable, puis la contrôler par la variable.

Tous les claviers ne sont pas forcément équipés de touches de crochets. Le cas échéant, utilisez la fonction `list` pour créer une liste linéaire.

Pour une liste de propriétés, créez les éléments de liste sous forme de chaîne avant de les convertir en une liste utile.

```
myListString = numToChar(91) & ":" & numToChar(93)
put myListString
-- "[:]"
myList = myListString.value
put myList
-- [:]
put myList.listP
-- 1
myList[#name] = "Brynn"
put myList
-- [#name: "Brynn"]
```

Exemple

L'instruction suivante définit une liste en rendant la variable `machine` égale à la liste :

```
-- Lingo syntax
machinery = [#gears:6, #balls:3, #ramps:8]

// JavaScript syntax
var machinery = propList("gears",6, "balls",3, "ramps",8);
```

Le gestionnaire ci-après trie la liste uneListe, puis affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
on sortList aList
    alist.sort()
    put(aList)
end sortList

// JavaScript syntax
function sortList(aList) {
    aList.sort();
    put(aList);
}
```

Si l'animation émet l'instruction `trierListe machine`, où `machine` est la liste de l'exemple précédent, le résultat est [#billes:3, #engrenages:6, #rampes:8].

Les instructions suivantes créent une liste linéaire vide :

```
-- Lingo syntax
x = [ ]
x = list()

// JavaScript syntax
var x = list();
```

Les instructions suivantes créent une liste de propriétés vide :

```
-- Lingo syntax
x = [:]
x = propList()

// JavaScript syntax
var x = propList();
```

Voir aussi

`add`, `addVertex()`, `append`, `count()`, `deleteAt`, `duplicate() (fonction de liste)`, `findPos`, `findPosNear`, `getProp()`, `getAt`, `getLast()`, `getPos()`, `ilk()`, `list()`, `max()`, `min`, `setAt`, `setaProp`, `sort`

@ (chemin d'accès)

Syntaxe

```
@pathReference
```

Description

Opérateur de chemin d'accès ; définit le chemin d'accès du dossier de l'animation en cours et offre l'avantage d'être compris par Windows® comme par Mac®.

Identifiez le dossier de l'animation en cours à l'aide du symbole `@` suivi de l'un des séparateurs de chemin suivants :

- `/` (barre oblique)
- `\` (barre oblique inverse)
- `:` (deux-points)

Lorsqu'une animation est interrogée pour en déterminer l'emplacement, la chaîne renvoyée inclut le symbole @.

Veillez à n'utiliser que le symbole @ lorsque vous passez d'une animation Director à une autre ou que vous modifiez la source d'un acteur média lié. Le symbole @ n'a aucun effet en cas d'utilisation de l'extension Xtra Fileio ou de fonctions non disponibles dans Director.

Vous pouvez vous baser sur ce chemin pour spécifier des dossiers placés en amont ou en aval du dossier de l'animation en cours. N'oubliez pas que la partie @ représente la position de l'animation en cours et pas nécessairement celle de la projection.

- Ajoutez un séparateur de chemin immédiatement à la suite du symbole `@` pour spécifier un dossier en amont d'un niveau dans la hiérarchie.

• Ajoutez des noms de dossier et de fichier (séparés par /, \ ou :) à la suite du nom du dossier actuel pour spécifier des dossiers et des fichiers placés en aval dans les dossiers.

Vous pouvez utiliser des chemins relatifs dans Lingo pour indiquer l'emplacement d'un fichier lié dans un dossier différent de celui contenant l'animation.

Exemple

Les expressions ci-après sont équivalentes et spécifient le sous-dossier grosDossier qui se trouve dans le dossier de l'animation en cours :

```
@/bigFolder
@:bigFolder
@\bigFolder
```

Les expressions ci-après sont équivalentes et spécifient le fichier fichierLié, dans le sous-dossier grosDossier, qui se trouve dans le dossier de l'animation en cours :

```
@:bigFolder:linkedFile
@\bigFolder\linkedFile
@/bigFolder/linkedFile
```

L'expression ci-après spécifie le fichier fichierLié, qui se trouve un niveau en amont du dossier de l'animation en cours :

```
@//linkedFile
```

L'expression ci-après spécifie le fichier fichierLié, qui se trouve deux niveaux en amont du dossier de l'animation en cours :

```
@:::linkedFile
```

Les expressions ci-après sont équivalentes et spécifient le fichier fichierLié, qui se trouve dans le dossier autreDossier. Le dossier autreDossier se trouve un niveau en amont du dossier de l'animation en cours.

```
@::otherFolder:linkedFile
@\\otherFolder\linkedFile
@//otherFolder/linkedFile
```

Voir aussi

searchPathList, fileName (distribution), fileName (acteur), fileName (fenêtre)

and

Syntaxe

```
-- Lingo syntax
logicalExpression1 and logicalExpression2

// JavaScript syntax
logicalExpression1 && logicalExpression2
```

Description

Opérateur logique ; détermine si *expressionLogique1* et *expressionLogique2* ont toutes les deux la valeur TRUE ou si l'une ou les deux expressions ont la valeur FALSE.

Cet opérateur logique a un niveau de priorité de 4.

Exemple

L'instruction suivante détermine si les deux expressions logiques ont la valeur TRUE et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(1 < 2 and 2 < 3)

// JavaScript syntax
put((1 < 2) && (2 < 3));
```

Le résultat est 1, équivalent numérique de TRUE.

La première expression logique de l'instruction suivante a la valeur TRUE ; la deuxième a la valeur FALSE. Puisque les deux expressions logiques n'ont pas toutes deux la valeur TRUE, l'opérateur logique renvoie le résultat 0, qui est l'équivalent numérique de la valeur FALSE.

```
-- Lingo syntax
put(1 < 2 and 2 < 1)
-- 0

// JavaScript syntax
put((1 < 2) && (2 < 1));
// 0
```

Voir aussi

not, or

contains

Syntaxe

```
-- Lingo syntax
stringExpression1 contains stringExpression2

// JavaScript syntax
stringExpression1.indexOf(stringExpression2);
```

Description

Opérateur ; compare deux chaînes et détermine si *expressionChaîne1* contient *expressionChaîne2* (TRUE) ou non (FALSE).

L'opérateur de comparaison contains a un niveau de priorité de 1.

L'opérateur contains est utile pour vérifier si l'utilisateur tape un caractère ou une chaîne de caractères spécifique. Vous pouvez également utiliser l'opérateur contains pour rechercher des chaînes de caractères spécifiques dans un ou plusieurs champs.

Exemple

L'exemple suivant détermine si un caractère transmis est un chiffre :

```
-- Lingo syntax
on isNumber aLetter
    digits = "1234567890"
    if digits contains aLetter then
        return TRUE
    else
        return FALSE
```

```
    end if
end

// JavaScript syntax
function isNumber(aLetter) {
    var digits = "1234567890"
    if (digits.indexOf(aLetter) >= 0) {
        return true;
    } else {
        return false;
    }
}
```

Remarque : *La comparaison de chaînes ne tient pas compte des majuscules ni des accents ; les lettres « a » et « Å » sont ainsi considérées comme identiques.*

Voir aussi

`offset() (fonction de chaîne)`, `starts`

mod

Syntaxe

```
-- Lingo syntax
integerExpression1 mod integerExpression2

// JavaScript syntax
integerExpression1 % integerExpression2
```

Description

Opérateur mathématique ; calcule le reste de la division de deux entiers. Dans l'opération suivante, *`expressionEntière1`* est divisée par *`expressionEntière2`*.

La valeur de l'expression entière obtenue correspond au reste de la division sous forme d'entier. Ce nombre est toujours accompagné du signe (positif ou négatif) de *`expressionEntière1`*.

Cet opérateur arithmétique a un niveau de priorité de 4.

Exemple

L'instruction suivante divise 7 par 4, puis affiche le reste dans la fenêtre Messages :

```
-- Lingo syntax
put(7 mod 4)

// JavaScript syntax
put(7 % 4);
```

Le résultat est 3.

Le gestionnaire suivant affecte à toutes les images-objets impaires l'effet d'encre `copy` (dont le numéro est 0). Le gestionnaire commence par vérifier si l'image-objet de la variable `monImageObjet` est une image-objet impaire en divisant son numéro par 2, puis vérifie si le reste de la division est 1. Si tel est le cas, le gestionnaire définit l'effet d'encre sur `copy`.

```
-- Lingo syntax
on setInk
    repeat with mySprite = 1 to _movie.lastChannel
```

```
        if (mySprite mod 2) = 1 then
            sprite(mySprite).ink = 0
        else
            sprite(mySprite).ink = 8
        end if
    end repeat
end setInk

// JavaScript syntax
function setInk() {
    for (mySprite=1; mySprite<=_movie.lastChannel; mySprite++) {
        if ((mySprite % 2) == 1) {
            sprite(mySprite).ink = 0;
        } else {
            sprite(mySprite).ink = 8;
        }
    }
}
```

Le gestionnaire suivant change régulièrement l'acteur d'une image-objet en choisissant un autre acteur parmi un certain nombre de bitmaps :

```
-- Lingo syntax
on exitFrame
    global gCounter
    -- These are sample values for bitmap cast member numbers
    theBitmaps = [2,3,4,5,6,7]
    -- Specify which sprite channel is affected
    theChannel = 1
    -- This cycles through the list
    gCounter = 1 + (gCounter mod theBitmaps.count)
    sprite(theChannel).memberNum = theBitmaps[gCounter]
    _movie.go(_movie.frame)
end

// JavaScript syntax
function exitFrame() {
    // these are sample values for bitmap cast member numbers
    theBitmaps = new Array(2,3,4,5,6,7);
    // specify which sprite channel is affected
    theChannel = 1;
    // this cycles through the list
    _global.gCounter = 1 + (_global.gCounter % theBitmaps.length);
    sprite(theChannel).memberNum = theBitmaps[_global.gCounter];
    _movie.go(_movie.frame);
}
```

not

Syntaxe

```
-- Lingo syntax
not logicalExpression

// JavaScript syntax
! logicalExpression
```

Description

Opérateur ; effectue la négation logique d'une expression logique. Elle revient à donner à une valeur TRUE la valeur FALSE et à donner à une valeur FALSE la valeur TRUE. Elle est pratique pour vérifier si une certaine condition connue existe ou non.

Cet opérateur logique a un niveau de priorité de 5.

Exemple

L'instruction suivante détermine si 1 n'est pas inférieur à 2 :

```
-- Lingo syntax
put(not (1 < 2))

// JavaScript syntax
put(!(1 < 2));
```

Puisque 1 est inférieur à 2, le résultat est 0, ce qui indique que la valeur de l'expression est FALSE.

L'instruction suivante détermine si 1 n'est pas supérieur à 2 :

```
-- Lingo syntax
put(not (1 > 2))

// JavaScript syntax
put(!(1 > 2));
```

Puisque 1 n'est pas supérieur à 2, le résultat est 1, ce qui indique que la valeur de l'expression est TRUE.

Le gestionnaire suivant attribue à la propriété `the checkMark` de l'élément Gras du menu Style l'inverse de sa valeur en cours :

```
-- Lingo syntax
on resetMenuItem
    menu("Style").menuItem("Bold").checkMark =
    not (menu("Style").menuItem("Bold").checkMark)
end resetMenuItem

// JavaScript syntax
function resetMenuItem() {
    menu("Style").menuItem("Bold").checkMark =
    !(menu("Style").menuItem("Bold").checkMark)
}
```

Voir aussi

and, or

or

Syntaxe

```
-- Lingo syntax
logicalExpression1 or logicalExpression2

// JavaScript syntax
logicalExpression1 || logicalExpression2
```

Description

Opérateur ; effectue une opération OR logique sur deux expressions logiques ou plus pour déterminer si une expression est TRUE.

Cet opérateur logique a un niveau de priorité de 4.

Exemple

L'instruction suivante indique dans la fenêtre Messages si l'une au moins des expressions 1 < 2 et 1 > 2 est TRUE :

```
-- Lingo syntax
put((1 < 2) or (1 > 2))

// JavaScript syntax
put((1 < 2) || (1 > 2));
```

Puisque la première expression est TRUE, le résultat est 1 (équivalent numérique de TRUE).

L'instruction suivante vérifie si le contenu de l'acteur champ appelé Département est Ariège ou Gironde et, le cas échéant, affiche un message d'alerte :

```
-- Lingo syntax
if member("State").text = "AK" or member("State").text = "HI" then
    _player.alert("You're off the map!")
end if

// JavaScript syntax
if (member("State").text == "AK" || member("State").text == "HI") {
    _player.alert("You're off the map!");
}
```

Voir aussi

and, not

starts

Syntaxe

```
-- Lingo syntax
string1 starts string2

// JavaScript syntax
string1.indexOf(string2) == 0;
```

Description

Opérateur ; permet de déterminer si la *chaîne1* commence par la *chaîne2* (TRUE) ou non (FALSE).

La comparaison des chaînes ne tient pas compte des majuscules ou des accents ; *a* et *Å* sont ainsi considérés comme identiques.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 1.

Exemple

L'instruction suivante indique dans la fenêtre Messages si le mot *Macrostuff* commence par la chaîne « Macro » :

```
-- Lingo syntax
put("Macrostuff" starts "Macro")
```

```
// JavaScript syntax
var string1 = "Macrostuff";
put(string1.indexOf("Macro") == 0);
```

Le résultat est 1, équivalent numérique de `TRUE`.

Voir aussi

`contains`

Chapitre 14 : Propriétés

Ce chapitre répertorie dans l'ordre alphabétique toutes les propriétés disponibles dans Director®.

_global

Syntaxe

```
-- Lingo syntax
_global

// JavaScript syntax
_global;
```

Description

Propriété de haut niveau ; fournit une référence à l'objet global qui stocke toutes les variables globales. Lecture seule.

Toutes les variables globales sont accessibles à la fois en syntaxe Lingo et en syntaxe JavaScript.

Exemple

Cette instruction définit la variable `objGlobal` sur la propriété `_global` :

```
-- Lingo syntax
objGlobal = _global

// JavaScript syntax
var objGlobal = _global;
```

L'instruction suivante utilise directement la propriété `_global` pour supprimer toutes les variables globales :

```
-- Lingo syntax
_global.clearGlobals()

// JavaScript syntax
_global.clearGlobals();
```

Voir aussi

Global

_key

Syntaxe

```
-- Lingo syntax
_key

// JavaScript syntax
_key;
```

Description

Propriété de haut niveau ; fournit une référence à l'objet touche utilisé pour surveiller les opérations effectuées au clavier par un utilisateur. Lecture seule.

Exemple

L'instruction suivante définit la variable `objKey` sur la propriété `_key` :

```
-- Lingo syntax
objKey = _key

// JavaScript syntax
var objKey = _key;
```

L'instruction suivante utilise directement la propriété `_key` pour accéder à la valeur de la propriété `key` :

```
-- Lingo syntax
theKey = _key.key

// JavaScript syntax
var theKey = _key.key;
```

Voir aussi

Touche

_mouse

Syntaxe

```
-- Lingo syntax
_mouse

// JavaScript syntax
_mouse;
```

Description

Propriété de haut niveau ; fournit une référence à l'objet souris qui permet d'accéder aux opérations effectuées par un utilisateur avec la souris, tels que les déplacements de la souris et les clics de souris. Lecture seule.

Exemple

L'instruction suivante définit la variable `objMouse` sur la propriété `_mouse` :

```
-- Lingo syntax
objMouse = _mouse

// JavaScript syntax
var objMouse = _mouse;
```

L'instruction suivante utilise directement la propriété `_mouse` pour accéder à la valeur de la propriété `mouseH` :

```
-- Lingo syntax
theMouseH = _mouse.mouseH

// JavaScript syntax
var theMouseH = _mouse.mouseH;
```

Voir aussi

Souris

_movie

Syntaxe

```
-- Lingo syntax
_movie

// JavaScript syntax
_movie;
```

Description

Propriété de haut niveau ; fournit une référence à l'objet animation qui représente l'animation active en cours dans le lecteur Director et permet d'accéder aux propriétés et méthodes disponibles au niveau animation. Lecture seule.

Exemple

L'instruction suivante définit la variable `objAnimation` sur la propriété `_movie` :

```
-- Lingo syntax
objMovie = _movie

// JavaScript syntax
var objMovie = _movie;
```

L'instruction suivante utilise directement la propriété `_movie` pour accéder à la valeur de la propriété `displayTemplate` :

```
-- Lingo syntax
theTemplate = _movie.displayTemplate

// JavaScript syntax
var theTemplate = _movie.displayTemplate;
```

Voir aussi

Animation

_player

Syntaxe

```
-- Lingo syntax
_player

// JavaScript syntax
_player;
```

Description

Propriété de haut niveau ; fournit une référence à l'objet lecteur qui gère et exécute toutes les animations, dont les animations dans une fenêtre. Lecture seule.

Exemple

L'instruction suivante définit la variable `objPlayer` sur la propriété `_player` :

```
-- Lingo syntax
objPlayer = _player
```

```
// JavaScript syntax
var objPlayer = _player;
```

L'instruction suivante utilise directement la propriété `_player` pour accéder à la valeur de la propriété `xtraList` :

```
-- Lingo syntax
theXtras = _player.xtraList

// JavaScript syntax
var theXtras = _player.xtraList;
```

Voir aussi

Lecteur

_sound

Syntaxe

```
-- Lingo syntax
_sound

// JavaScript syntax
_sound;
```

Description

Propriété de haut niveau ; fournit une référence à l'objet son qui contrôle la lecture audio des huit pistes audio disponibles. Lecture seule.

Exemple

L'instruction suivante définit la variable `objSound` sur la propriété `_sound` :

```
-- Lingo syntax
objSound = _sound

// JavaScript syntax
var objSound = _sound;
```

L'instruction suivante utilise directement la propriété `_sound` pour accéder à la propriété `soundLevel` :

```
-- Lingo syntax
theLevel = _sound.soundLevel

// JavaScript syntax
var theLevel = _sound.soundLevel;
```

Voir aussi

Son

_system

Syntaxe

```
-- Lingo syntax
_system
```

```
// JavaScript syntax
_system;
```

Description

Propriété de haut niveau ; fournit une référence à l'objet système qui permet d'accéder aux informations du système et de l'environnement, telles que les méthodes système. Lecture seule.

Exemple

L'instruction suivante définit la variable `objSystème` sur la propriété `_system` :

```
-- Lingo syntax
objSystem = _system

// JavaScript syntax
var objSystem = _system;
```

L'instruction suivante utilise directement la propriété `_system` pour accéder à la propriété `freeBytes` :

```
-- Lingo syntax
theBytes = _system.freeBytes

// JavaScript syntax
var theBytes = _system.freeBytes;
```

Voir aussi

Système

aboutInfo

Syntaxe

```
-- Lingo syntax
_movie.aboutInfo

// JavaScript syntax
_movie.aboutInfo;
```

Description

Propriété d'animation ; permet de saisir une chaîne dans la boîte de dialogue Propriétés de l'animation en phase de création. Lecture seule.

Exemple

Les instructions suivantes affichent des informations sur l'animation dans la fenêtre Messages.

```
-- Lingo syntax
trace(_movie.aboutInfo)

// JavaScript syntax
trace(_movie.aboutInfo);
```

Voir aussi

copyrightInfo (animation), Animation

actionsEnabled

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.actionsEnabled

// JavaScript syntax
memberOrSpriteObjRef.actionsEnabled;
```

Description

Propriété d'acteur et propriété d'image-objet ; contrôle si les actions d'un contenu Adobe® Flash® sont activées (`TRUE`, valeur par défaut) ou désactivées (`FALSE`).

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant accepte une référence d'image-objet en tant que paramètre, puis active ou désactive la propriété `actionsEnabled` de l'image-objet.

```
-- Lingo syntax
on ToggleActions(whichSprite)
     sprite(whichSprite).actionsEnabled = not(sprite(whichSprite).actionsEnabled)
end

// JavaScript syntax
function ToggleActions(whichSprite) {
     switch(sprite(whichSprite).actionsEnabled){
        case 0:
             sprite(whichSprite).actionsEnabled = 1;
            break;
        case 1:
            sprite(whichSprite).actionsEnabled = 0;
            break;
    }
}
```

active3dRenderer

Syntaxe

```
-- Lingo syntax
_movie.active3dRenderer

// JavaScript syntax
_movie.active3dRenderer;
```

Description

Propriété d'animation ; indique le moteur de rendu utilisé par l'animation pour le dessin des images-objets 3D. Cette propriété est l'équivalent de la propriété `getRendererServices().renderer`. Lecture seule.

Les valeurs possibles de la propriété `active3dRenderer` sont `#openGL`, `#directX7_0`, `#directx9`, `#directX5_2` et `#software`. Les valeurs `#openGL`, `#directX7_0`, `#directx9` et `#directX5_2`, qui représentent des pilotes de carte vidéo, permettent d'obtenir de meilleures performances que la valeur `#software` qui correspond à un moteur de rendu logiciel utilisé lorsque aucune des trois premières options n'est disponible.

Utilisez `getRendererServices().renderer` pour définir cette propriété.

Exemple

Les exemples ci-dessous indiquent les deux façons de déterminer le moteur de rendu en cours d'utilisation.

```
-- Lingo syntax
put(_movie.active3dRenderer)
put(getRendererServices().renderer)

// JavaScript syntax
put(_movie.active3dRenderer);
put(getRendererServices().renderer);
```

Voir aussi

Animation, renderer

activeCastLib

Syntaxe

```
-- Lingo syntax
_movie.activeCastLib

// JavaScript syntax
_movie.activeCastLib;
```

Description

Propriété d'animation ; indique la bibliothèque de distribution la plus récemment activée. Lecture seule.

La valeur de la propriété `activeCastLib` correspond au numéro de cette bibliothèque de distribution.

La propriété `activeCastLib` est utile lors de l'utilisation de la propriété `selection` de l'objet distribution. Utilisez-la pour déterminer la bibliothèque de distribution à laquelle la sélection fait référence.

Exemple

Les instructions suivantes affectent les acteurs sélectionnés de la distribution la plus récemment activée à la variable `acteursSélectionnés` :

```
-- Lingo syntax
castLibOfInterest = _movie.activeCastLib
selectedMembers = castLib(castLibOfInterest).selection

// JavaScript syntax
var castLibOfInterest = _movie.activeCastLib;
var selectedMembers = castLib(castLibOfInterest).selection;
```

Voir aussi

Lecteur, selection

activeWindow

Syntaxe

```
-- Lingo syntax
```

```
_player.activeWindow

// JavaScript syntax
_player.activeWindow;
```

Description

Propriété de lecteur ; indique la fenêtre d'animation active. Lecture seule.

Dans le cas de l'animation principale, `activeWindow` correspond à la scène. Dans le cas d'une animation dans une fenêtre (MIAW), `activeWindow` désigne l'animation dans la fenêtre.

Exemple

L'exemple suivant place le mot Active dans la barre de titre de la fenêtre sur laquelle l'utilisateur a cliqué et place le mot Inactive dans la barre de titre de toutes les autres fenêtres ouvertes :

```
-- Lingo syntax
on activateWindow
    clickedWindow = _player.windowList.getPos(_player.activeWindow)
    windowCount = _player.windowList.count
    repeat with x = 1 to windowCount
        if (x = clickedWindow) then
            _player.window[clickedWindow].title = "Active"
        else
            _player.windowList[x].title = "Inactive"
        end if
    end repeat
end activateWindow

// JavaScript syntax
function activateWindow() {
    var clickedWindow = _player.windowList.getPos(_player.activeWindow);
    var windowCount = _player.windowList.count;
    for (var x = 1; x <= windowCount; x++) {
        if (x == clickedWindow) {
            _player.window[clickedWindow].title = "Active"
        }
        else {
            _player.windowList[x].title = "Inactive"
        }
    }
}
```

Voir aussi

Lecteur

actorList

Syntaxe

```
-- Lingo syntax
_movie.actorList

// JavaScript syntax
_movie.actorList;
```

Description

Propriété d'animation ; liste d'objets enfants explicitement ajoutés à cette liste. Lecture/écriture.

Les objets d'`actorList` reçoivent un message `stepFrame` à chaque passage de la tête de lecture sur une image.

Pour ajouter un objet dans `actorList`, utilisez `_movie.actorList.append(`*`newScriptObjRef`*`)`. Le gestionnaire `stepFrame` de l'objet dans son script parent ou ancêtre est alors exécuté automatiquement à chaque avancée d'image.

Pour supprimer les objets de la liste `actorList`, affectez à celle-ci la valeur `[ ]` qui vide la liste.

Director n'efface pas le contenu d'`actorList` lorsqu'il passe à une autre animation, ce qui peut provoquer un comportement imprévisible dans cette dernière. Pour empêcher le transfert des objets enfants de l'animation en cours à la nouvelle animation, insérez l'instruction `actorList = [ ]` dans le gestionnaire `prepareMovie` de la nouvelle animation.

Exemple

L'instruction suivante ajoute un objet enfant créé à partir du script parent Balle roulante. Les trois valeurs sont des paramètres dont le script a besoin.

L'instruction suivante affiche le contenu d'`actorList` dans la fenêtre Messages :

```
-- Lingo syntax
put(_movie.actorList)

// JavaScript syntax
put(_movie.actorList);
```

L'instruction suivante efface les objets d'`actorList`.

```
-- Lingo syntax
_movie.actorList = [] -- using brackets
_movie.actorList = list() -- using list()

// JavaScript syntax
_movie.actorList = list();
```

Voir aussi

Animation, on prepareMovie, on stepFrame

alertHook

Syntaxe

```
-- Lingo syntax
_player.alertHook

// JavaScript syntax
_player.alertHook;
```

Description

Propriété de lecteur ; spécifie un script parent contenant le gestionnaire `alertHook`. Lecture/écriture.

Utilisez `alertHook` pour contrôler l'affichage des messages d'alerte concernant les erreurs de fichier ou de script. Si une erreur survient alors qu'un script parent est affecté à `alertHook`, Director exécute le gestionnaire `alertHook` dans le script parent.

Bien qu'il soit possible de placer des gestionnaires `alertHook` dans des scripts d'animation, il est vivement recommandé de placer un gestionnaire `alertHook` dans un script parent ou de comportement afin d'éviter tout risque d'appel accidentel du gestionnaire depuis différents emplacements, ce qui risquerait de créer une certaine confusion concernant l'emplacement de l'erreur.

Le gestionnaire `alertHook` étant exécuté en cas d'erreur, évitez de l'utiliser pour un script non utilisé pour traiter les erreurs. Par exemple, le gestionnaire `alertHook` ne constitue pas l'emplacement approprié pour une instruction `go()`.

Le gestionnaire `alertHook` reçoit un argument d'instance, deux arguments de chaîne décrivant l'erreur, ainsi qu'un argument facultatif spécifiant un événement supplémentaire invoquant le gestionnaire.

Le quatrième argument peut avoir une de ces quatre valeurs :

- `#alert` : entraîne le déclenchement du gestionnaire par la méthode `alert()`.
- `#movie` : entraîne le déclenchement du gestionnaire par une erreur de fichier introuvable lors de l'exécution d'une commande `go()`.
- `#script` : entraîne le déclenchement du gestionnaire par une erreur de script.
- `#safeplayer` : entraîne le déclenchement du gestionnaire par la vérification de la propriété `safePlayer`.

Selon le script qu'il contient, le gestionnaire `alertHook` peut ignorer l'erreur ou la signaler d'une autre façon.

Exemple

L'instruction suivante spécifie que le script parent Alerte est le script déterminant si une alerte doit être affichée lorsqu'une erreur se produit. Si une erreur se produit, le script affecte les chaînes d'erreur et de message à l'acteur champ Sortie et renvoie la valeur 1.

```
-- Lingo syntax
on prepareMovie
    _player.alertHook = script("Alert")
end

-- "Alert" script
on alertHook me, err, msg
    member("Output").text = err && msg
    return 1
end

// JavaScript syntax
function prepareMovie() {
     _player.alertHook = "alert("Error type", "Error message");"
}

// alert handler
function alert(err, msg) {
    member("Output").text = err + " " + msg; return 1;
}
```

Voir aussi

alertHook, Lecteur, safePlayer

alignment

Syntaxe

```
-- Lingo syntax
memberObjRef.alignment

// JavaScript syntax
memberObjRef.alignment;
```

Description

Propriété d'acteur ; détermine l'alignement utilisé pour afficher des caractères dans l'acteur champ spécifié. Cette propriété s'applique uniquement aux acteurs champ et texte qui contiennent des caractères (un espace suffit).

Pour les acteurs champ, la valeur de la propriété est une chaîne contenant l'un des termes suivants : left, center ou right.

Pour les acteurs texte, la valeur de la propriété est un symbole contenant l'un des éléments suivants : #left, #center, #right ou #full.

Le paramètre *quelActeur* peut indiquer le nom ou le numéro d'un acteur.

Cette propriété peut être testée et définie. Pour les acteurs texte, la propriété peut être définie paragraphe par paragraphe.

Exemple

L'instruction suivante définit la variable `characterAlign` sur la valeur en cours de la propriété `alignment` pour l'acteur champ Pierre :

```
--Lingo syntax
characterAlign = member("Rokujo Speaks").alignment

// JavaScript syntax
var characterAlign = member("Rokujo Speaks").alignment;
```

Voir aussi

`text`, `font`, `lineHeight`, `fontSize`, `fontStyle`, `&`, `+ (opérateur de concaténation)`, `&&`, `+ (opérateur de concaténation)`

allowCustomCaching

Syntaxe

```
-- Lingo syntax
_movie.allowCustomCaching

// JavaScript syntax
_movie.allowCustomCaching;
```

Description

Propriété d'animation ; contiendra les informations concernant un cache privée dans les futures versions de Director. Lecture/écriture.

La valeur par défaut de cette propriété est `TRUE`.

Voir aussi

allowGraphicMenu, allowSaveLocal, allowTransportControl, allowVolumeControl, allowZooming, Animation

allowGraphicMenu

Syntaxe

```
-- Lingo syntax
_movie.allowGraphicMenu

// JavaScript syntax
_movie.allowGraphicMenu;
```

Description

Propriété d'animation ; définit la disponibilité des contrôles graphiques dans le menu contextuel lors de la lecture de l'animation dans un environnement Adobe Shockwave®. Lecture/écriture.

Affectez à cette propriété la valeur FALSE si vous préférez afficher un menu textuel plutôt que le menu contextuel graphique.

La valeur par défaut de cette propriété est TRUE.

Voir aussi

allowCustomCaching, allowSaveLocal, allowTransportControl, allowVolumeControl, allowZooming, Animation

allowSaveLocal

Syntaxe

```
-- Lingo syntax
_movie.allowSaveLocal

// JavaScript syntax
_movie.allowSaveLocal;
```

Description

Propriété d'animation ; définit la disponibilité du contrôle d'enregistrement dans le menu contextuel lors de la lecture de l'animation dans un environnement Shockwave® Player. Lecture/écriture.

Cette propriété permettra d'effectuer des améliorations dans les futures versions de Shockwave Player.

La valeur par défaut de cette propriété est TRUE.

Voir aussi

allowCustomCaching, allowGraphicMenu, allowTransportControl, allowVolumeControl, allowZooming, Animation

allowTransportControl

Syntaxe

```
-- Lingo syntax
_movie.allowTransportControl

// JavaScript syntax
_movie.allowTransportControl;
```

Description

Propriété d'animation ; cette propriété permettra d'effectuer des améliorations dans les futures versions de Shockwave Player. Lecture/écriture.

La valeur par défaut de cette propriété est `TRUE`.

Voir aussi

`allowCustomCaching`, `allowGraphicMenu`, `allowSaveLocal`, `allowVolumeControl`, `allowZooming`, `Animation`

allowVolumeControl

Syntaxe

```
-- Lingo syntax
_movie.allowVolumeControl

// JavaScript syntax
_movie.allowVolumeControl;
```

Description

Propriété d'animation ; définit la disponibilité du contrôle du volume dans le menu contextuel lors de la lecture de l'animation dans un environnement Shockwave Player. Lecture/écriture.

L'un des contrôles du volume est activé ou désactivé selon que vous affectez la valeur `TRUE` ou `FALSE` à cette propriété.

La valeur par défaut de cette propriété est `TRUE`.

Voir aussi

`allowCustomCaching`, `allowGraphicMenu`, `allowSaveLocal`, `allowTransportControl`, `allowZooming`, `Animation`

allowZooming

Syntaxe

```
-- Lingo syntax
_movie.allowZooming

// JavaScript syntax
_movie.allowZooming;
```

Description

Propriété d'animation ; détermine si l'utilisateur peut étirer l'animation ou zoomer sur cette dernière lors de la lecture dans Shockwave Player et ShockMachine. Lecture/écriture.

Affectez la valeur `FALSE` à cette propriété pour empêcher le redimensionnement de l'animation dans un navigateur ou dans ShockMachine.

La valeur par défaut de cette propriété est `TRUE`.

Voir aussi

`allowCustomCaching`, `allowGraphicMenu`, `allowSaveLocal`, `allowTransportControl`, `allowVolumeControl`, `Animation`

alphaThreshold

Syntaxe

```
-- Lingo syntax
memberObjRef.alphaThreshold

// JavaScript syntax
memberObjRef.alphaThreshold;
```

Description

Propriété d'acteur bitmap ; régit l'incidence de la couche alpha du bitmap sur la détection des clics de souris. Cette propriété peut prendre une valeur comprise entre 0 et 255 correspondant exactement aux valeurs alpha de la couche alpha pour une image bitmap 32 bits.

Pour un paramètre `alphaThreshold` donné, Director détecte un clic de souris si la valeur du pixel correspondant à ce point dans la couche alpha est égale ou supérieure au seuil. L'affectation de la valeur 0 à `alphaThreshold` rend tous les pixels opaques à la détection des clics de souris, quel que soit le contenu de la couche alpha.

Voir aussi

`useAlpha`

ambient

Syntaxe

```
member(whichCastmember).shader(whichShader).ambient
member(whichCastmember).model(whichModel).shader.ambient
member(whichCastmember).model(whichModel).shaderList{[index]}.ambient
```

Description

Propriété 3D de matériau `#standard` ; indique la quantité de chaque composante de couleur de la lumière ambiante de l'acteur reflétée par le matériau.

Par exemple, si la couleur de la lumière ambiante est `rgb(255, 255, 255)` et la valeur de la propriété `ambient` du matériau est `rgb(255, 0, 0)`, le matériau reflète toute la composante rouge de la lumière que les couleurs du matériau peuvent refléter. Toutefois, quelles que soient les couleurs du matériau, il ne reflète pas les composantes bleue et verte de la lumière. Dans ce cas, s'il n'y a pas d'autres lumières dans la scène, le bleu et le vert du matériau ne reflètent aucune couleur et apparaissent en noir.

La valeur par défaut de cette propriété est `rgb(63,63,63)`.

Exemple

L'instruction suivante donne à la propriété `ambient` du modèle Fauteuil la valeur `rgb(255, 255, 0)`. Le modèle Fauteuil reflète entièrement les composantes rouge et verte de la lumière ambiante et ignore la composante bleue.

```
-- Lingo syntax
member("Room").model("Chair").shader.ambient = rgb(255, 0, 0)

// JavaScript syntax
member("Room").getPropRef("model",1).shader.ambient = color(255,0,0);
```

Voir aussi

`ambientColor`, `newLight`, `type (lumière)`, `diffuse`, `specular (matériau)`

ambientColor

Syntaxe

```
member(whichCastmember).ambientColor
```

Description

Propriété 3D d'acteur ; indique la couleur rvb de la lumière ambiante par défaut de l'acteur.

La valeur par défaut de cette propriété est `rgb(0, 0, 0)`. Cela n'ajoute aucune lumière à la scène.

Exemple

L'instruction suivante définit la propriété `ambientColor` de l'acteur Pièce sur la valeur rgb(255, 0, 0). La lumière ambiante par défaut de l'acteur est rouge. Cette propriété peut également être définie dans l'Inspecteur des propriétés.

```
-- Lingo syntax
member("Room").ambientColor = rgb(255, 0, 0)

// JavaScript syntax
member("Room").ambientColor = color(255,0,0);
```

Voir aussi

`directionalColor`, `directionalPreset`, `ambient`

ancestor

Syntaxe

```
property {optionalProperties} ancestor
```

Description

Propriété d'objet ; permet aux objets enfants et aux comportements d'utiliser des gestionnaires qui ne sont pas contenus dans le script parent ou le comportement.

La propriété `ancestor` est généralement utilisée avec deux scripts parents ou plus. Vous pouvez utiliser cette propriété pour permettre aux objets enfants et aux comportements de partager certains comportements hérités d'un ancêtre, tout en ayant des comportements différents hérités de leurs parents.

Dans le cas des objets enfants, la propriété `ancestor` est généralement affectée dans le gestionnaire `on new` du script parent. Lorsque vous envoyez un message à un objet enfant ne disposant d'aucun gestionnaire, ce message est envoyé directement au script défini par la propriété `ancestor`.

Si un comportement comporte un ancêtre, cet ancêtre reçoit des événements de souris tels que `mouseDown` et `mouseWithin`.

La propriété `ancestor` vous permet de modifier le comportement et les propriétés d'un grand nombre d'objets à l'aide d'une même commande.

Le script ancêtre peut contenir des variables de propriétés indépendantes accessibles par les objets enfants. Pour faire référence aux variables de propriétés du script ancêtre, respectez la syntaxe suivante :

```
me.variableDePropriété = valeur
```

Par exemple, l'instruction suivante remplace la valeur de la variable de propriété `nombreDePattes` d'un script ancêtre par la valeur 4 :

```
me.legCount = 4
```

Utilisez la syntaxe `the nomDeVariable of nomDeScript` pour accéder aux variables de propriétés non contenues dans l'objet en cours. L'instruction suivante permet à la variable `monNombreDePattes` de l'objet enfant d'accéder à la variable de propriété `nombreDePattes` du script ancêtre :

```
set myLegCount to the legCount of me
```

Exemple

Chacun des scripts suivants est un acteur. Le script ancêtre Animal et les scripts parents Chien et Homme interagissent pour définir des objets.

Le premier script, Chien, définit la variable de propriété `breed` sur `Mutt`, affecte à `the ancestor of Dog` le script Animal et attribue à la variable `legCount` du script ancêtre la valeur 4 :

```
property breed, ancestor

on new me
    set breed = "Mutt"
    set the ancestor of me to new(script "Animal")
    set the legCount of me to 4
    return me
end
```

Le second script, Homme, paramètre la variable de propriété `race` sur Caucasien, affecte à `the ancestor of Mann` le script Animal et attribue à la variable `legCount` du script ancêtre la valeur 2 :

```
property race, ancestor
on new me
    set race to "Caucasian"
    set the ancestor of me to new(script "Animal")
    set the legCount of me to 2
    return me
```

```
end
```

Voir aussi

new(), menu, property

angle (3D)

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).emitter.angle
```

Description

Propriété 3D d'émetteur ; décrit la région dans laquelle les particules d'un système de particules sont émises. Un système de particules est une ressource de modèle de type `#particle`.

La principale direction de l'émission des particules est le vecteur défini par la propriété `direction` de l'émetteur. Toutefois, la direction de l'émission d'une particule donnée dévie de ce vecteur selon un angle aléatoire compris entre 0 et la valeur de la propriété `angle` de l'émetteur.

La plage de cette propriété s'étend de 0,0 à 180,0. La valeur par défaut est 180,0.

Exemple

L'instruction suivante donne à l'angle d'émission de la ressource de modèle mrFont la valeur 1, ce qui entraîne l'émission des particules en une fine ligne.

```
member("fountain").modelResource("mrFount").emitter.angle = 1
```

Voir aussi

emitter, direction

angle (DVD)

Syntaxe

```
-- Lingo syntax
dvdObjRef.angle

// JavaScript syntax
dvdObjRef.angle;
```

Description

Propriété de DVD ; renvoie le nombre d'angles en cours de la caméra. Lecture/écriture.

La valeur renvoyée est un nombre entier.

Exemple

L'instruction suivante renvoie le nombre d'angles en cours de la caméra :

```
-- Lingo syntax
put(member(1).angle)-- 1

// JavaScript syntax
put(member(1).angle);// 1
```

Voir aussi

DVD

angleCount

Syntaxe

```
-- Lingo syntax
dvdObjRef.angleCount

// JavaScript syntax
dvdObjRef.angleCount;
```

Description

Propriété de DVD ; renvoie le nombre d'angles de caméra disponibles dans le titre en cours. Lecture seule.

La valeur renvoyée est un nombre entier compris entre 1 et 9.

Exemple

L'instruction suivante renvoie le nombre d'angles de caméra disponibles :

```
-- Lingo syntax
put(member(12).angleCount)-- 2

// JavaScript syntax
put(member(12).angleCount);// 2
```

Voir aussi

DVD

animationEnabled

Syntaxe

```
member(whichCastmember).animationEnabled
```

Description

Propriété 3D d'acteur ; indique si les mouvements sont exécutés (TRUE) ou ignorés (FALSE). Cette propriété peut également être définie dans l'Inspecteur des propriétés.

La valeur par défaut de cette propriété est TRUE.

Exemple

L'instruction suivante désactive l'animation pour l'acteur Séquence.

```
member("Scene").animationEnabled = FALSE
```

antiAlias

Syntaxe

```
-- Lingo syntax
```

```
memberOrSpriteObjRef.antiAlias

// JavaScript syntax
memberOrSpriteObjRef.antiAlias;
```

Description

Propriété d'acteur ; contrôle si le rendu d'un acteur forme vectorielle ou Flash® repose sur l'anti-aliasing pour assurer une qualité élevée de rendu au détriment de la vitesse de lecture de l'animation. La propriété `antiAlias` présente la valeur `TRUE` par défaut.

Dans le cas des formes vectorielles, la valeur `TRUE` correspond au paramètre de qualité `#high` (qualité supérieure) d'un élément Flash, tandis que la valeur `FALSE` correspond au paramètre `#low` (basse qualité).

La propriété `antiAlias` est également utilisable comme propriété d'image-objet pour les images-objets forme vectorielle uniquement.

Cette propriété peut être testée et définie.

Remarque : *La propriété `antiAlias` n'est pas prise en charge pour les acteurs texte dans Director 11. Pour effectuer l'anti-aliasing des acteurs texte, utilisez la propriété `antiAliasType`.*

Exemple

Le comportement suivant vérifie le codage des couleurs du moniteur sur lequel l'animation est en cours de lecture. Si ce codage est défini sur 8 bits ou moins (256 couleurs), le script affecte la valeur `FALSE` à la propriété `antiAlias` de l'image-objet.

```
--Lingo syntax
property spriteNum

on beginsprite me
    if _system.colorDepth <= 8 then
        sprite(spriteNum).antiAlias = FALSE
    end if
end

// JavaScript syntax
function beginsprite() {
    var cd = _system.colorDepth;
    if (cd <= 8 ) {
        sprite(this.spriteNum).antiAlias = 0;
    }
}
```

Voir aussi

`antiAliasThreshold`, `quality`

antiAliasingEnabled

Syntaxe

```
sprite(whichSprite).antiAliasingEnabled
```

Description

Propriété 3D d'image-objet ; indique si l'univers 3D de l'image-objet *quelleImageObjet* est anti-aliasé. Elle peut être testée et définie. La valeur par défaut de cette propriété est FALSE, ce qui indique que l'anti-aliasing est désactivé. Si la propriété antiAliasingEnabled a pour valeur TRUE et que le moteur de rendu 3D est remplacé par un moteur ne prenant pas en charge l'anti-aliasing, la propriété prend la valeur FALSE. La valeur de cette propriété n'est pas enregistrée avec l'animation.

Les images-objets anti-aliasées imposent une charge supplémentaire au processeur et ont besoin d'une plus grande quantité de mémoire. La désactivation temporaire de l'anti-aliasing peut améliorer les performances des effets d'animation et l'interaction avec l'utilisateur.

Exemple

L'instruction Lingo suivante vérifie si le moteur de rendu 3D en cours de l'image-objet 2 prend en charge l'anti-aliasing à l'aide de la propriété antiAliasingSupported. Si l'anti-aliasing est pris en charge, la seconde instruction active l'anti-aliasing pour l'image-objet avec la propriété antiAliasingEnabled.

```
if sprite(2).antiAliasingSupported = TRUE then
    sprite(2).antiAliasingEnabled = TRUE
end if
```

Voir aussi

antiAliasingSupported, renderer, rendererDeviceList

antiAliasingSupported

Syntaxe

```
sprite(whichSprite).antiAliasingSupported
```

Description

Propriété 3D d'image-objet ; indique si l'anti-aliasing est pris en charge par le moteur de rendu 3D en cours. Cette propriété peut être testée, mais pas définie. Cette propriété renvoie la valeur TRUE ou FALSE.

Exemple

L'instruction Lingo suivante vérifie si le moteur de rendu 3D actuel de l'image-objet 3 supporte l'anti-aliasing. Si l'anti-aliasing est pris en charge, la seconde instruction active l'anti-aliasing pour l'image-objet avec la propriété antiAliasingEnabled.

```
if sprite(3).antiAliasingSupported = TRUE then
    sprite(3).antiAliasingEnabled = TRUE
end if
```

Voir aussi

antiAliasingEnabled, renderer, rendererDeviceList

antiAliasThreshold

Syntaxe

```
-- Lingo syntax
memberObjRef.antiAliasThreshold
```

```
// JavaScript syntax
memberObjRef.antiAliasThreshold;
```

Description

Propriété d'acteur texte ; contrôle la taille de point à laquelle l'anti-aliasing est automatiquement appliqué à un acteur texte. Cette propriété n'a d'effet que lorsque la propriété `antiAlias` de l'acteur texte présente la valeur `TRUE`.

Le paramètre est un nombre entier indiquant la taille (en points) pour laquelle l'anti-aliasing est exécuté.

Cette propriété a une valeur par défaut de 14 points.

Voir aussi

antiAliasType

antiAliasType

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.antiAliasType
```

```
// JavaScript syntax
memberOrSpriteObjRef.antiAliasType;
```

Description

Propriété d'acteur ; contrôle si le rendu d'un acteur texte repose sur l'anti-aliasing pour assurer une qualité élevée de rendu. La propriété `antiAliasType` présente la valeur `#autoAlias` par défaut.

Si vous effectuez une mise à niveau à partir d'une version précédente de Director, l'option auto est mappée sur Niveaux de gris supérieurs à avec la propriété `antiAliasThreshold` définie sur zéro. L'option Supérieur à est mappée sur Niveaux de gris supérieurs à avec le seuil correspondant.

Utilisez les symboles ci-après pour définir l'option antialiasing correspondante :

`Auto - #AutoAlias (utilise les informations du fichier de polices pour l'anti-aliasing).`

`Appliquer globalement les niveaux de gris - #GrayScaleAllAlias (active l'anti-aliasing en niveaux de gris pour tous les acteurs texte).`

`Appliquer globalement la sous-pixellisation - #SubpixelAllAlias (active l'anti-aliasing par sous-pixellisation pour tous les acteurs texte).`

`Niveaux de gris supérieurs à - #GrayscaleLargerThanAlias (active l'anti-aliasing en niveaux de gris pour les tailles de polices supérieures au seuil spécifié).`

`Aucun - #NoneAlias (désactive l'anti-aliasing pour l'acteur en cours).`

Cette propriété peut être testée et définie.

Exemple

Le comportement suivant vérifie le codage des couleurs du moniteur sur lequel l'animation est en cours de lecture. Si ce codage est défini sur 8 bits ou moins (256 couleurs), le script affecte la valeur `FALSE` à la propriété `antiAlias` de l'image-objet.

```
--Lingo syntax
property spriteNum
```

```
on beginsprite me
    if _system.colorDepth <= 8 then
        sprite(spriteNum).antiAliasType = #noneAlias
    end if
end

// JavaScript syntax
function beginsprite() {
    var cd = _system.colorDepth;
    if (cd <= 8 ) {
        sprite(this.spriteNum).antiAliasType = symbol("noneAlias");
    }
}
```

Voir aussi

antiAliasThreshold

appearanceOptions

Syntaxe

```
-- Lingo syntax
windowObjRef.appearanceOptions

// JavaScript syntax
windowObjRef.appearanceOptions;
```

Description

Propriété de fenêtre ; spécifie une liste de propriétés stockant les options d'aspect d'une fenêtre. Lecture/écriture.

Cette liste contient les propriétés ci-après.

Propriété	Description
`#mask`	Spécifie l'acteur 1 bit à utiliser en tant que masque pour la fenêtre.
`#border`	Spécifie le type de bordure de la fenêtre. Cette propriété peut prendre l'une des valeurs suivantes : • `#none` : indique que la fenêtre ne comporte aucune bordure. • `#line` : indique que la fenêtre comporte une bordure noire de 1 pixel. Les propriétés #none et #line ne sont effectives que si la propriété `titlebarOptions.visible` présente la valeur `FALSE`.
`#metal`	(Mac uniquement) Indique si la fenêtre doit présenter un aspect métallisé (`TRUE`). Si cette propriété est définie sur la valeur `FALSE`, la fenêtre a un aspect glacé.
`#dragRegionMask`	Spécifie l'acteur 1 bit à utiliser en tant que masque pour une zone de la fenêtre.
`#shadow`	(Mac uniquement) Indique si la fenêtre doit comporter une ombre. Les fenêtres Mac comportent généralement une ombre.
`#liveresize`	(Mac uniquement) Indique si la fenêtre doit faire l'objet d'un redimensionnement dynamique. Si cette propriété présente la valeur `TRUE`, le redimensionnement dynamique est activé. Si elle présente la valeur `FALSE`, le redimensionnement dynamique est désactivé.

Ces propriétés sont également accessibles par l'intermédiaire de la propriété `displayTemplate` de l'objet animation.

Exemple

L'instruction suivante affiche dans la fenêtre Messages toutes les options d'aspect en cours pour la fenêtre intitulée Tableau de commande :

```
-- Lingo syntax
put(window("Control Panel").appearanceOptions)

// JavaScript syntax
put(window("Control Panel").appearanceOptions);
```

L'instruction suivante définit la propriété `border` de façon à faire apparaître une bordure de 1 pixel autour de la fenêtre Tableau de commande :

```
-- Lingo syntax
window("Control Panel").appearanceOptions.border = #line

// JavaScript syntax
window("Control Panel").appearanceOptions.border = symbol("line");
```

Voir aussi

displayTemplate, titlebarOptions, visible, Fenêtre

applicationName

Syntaxe

```
-- Lingo syntax
_player.applicationName

// JavaScript syntax
_player.applicationName;
```

Description

Propriété de lecteur ; indique le nom de la copie en cours d'exécution de l'application Director en phase de création ou le nom d'un fichier de projection lors de l'exécution. Lecture seule.

La valeur de la propriété est une chaîne.

Shockwave Player ne prend pas en charge cette propriété.

Exemple

L'instruction suivante affiche le nom de l'application Director, Director.exe.

```
-- Lingo syntax
put(_player.applicationName)

// JavaScript syntax
put(_player.applicationName);
```

Voir aussi

applicationPath, Lecteur

applicationPath

Syntaxe

```
-- Lingo syntax
_player.applicationPath

// JavaScript syntax
_player.applicationPath;
```

Description

Propriété de lecteur ; détermine le chemin d'accès ou l'emplacement du dossier contenant la copie en cours d'exécution de l'application Director en phase de création ou le dossier contenant la projection lors de l'exécution. Lecture seule.

La valeur de la propriété est une chaîne.

Si vous utilisez `applicationPath` suivi de `&` et d'un chemin de sous-dossier, indiquez la totalité de l'expression entre parenthèses afin que le script l'analyse comme un tout.

Shockwave Player ne prend pas en charge cette propriété.

Exemple

L'instruction suivante affiche le chemin du dossier contenant l'application Director.

```
-- Lingo syntax
put(_player.applicationPath)

// JavaScript syntax
put(_player.applicationPath);
```

L'instruction suivante ouvre l'animation Champs Elysées dans une fenêtre (sur un ordinateur Windows) :

```
-- Lingo syntax
window(_player.applicationPath & "Film Noir\Sunset Boulevard").open()

// JavaScript syntax
window(_player.applicationPath + "Film Noir\Sunset Boulevard").open();
```

Voir aussi

`applicationName`, `Lecteur`

aspectRatio

Syntaxe

```
-- Lingo syntax
dvdObjRef.aspectRatio

// JavaScript syntax
dvdObjRef.aspectRatio;
```

Description

Propriété de DVD. Renvoie une liste de propriétés spécifiant la largeur et la hauteur de l'acteur DVD. Lecture seule.

La largeur et la hauteur sont renvoyées sous forme de nombres entiers.

Exemple

L'instruction suivante renvoie la valeur aspectRatio de l'acteur 1 :

```
-- Lingo syntax
trace(member(1).aspectRatio) -- [#width: 16, #height:9]

// JavaScript syntax
trace(member(1).aspectRatio); // ["width": 16, "height":9];
```

Voir aussi

DVD

attenuation

Syntaxe

```
member(whichCastMember).light(whichLight).attenuation
```

Description

Propriété 3D de lumière ; indique les facteurs d'atténuation constante, linéaire et quadratique pour les projecteurs et les points lumineux.

La valeur par défaut de cette propriété est vector(1,0, 0,0, 0,0).

Exemple

L'instruction suivante définit la propriété `attenuation` de la lumière Lampe sur la valeur vector(0,5, 0, 0), ce qui l'assombrit légèrement.

```
-- Lingo syntax
member("3d world").light("HouseLight").attenuation = vector(.5, 0, 0)

// JavaScript syntax
member("3d world").getProp("light",1).attenuation = vector(.5, 0, 0);
```

Voir aussi

color (lumière)

attributeName

Syntaxe

```
XMLnode.attributeName[ attributeNumber ]
```

Description

Propriété XML ; renvoie le nom du nœud enfant spécifié d'un document XML analysé.

Exemple

Avec le code XML suivant :

```
<?xml version="1.0"?>
    <e1>
        <tagName attr1="val1" attr2="val2"/>
        <e2> element 2</e2>
```

```
        <e3>element 3</e3>
        here is some text
    </e1>
```

L'instruction Lingo suivante renvoie le nom du premier attribut de la balise appelée `tagName` :

```
put gParserObject.child[1].child[1].attributeName[1]
-- "attr1"
```

Voir aussi

attributeValue

attributeValue

Syntaxe

```
XMLnode.attributeValue[ attributeNameOrNumber ]
```

Description

Propriété XML ; renvoie la valeur du nœud enfant spécifié d'un document XML analysé.

Exemple

Avec le code XML suivant :

```
<?xml version="1.0"?>
    <e1>
        <tagName attr1="val1" attr2="val2">
        <e2> element 2</e2>
        <e3>element 3</e3>
        here is some text
    </e1>
```

L'instruction Lingo suivante renvoie la valeur du premier attribut de la balise appelée `tagName` :

```
put gParserObject.child[1].child[1].attributeValue[1]
-- "val1"
```

Voir aussi

attributeName

audio (DVD)

Syntaxe

```
-- Lingo syntax
dvdObjRef.audio

// JavaScript syntax
dvdObjRef.audio;
```

Description

Propriété de DVD. Détermine si la fonction audio est activée (`TRUE`, valeur par défaut) ou non (`FALSE`). Lecture/écriture.

Exemple

L'instruction suivante désactive la fonction audio :

```
-- Lingo syntax
member(14).audio = 0

// JavaScript syntax
member(14).audio = 0;
```

Voir aussi

DVD

audio (RealMedia)

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.audio

// JavaScript syntax
memberOrSpriteObjRef.audio;
```

Description

Propriété d'acteur ou d'image-objet RealMedia ; permet de lire (`TRUE`) ou de mettre en sourdine (`FALSE`) la fonction audio du flux RealMedia. Le paramétrage par défaut de cette propriété est `TRUE` (1). Les valeurs entières autres que 1 ou 0 sont traitées comme `TRUE` (1). La définition de cette propriété n'a aucun effet si la méthode `realPlayerNativeAudio()` présente la valeur `TRUE`.

Si la propriété `audio` présente la valeur `FALSE` lorsque la lecture d'un acteur RealMedia démarre, la piste audio reste affectée, ce qui vous permet d'activer ou de désactiver le son lors de la lecture.

Une certaine latence peut se produire lors de la définition de cette propriété, ce qui signifie qu'un léger délai peut être observé avant l'activation ou la désactivation du son.

Exemple

Les exemples suivants définissent la propriété audio de l'image-objet 2 et de l'acteur Real sur la valeur `TRUE`, ce qui signifie que la portion audio du flux RealMedia est lue.

```
-- Lingo syntax
put(sprite(2).audio) -- 1
put(member("Real").audio) -- 1

// JavaScript syntax
put(sprite(2).audio); // 1
put(member("Real").audio); // 1
```

Le code Lingo suivant définit la propriété `audio` de l'image-objet 2 et de l'acteur Real sur la valeur `FALSE`, ce qui signifie que la portion audio du flux RealMedia n'est pas lue en même temps que l'animation.

```
-- Lingo syntax
sprite(2).audio = FALSE
member("Real").audio = FALSE

// JavaScript syntax
sprite(2).audio = 0;
member("Real").audio = 0;
```

Voir aussi

soundChannel (RealMedia), video (RealMedia, Windows Media), sound (lecteur)

audio (Windows Media)

Syntaxe

```
-- Lingo syntax
windowsMediaObjRef.audio

// JavaScript syntax
windowsMediaObjRef.audio;
```

Description

Propriété Windows Media. Détermine si la fonction audio est activée (TRUE, valeur par défaut) ou non (FALSE) pendant la lecture. Lecture/écriture.

Exemple

L'instruction suivante indique dans la fenêtre Messages si la fonction audio est activée pour l'acteur 5 :

```
-- Lingo syntax
trace(member(5).audio)

// JavaScript syntax
trace(member(5).audio);
```

Voir aussi

Windows Media

audioChannelCount

Syntaxe

```
-- Lingo syntax
dvdObjRef.audioChannelCount

// JavaScript syntax
dvdObjRef.audioChannelCount;
```

Description

Propriété de DVD ; renvoie le nombre de pistes audio. Lecture seule.

Exemple

L'instruction suivante renvoie le nombre de pistes audio :

```
-- Lingo syntax
member(1).audioChannelCount

// JavaScript syntax
member(1).audioChannelCount;
```

Voir aussi

DVD

audioExtension

Syntaxe

```
-- Lingo syntax
dvdObjRef.audioExtension

// JavaScript syntax
dvdObjRef.audioExtension;
```

Description

Propriété de DVD. Renvoie un symbole indiquant les éventuelles extensions audio d'un flux audio. Lecture seule.

Les valeurs pouvant être renvoyées sont les suivantes :

Symbole	Description
#caption	Le flux audio contient des sous-titres.
#lowvision	Le flux audio contient du contenu pour les personnes malvoyantes.
#directorcomments1	Le flux audio contient l'indication « director comments 1 ».
#directorcomments2	Le flux audio contient l'indication « director comments 2 ».
#none	L'extension audio pour ce flux audio n'est pas spécifiée par le DVD ou n'a pu être déterminée.

Voir aussi

DVD

audioFormat

Syntaxe

```
-- Lingo syntax
dvdObjRef.audioFormat

// JavaScript syntax
dvdObjRef.audioFormat;
```

Description

Propriété de DVD. Renvoie un symbole indiquant le format (mode de codage) d'un flux audio. Lecture seule.

Les valeurs pouvant être renvoyées sont les suivantes :

Symbole	Description
#AC3	Le format audio est Dolby AC-3.
#MPEG1	Le format audio est MPEG-1.
#MPEG1DRC	Le format audio est MPEG-1 avec contrôle de plage dynamique.
#MPEG2	Le format audio est MPEG-2.
#MPEG2DRC	Le format audio est MPEG-2 avec contrôle de plage dynamique.

Symbole	Description
#LPCM	Le format audio est LPCM (Linear Pulse Code Modulated, modulation par impulsion et codage linéaire).
#DTS	Le format audio est DTS (Digital Theater Systems).
#SDDS	Le format audio est SDDS (Sony Dynamic Digital Sound).

Voir aussi

DVD

audioSampleRate

Syntaxe

```
-- Lingo syntax
dvdObjRef.audioSampleRate

// JavaScript syntax
dvdObjRef.audioSampleRate;
```

Description

Propriété de DVD ; renvoie la fréquence d'un flux audio en hertz. Lecture seule.

Voir aussi

DVD

audioStream

Syntaxe

```
-- Lingo syntax
dvdObjRef.audioStream

// JavaScript syntax
dvdObjRef.audioStream;
```

Description

Propriété de DVD. Renvoie le flux audio actif. Lecture/écriture.

Les valeurs possibles sont comprises entre 1 et 8.

Voir aussi

DVD

audioStreamCount

Syntaxe

```
-- Lingo syntax
dvdObjRef.audioStreamCount
```

```
// JavaScript syntax
dvdObjRef.audioStreamCount;
```

Description

Propriété de DVD ; renvoie le nombre de flux audio disponibles dans le titre en cours. Lecture seule.

Le nombre de flux audio disponibles peut être compris entre 1 et 8.

Voir aussi

DVD

auto

Syntaxe

```
member(whichCastmember).model(whichModel).lod.auto
```

Description

Propriété 3D de modificateur `lod` ; permet au modificateur de gérer la réduction des détails dans le modèle à mesure que la distance entre le modèle et la caméra change.

Le paramétrage de la propriété `bias` du modificateur détermine la mesure dans laquelle le modificateur supprime les détails du modèle lorsque la propriété `auto` présente la valeur `TRUE`.

Le modificateur met à jour sa propriété `level` en même temps qu'il ajuste le niveau de détail du modèle. Le paramétrage de la propriété `level` n'a aucun effet, sauf lorsque la propriété `auto` présente la valeur `FALSE`.

Le modificateur `#lod` ne peut être ajouté qu'aux modèles créés dans un programme de modélisation 3D autre que Director. La valeur de la propriété `type` des ressources de modèle utilisées par ces modèles est `#fromFile`. Le modificateur ne peut pas être ajouté aux primitives créées dans Director.

Exemple

L'instruction suivante définit la propriété `auto` du modificateur lod du modèle vaisseauSpatial sur la valeur `TRUE`. Le modificateur définira automatiquement le niveau de détail du modèle.

```
-- Lingo syntax
member("3D World").model("Spaceship").lod.auto = TRUE

// Java Script
member("3D World").getPropRef("model", 1).getPropRef("lod", 1).auto = true;
```

Voir aussi

lod (modificateur), bias, level

autoblend

Syntaxe

```
member(whichCastmember).model(whichModel).keyframePlayer.autoblend
member(whichCastmember).model(whichModel).bonesPlayer.autoblend
```

Description

Propriété 3D de modificateur `keyframePlayer` et `bonesPlayer` ; indique si le modificateur crée une transition linéaire entre le mouvement en cours de lecture et le mouvement précédent `(TRUE)` ou non `(FALSE)`. Si `autoBlend` présente la valeur `TRUE`, la durée de la transition est définie par la propriété `blendTime` du modificateur. Si `autoBlend` présente la valeur `FALSE`, la transition est contrôlée par la propriété `blendFactor` du modificateur et `blendTime` est ignoré.

Le fondu des mouvements est totalement désactivé lorsque `blendTime` a pour valeur 0 et qu'`autoBlend` présente la valeur `TRUE`.

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante désactive autoblend pour le modèle Martien3. Le paramètre `blendFactor` du modèle est utilisé pour fusionner plusieurs mouvements successifs de la liste de lecture.

```
-- Lingo syntax
member("newaliens").model("Alien3").addModifier(#keyframeplayer)
member("newaliens").model("Alien3").keyframePlayer.autoblend = FALSE

// JavaScript syntax
member("newaliens").getPropRef("model",1).addModifier(symbol("keyframeplayer"));
member("newaliens").getPropRef("model",1).getPropRef("keyframeplayer",1).autoBlend =
false;
```

Voir aussi

`blendFactor`, `blendTime`

autoCameraPosition

Syntaxe

```
member(whichTextCastmember).autoCameraPosition
```

Description

Propriété 3D de caméra ; indique si la caméra de l'acteur texte 3D est automatiquement positionnée pour afficher tout le texte (`TRUE`) ou non (`FALSE`). Cela est utile lors de la modification du texte, de la police ou de sa taille, et d'autres propriétés de l'acteur.

Cette propriété n'est pas valide avec d'autres types d'acteurs 3D.

Exemple

L'instruction suivante définit la propriété `autoCameraPosition` de l'acteur Titres sur la valeur `FALSE`. Lorsque l'acteur est affiché en mode 3D, la caméra n'est pas positionnée automatiquement.

```
-- Lingo syntax
member("Headline").autoCameraPosition = FALSE

// JavaScript syntax
member("Headline").autoCameraPosition = false;
```

Voir aussi

`displayMode`

autoMask

Syntaxe

```
member(whichCursorCastMember).autoMask
the autoMask of member whichCastMember
```

Description

Propriété d'acteur ; spécifie si les pixels blancs de l'acteur curseur couleur animé *quelActeurCurseur* sont transparents de façon à permettre l'affichage de l'arrière-plan (TRUE, valeur par défaut) ou opaques (FALSE).

Exemple

Dans le script suivant, lorsque le curseur animé personnalisé stocké dans l'acteur 5 entre dans l'image-objet, la fonction de masque automatique est activée de sorte que l'arrière-plan de l'image-objet s'affiche au travers des pixels blancs. La fonction de masque automatique est désactivée dès que le curseur quitte l'image-objet.

```
-- Lingo syntax
on mouseEnter
    member 5.autoMask = TRUE
end

on mouseLeave
    member 5.autoMask = FALSE
end
```

En syntaxe Lingo traditionnelle, le script s'écrit sous la forme suivante :

```
on mouseEnter
    set the autoMask of member 5 = TRUE
end

on mouseLeave
    set the autoMask of member 5 = FALSE
end
```

autoTab

Syntaxe

```
-- Lingo syntax
memberObjRef.autoTab

// JavaScript syntax
memberObjRef.autoTab;
```

Description

Propriété d'acteur ; détermine l'effet d'une pression de la touche de tabulation sur le champ modifiable ou l'acteur texte spécifié par *quelActeur*. Cette propriété peut être rendue active (TRUE) ou inactive (FALSE). L'ordre de tabulation dépend du numéro des images-objets et non de leur position sur la scène.

Exemple

L'instruction suivante entraîne l'acteur Commentaires à avancer automatiquement le point d'insertion au champ modifiable ou à l'image-objet texte suivant(e) lorsque l'utilisateur appuie sur Tabulation :

```
--Lingo syntax
member ("Comments").autotab = TRUE
```

```
// JavaScript syntax
member ("Comments").autotab = true;
```

axisAngle

Syntaxe

```
member(whichCastmember).model(whichModel).transform.axisAngle
member(whichCastmember).camera(whichCamera).transform.axisAngle
member(whichCastmember).light(whichLight).transform.axisAngle
member(whichCastmember).group(whichGroup).transform.axisAngle
transformReference.axisAngle
```

Description

Propriété 3D de transformation ; décrit la rotation de la transformation sous la forme d'une paire axe/angle.

La propriété `axisAngle` est une liste linéaire contenant un vecteur (l'axe) et une valeur à virgule flottante (l'angle). Le vecteur est l'axe autour duquel la transformation est pivotée. La valeur à virgule flottante est l'importance, en degrés, de la rotation.

La valeur par défaut de cette propriété est `[vector( 1.0000, 0.0000, 0.0000 ), 0.0000]`.

Exemple

L'instruction suivante indique la rotation du modèle boîteAuxLettres. Le modèle pivote de 145,5 degrés dans le sens opposé aux aiguilles d'une montre autour de l'axe des y.

```
-- Lingo syntax
put member("Yard").model("Mailbox").transform.axisAngle
-- [vector( 0.0000, 1.0000, 0.0000 ), -145.5000]

// JavaScript syntax
put(member("Yard").getProp("model",1).transform.axisAngle);
// <[vector( 0.0000, 1.0000, 0.0000 ), -145.5000]>
```

Voir aussi

rotation (transformation)

back

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).back
```

Description

Propriété 3D de ressource de modèle `#box` ; indique si le côté de la boîte coupé par son axe des z positif est fermé (`TRUE`) ou ouvert (`FALSE`).

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante définit la propriété `back` de la ressource de modèle Caisse sur la valeur `FALSE`, ce qui signifie que l'arrière de la caisse est ouvert.

```
-- Lingo syntax
nmr = member("3D World").newModelResource("Crate", symbol("box"))
member("3D World").modelResource("Crate").back = FALSE

// JavaScript syntax
nmr = member("3D World").newModelResource("Crate", symbol("box"));
member("3D World").getProp("modelresource",10).back = false;
```

Voir aussi

bottom (3D), front, top (3D), left (3D), right (3D)

backColor

Syntaxe

```
-- Lingo syntax
spriteObjRef.backColor

// JavaScript syntax
spriteObjRef.backColor;
```

Description

Propriété d'image-objet ; définit la couleur d'arrière-plan d'une image-objet spécifiée en fonction de la valeur de couleur affectée. Lecture/écriture.

La définition de la propriété `backColor` d'une image-objet équivaut à choisir la couleur d'arrière-plan dans la Palette des outils lorsque l'image-objet est sélectionnée sur la scène. Pour que la valeur définie par un script dure au-delà de l'image-objet en cours, l'image-objet doit être contrôlée par un script. La couleur d'arrière-plan s'applique à tous les acteurs bitmap, ainsi qu'aux acteurs champ, bouton, case à cocher et bouton radio.

La valeur de la propriété `backColor` peut être comprise entre 0 et 255 pour un codage couleur sur 8 bits et entre 0 et 15 pour un codage sur 4 bits. Ces valeurs correspondent au numéro d'index de la couleur d'arrière-plan dans la palette en cours. Ce numéro apparaît dans l'angle inférieur gauche de la palette de couleurs lorsque vous cliquez sur la couleur.

Si cette propriété est définie sur des acteurs bitmap supérieurs à 1 bit, la couleur définie par `backColor` risque de ne pas apparaître si l'arrière-plan du bitmap n'est pas visible.

Si l'opacité d'une image-objet est inférieure à 100 mais supérieure à 0, la couleur définie par `backColor` se mélange avec les couleurs transparentes.

Remarque : *il est recommandé d'utiliser la nouvelle propriété `bgColor` plutôt que la propriété `backColor`.*

Exemple

L'instruction suivante définit la variable `ancienneCouleur` sur la couleur d'arrière-plan de l'image-objet 5 :

```
-- Lingo syntax
oldColor = sprite(5).backColor

// JavaScript syntax
var oldColor = sprite(5).backColor;
```

L'instruction suivante modifie de façon aléatoire la couleur d'arrière-plan d'une image-objet choisie au hasard entre les images-objets 11 et 13 en lui affectant la couleur 36 :

```
-- Lingo syntax
```

```
sprite(10 + random(3)).backColor = 36

// JavaScript syntax
sprite(10 + random(3)).backColor = 36;
```

Voir aussi

Image-objet

backdrop

Syntaxe

```
sprite(whichSprite).camera{(index)}.backdrop[index].loc
member(whichCastmember).camera(whichCamera).backdrop[index].loc
sprite(whichSprite).camera{(index)}.backdrop[index].source
member(whichCastmember).camera(whichCamera).backdrop[index].source
sprite(whichSprite).camera{(index)}.backdrop[index].scale
member(whichCastmember).camera(whichCamera).backdrop[index].scale
sprite(whichSprite).camera{(index)}.backdrop[index].rotation
member(whichCastmember).camera(whichCamera).backdrop[index].rotation
sprite(whichSprite).camera{(index)}.backdrop[index].regPoint
member(whichCastmember).camera(whichCamera).backdrop[index].regPoint
sprite(whichSprite).camera{(index)}.backdrop[index].blend
member(whichCastmember).camera(whichCamera).backdrop[index].blend
sprite(whichSprite).camera{(index)}.backdrop.count
member(whichCastmember).camera(whichCamera).backdrop.count
```

Description

Propriété 3D de caméra ; image 2D rendue sur le plan de projection de la caméra. Tous les modèles présents dans la vue de la caméra apparaissent devant le fond.

Les fonds ont les propriétés suivantes :

Remarque : *ces propriétés peuvent aussi être utilisées pour obtenir, définir et manipuler les recouvrements. Pour plus d'informations, reportez-vous aux entrées des différentes propriétés.*

loc (fond et recouvrement) indique l'emplacement 2D du fond, mesuré à partir du coin supérieur gauche de l'image-objet.

source indique la texture utilisée pour le fond.

scale (fond et recouvrement) est le nombre par lequel la hauteur et la largeur de la texture sont multipliées pour déterminer les dimensions du fond.

rotation (fond et recouvrement) est le nombre qui détermine la rotation du fond par rapport à son point d'alignement.

regPoint (3D) indique le point d'alignement du fond.

blend (3D) indique l'opacité du fond.

count (3D) indique le nombre d'éléments dans la liste de fonds de la caméra.

Utilisez les commandes suivantes pour créer et retirer des fonds :

addBackdrop crée un fond à partir d'une texture et l'ajoute à la fin de la liste de fonds de la caméra.

insertBackdrop crée un fond à partir d'une texture et l'ajoute à la liste de fonds de la caméra au niveau d'une position d'index spécifique.

removeBackdrop supprime le fond.

Voir aussi

overlay

backgroundColor

Syntaxe

```
-- Lingo syntax
memberObjRef.backgroundColor

// JavaScript syntax
memberObjRef.backgroundColor;
```

Description

Propriété d'acteur forme vectorielle ; affecte à la couleur d'arrière-plan de l'acteur ou de l'image-objet spécifié la valeur de couleur RVB indiquée.

Cette propriété peut être testée et définie.

Exemple

```
-- Lingo syntax
member("Archie").backgroundColor= color(255,255,255)

// JavaScript syntax
member("Archie").backgroundColor= color(255,255,255);
```

Voir aussi

bgColor (fenêtre)

beepOn

Syntaxe

```
-- Lingo syntax
_movie.beepOn

// JavaScript syntax
_movie.beepOn;
```

Description

Propriété d'animation ; détermine si l'ordinateur émet automatiquement un signal sonore lorsque l'utilisateur clique en dehors de toute image-objet active (`TRUE`) ou non (`FALSE`, valeur par défaut). Lecture/écriture.

Les scripts définissant la propriété `beepOn` doivent être placés dans les scripts d'image ou d'animation.

Exemple

L'instruction suivante définit la propriété `beepOn` sur la valeur `TRUE` :

```
-- Lingo syntax
_movie.beepOn = TRUE

// JavaScript syntax
```

```
_movie.beepOn = true;
```

L'instruction suivante inverse la valeur de la propriété beepOn :

```
-- Lingo syntax
_movie.beepOn = not(_movie.beepOn)

// JavaScript syntax
_movie.beepOn = !(_movie.beepOn);
```

Voir aussi

Animation

bevelDepth

Syntaxe

```
member(whichTextCastmember).bevelDepth
member(which3DCastmember).modelResource(whichModelResource).bevelDepth
```

Description

Propriété 3D de texte ; indique la taille du biseau du texte 3D.

Dans le cas d'un acteur texte, cette propriété n'a aucun effet, sauf si la propriété `displayMode` de l'acteur présente la valeur `#mode3D` et que sa propriété `bevelType` a pour valeur `#miter` ou `#round`.

Dans le cas d'un texte extrudé d'un acteur 3D, cette propriété n'a aucun effet, sauf si la propriété `bevelType` de la ressource de modèle a pour valeur `#miter` ou `#round`.

La plage de cette propriété s'étend de 0,0 à 10,0, la valeur par défaut étant 10,0.

Exemple

Dans l'exemple suivant, l'acteur Logo est un acteur texte. L'instruction suivante définit la propriété bevelDepth de Logo sur la valeur 5,5. Lorsque l'acteur Logo est affiché en mode 3D et que sa propriété `bevelType` présente la valeur #miter ou #round, les bords de ses lettres sont fortement biseautés.

```
-- Lingo syntax
member("Logo").bevelDepth = 5.5

// JavaScript syntax
member("Logo").bevelDepth = 5.5;
```

Dans l'exemple suivant, la ressource du modèle Slogan correspond à du texte extrudé. L'instruction suivante définit la propriété bevelDepth de la ressource du modèle Slogan sur la valeur 5. Si la propriété `bevelType` de Slogan présente la valeur `#miter` ou `#round`, les bords de ses lettres sont fortement biseautés.

```
-- Lingo syntax
member("scene").model("Slogan").resource.bevelDepth = 5
```

Voir aussi

bevelType, extrude3D, displayMode

bevelType

Syntaxe

```
member(whichTextCastmember).bevelType
member(which3DCastmember).modelResource(whichModelResource).bevelType
```

Description

Propriété 3D de texte ; indique le style de biseau appliqué au texte 3D.

Pour les acteurs texte, il s'agit d'une propriété d'acteur. Pour le texte extrudé d'un acteur 3D, il s'agit d'une propriété de ressource de modèle.

La propriété `bevelType` peut prendre les valeurs suivantes :

- `#none`
- `#miter` (valeur par défaut)
- `#round`

Exemple

Dans l'exemple suivant, l'acteur Logo est un acteur texte. L'instruction suivante définit la propriété `bevelType` de Logo sur la valeur `#round`.

```
member("logo").beveltype = #round
```

Dans l'exemple suivant, la ressource du modèle Slogan correspond à du texte extrudé. L'instruction suivante définit la propriété `bevelType` de la ressource du modèle Slogan sur la valeur `#miter`.

```
member("scene").model("Slogan").resource.bevelType = #miter
```

Voir aussi

`bevelDepth`, `extrude3D`, `displayMode`

bgColor (fenêtre)

Syntaxe

```
-- Lingo syntax
windowObjRef.bgColor

// JavaScript syntax
windowObjRef.bgColor;
```

Description

Propriété de fenêtre ; détermine la couleur d'arrière-plan d'une fenêtre. Lecture/écriture.

La définition de la propriété `bgColor` équivaut à définir la couleur dans la boîte de dialogue Propriétés de l'animation.

Exemple

L'exemple suivant définit la couleur de la fenêtre Animaux sur une valeur RVB.

```
-- Lingo syntax
window("Animals").bgColor = color(255, 153, 0)
```

```
// JavaScript syntax
window("Animals").bgColor = color(255, 153, 0);
```

Voir aussi

Fenêtre

bgColor (image-objet, acteur 3D)

Syntaxe

```
sprite(whichSpriteNumber).bgColor
the bgColor of sprite whichSpriteNumber
the bgColor of the stage
(the stage).bgColor
member(which3dMember).bgcolor
```

Description

Propriété d'image-objet, propriété système et propriété 3D d'acteur ; détermine la couleur d'arrière-plan de l'image-objet spécifiée par *quelleImageObjet*, la couleur de la scène ou la couleur d'arrière-plan de l'acteur 3D. La définition de la propriété d'image-objet `bgColor` équivaut à choisir la couleur d'arrière-plan dans la fenêtre Outils lorsque l'image-objet est sélectionnée sur la scène. La définition de la propriété `bgColor` pour la scène équivaut à définir la couleur dans la boîte de dialogue Propriétés de l'animation.

Cette propriété d'image-objet a une fonction équivalente à celle de `backColor`, mais la valeur de couleur renvoyée est un objet couleur du type défini pour cette image-objet.

Cette propriété peut être testée et définie.

Exemple

L'exemple suivant définit la couleur de la scène sur une valeur RVB.

Syntaxe à point :

```
(the stage).bgColor = rgb(255, 153, 0)
```

Syntaxe verbose :

```
set the bgColor of the stage = rgb(255, 153, 0)
```

Voir aussi

color(), backColor, backgroundColor

bias

Syntaxe

```
member(whichCastmember).model(whichModel).lod.bias
```

Description

Propriété 3D de modificateur `lod` ; indique la mesure dans laquelle le modificateur supprime les détails du modèle lorsque sa propriété `auto` présente la valeur `TRUE`. Cette propriété n'a aucun effet lorsque la propriété `auto` du modificateur présente la valeur `FALSE`.

La plage de cette propriété s'étend de 0,0 (qui supprime tous les polygones) à +100,0 (qui ne supprime aucun polygone). Le paramètre par défaut est 100,0.

Le modificateur `#lod` ne peut être ajouté qu'aux modèles créés dans un programme de modélisation 3D autre que Director. La valeur de la propriété `type` des ressources de modèle utilisées par ces modèles est `#fromFile`. Le modificateur ne peut pas être ajouté aux primitives créées dans Director.

Exemple

L'instruction suivante définit la propriété `bias` du modificateur lod du modèle vaisseauSpatial sur la valeur 10. Si la propriété `auto` du modificateur lod présente la valeur `TRUE`, ce modificateur réduira considérablement le niveau de détail de vaisseauSpatial à mesure que celui-ci s'éloigne de la caméra.

```
-- Lingo syntax
member("3D World").model("Spaceship").lod.bias = 10

// Java Script
member("3D World").getPropRef("model", 1).getPropRef("lod", 1).bias = 10;
```

Voir aussi

lod (modificateur), auto, level

bitmapSizes

Syntaxe

```
-- Lingo syntax
memberObjRef.bitmapSizes

// JavaScript syntax
memberObjRef.bitmapSizes;
```

Description

Propriété d'acteur police ; renvoie une liste des tailles de bitmap, en points, incluses lorsque l'acteur police a été créé.

Exemple

L'instruction suivante affiche les tailles de bitmap, en points, incluses lorsque l'acteur 11 a été créé :

```
-- Lingo syntax
put(member(11).bitmapSizes)

// JavaScript syntax
put(member(11).bitmapSizes);
```

Voir aussi

recordFont, characterSet, originalFont

bitRate

Syntaxe

```
-- Lingo syntax
memberObjRef.bitRate
```

```
// JavaScript syntax
memberObjRef.bitRate;
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; renvoie le débit de téléchargement (en Kbps) de l'acteur SWA spécifié préchargé depuis le serveur.

La propriété d'acteur `bitRate` renvoie 0 tant que la lecture en flux continu n'a pas commencé.

Exemple

Le comportement suivant affiche le débit de téléchargement d'un acteur SWA lors de la première apparition de l'image-objet.

```
-- Lingo syntax
property spriteNum

on beginSprite (me)
    memName = sprite(spriteNum).member.name
    put("The bitRate of member"&&memName&&"is"&&member(memName).bitRate)
end

// JavaScript syntax
function beginSprite() {
    var memName = sprite(spriteNum).member.name;
    put("The bitRate of member " + memName +" is " + member(memName).bitRate);
}
```

bitsPerSample

Syntaxe

```
-- Lingo syntax
memberObjRef.bitsPerSample

// JavaScript syntax
memberObjRef.bitsPerSample;
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; indique le codage du fichier d'origine qui a été encodé pour Shockwave Audio (SWA). Cette propriété n'est disponible qu'après le début de la lecture du son SWA ou après le préchargement du fichier à l'aide de la commande `preLoadBuffer`.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affecte le codage d'origine du fichier utilisé pour l'acteur SWA en flux continu Paul Robin à l'acteur champ Codage :

```
-- Lingo syntax
member("How Deep").text = member("Paul Robeson").bitsPerSample

// JavaScript syntax
member("How Deep").text = member("Paul Robeson").bitsPerSample;
```

blend (3D)

Syntaxe

```
sprite(whichSprite).camera{(index)}.backdrop[index].blend
member(whichCastmember).camera(whichCamera).backdrop[index].blend
sprite(whichSprite).camera{(index)}.overlay[index].blend
member(whichCastmember).camera(whichCamera).overlay[index].blend
member(whichCastmember).shader(whichShader).blend
member(whichCastmember).model(whichModel).shader.blend
member(whichCastmember).model(whichModel).shaderList{[index]}.blend
```

Description

Propriété 3D de fond, de recouvrement et de matériau `#standard` ; indique l'opacité du fond, du recouvrement ou du matériau.

La définition de la propriété `blend` d'un matériau n'a aucun effet, sauf si la propriété `transparent` de ce matériau présente la valeur `TRUE`.

La plage de cette propriété s'étend de 0 à 100, la valeur par défaut étant 100.

Exemple

L'instruction suivante définit la propriété `blend` du matériau du modèle Fenêtre sur la valeur 80. Si la propriété `transparent` du matériau de Fenêtre présente la valeur TRUE, le modèle est légèrement transparent.

```
-- Lingo syntax
member("House").model("Window").shader.blend = 80

// Java Script
member("House").getPropRef("model", 1).shaderList[1].blend = 80;
```

Voir aussi

`bevelDepth`, `overlay`, `shadowPercentage`, `transparent`

blend (image-objet)

Syntaxe

```
-- Lingo syntax
spriteObjRef.blend

// JavaScript syntax
spriteObjRef.blend;
```

Description

Propriété d'image-objet ; renvoie ou définit la valeur d'opacité d'une image-objet, comprise entre 0 et 100, correspondant aux valeurs d'opacité de la boîte de dialogue Propriétés de l'image-objet. Lecture/écriture.

Les couleurs possibles dépendent des couleurs disponibles dans la palette, quel que soit le codage de couleurs du moniteur.

Pour assurer des résultats optimaux, utilisez l'encre Opacité avec des images possédant un codage de couleurs supérieur à 8 bits.

Exemple

L'instruction suivante affecte une valeur d'opacité de 40 pour cent à l'image-objet 3.

```
-- Lingo syntax
sprite(3).blend = 40

// JavaScript syntax
sprite(3).blend = 40;
```

L'instruction suivante affiche la valeur d'opacité de l'image-objet 3 dans la fenêtre Messages :

```
-- Lingo syntax
put(sprite(3).blend)

// JavaScript syntax
put(sprite(3).blend);
```

Voir aussi

blendLevel, Image-objet

blendConstant

Syntaxe

```
member(whichCastmember).shader(whichShader).blendConstant
member(whichCastmember).model(whichModel).shader.blendConstant
member(whichCastmember).model(whichModel).shaderList{[index]}.blendConstant
```

Description

Propriété 3D de matériau `#standard` ; indique le taux de fusion utilisé pour la première couche de texture du matériau.

Si la propriété `useDiffuseWithTexture` du matériau présente la valeur `TRUE`, la texture se mélange à la couleur définie par la propriété `diffuse` du matériau. Si la propriété `useDiffuseWithTexture` présente la valeur `FALSE`, le blanc est utilisé pour la fusion.

Chacune des autres couches de texture se mélange à la couche inférieure. Utilisez la propriété `blendConstantList` pour contrôler la fusion dans ces couches de texture.

La propriété `blendConstant` ne fonctionne que lorsque la propriété `blendSource` du matériau présente la valeur `#constant`. Pour plus d'informations, reportez-vous aux entrées `blendSource` et `blendSourceList`.

La plage de cette propriété va de 0 à 100, la valeur par défaut étant 50.

Exemple

Dans l'exemple suivant, la liste des matériaux du modèle Mystère contient six matériaux. L'instruction suivante définit la propriété `blendConstant` du second matériau sur 20. Cette propriété est affectée par le paramétrage des propriétés `blendFunction`, `blendFunctionList`, `blendSource` et `blendSourceList`.

```
member("Level2").model("MysteryBox").shaderList[2].blendConstant = 20
```

Voir aussi

blendConstantList, blendFunction, blendFunctionList, blendSource, blendSourceList, useDiffuseWithTexture, diffuse, diffuseColor

blendConstantList

Syntaxe

```
member(whichCastmember).shader(whichShader).blendConstantList
member(whichCastmember).model(whichModel).shader.blendConstantList{[index]}
member(whichCastmember).model(whichModel).shaderList{[index]}.blendConstantList{[index]}
```

Description

Propriété 3D de matériau `#standard` ; indique le taux utilisé pour fusionner une couche de texture du matériau avec la couche de texture inférieure.

La liste des textures et la liste des constantes de fusion du matériau comportent huit positions d'index chacune. Chaque position d'index de la liste des constantes de fusion contrôle la fusion de la texture à la position d'index correspondante de la liste des textures. Vous pouvez attribuer la même valeur à toutes les positions d'index de la liste simultanément en omettant de spécifier le paramètre facultatif *`index`*. Utilisez le paramètre *`index`* pour définir les positions d'index une par une.

La propriété `blendConstantList` ne fonctionne que lorsque la propriété `blendSource` de la couche de texture correspondante présente la valeur `#constant`.

La plage de cette propriété va de 0 à 100, la valeur par défaut étant 50.

Exemple

Dans l'exemple suivant, la liste des matériaux du modèle Mystère contient six matériaux. L'instruction suivante indique la propriété `blendConstant` de chacune des textures utilisées par le second matériau. Cette propriété est affectée par le paramétrage des propriétés `blendFunction`, `blendFunctionList`, `blendSource` et `blendSourceList`.

```
-- Lingo syntax
put member("Level2").model("MysteryBox").shaderList[2].blendConstantList
-- [20.0000, 50.0000, 50.0000, 50.0000, 20.0000, 50.0000, 50.0000, 50.0000]

// JavaScript syntax
put(member("Level2").getPropRef("model",3).shaderList[1].blendConstantList);
// <[20.0000, 50.0000, 50.0000, 50.0000, 20.0000, 50.0000, 50.0000, 50.0000]>
put member("Level2").model("MysteryBox").shaderList[2].blendConstantList
-- [20.0000, 50.0000, 50.0000, 50.0000, 20.0000, 50.0000, 50.0000, 50.0000]
```

Voir aussi

`blendConstant`, `blendFunction`, `blendFunctionList`, `blendSource`, `blendSourceList`, `useDiffuseWithTexture`, `diffuse`, `diffuseColor`

blendFactor

Syntaxe

```
member(whichCastmember).model(whichModel).keyframePlayer.blendFactor
member(whichCastmember).model(whichModel).bonesPlayer.blendFactor
```

Description

Propriété 3D de modificateur `keyframePlayer` et `bonesPlayer` ; indique le degré de combinaison d'un mouvement au mouvement qui le précède.

La plage de cette propriété s'étend de 0 à 100, la valeur par défaut étant 0.

`BlendFactor` n'est utilisé que lorsque la propriété `autoblend` du modificateur présente la valeur `FALSE`. Si la valeur de la propriété `blendFactor` est 100, le mouvement en cours ne possède aucune des caractéristiques du mouvement qui le précède. Si la valeur de `blendFactor` est 0, le mouvement en cours possède toutes les caractéristiques du mouvement qui le précède et aucune valeur qui lui soit propre. Si la valeur de `blendFactor` est 50, le mouvement en cours constitue une synthèse composée, dans les mêmes proportions, de ses propres caractéristiques et de celles du mouvement qui le précède. La valeur de `blendFactor` peut être modifiée pour créer des transitions, contrairement à la transition linéaire créée lorsque la propriété `autoblend` du modificateur présente la valeur `TRUE`.

Exemple

L'instruction suivante définit la propriété `blendFactor` du modèle Martien3 sur la valeur 50. Si la propriété `autoblend` du modificateur présente la valeur `FALSE`, chaque mouvement de la liste de lecture du `keyframePlayer` pour Martien3 constitue un mélange, à parts égales, du mouvement en cours et de celui qui le précède.

```
member("newaliens").model("Alien3").keyframePlayer.blendFactor = 50
```

Voir aussi

`autoblend`, `keyframePlayer (modificateur)`

blendFunction

Syntaxe

```
member(whichCastmember).shader(whichShader).blendFunction
member(whichCastmember).model(whichModel).shader.blendFunction
member(whichCastmember).model(whichModel).shaderList{[index]}.blendFunction
```

Description

Propriété 3D de matériau `#standard` ; indique le type de fusion utilisé pour la première couche de texture du matériau.

Si la propriété `useDiffuseWithTexture` du matériau présente la valeur `TRUE`, la texture se mélange à la couleur définie par la propriété `diffuse` du matériau. Si la propriété `useDiffuseWithTexture` présente la valeur `FALSE`, le blanc est utilisé pour la fusion.

Chacune des autres couches de texture se mélange à la couche inférieure. Utilisez la propriété `blendFunctionList` pour contrôler la fusion dans ces couches de texture.

La propriété `blendFunction` peut prendre les valeurs suivantes :

`#multiply` multiplie les valeurs RVB de la couche de texture par la couleur utilisée pour la fusion (voir ci-dessus).

`#add` ajoute les valeurs RVB de la couche de texture à la couleur utilisée pour la fusion, puis se verrouille sur 255.

`#replace` empêche la fusion de la texture avec la couleur définie par la propriété `diffuse` du matériau.

`#blend` combine les couleurs de la couche de texture avec la couleur utilisée pour la fusion selon le taux défini par la propriété `blendConstant`.

La valeur par défaut de cette propriété est `#multiply`.

Exemple

Dans l'exemple suivant, la liste des matériaux du modèle Mystère contient six matériaux. L'instruction suivante définit la propriété `blendFunction` du second matériau sur la valeur `#blend`. Cette opération active le paramétrage des propriétés `blendSource`, `blendSourceList`, `blendConstant` et `blendConstantList`.

```
member("Level2").model("MysteryBox").shaderList[2].blendFunction = #blend
```

Voir aussi

`blendConstant`, `blendConstantList`, `blendFunctionList`, `blendSource`, `blendSourceList`, `useDiffuseWithTexture`, `diffuse`, `diffuseColor`

blendFunctionList

Syntaxe

```
member(whichCastmember).shader(whichShader).blendFunctionList{[index]}
member(whichCastmember).model(whichModel).shader.blendFunctionList{[index]}
member(whichCastmember).model(whichModel).shaderList{[index]}.blendFunctionList{[index]}
```

Description

Propriété 3D de matériau `#standard` ; liste linéaire indiquant la façon dont chaque couche de texture se mélange à la couche inférieure.

La liste des textures et la liste des fonctions de fusion du matériau comportent huit positions d'index chacune. Chaque position d'index de la liste des fonctions de fusion contrôle la fusion de la texture à la position d'index correspondante de la liste de textures. Vous pouvez attribuer la même valeur à toutes les positions d'index de la liste simultanément en omettant de spécifier le paramètre facultatif *`index`*. Utilisez le paramètre *`index`* pour définir les positions d'index une par une.

Chaque position d'index de la liste des fonctions de fusion peut avoir une des valeurs suivantes :

`#multiply` multiplie les valeurs RVB de la couche de texture par les valeurs RVB de la couche de texture inférieure.

`#add` ajoute les valeurs RVB de la couche de texture aux valeurs RVB de la couche de texture inférieure, puis se verrouille sur 255.

`#replace` entraîne le recouvrement par la texture de la couche de texture inférieure. Aucune fusion n'a lieu.

`#blend` entraîne le contrôle de la fusion par la valeur de la propriété `blendSource`, permettant ainsi une fusion alpha.

La valeur par défaut de cette propriété est `#multiply`.

Exemple

Dans l'exemple suivant, la propriété `shaderList` du modèle Mystère contient six matériaux. L'instruction suivante indique que la quatrième position d'index de la propriété `blendFunctionList` du deuxième matériau présente la valeur `#blend`. La fusion de la quatrième couche de texture du deuxième matériau du modèle est contrôlée par le paramétrage des propriétés `blendSource`, `blendSourceList`, `blendConstant`, `blendConstantList`, `diffuse`, `diffuseColor` et `useDiffuseWithTexture`.

```
put member("Level2").model("MysteryBox").shaderList[2].blendFunctionList[4]
-- #blend
```

Voir aussi

`blendConstant`, `blendConstantList`, `blendFunction`, `blendSource`, `blendSourceList`, `diffuse`, `diffuseColor`, `useDiffuseWithTexture`

blendLevel

Syntaxe

```
sprite(whichSpriteNumber).blendLevel
the blendLevel of sprite whichSpriteNumber
```

Description

Propriété d'image-objet ; permet de définir la valeur d'opacité en cours d'une image-objet ou d'accéder à cette valeur. Les valeurs possibles sont comprises entre 0 et 255. Cette plage diffère de celle de l'Inspecteur d'image-objet, qui est comprise entre 0 et 100. Toutefois, elles ont toutes deux le même résultat, la seule différence portant sur l'échelle des valeurs.

Cette propriété équivaut à la propriété d'image-objet `blend`.

Exemple

```
sprite(3).blendlevel = 99
```

Voir aussi

`blend (image-objet)`

blendRange

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).blendRange.start
modelResourceObjectReference.blendRange.end
member(whichCastmember).modelResource(whichModelResource).blendRange.start
modelResourceObjectReference.blendRange.end
```

Description

Propriété 3D ; lorsqu'elle est utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir ou de définir le début et la fin de la plage d'opacité de la ressource.

L'opacité des particules du système est interpolée de façon linéaire entre `blendRange.start` et `blendRange.end` pendant toute la durée de vie de chaque particule.

La valeur de cette propriété doit toujours être supérieure ou égale à 0,0 et inférieure ou égale à 100,0. La valeur par défaut de cette propriété est 100,0.

Exemple

L'instruction suivante définit les propriétés blendRange de la ressource de modèle systèmeThermique, qui est du type #particle.

La première ligne donne la valeur de départ 100 et la seconde donne la valeur de fin 0. Cette instruction rend les particules de systèmeThermique complètement opaques lorsqu'elles apparaissent pour la première fois, puis elles s'estompent petit à petit jusqu'à devenir transparentes.

```
member("Heater").modelResource("ThermoSystem").blendRange.start = 100.0
member("Heater").modelResource("ThermoSystem").blendRange.end = 0.0
```

blendSource

Syntaxe

```
member(whichCastmember).shader(whichShader).blendSource
member(whichCastmember).model(whichModel).shader.blendSource
member(whichCastmember).model(whichModel).shaderList{[index]}.blendSource
```

Description

Propriété 3D de matériau `#standard` ; indique si la fusion de la première couche de texture de la liste des textures du matériau repose sur l'information alpha de la texture ou sur un taux constant.

Si la propriété `useDiffuseWithTexture` du matériau présente la valeur `TRUE`, la texture se mélange à la couleur définie par la propriété `diffuse` du matériau. Si la propriété `useDiffuseWithTexture` présente la valeur `FALSE`, le blanc est utilisé pour la fusion.

Chacune des autres couches de texture se mélange à la couche inférieure. Utilisez la propriété `blendSourceList` pour contrôler la fusion dans ces couches de texture.

La propriété `blendSource` ne fonctionne que lorsque la propriété `blendFunction` du matériau présente la valeur `#blend`.

Les valeurs possibles de cette propriété sont les suivantes :

`#alpha` indique que c'est l'information alpha de la texture qui détermine le taux de fusion de chaque pixel de la texture avec la couleur utilisée (voir ci-dessus).

`#constant` indique que la valeur de la propriété `blendConstant` du matériau est utilisée comme taux de fusion pour tous les pixels de la texture.

La valeur par défaut de cette propriété est `#constant`.

Exemple

Dans l'exemple suivant, la liste des matériaux du modèle Mystère contient six matériaux. L'instruction suivante définit la propriété `blendSource` de la première texture utilisée par le second matériau sur la valeur `#constant`. Cette opération active le paramétrage des propriétés `blendConstant` et `blendConstantList`.

```
-- Lingo syntax
member("Level2").model("MysteryBox").shaderList[2].blendSource = #constant

// JavaScript syntax
member("Level2").getPropRef("model",3).shaderList[2].blendSource = symbol("constant");
```

Voir aussi

`blendSourceList`, `blendFunction`, `blendFunctionList`, `blendConstant`, `blendConstantList`, `useDiffuseWithTexture`, `diffuse`, `diffuseColor`

blendSourceList

Syntaxe

```
member(whichCastmember).shader(whichShader).blendSourceList[index]
member(whichCastmember).model(whichModel).shader.blendSourceList{[index]}
member(whichCastmember).model(whichModel).shaderList{[index]}.blendSourceList{[index]}
```

Description

Propriété 3D de matériau `#standard` ; indique si la fusion d'une couche de texture avec les couches de texture inférieures repose sur l'information alpha de la texture ou sur un taux constant.

La liste des textures et la liste des sources de fusion du matériau comportent huit positions d'index chacune. Chaque position d'index de la liste des sources de fusion contrôle la fusion de la texture à la position d'index correspondante de la liste de textures. Vous pouvez attribuer la même valeur à toutes les positions d'index de la liste simultanément en omettant de spécifier le paramètre facultatif *index*. Utilisez le paramètre *index* pour définir les positions d'index une par une.

La propriété `blendSourceList` ne fonctionne que lorsque la propriété `blendSource` de la couche de texture correspondante présente la valeur `#blend`. Pour plus d'informations, reportez-vous aux entrées `blendFunction` et `blendFunctionList`.

Les valeurs possibles de cette propriété sont les suivantes :

`#alpha` indique que c'est l'information alpha de la texture qui détermine le taux de fusion de chaque pixel de la couche de texture avec la couche inférieure.

`#constant` indique que la valeur de la propriété `blendConstant` de la couche de texture correspondante est utilisée comme taux de fusion pour tous les pixels de la couche de texture. Pour plus d'informations, reportez-vous aux entrées `blendConstant` et `blendConstantList`.

La valeur par défaut de cette propriété est `#constant`.

Exemple

Dans l'exemple suivant, la liste des matériaux du modèle Mystère contient six matériaux. Chaque matériau possède une liste de textures pouvant contenir jusqu'à huit textures. L'instruction suivante indique que la propriété `blendSource` de la quatrième texture utilisée par le second matériau présente la valeur `#constant`. Cette opération active le paramétrage des propriétés `blendConstant`, `blendConstantList` et `useDiffuseWithTexture`.

```
member("Level2").model("MysteryBox").shaderList[2].blendSourceList[4] = #constant
```

Voir aussi

`blendSource`, `blendFunction`, `blendFunctionList`, `blendConstant`, `blendConstantList`, `useDiffuseWithTexture`, `diffuse`, `diffuseColor`

blendTime

Syntaxe

```
member(whichCastmember).model(whichModel).keyframePlayer.blendTime
member(whichCastmember).model(whichModel).bonesPlayer.blendTime
```

Description

Propriété 3D de modificateur `keyframePlayer` et `bonesPlayer` ; détermine la durée, en millisecondes, de la transition entre les mouvements de la liste de lecture du modificateur pour le modèle.

La propriété `blendTime` fonctionne en conjonction avec la propriété `autoBlend` du modificateur. Lorsque la propriété `autoBlend` présente la valeur `TRUE`, le modificateur crée une transition linéaire entre le mouvement en cours de lecture du modèle et le mouvement précédent. La valeur de la propriété `blendTime` correspond à la durée de cette transition. La propriété `blendTime` est ignorée lorsque `autoBlend` présente la valeur `FALSE`.

Le paramètre par défaut de cette propriété est 500.

Exemple

L'instruction suivante définit la durée de la transition entre les mouvements de la liste de lecture du modificateur du modèle Martien5 à 1 200 millisecondes.

```
member("newaliens").model("Alien5").keyframePlayer.blendTime = 1200
```

Voir aussi

`autoblend`, `blendFactor`

bone

Syntaxe

```
-- Lingo Usage
member(whichCastmember).modelResource(whichModelResource).bone.count
member(whichCastmember).model(whichModel).bonesPlayer.bone[index].transform
member(whichCastmember).model(whichModel).bonesPlayer.bone[index].worldTransform

// JavaScript Usage
member(whichCastmember).getProp("model",
whichModelIndex).getPropRef("bonesplayer").getPropRef("bone", whichBoneIndex).transform
member(whichCastmember).getProp("model",
whichModelIndex).getPropRef("bonesplayer").getPropRef("bone",
whichBoneIndex).worldTransform
```

Description

Elément 3D ; un segment est un élément structurel d'une ressource de modèle créée dans un programme de modélisation 3D. Les segments ne peuvent pas être créés, supprimés ou réarrangés dans Director.

Les mouvements de segments (`#bones`), qui doivent également être contrôlés par un script dans un programme de modélisation 3D, agissent sur la structure des segments d'une ressource de modèle et sont gérés dans Director par le modificateur `bonesPlayer`.

Voir aussi

`count (3D)`, `bonesPlayer (modificateur)`, `transform (propriété)`, `worldTransform`

bonesPlayer (modificateur)

Syntaxe

```
member(whichCastmember).model(whichModel).bonesPlayer.whichBonesPlayerProperty
```

Description

Modificateur 3D ; gère l'utilisation des mouvements par les modèles. Les mouvements gérés par le modificateur `bonesPlayer` animent les segments du modèle.

Les mouvements et les modèles qui les utilisent doivent être créés dans un programme de modélisation 3D, exportés au format *.w3d, puis importés dans une animation. Les mouvements ne peuvent pas être appliqués aux primitives de modèle créées dans Director.

L'ajout du modificateur `bonesPlayer` à un modèle à l'aide de la commande `addModifier` permet d'accéder aux propriétés suivantes du modificateur `bonesPlayer` :

`playing (3D)` indique qu'un modèle exécute un mouvement.

`playlist` est une liste linéaire de listes de propriétés contenant les paramètres de lecture des mouvements d'un modèle en file d'attente.

`currentTime (3D)` indique la position, en millisecondes, du mouvement en cours de lecture ou en pause.

`playRate (3D)` est un nombre multiplié par le paramètre *`échelle`* de la commande `play()` ou `queue()` pour déterminer la cadence de lecture du mouvement.

`playlist.count (3D)` renvoie le nombre de mouvements en file d'attente dans la liste de lecture.

`rootLock` indique si le composant de translation du mouvement est utilisé ou ignoré.

`currentLoopState` indique si le mouvement est lu une seule fois ou répété de façon continue.

`blendTime` indique la durée de la transition créée par le modificateur entre les mouvements lorsque la propriété `autoblend` du modificateur présente la valeur `TRUE`.

`autoblend` indique si le modificateur crée une transition linéaire entre le mouvement en cours de lecture et le mouvement précédent.

`blendFactor` indique le degré de fusion entre les mouvements lorsque la propriété `autoBlend` du modificateur présente la valeur `FALSE`.

`bone[IdDeSegment].transform` indique la transformation du segment par rapport au segment parent. Vous pouvez trouver la valeur de `IdDeSegment` en testant la propriété `getBoneID` de la ressource de modèle. Lorsque vous définissez la transformation du segment, il n'est plus contrôlé par le mouvement en cours. Le contrôle manuel s'arrête avec le mouvement en cours.

`bone[IdDeSegment].getWorldTransform` renvoie la transformation du segment par rapport à l'univers.

`lockTranslation` indique si le modèle peut être déplacé à partir des plans spécifiés.

`positionReset` indique si le modèle retourne à sa position de départ à la fin d'un mouvement ou de chaque itération d'une boucle.

`rotationReset` indique l'élément de rotation d'une transition d'un mouvement à un autre ou de la boucle d'un seul mouvement.

Remarque : *pour plus d'informations, consultez les entrées des différentes propriétés.*

Le modificateur `bonesPlayer` utilise les commandes suivantes :

`pause() (3D)` stoppe le mouvement du modèle en cours d'exécution.

`play() (3D)` entraîne ou reprend l'exécution d'un mouvement.

`playNext() (3D)` entraîne la lecture du mouvement suivant de la liste de lecture.

queue() (3D) ajoute un mouvement à la fin de la liste de lecture.

Le modificateur bonesPlayer génère les événements suivants, qui sont utilisés par les gestionnaires déclarés dans les commandes registerForEvent() et registerScript(). L'appel au gestionnaire déclaré contient trois arguments : le type d'événement (#animationStarted ou #animationEnded), le nom du mouvement, ainsi que sa position en cours. Pour plus d'informations sur les événements de notification, reportez-vous à l'entrée registerForEvent().

#animationStarted est envoyé au début de la lecture d'un mouvement. Si la fusion est utilisée entre des mouvements, l'événement est envoyé au début de la transition.

#animationEnded est envoyé à la fin d'un mouvement. Si la fusion est utilisée entre des mouvements, l'événement est envoyé à la fin de la transition.

Voir aussi

keyframePlayer (modificateur), addModifier, modifiers, modifier

border

Syntaxe

```
-- Lingo syntax
memberObjRef.border

// JavaScript syntax
memberObjRef.border;
```

Description

Propriété d'acteur champ ; indique l'épaisseur, en pixels, de la bordure entourant l'acteur champ spécifié.

Exemple

L'instruction suivante attribue à la bordure de l'acteur champ Titre une épaisseur de 10 pixels :

```
--Lingo syntax
member("Title").border = 10

// JavaScript syntax
member("Title").border = 10;
```

bottom

Syntaxe

```
-- Lingo syntax
spriteObjRef.bottom

// JavaScript syntax
spriteObjRef.bottom;
```

Description

Propriété d'image-objet ; spécifie la coordonnée verticale du bord inférieur du rectangle de délimitation d'une image-objet. Lecture/écriture.

Exemple

L'instruction suivante affecte la coordonnée verticale de la partie inférieure de l'image-objet numérotée (i + 1) à la variable `lowest` :

```
-- Lingo syntax
lowest = sprite(i + 1).bottom

// JavaScript syntax
var lowest = sprite(i + 1).bottom;
```

Voir aussi

Image-objet

bottom (3D)

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).bottom
```

Description

Propriété 3D de ressource de modèle `#box` ; indique si le côté de la boîte coupé par son axe des y négatif est fermé (`TRUE`) ou ouvert (`FALSE`).

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante attribue à la propriété `bottom` de la ressource de modèle boîteAcadeau la valeur `TRUE`, ce qui signifie que le fond de cette boîte est fermé.

```
-- Lingo syntax
nmr = member("3D World").newModelresource("GiftBox", #box)
member("3D World").modelResource("GiftBox").bottom = TRUE

// JavaScript syntax
nmr = member("3D World").newModelresource("GiftBox",symbol("box"));
member("3D World").getProp("modelresource",10).bottom = true;
```

Voir aussi

back, front, top (3D), left (3D), right (3D), bottomCap

bottomCap

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).bottomCap
```

Description

Propriété 3D de ressource de modèle `#cylinder` ; indique si le fond du cylindre coupé par son axe des y négatif est fermé (`TRUE`) ou ouvert (`FALSE`).

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante attribue à la propriété `bottomCap` de la ressource de modèle Cylindre11 la valeur FALSE, ce qui signifie que le fond de ce cylindre est ouvert.

```
-- Lingo syntax
nmr = member("3D World").newModelresource("Cylinder11", #cylinder);
member("3D World").modelResource("Cylinder11").bottomCap = FALSE

// JavaScript syntax
nmr = member("3D World").newModelresource("Cylinder11", symbol("cylinder"));
member("3D World").getPropRef("modelResource", 11).bottomCap = false;
```

Voir aussi

topCap, bottomRadius, bottom (3D)

bottomRadius

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).bottomRadius
```

Description

Propriété 3D de ressource de modèle `#cylinder` ; indique le rayon, en unités de l'univers, du fond du cylindre coupé par son axe des y négatif.

La valeur par défaut de cette propriété est 25.

Exemple

L'instruction suivante attribue à la propriété `bottomRadius` de la ressource de modèle Tube la valeur 38,5.

```
member("3D World").modelResource("Tube").bottomRadius = 38.5
```

Voir aussi

topRadius, bottomCap

bottomSpacing

Syntaxe

```
-- Lingo syntax
chunkExpression.bottomSpacing

// JavaScript syntax
chunkExpression.bottomSpacing;
```

Description

Propriété d'acteur texte ; permet de spécifier tout espacement supplémentaire applicable au bas de chaque paragraphe dans la partie *`expressionSousChaîne`* de l'acteur texte.

La valeur même est un entier, qui indique un espacement moindre entre les paragraphes s'il est inférieur à 0 et un espacement plus important s'il est supérieur à 0.

La valeur par défaut est 0 ; elle correspond à l'espacement par défaut entre les paragraphes.

Remarque : *cette propriété, comme toutes les propriétés d'acteur texte, ne supporte que la syntaxe à point.*

Exemple

Dans l'exemple suivant, un espace est ajouté après le premier paragraphe de l'acteur texte Nouvelles du jour.

```
--Lingo syntax
member("News Items").paragraph[1].bottomSpacing=20

// JavaScript syntax
member("News Items").getPropRef("paragraph", 1).bottomSpacing=20;
```

Voir aussi

top (3D)

boundary

Syntaxe

```
member(whichCastmember).model(whichModel).inker.boundary
member(whichCastmember).model(whichModel).toon.boundary
```

Description

Propriété 3D de modificateur `inker` et `toon` ; permet de définir si une ligne est tracée au niveau des bords d'un modèle.

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante attribue à la propriété `boundary` du modificateur inker appliqué au modèle Boîte la valeur `TRUE`. Les lignes sont tracées aux bords de la surface du modèle.

```
-- Lingo syntax
member("shapes").model("Box").addModifier(#inker)
member("shapes").model("Box").inker.boundary = TRUE

// JavaScript syntax
member("shapes").getProp("model",1).addModifier(symbol("inker"));
member("shapes").getProp("model",1).getPropRef("inker").boundary = true;
```

Voir aussi

lineColor, lineOffset, silhouettes, creases

boundingSphere

Syntaxe

```
member(whichCastmember).model(whichModel).boundingSphere
member(whichCastmember).group(whichGroup).boundingSphere
member(whichCastmember).light(whichLight).boundingSphere
member(whichCastmember).camera(whichCamera).boundingSphere
```

Description

Propriété 3D de modèle, de groupe, de lumière et de caméra ; décrit une sphère contenant le modèle, le groupe, la lumière ou la caméra et ses enfants.

La valeur de cette propriété est une liste contenant la position vectorielle du centre de la sphère et la longueur, exprimée en valeur à virgule flottante, du rayon de la sphère.

Cette propriété peut être testée, mais pas définie.

Exemple

L'exemple suivant affiche la sphère de délimitation d'une lumière dans la fenêtre Messages.

```
-- Lingo syntax
put member("newAlien").light[5].boundingSphere
-- [vector(166.8667, -549.6362, 699.5773), 1111.0039]

// JavaScript syntax
put(member("newAlien").getProp("light",5).boundingSphere);
// <[vector(166.8667, -549.6362, 699.5773), 1111.0039]>
```

Voir aussi

debug

boxDropShadow

Syntaxe

```
-- Lingo syntax
memberObjRef.boxDropShadow

// JavaScript syntax
memberObjRef.boxDropShadow;
```

Description

Propriété d'acteur ; détermine la taille, en pixels, de l'ombre portée du cadre de l'acteur champ spécifié par *quelActeur*.

Exemple

L'instruction suivante donne à l'ombre portée de l'acteur champ Titre une largeur de 10 pixels :

```
--Lingo syntax
member("Title").boxDropShadow = 10

// JavaScript syntax
member("Title").boxDropShadow = 10;
```

boxType

Syntaxe

```
-- Lingo syntax
memberObjRef.boxType

// JavaScript syntax
```

```
memberObjRef.boxType;
```

Description

Propriété d'acteur ; détermine le type de zone de texte utilisé pour l'acteur spécifié. Les valeurs possibles sont `#adjust` (ajustable), `#scroll` (défilant), `#fixed` (fixe) et `#limit` (limité).

Exemple

L'instruction suivante fait du cadre de l'acteur champ Editorial un champ défilant.

```
--Lingo syntax
member("Editorial").boxType = #scroll

// JavaScript syntax
member("Editorial").boxType = symbol("scroll");
```

brightness

Syntaxe

```
member(whichCastmember).shader(whichShader).brightness
member(whichCastmember).model(whichModel).shader.brightness
member(whichCastmember).model(whichModel).shaderList{[index]}.brightness
```

Description

Propriété 3D de matériau `#newsprint` et `#engraver` ; indique la quantité de blanc fusionnée dans le matériau.

La plage de cette propriété va de 1 à 100, la valeur par défaut étant 0.

Exemple

L'instruction suivante définit la luminosité du matériau utilisé par le modèle gbCyl2 à la moitié de sa valeur maximum.

```
-- Lingo syntax
member("scene").model("gbCyl2").shader.brightness = 50

// JavaScript syntax
member("scene").getProp("shader",1).brightness = 50;
```

Voir aussi

newShader

broadcastProps

Syntaxe

```
-- Lingo syntax
memberObjRef.broadcastProps

// JavaScript syntax
memberObjRef.broadcastProps;
```

Description

Propriété d'acteur ; contrôle si les modifications apportées à un acteur Flash ou forme vectorielle sont immédiatement transmises à toutes ses images-objets présentes sur la scène (`TRUE`) ou non (`FALSE`).

Lorsque cette propriété présente la valeur `FALSE`, les modifications apportées à l'acteur sont utilisées comme valeurs par défaut pour les nouvelles images-objets et n'affectent pas les images-objets sur la scène.

Cette propriété présente la valeur par défaut `TRUE` et peut être testée et définie.

Exemple

Le script d'image suivant suppose que l'acteur Animation de navigation d'une animation Flash a été configuré avec une propriété `broadcastProps` définie sur la valeur `FALSE`. Il permet provisoirement de modifier un acteur animation Flash à diffuser aux images-objets placées sur la scène. Il définit ensuite la propriété `viewScale` de l'acteur animation Flash et cette modification est transmise à son image-objet. Le script interdit alors à l'animation Flash de diffuser les modifications ultérieures à ses images-objets.

```
-- Lingo syntax
on enterFrame
    member("Navigation Movie").broadcastProps = TRUE
    member("Navigation Movie").viewScale = 200
    member("Navigation Movie").broadcastProps = FALSE
end

// JavaScript syntax
function enterFrame() {
    member("Navigation Movie").broadcastProps = 1;
    member("Navigation Movie").viewScale = 200;
    member("Navigation Movie").broadcastProps = 0;
}
```

bufferSize

Syntaxe

```
-- Lingo syntax
memberObjRef.bufferSize

// JavaScript syntax
memberObjRef.bufferSize;
```

Description

Propriété d'acteur Flash ; contrôle le nombre d'octets d'une animation Flash liée qui sont passés en mémoire en une seule fois. La valeur de la propriété d'acteur `bufferSize` ne peut être qu'un nombre entier. Cette propriété n'est effective que lorsque la propriété `preload` de l'acteur présente la valeur `FALSE`.

Cette propriété peut être testée et définie. La valeur par défaut est 32 768 octets.

Exemple

Le gestionnaire `startMovie` suivant définit un acteur animation Flash pour une lecture en flux continu, puis définit sa propriété `bufferSize`.

```
-- Lingo syntax
on startMovie
    member.("Flash Demo").preload = FALSE
    member.("Flash Demo").bufferSize = 65536
```

```
end

// JavaScript syntax
function startMovie() {
    member.("Flash Demo").preload = 0;
    member.("Flash Demo").bufferSize = 65536;
}
```

Voir aussi

bytesStreamed, preLoadRAM, stream(), streamMode

buttonCount

Syntaxe

```
-- Lingo syntax
dvdObjRef.buttonCount

// JavaScript syntax
dvdObjRef.buttonCount;
```

Description

Propriété de DVD ; renvoie le nombre de boutons disponibles sur le menu DVD en cours. Lecture seule.

Propriété non prise en charge sur Mac à ce jour.

Voir aussi

DVD

buttonsEnabled

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.buttonsEnabled

// JavaScript syntax
memberOrSpriteObjRef.buttonsEnabled;
```

Description

Propriété d'acteur Flash et propriété d'image-objet ; contrôle si les boutons d'une animation Flash sont actifs (`TRUE`, valeur par défaut) ou inactifs (`FALSE`). Les actions de bouton sont uniquement déclenchées lorsque la propriété `actionsEnabled` reçoit la valeur `TRUE`.

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant accepte une référence d'image-objet et permet d'activer ou de désactiver la propriété `buttonsEnabled` de l'image-objet.

```
-- Lingo syntax
on ToggleButtons(whichSprite)
    sprite(whichSprite).buttonsEnabled = not(sprite(whichSprite).buttonsEnabled)
end
```

```
// JavaScript syntax
function ToggleActions(whichSprite) {
    sprite(whichSprite).buttonsEnabled = !(sprite(whichSprite).buttonsEnabled);
}
```

Voir aussi

actionsEnabled

buttonStyle

Syntaxe

```
-- Lingo syntax
_movie.buttonStyle

// JavaScript syntax
_movie.buttonStyle;
```

Description

Propriété d'animation ; détermine la réponse visuelle des boutons lorsque l'utilisateur maintient le bouton de la souris enfoncé. Lecture/écriture.

Elle s'applique uniquement aux boutons créés avec l'outil Bouton de la palette des outils.

La propriété `buttonStyle` peut prendre les valeurs suivantes :

- `0` (style de liste : valeur par défaut) : les boutons suivants sont mis en surbrillance lorsque le pointeur les survole. Si l'utilisateur relâche le bouton de la souris, le script associé à ce dernier est activé.
- `1` (style de boîte de dialogue) : seul le premier bouton sur lequel l'utilisateur clique est mis en surbrillance. Les boutons suivants ne le sont pas. Si l'utilisateur relâche le bouton de la souris alors que le pointeur survole un bouton autre que celui sur lequel il a cliqué initialement, le script associé à ce bouton n'est pas activé.

Exemple

L'instruction suivante attribue à la propriété `buttonStyle` la valeur 1 :

```
-- Lingo syntax
_movie.buttonStyle = 1

// JavaScript syntax
_movie.buttonStyle = 1;
```

L'instruction suivante mémorise le paramètre en cours de la propriété `buttonStyle` en le stockant dans la variable `valeurDeStyleDeBouton` :

```
-- Lingo syntax
buttonStyleValue = _movie.buttonStyle

// JavaScript syntax
var buttonStyleValue = _movie.buttonStyle;
```

Voir aussi

Animation

buttonType

Syntaxe

```
member(whichCastMember).buttonType
the buttonType of member whichCastMember
```

Description

Propriété d'acteur bouton ; indique le type de l'acteur bouton spécifié. Les valeurs possibles sont `#pushButton` (bouton-poussoir), `#checkBox` (case à cocher) et `#radioButton` (bouton radio). Cette propriété s'applique uniquement aux boutons créés avec l'outil Bouton de la palette des outils.

Exemple

L'instruction suivante transforme l'acteur bouton Editorial en case à cocher :

```
--Lingo Dot syntax:
member("Editorial").buttonType = #checkBox

--Lingo Verbose syntax:
set the buttonType of member "Editorial" to #checkBox

// JavaScript syntax
member("Editorial").buttonType = symbol("checkBox");
```

bytesStreamed

Syntaxe

```
-- Lingo syntax
memberObjRef.bytesStreamed

// JavaScript syntax
memberObjRef.bytesStreamed;
```

Description

Propriété d'acteur Flash et Shockwave Audio ; indique le nombre d'octets de l'acteur spécifié qui ont été chargés en mémoire. La propriété `bytesStreamed` renvoie une valeur uniquement pendant la lecture de l'animation Director. Elle renvoie un nombre entier.

Cette propriété peut être testée, mais pas définie.

Exemple

Le gestionnaire suivant accepte une référence d'acteur en tant que paramètre, puis utilise la commande `stream` pour charger l'acteur en mémoire. Chaque fois qu'il transfère une partie de l'acteur en mémoire, il utilise la propriété `bytesStreamed` pour indiquer dans la fenêtre Messages le nombre d'octets transmis.

```
-- Lingo syntax
on fetchMovie(whichFlashMovie)
    repeat while member(whichFlashMovie).percentStreamed < 100
        stream(member(whichFlashMovie))
        put("Number of bytes streamed:" && member(whichFlashMovie).bytesStreamed)
    end repeat
end

// JavaScript syntax
```

```
function fetchMovie(whichFlashMovie)
    var i = member(whichFlashMovie).percentStreamed;
    while(i < 100) {
        stream(member(whichFlashMovie));
        trace( "Number of bytes streamed: " + member(whichFlashMovie).bytesStreamed);
    }
}
```

Voir aussi

bufferSize, percentStreamed (acteur), stream()

bytesStreamed (3D)

Syntaxe

```
member(whichCastMember).bytesStreamed
```

Description

Propriété 3D d'acteur ; indique la quantité qui a été chargée du fichier initial ou du dernier fichier demandé.

Exemple

L'instruction suivante indique que 325 300 octets de l'acteur Séquence ont été chargés.

```
put member("Scene").bytesStreamed
-- 325300
```

Voir aussi

streamSize (3D), state (3D)

camera

Syntaxe

```
member(whichCastMember).camera(whichCamera)
member(whichCastMember).camera[index]
member(whichCastMember).camera(whichCamera).whichCameraProperty
member(whichCastMember).camera[index].whichCameraProperty
sprite(whichSprite).camera{(index)}
sprite(whichSprite).camera{(index)}.whichCameraProperty
```

Description

Elément 3D ; objet à une position de vecteur à partir de laquelle l'univers 3D est observé.

Chaque image-objet possède une liste de caméras. Les vues des différentes caméras de la liste s'affichent au-dessus de celles des caméras en position *index* inférieures. Vous pouvez définir la propriété rect (caméra) de chaque caméra afin d'afficher plusieurs vues au sein de l'image-objet.

Les caméras sont enregistrées dans la palette des caméras de l'acteur. Utilisez les commandes `newCamera` et `deleteCamera` pour créer et supprimer les caméras d'un acteur 3D.

La propriété `camera` d'une image-objet est la première caméra de la liste des caméras de l'image-objet. La caméra référencée par `sprite(`*`quelleImageObjet`*`).camera` est la même que `sprite(`*`quelleImageObjet`*`).camera(1)`. Utilisez les commandes `addCamera` et `deleteCamera` pour construire la liste des caméras d'une image-objet 3D.

Pour obtenir la liste complète des propriétés et commandes de caméra, reportez-vous aux rubriques Utilisation de Director dans l'Aide de Director.

Exemple

L'instruction suivante affecte à l'image-objet 1 la caméra camArbre de l'acteur Picnic.

```
-- Lingo syntax
sprite(1).camera = member("Picnic").camera("TreeCam")
```

L'instruction suivante affecte à l'image-objet 1 la caméra 2 de l'acteur Picnic.

```
-- Lingo syntax
sprite(1).camera = member("Picnic").camera[2]

// JavaScript syntax
sprite(1).camera = member("Picnic").getProp("camera", 2);
```

Voir aussi

bevelDepth, overlay, modelUnderLoc, spriteSpaceToWorldSpace, fog, clearAtRender

cameraPosition

Syntaxe

```
member(whichCastMember).cameraPosition
sprite(whichSprite).cameraPosition
```

Description

Propriété 3D d'acteur et d'image-objet ; indique la position de la caméra par défaut.

La valeur par défaut de cette propriété est vector(0, 0, 250). Il s'agit de la position de la caméra par défaut dans le nouvel acteur 3D.

Exemple

L'instruction suivante indique que la position de la caméra par défaut de l'acteur LileAuxEnfants est vector(-117,5992, -78,9491, 129,0254).

```
-- Lingo syntax
member("Babyland").cameraPosition = vector(-117.5992, -78.9491, 129.0254)

// JavaScript syntax
member("Babyland").cameraPosition = vector(-117.5992, -78.9491, 129.0254);
```

Voir aussi

cameraRotation, autoCameraPosition

cameraRotation

Syntaxe

```
member(whichCastMember).cameraRotation
sprite(whichSprite).cameraRotation
```

Description

Propriété 3D d'acteur et d'image-objet ; indique la position de la caméra par défaut.

La valeur par défaut de cette propriété est vector(0, 0, 0). Il s'agit de la rotation de la caméra par défaut dans le nouvel acteur 3D.

Exemple

L'instruction suivante indique que la rotation de la caméra par défaut de l'acteur lileAuxEnfants est vector(82,6010, -38,8530, -2,4029).

```
member("babyland").cameraRotation = vector(82.6010, -38.8530, -2.4029)
```

Voir aussi

cameraPosition, autoCameraPosition

castLib

Syntaxe

```
-- Lingo syntax
_movie.castLib[castNameOrNum]

// JavaScript syntax
_movie.castLib[castNameOrNum];
```

Description

Propriété d'animation ; permet d'accéder par nom ou par index aux bibliothèques de distribution d'une animation, que cette animation soit ou non active. Lecture seule.

L'argument *nomOuNumDistribution* peut être une chaîne spécifiant le nom de l'animation à laquelle accéder ou un nombre entier indiquant le numéro de cette animation.

Cette propriété a la même fonction que la méthode de haut niveau `castLib()`, à ceci près que la méthode `castLib()` ne s'applique qu'à l'animation active.

Exemple

L'instruction suivante affiche le numéro de la distribution Boutons dans la fenêtre Messages :

```
-- Lingo syntax
put(_movie.castLib["Buttons"].number)

// JavaScript syntax
put(_movie.castLib["Buttons"].number);
```

Voir aussi

castLib(), Animation

castLibNum

Syntaxe

```
-- Lingo syntax
memberObjRef.castLibNum
```

```
// JavaScript syntax
memberObjRef.castLibNum;
```

Description

Propriété d'acteur ; détermine le numéro de la bibliothèque de distribution à laquelle un acteur appartient. Lecture seule.

Exemple

L'instruction suivante détermine le numéro de la distribution à laquelle l'acteur Jazz est affecté.

```
-- Lingo syntax
put(member("Jazz").castLibNum)

// JavaScript syntax
put(member("Jazz").castLibNum);
```

L'instruction suivante modifie l'acteur affecté à l'image-objet 5 en remplaçant sa distribution par la distribution Mercredi.

```
-- Lingo syntax
sprite(5).castLibNum = castLib("Wednesday Schedule").number

// JavaScript syntax
sprite(5).castLibNum = castLib("Wednesday Schedule").number;
```

Voir aussi

Bibliothèque de distribution, Acteur

castMemberList

Syntaxe

```
-- Lingo syntax
memberObjRef.castMemberList

// JavaScript syntax
memberObjRef.castMemberList;
```

Description

Propriété d'acteur curseur ; spécifie la liste des acteurs composant les images d'un curseur. Remplacez *quelActeurCurseur* par le nom (entre guillemets) ou le numéro d'un acteur. Vous pouvez spécifier des acteurs appartenant à différentes distributions.

Le premier acteur de la liste est la première image du curseur, le deuxième acteur est la deuxième image, et ainsi de suite.

Si vous spécifiez des acteurs impossibles à utiliser dans un curseur, ils sont ignorés et les acteurs restants sont utilisés.

Cette propriété peut être testée et définie.

Exemple

La commande suivante crée une série de quatre acteurs pour l'acteur curseur couleur animé monCurseur.

```
-- Lingo syntax
member("myCursor").castmemberList = [member(1), member(2), member(1, 2), member(2, 2)]

// JavaScript syntax
```

```
member("myCursor").castmemberList = list(member(1), member(2), member(1, 2), member(2, 2));
```

center

Syntaxe

```
member(whichCastMember).center
the center of member whichCastMember
```

Description

Propriété d'acteur ; interagit avec la propriété d'acteur `crop`.

- Lorsque la propriété `crop` présente la valeur `FALSE`, la propriété `center` n'a aucun effet.
- Lorsque `crop` et `center` présentent la valeur `TRUE`, un recadrage se produit autour du centre de l'acteur vidéo numérique.
- Lorsque `crop` présente la valeur `TRUE` et que `center` présente la valeur `FALSE`, les côtés droit et inférieur de la vidéo numérique sont rognés.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante provoque l'affichage, dans le coin supérieur gauche de l'image-objet, de l'acteur vidéo numérique Entrevue :

```
-- Lingo Dot syntax:
member("Interview").center = FALSE

-- Lingo Verbose syntax:
set the center of member "Interview" to FALSE

// JavaScript syntax
member("Interview").center = false
```

Voir aussi

`crop`, `centerRegPoint`, `regPoint`, `scale (acteur)`

centerRegPoint

Syntaxe

```
-- Lingo syntax
memberObjRef.centerRegPoint

// JavaScript syntax
memberObjRef.centerRegPoint;
```

Description

Propriété d'acteur Flash, forme vectorielle et bitmap ; centre automatiquement le point d'alignement de l'acteur lorsque vous redimensionnez l'image-objet (`TRUE`, valeur par défaut) ou repositionne le point d'alignement sur sa valeur en cours lorsque vous redimensionnez l'image-objet ou que vous définissez la propriété `defaultRect` ou la propriété `regPoint` (`FALSE`).

Cette propriété peut être testée et définie.

Exemple

Le script suivant vérifie si la propriété `centerRegPoint` d'une animation Flash présente la valeur `TRUE`. Le cas échéant, le script utilise la propriété `regPoint` pour repositionner le point d'alignement de l'image-objet dans son coin supérieur gauche. Lorsqu'il vérifie la propriété `centerRegPoint`, le script s'assure qu'il ne repositionne pas un point d'alignement précédemment défini au moyen de la propriété `regPoint`.

```
-- Lingo syntax
property spriteNum

on beginSprite me
    if sprite(spriteNum).member.centerRegPoint = TRUE then
        sprite(spriteNum).member.regPoint = point(0,0)
    end if
end

// JavaScript syntax
function beginSprite() {
    var ctrRg = sprite(this.spriteNum).member.centerRegPoint;
    if (ctrRg == 1) {
        sprite(this.spriteNum).member.regPoint = point(0,0);
    }
}
```

Voir aussi

regPoint

centerStage

Syntaxe

```
-- Lingo syntax
_movie.centerStage

// JavaScript syntax
_movie.centerStage;
```

Description

Propriété d'animation ; détermine si la scène est centrée sur le moniteur lors du chargement de l'animation (`TRUE`, valeur par défaut) ou non (`FALSE`). Lecture/écriture.

Placez l'instruction incluant cette propriété dans l'animation précédant celle qui doit être affectée.

Cette propriété est utile pour vérifier l'emplacement de la scène avant de lire une projection.

Remarque : *veuillez noter que le comportement constaté pendant la lecture d'une projection peut varier selon que vous utilisez un système Windows ou Mac. Les paramètres sélectionnés pendant la création de la projection peuvent remplacer cette propriété.*

Exemple

L'instruction suivante déplace l'animation vers une image spécifique si la scène n'est pas centrée :

```
-- Lingo syntax
if (_movie.centerStage = FALSE) then
     _movie.go("Off Center")
```

```
end if

// JavaScript syntax
if (_movie.centerStage == false) {
    _movie.go("Off Center");
}
```

L'instruction suivante inverse la valeur en cours de la propriété `centerStage` :

```
-- Lingo syntax
_movie.centerStage = not(_movie.centerStage)

// JavaScript syntax
_movie.centerStage = !(_movie.centerStage)
```

Voir aussi

fixStageSize, Animation

changeArea

Syntaxe

```
member(whichCastMember).changeArea
the changeArea of member whichCastMember
```

Description

Propriété d'acteur transition ; détermine si une transition s'applique uniquement à la zone modifiée de la scène (`TRUE`) ou à la scène entière (`FALSE`). Son effet est semblable à celui de l'option Zone modifiée seulement de la boîte de dialogue Propriétés de l'image : Transition.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante permet à l'acteur transition Vague de ne s'appliquer qu'à la zone modifiée de la scène.

```
-- Lingo Dot syntax:
member("Wave").changeArea = TRUE

-- Lingo Verbose syntax:
set the changeArea of member "Wave" to TRUE

// JavaScript syntax
member("Wave").changeArea = true;
```

channelCount

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.channelCount

// JavaScript syntax
soundChannelObjRef.channelCount;
```

Description

Propriété de piste audio ; détermine le nombre de pistes du son en cours de lecture ou en pause dans une piste audio donnée. Lecture seule.

Cette propriété est utile pour savoir si un son est mono ou stéréo.

Exemple

L'instruction suivante détermine le nombre de pistes contenues dans l'acteur son Jazz.

```
-- Lingo syntax
put(member("Jazz").channelCount)

// JavaScript syntax
put(member("Jazz").channelCount);
```

L'instruction suivante détermine le nombre de pistes contenues dans l'acteur son en cours de lecture dans la piste audio 2.

```
-- Lingo syntax
put(sound(2).channelCount)

// JavaScript syntax
put(sound(2).channelCount);
```

Voir aussi

Piste audio

chapter

Syntaxe

```
-- Lingo syntax
dvdObjRef.chapter

// JavaScript syntax
dvdObjRef.chapter;
```

Description

Propriété de DVD ; renvoie le numéro du chapitre en cours. Lecture/écriture.

Exemple

L'instruction suivante renvoie le chapitre en cours :

```
-- Lingo syntax
trace (member(1).chapter)-- 1

// JavaScript syntax
trace (member(1).chapter);// 1
```

Voir aussi

DVD

chapterCount

Syntaxe

```
-- Lingo syntax
dvdObjRef.chapterCount

// JavaScript syntax
dvdObjRef.chapterCount;
```

Description

Propriété de DVD ; renvoie le nombre de chapitres disponibles dans un titre. Lecture seule.

Exemple

L'instruction suivante renvoie le nombre de chapitres du titre en cours :

```
-- Lingo syntax
trace (member(1).chapterCount)-- 17

// JavaScript syntax
trace (member(1). chapterCount);// 17
```

Voir aussi

chapterCount(), DVD

characterSet

Syntaxe

```
-- Lingo syntax
memberObjRef.characterSet

// JavaScript syntax
memberObjRef.characterSet;
```

Description

Propriété d'acteur police ; renvoie une chaîne contenant les caractères inclus pour l'importation lors de la création de l'acteur. Si tous les caractères de la police d'origine étaient inclus, le résultat est une chaîne vide.

Exemple

L'instruction suivante affiche les caractères inclus lorsque l'acteur 11 a été créé. Les caractères inclus durant l'importation étaient des caractères numériques et romains.

```
-- Lingo syntax
put(member(11).characterSet)

// JavaScript syntax
put(member(11).characterSet);
```

Voir aussi

recordFont, bitmapSizes, originalFont

charSpacing

Syntaxe

```
-- Lingo syntax
chunkExpression.charSpacing

// JavaScript syntax
chunkExpression.charSpacing;
```

Description

Propriété d'acteur texte ; permet de spécifier tout espacement supplémentaire à appliquer à chaque lettre de la partie *expressionSousChaîne* de l'acteur texte.

Une valeur inférieure à 0 indique un espacement plus réduit entre les lettres. Une valeur supérieure à 0 indique un plus grand espacement entre les lettres.

La valeur par défaut est 0, ce qui active l'espacement par défaut entre les lettres.

Exemple

Le gestionnaire suivant augmente l'espacement des caractères actuel du troisième au cinquième mot de l'acteur texte monTitre par une valeur de 2 :

```
--Lingo syntax
on myCharSpacer
    mySpaceValue = member("myCaption").word[3..5].charSpacing
    member("myCaption").word[3..5].charSpacing = (mySpaceValue + 2)
end

// JavaScript syntax
function myCharSpacer() {
    var i = 3;
    while (i < 6) {
        var mySpaceValue = member("myCaption").getPropRef("word", i).charSpacing;
        member("myCaption").getPropRef("word", i).charSpacing = (mySpaceValue + 2);
    }
}
```

checkMark

Syntaxe

```
the checkMark of menuItem whichItem of menu whichMenu
```

Description

Propriété d'élément de menu ; détermine si l'élément de menu personnalisé spécifié s'affiche avec une coche (TRUE) ou non (FALSE, valeur par défaut).

La valeur *quelElément* peut être un nom ou un numéro d'élément de menu. La valeur *quelMenu* peut être un nom ou un numéro de menu.

Cette propriété peut être testée et définie.

Remarque : *les menus ne sont pas disponibles dans Shockwave Player.*

Exemple

Le gestionnaire suivant désactive tout élément activé du menu personnalisé spécifié par l'argument `theMenu`. Par exemple, `unCheck ("Format")` désactive tous les éléments du menu Format.

```
-- Lingo syntax
on unCheck theMenu
    set n = the number of menuItems of menu theMenu
    repeat with i = 1 to n
        set the checkMark of menuItem i of menu theMenu to FALSE
    end repeat
end unCheck

// JavaScript syntax
function unCheck (theMenu) {
    var n = _menuBar.menu[theMenu].item.count; //the number of menuItems of menu theMenu
    for( i = 1 ; i <= n ; i++ )
        _menuBar.menu[theMenu].item[i].checkMark = false;
}
```

Voir aussi

`installMenu`, `enabled`, `name (propriété d'élément de menu)`, `number (éléments de menu)`, `script`, `menu`

child (3D)

Syntaxe

```
member(whichCastmember).model(whichParentNode).child(whichChildNodeName)
member(whichCastmember).model(whichParentNode).child[index]
```

Description

Propriété 3D de modèle, de groupe, de lumière et de caméra ; renvoie le nœud enfant nommé *`quelNœudEnfant`* ou au niveau de l'index spécifié dans la liste d'enfants du nœud parent. Un nœud est un modèle, un groupe, une caméra ou une lumière.

La transformation d'un nœud est relative au parent. Si vous modifiez la position du parent, ses enfants se déplacent avec lui et leur position relative au parent est conservée. De même, la modification des propriétés de rotation et d'échelle du parent est également reflétée dans ses enfants.

Utilisez la méthode `addChild` du nœud parent ou définissez la propriété `parent` du nœud enfant pour l'ajouter à la liste d'enfants du parent. Alors qu'un enfant ne peut avoir qu'un parent, un parent peut avoir un nombre illimité d'enfants. Un enfant peut lui-même avoir des enfants.

Exemple

L'instruction suivante indique que le second enfant du modèle Véhicule est le modèle Pneu.

```
-- Lingo syntax
put member("3D").model("Car").child[2]
-- model("Tire")

// JavaScript syntax
put ( member("3D").getProp("model", 1).child[2] );
// model("Tire")
```

Voir aussi

addChild, parent

child (XML)

Syntaxe

```
XMLnode.child[ childNumber ]
```

Description

Propriété XML ; fait référence au nœud enfant spécifié de la structure imbriquée d'un document XML analysé.

Exemple

Avec le code XML suivant :

```
<?xml version="1.0"?>
    <e1>
        <tagName attr1="val1" attr2="val2"/>
        <e2> element 2</e2>
        <e3>element 3</e3>
        here is some text
    </e1>
```

L'instruction Lingo suivante renvoie le nom du premier nœud enfant du code XML précédent :

```
put gParserObject.child[1].name
-- "e1"
```

chunkSize

Syntaxe

```
member(whichCastMember).chunkSize
the chunkSize of member whichCastMember
```

Description

Propriété d'acteur transition ; détermine la taille des blocs de la transition (entre 1 et 128 pixels) et équivaut à définir le curseur de fluidité dans la boîte de dialogue Propriétés de l'image : Transition. Plus la taille des blocs est petite, plus la transition est fluide.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante paramètre la taille des blocs de l'acteur transition Brouillard sur 4 pixels :

Syntaxe à point :

```
member("Fog").chunkSize = 4
```

Syntaxe verbose :

```
set the chunkSize of member "Fog" to 4
```

clearAtRender

Syntaxe

```
member(whichCastmember).camera(whichCamera).colorBuffer.clearAtRender
sprite(whichSprite).camera{(index)}.colorBuffer.clearAtRender
```

Description

Propriété 3D ; indique si le tampon des couleurs est vidé après chaque image. La valeur `FALSE`, qui signifie que le tampon n'est pas vidé, produit un effet similaire à des traces d'encre. La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante empêche Director d'effacer les images précédentes de la vue de la caméra. Les modèles en mouvement laissent une trace sur la scène.

```
-- Lingo syntax
sprite(1).camera.colorBuffer.clearAtRender = 0

// JavaScript syntax
sprite(1).camera.getPropRef("colorBuffer").clearAtRender = 0;
```

Voir aussi

`clearValue`

clearValue

Syntaxe

```
member(whichCastmember).camera(whichCamera).colorBuffer.clearValue
sprite(whichSprite).camera{(index)}.colorBuffer.clearValue
```

Description

Propriété 3D ; spécifie la couleur utilisée pour vider le tampon des couleurs si `colorBuffer.clearAtRender` présente la valeur `TRUE`. Le paramétrage par défaut de cette propriété est `rgb(0, 0, 0)`.

Exemple

L'instruction suivante définit la propriété `clearValue` de la caméra sur la valeur rgb(255, 0, 0). L'espace de l'univers 3D qui n'est pas occupé par les modèles apparaîtra en rouge.

```
-- Lingo syntax
sprite(1).camera.colorBuffer.clearValue= rgb(255, 0, 0)

// JavaScript syntax
sprite(1).camera.getPropRef("colorBuffer").clearValue= color(255, 0, 0);
```

Voir aussi

`clearAtRender`

clickLoc

Syntaxe

```
-- Lingo syntax
_mouse.clickLoc

// JavaScript syntax
_mouse.clickLoc;
```

Description

Propriété de souris ; identifie le dernier endroit de l'écran sur lequel l'utilisateur a cliqué avec la souris. Lecture seule.

Exemple

Le gestionnaire `mouseDown` suivant affiche l'emplacement du dernier clic de la souris :

```
-- Lingo syntax
on mouseDown
    put(_mouse.clickLoc)
end mouseDown

// JavaScript syntax
function mouseDown() {
    put(_mouse.clickLoc);
}
```

Si l'utilisateur a cliqué sur un emplacement de la scène situé à 50 pixels de son bord gauche et à 100 pixels de son bord supérieur, la fenêtre Messages affiche :

```
point(50, 100)
```

Voir aussi

`clickOn`, `Souris`

clickMode

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.clickMode

// JavaScript syntax
memberOrSpriteObjRef.clickMode;
```

Description

Propriété d'image-objet et d'acteur Flash ; contrôle le moment où l'image-objet de l'animation Flash détecte des événements de type clic de souris (`mouseUp` et `mouseDown`), ainsi que des survols (`mouseEnter`, `mouseWithin` et `mouseLeave`). La propriété `clickMode` peut prendre l'une des valeurs suivantes :

- `#boundingBox` : détecte les événements de type clic de souris à un endroit quelconque du rectangle de délimitation de l'image-objet, ainsi que les survols aux limites de l'image-objet.
- `#opaque` (valeur par défaut) : détecte les événements de type clic de souris uniquement lorsque le pointeur est positionné sur une portion opaque de l'image-objet et détecte les survols aux limites des portions opaques de l'image-objet si l'effet d'encre de cette dernière est configuré sur Fond transparent. Si l'effet d'encre de l'image-objet est défini sur une autre valeur, ce paramètre produit le même effet que `#boundingBox`.

• `#object` : Détecte les événements de type clic de souris lorsque le pointeur de la souris est positionné sur une zone remplie (n'appartenant pas à l'arrière-plan) de l'image-objet et détecte les survols aux limites de toute zone remplie. Ce paramètre fonctionne quel que soit l'effet d'encre de l'image-objet.

Cette propriété peut être testée et définie.

Exemple

Le script suivant vérifie si l'image-objet spécifiée avec l'effet d'encre Fond transparent est définie pour un affichage au premier plan sur la scène. Si tel n'est pas le cas, la propriété `clickMode` prend la valeur `#opaque`. Dans les autres cas (puisque les effets d'encre sont ignorés pour les images-objets d'animation Flash affichées au premier plan sur la scène), la propriété `clickMode` de l'image-objet prend la valeur `#boundingBox`.

```
-- Lingo syntax
property spriteNum

on beginSprite me
    if sprite(spriteNum).directToStage = FALSE then
        sprite(spriteNum).clickMode = #opaque
    else
        sprite(spriteNum).clickMode = #boundingBox
    end if
end

// JavaScript syntax
function beginSprite(me){
    var dts = sprite(this.spriteNum).directToStage;
    if (dts == 0) {
        sprite(this.spriteNum).clickMode = symbol("opaque");
    } else {
        sprite(this.spriteNum).clickMode = symbol("boundingBox");
    }
}
```

clickOn

Syntaxe

```
-- Lingo syntax
_mouse.clickOn

// JavaScript syntax
_mouse.clickOn;
```

Description

Propriété de souris ; renvoie la dernière image-objet active sur laquelle l'utilisateur a cliqué. Lecture seule.

Une image-objet active est une image-objet à laquelle un script d'image-objet ou d'acteur est associé.

Lorsque l'utilisateur clique sur la scène, `clickOn` renvoie 0. Pour faire en sorte que la propriété `clickOn` détecte un clic de l'utilisateur sur une image-objet à laquelle aucun script n'est associé, vous devez affecter un script d'événement de souris à cette image-objet. Par exemple :

```
-- Lingo syntax
on mouseUp me
    ...
end
```

La propriété `clickOn` détecte l'utilisation des boutons, cases à cocher et boutons radio, même si aucun script ne leur est associé.

La propriété `clickOn` peut être testée au sein d'une boucle. Toutefois, ni `clickOn` ni `clickLoc` ne change de valeur lors de l'exécution du gestionnaire. La valeur que vous obtenez est la valeur précédant le démarrage du gestionnaire.

Exemple

L'instruction suivante vérifie si l'image-objet 7 est la dernière image-objet active sur laquelle l'utilisateur a cliqué :

```
-- Lingo syntax
if (_mouse.clickOn = 7) then
    _player.alert("Sorry, try again.")
end if

// JavaScript syntax
if (_mouse.clickOn == 7) {
    _player.alert("Sorry, try again.");
}
```

L'instruction suivante affecte une couleur aléatoire à la propriété `foreColor` de la dernière image-objet active sur laquelle l'utilisateur a cliqué :

```
-- Lingo syntax
sprite(_mouse.clickOn).foreColor = (random(255) - 1)

// JavaScript syntax
sprite(_mouse.clickOn).foreColor = (random(255) - 1);
```

Voir aussi

`clickLoc`, `Souris`

closed

Syntaxe

```
-- Lingo syntax
memberObjRef.closed

// JavaScript syntax
memberObjRef.closed;
```

Description

Propriété d'acteur forme vectorielle ; indique si les extrémités du contour sont ouvertes ou fermées.

Les formes vectorielles doivent être fermées pour leur remplissage.

La valeur peut être :

- `TRUE` : les extrémités sont fermées.
- `FALSE` : les extrémités sont ouvertes.

closedCaptions

Syntaxe

```
-- Lingo syntax
dvdObjRef.closedCaptions

// JavaScript syntax
dvdObjRef.closedCaptions;
```

Description

Propriété de DVD. Détermine si l'insertion de sous-titres codés est activée (`TRUE`) ou bien désactivée ou impossible à activer (`FALSE`). Propriété actuellement non prise en charge sur Mac. Lecture/écriture.

Exemple

Les instructions suivantes tentent de définir la propriété closedCaptions sur TRUE et reçoivent un message d'alerte si l'activation échoue :

```
-- Lingo syntax
member(3).closedCaptions = TRUE
if (member(3).closedCaptions = FALSE) then
    _player.alert("Closed captions cannot be enabled.")
end if

// JavaScript syntax
member(3).closedCaptions = true
if (member(3).closedCaptions == false) {
    _player.alert("Closed captions cannot be enabled.");
}
```

Propriété actuellement non prise en charge sur Mac.

Voir aussi

DVD

collision (modificateur)

Syntaxe

```
member(whichCastmember).model(whichModel).collision.collisionModifierProperty
```

Description

Modificateur 3D ; gère la détection et la résolution des collisions. L'ajout du modificateur `collision` à un modèle à l'aide de la commande `addModifier` permet d'accéder aux propriétés suivantes du modificateur `collision` :

enabled (collision) indique si des collisions avec le modèle sont détectées.

resolve indique si les collisions avec le modèle sont résolues.

immovable indique si un modèle peut être déplacé d'une image à l'autre.

mode (collision) indique la géométrie utilisée pour la détection des collisions.

Remarque : *pour plus d'informations, consultez les entrées des différentes propriétés.*

Le modificateur de collision génère les événements suivants. Pour plus d'informations sur les événements de collision, reportez-vous à l'entrée `registerForEvent()`.

Un événement #collideAny est généré lorsqu'une collision survient entre des modèles auxquels le modificateur collision a été associé.

Un événement #collideWith est généré lorsqu'une collision survient avec un modèle spécifique auquel le modificateur collision a été associé.

L'objet collisionData est envoyé comme argument avec les événements #collideAny et #collideWith. Pour plus d'informations sur ses propriétés, reportez-vous à l'entrée collisionData.

Voir aussi

addModifier, removeModifier, modifiers

collisionData

Syntaxe

```
on myHandlerName me, collisionData
```

Description

Objet de données 3D ; envoyé comme argument avec les événements #collideWith et #collideAny au gestionnaire spécifié dans les commandes registerForEvent, registerScript et setCollisionCallback. L'objet collisionData présente les propriétés suivantes :

modelA est l'un des modèles impliqués dans la collision.

modelB est l'autre modèle impliqué dans la collision.

pointOfContact est la position de la collision dans l'univers.

collisionNormal est la direction de la collision.

Exemple

L'exemple suivant est constitué de trois parties. La première partie constitue la première ligne de code, qui enregistre le gestionnaire #putDetailspour l'événement #collideAny. La deuxième partie correspond au gestionnaire #putDetail. Lorsque deux modèles de l'acteur maSéquence entrent en collision, le gestionnaire #putDetail est appelé et l'argument collisionData lui est envoyé. Ce gestionnaire affiche les quatre propriétés de l'objet collisionData dans la fenêtre Messages. La troisième partie de l'exemple affiche les résultats de la fenêtre Messages. Les deux premières lignes indiquent que les modèles impliqués dans la collision étaient balleVerte, modèle A, et balleJaune, modèle B. La troisième ligne indique le point de contact des deux modèles. La dernière ligne indique la direction de la collision.

```
-- Lingo syntax

member("MyScene").registerForEvent(#collideAny, #putDetails, 0)

on putDetails me, collisionData
    put collisionData.modelA
    put collisionData.modelB
    put collisionData.pointOfContact
    put collisionData.collisionNormal
end
-- model("GreenBall")
-- model("YellowBall")
-- vector( 24.800, 0.000, 0.000 )
-- vector( -1.000, 0.000, 0.000 )
```

```
// JavaScript syntax

member("MyScene").registerForEvent(symbol("collideAny"),symbol("putDetails"), 0);

function putDetails (me, collisionData)
{
    put (collisionData.modelA);
    put (collisionData.modelB);
    put (collisionData.pointOfContact);
    put (collisionData.collisionNormal);
}
// model("GreenBall")
// model("YellowBall")
// vector( 24.800, 0.000, 0.000 )
// vector( -1.000, 0.000, 0.000 )
```

Voir aussi

Propriétés `collisionData : modelA, modelB, pointOfContact, collisionNormal`

Méthodes collisionData : `resolveA`, `resolveB`, `collision (modificateur)`

collisionNormal

Syntaxe

`collisionData.collisionNormal`

Description

Propriété 3D `collisionData` ; vecteur indiquant la direction de la collision.

L'objet `collisionData` est envoyé comme argument avec les événements `#collideWith` et `#collideAny` au gestionnaire spécifié dans les commandes `registerForEvent`, `registerScript` et `setCollisionCallback`.

Les événements `#collideWith` et `#collideAny` sont envoyés en cas de collision entre deux modèles associés à des modificateur de collision. La propriété `resolve` des modificateurs des modèles doit présenter la valeur `TRUE`.

Cette propriété peut être testée, mais pas définie.

Exemple

L'exemple suivant est constitué de deux parties. La première partie constitue la première ligne de code, qui enregistre le gestionnaire #explose pour l'événement `#collideAny`. La seconde partie correspond au gestionnaire `#explode`. Lorsque deux modèles de l'acteur maSéquence entrent en collision, le gestionnaire `#explode` est appelé et l'argument `collisionData` lui est envoyé. Les dix premières lignes du gestionnaire `#explose` créent la ressource de modèle sourceDétincelles et en définissent les propriétés. Cette ressource de modèle est une simple explosion de particules. La dixième ligne définit la direction de l'explosion sur la valeur `collisionNormal`, qui correspond à la direction de la collision. La onzième ligne du gestionnaire crée un modèle modèleDétincelles à l'aide de la ressource de modèle sourceDétincelles. La dernière ligne du gestionnaire définit la position de modèleDétincelles à l'emplacement de la collision. L'effet obtenu est une collision qui entraîne une explosion d'étincelles qui volent dans la direction de la collision, à partir du point de contact.

```
-- Lingo syntax
member("MyScene").registerForEvent(#collideAny, #explode, 0)
on explode me, collisionData
    nmr = member("MyScene").newModelResource("SparkSource", #particle)
```

```
    nmr.emitter.mode = #burst
    nmr.emitter.loop = 0
    nmr.emitter.minSpeed = 30
    nmr.emitter.maxSpeed = 50
    nmr.emitter.angle = 45
    nmr.colorRange.start = rgb(0, 0, 255)
    nmr.colorRange.end = rgb(255, 0, 0)
    nmr.lifetime = 5000
    nmr.emitter.direction = vector(0,0,-1)
    nm = member("MyScene").newModel("SparksModel", nmr)
    nm.transform.position = collisionData.pointOfContact
    nm.pointAt(collisionData.pointOfContact + collisionData.collisionNormal)
end

// JavaScript syntax
member("MyScene").registerForEvent(symbol("collideAny"), symbol("explode"), 0);
function explode (me, collisionData) {
    nmr = member("MyScene").newModelResource("SparkSource", symbol("particle"));
    nmr.getPropRef("emitter").mode = symbol("burst");
    nmr.getPropRef("emitter").loop = 0;
    nmr.getPropRef("emitter").minSpeed = 30;
    nmr.getPropRef("emitter").maxSpeed = 50;
    nmr.getPropRef("emitter").angle = 45;
    nmr.getPropRef("colorRange").start = color(0, 0, 255);
    nmr.getPropRef("colorRange").end = color(255, 0, 0);
    nmr.lifetime = 5000;
    nmr.getPropRef("emitter").direction = vector(0,0,-1);
    nm = member("MyScene").newModel("SparksModel", nmr);
    nm.transform.position = collisionData.pointOfContact;
    nm.pointAt(collisionData.pointOfContact + collisionData.collisionNormal);
}
```

Voir aussi

pointOfContact, modelA, modelB, resolveA, resolveB, collision (modificateur)

color()

Syntaxe

```
color(#rgb, redValue, greenValue, blueValue)
color(#paletteIndex, paletteIndexNumber)
rgb(rgbHexString)
rgb(redValue, greenValue, blueValue)
paletteIndex(paletteIndexNumber)
```

Description

Fonction et type de données ; détermine la couleur d'un objet sous la forme de valeurs RVB ou de valeurs d'un index de palette de 8 bits. Ces valeurs sont identiques à celles utilisées dans la propriété d'acteur `color` et dans la propriété d'image-objet `color`, dans la propriété d'acteur `bgColor` et dans la propriété d'image-objet `bgColor`, ainsi que dans la propriété de scène `bgColor`.

La fonction `color` permet de manipuler les valeurs de couleur de 24 bits ou de 8 bits et de les appliquer aux acteurs, aux images-objets et à la scène.

Dans le cas des valeurs rvb, chaque composant de couleur possède une gamme de valeurs comprises entre 0 et 255, toutes les autres valeurs étant tronquées. Dans le cas des types `paletteIndex`, un entier compris entre 0 et 255 est utilisé pour indiquer le numéro d'index dans la palette en cours et toutes les autres valeurs sont tronquées.

Exemple

L'instruction suivante effectue une opération mathématique :

```
palColorObj = paletteIndex(20)
put palColorObj
-- paletteIndex(20)
put palColorObj / 2
-- paletteIndex(10)
```

L'instruction suivante convertit un type de couleur en un autre type :

```
newColorObj = color(#rgb, 155, 0, 75)
put newColorObj
-- rgb(155, 0, 75)
newColorObj.colorType = #paletteIndex
put newColorObj
-- paletteIndex(106)
```

L'instruction suivante obtient la représentation hexadécimale d'une couleur, quel que soit son type :

```
someColorObj = color(#paletteIndex, 32)
put someColorObj.hexString()
-- "#FF0099"
```

L'instruction suivante détermine les composants RVB individuels et la valeur paletteIndex d'une couleur, quel que soit son type :

```
newColorObj = color(#rgb, 155, 0, 75)
put newColorObj.green
-- 0
put newColorObj.paletteIndex
-- 106
newColorObj.green = 100
put newColorObj.paletteIndex
-- 94
put newColorObj
-- rgb(155, 100, 75)
newColorObj.paletteIndex = 45
put newColorObj
-- paletteIndex(45)
```

L'instruction suivante modifie la couleur des caractères 4 à 7 de l'acteur texte mesCitations :

```
member("myQuotes").char[4..7].color = rgb(200, 150, 75)
```

L'instruction Lingo suivante affiche la couleur de l'image-objet 6 dans la fenêtre Messages, puis définit cette couleur sur une nouvelle valeur RVB :

```
put sprite(6).color
-- rgb( 255, 204, 102 )
sprite(6).color = rgb(122, 98, 210)
```

Remarque : *la définition de la valeur* `paletteIndex` *d'un type de couleur RVB remplace* `colorType` *par* `paletteIndex`*. La définition du type de couleur RVB d'une couleur* `paletteIndex` *définit sa valeur* `colorType` *sur RVB.*

Voir aussi

`bgColor (fenêtre)`

color (brouillard)

Syntaxe

```
member(whichCastmember).camera(whichCamera).fog.color
sprite(whichSprite).camera{(index)}.fog.color
```

Description

Propriété 3D ; indique la couleur introduite dans la scène par la caméra lorsque la propriété `fog.enabled` de cette dernière présente la valeur `TRUE`.

Le paramètre par défaut de cette propriété est `rgb(0, 0, 0)`.

Exemple

L'instruction suivante attribue à la couleur du brouillard de la caméra vueDeLaBaie la valeur `rgb(255, 0, 0)`. Si la propriété `fog.enabled` de la caméra présente la valeur `TRUE`, les modèles dans le brouillard prennent une teinte rouge.

```
member("MyYard").camera("BayView").fog.color = rgb(255, 0, 0)
```

Voir aussi

`fog`

color (lumière)

Syntaxe

```
member(whichCastmember).light(whichLight).color
```

Description

Propriété 3D de lumière ; indique la valeur rvb de la lumière.

La valeur par défaut de cette propriété est `rgb(191,191,191)`.

Exemple

L'instruction suivante attribue à la couleur de la lumière lumièreDeLaPièce la valeur `rgb(255, 0, 255)`.

```
member("Room").light("RoomLight").color = rgb(255,0,255)
```

Voir aussi

`fog`

colorBufferDepth

Syntaxe

```
getRendererServices().colorBufferDepth
```

Description

Propriété 3D `rendererServices` ; indique la précision des couleurs du tampon de sortie du matériel du système de l'utilisateur. Sa valeur est de 16 ou de 32, selon le matériel de l'utilisateur.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante définit la valeur `colorBufferDepth` de la carte vidéo de l'utilisateur sur 32.

```
-- Lingo syntax
put getRendererServices().colorBufferDepth
-- 32

// JavaScript syntax
put (getRendererServices().colorBufferDepth);
// 32
```

Voir aussi

`getRendererServices()`, `getHardwareInfo()`, `depthBufferDepth`

colorDepth

Syntaxe

```
-- Lingo syntax
_system.colorDepth

// JavaScript syntax
_system.colorDepth;
```

Description

Propriété système ; paramètre le codage des couleurs du moniteur. Lecture/écriture.

- Sous Windows, cette propriété vous permet de vérifier et définir le codage des couleurs du moniteur. La définition de la propriété `colorDepth` se révèle parfois impossible pour certaines combinaisons de cartes vidéo et de pilotes. Vérifiez toujours que le codage des couleurs a bien été changé après votre essai.
- Sur le Mac, cette propriété vous permet de vérifier le codage des couleurs des différents moniteurs et de le modifier si nécessaire.

Les valeurs possibles sont :

`1`	Noir et blanc
`2`	4 couleurs
`4`	16 couleurs
`8`	256 couleurs
`16`	32 768 ou 65 536 couleurs
`32`	16 777 216 couleurs

Si vous tentez de modifier le codage des couleurs avec une valeur non prise en charge par le moniteur, le codage reste inchangé.

Dans le cas des ordinateurs disposant de plusieurs moniteurs, la propriété `colorDepth` se réfère au moniteur sur lequel la scène est affichée. Si la scène s'étend sur plusieurs moniteurs, la propriété `colorDepth` indique le codage maximal de ces moniteurs ; `colorDepth` tente d'affecter à tous ces moniteurs le codage spécifié.

Exemple

L'instruction suivante indique à Director de n'ouvrir l'animation Couleurs complètes que si le codage des couleurs du moniteur est de 256 couleurs :

```
-- Lingo syntax
if (_system.colorDepth = 8) then
    window("Full color").open()
end if

// JavaScript syntax
if (_system.colorDepth == 8) {
    window("Full color").open()
}
```

Le gestionnaire suivant tente de modifier le codage des couleurs et, s'il échoue, affiche un message d'alerte :

```
-- Lingo syntax
on tryToSetColorDepth(desiredDepth)
    _system.colorDepth = desiredDepth
    if (_system.colorDepth = desiredDepth) then
        return true
    else
        _player.alert("Please change your system to" && desiredDepth && "color depth and
reboot.")
        return false
    end if
end

// JavaScript syntax
function tryToSetColorDepth(desiredDepth) {
    _system.colorDepth = desiredDepth;
    if (_system.colorDepth == desiredDepth) {
        return true;
    }
    else {
        _player.alert("Please change your system to " + desiredDepth + " color depth and
reboot.");
        return false;
    }
}
```

Voir aussi

Système

colorList

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).colorList
member(whichCastmember).modelResource(whichModelResource).colorList[index]
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].colorList
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].colorList[index]
```

Description

Propriété 3D ; permet d'obtenir ou de définir chaque couleur utilisée dans une maille. Cette commande ne peut être utilisée que pour les ressources de modèle de type #mesh. Chaque couleur peut être partagée par plusieurs sommets (faces) de la maille. Vous avez également la possibilité de spécifier les coordonnées de texture des faces de la maille et d'appliquer un matériau aux modèles qui utilisent cette ressource de modèle.

Cette commande doit être définie avec une liste comportant le même nombre de valeurs de couleur Lingo que dans l'appel `newMesh`.

Exemple

L'instruction suivante indique que la troisième couleur de la liste `colorList` de la ressource de modèle Maille2 est rgb(255, 0, 0).

```
-- Lingo syntax
put member("shapes").modelResource("mesh2").colorlist[3]
-- rgb(255,0,0)

// JavaScript syntax
put ( member("shapes").getProp("modelResource" , 1).colorlist[3] );
// color(255,0,0)
```

Voir aussi

`face[ ]`, `colors`

colorRange

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).colorRange.start
member(whichCastmember).modelResource(whichModelResource).colorRange.end
```

Description

Propriétés 3D de ressource de modèle `#particle` ; indiquent les couleurs de début et de fin des particules d'un système de particules.

La propriété `start` définit la couleur des particules lorsqu'elles sont créées. La propriété `end` définit la couleur des particules à la fin de leur vie. La couleur de chaque particule évolue progressivement de la valeur `start` à la valeur `end`.

Par défaut, les propriétés `start` et `end` présentent la valeur `rgb(255, 255, 255)`.

Exemple

L'instruction suivante définit les propriétés `colorRange` de la ressource de modèle systèmeThermique. La première ligne attribue la valeur de début `rgb(255, 0, 0)` et la seconde ligne définit la valeur de fin `rgb(0, 0, 255)`. Cette instruction fait progressivement évoluer les particules de systèmeThermique du rouge au bleu, entre le moment où elles apparaissent et la fin de leur vie.

```
member(8,2).modelResource("ThermoSystem").colorRange.start =rgb(255,0,0)
member(8,2).modelResource("ThermoSystem").colorRange.end = rgb(0,0,255)
```

Voir aussi

`emitter`, `blendRange`, `sizeRange`

colors

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).face[faceIndex].colors
```

Description

Propriété 3D de face ; liste linéaire de trois nombres entiers indiquant les positions d'index de la liste des couleurs de la ressource de modèle à utiliser pour les trois sommets de la face. La liste des couleurs est une liste linéaire de valeurs `rvb`.

La propriété `colors` n'est utilisée qu'avec les ressources de modèle de type `#mesh`.

Vous devez utiliser la commande `build()` de la ressource de modèle après avoir défini cette propriété ; dans le cas contraire, les modifications ne prennent pas effet.

Exemple

L'exemple suivant crée une ressource de modèle de type #mesh, en spécifie les propriétés, puis l'utilise pour créer un nouveau modèle.

La ligne 1 utilise la commande newMesh() pour créer une ressource de modèle #mesh nommée Triangle, qui consiste en une face, trois sommets et un maximum de trois couleurs. Le nombre de normales et de coordonnées de textures n'est pas défini.

La ligne 2 définit la propriété `vertexList` sur une liste de trois vecteurs.

La ligne 3 affecte les vecteurs de la propriété `vertexList` aux sommets de la première face de Triangle.

La ligne 4 attribue à la liste des couleurs trois valeurs `rvb`.

La ligne 5 affecte les couleurs à la première face de Triangle. La troisième couleur de la liste est appliquée au premier sommet de Triangle, la deuxième couleur au deuxième sommet, et la première couleur au troisième sommet. Les couleurs sont étalées sur la première face de Triangle en dégradés.

La ligne 6 crée les normales de Triangle avec la commande `generateNormals()`.

La ligne 7 utilise la commande `build()` pour construire la maille.

La ligne 8 crée un nouveau modèle nommé triModèle, qui utilise la nouvelle maille.

```
nm = member("Shapes").newMesh("Triangle",1,3,0,3,0)
nm.vertexList = [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)]
nm.face[1].vertices = [1,2,3]
nm.colorList = [rgb(255,255,0), rgb(0, 255, 0), rgb(0,0,255)]
nm.face[1].colors = [3,2,1]
nm.generateNormals(#smooth)
nm.build()
nm = member("Shapes").newModel("TriModel", nm)
```

Voir aussi

face, vertices, vertices, flat

colorSteps

Syntaxe

```
member(whichCastmember).model(whichModel).toon.colorSteps
```

```
member(whichCastmember).model(whichModel).shader.colorSteps
member(whichCastmember).shader(whichShader).colorSteps
```

Description

Propriété 3D de modificateur `toon` et de matériau `painter` ; nombre maximal de couleurs disponibles pour le modificateur `toon` ou pour le matériau `painter`. Les valeurs possibles de cette propriété sont 2, 4, 8 ou 16. Si vous attribuez une autre valeur à `colorSteps`, cette valeur est arrondie à l'une de ces valeurs autorisées.

La valeur par défaut est 2.

Exemple

L'instruction suivante limite le nombre de couleurs disponibles pour le modificateur toon du modèle Théière à 8. Le rendu de la théière est composé de huit couleurs au maximum.

```
-- Lingo syntax
member("shapes").model("Teapot").toon.colorSteps = 8

// JavaScript syntax
member("shapes").getProp("model", 1).getPropRef("toon").colorSteps = 8;
```

Voir aussi

highlightPercentage, shadowPercentage

commandDown

Syntaxe

```
-- Lingo syntax
_key.commandDown

// JavaScript syntax
_key.commandDown;
```

Description

Propriété de touche ; détermine si l'utilisateur appuie sur la touche Ctrl (Windows) ou Cmd (Mac). Lecture seule.

Cette propriété renvoie la valeur `TRUE` si la touche Ctrl ou Cmd est enfoncée ; dans le cas contraire, elle renvoie la valeur `FALSE`.

Vous pouvez utiliser la propriété `commandDown` avec la propriété `key` pour déterminer si l'utilisateur appuie sur la touche Ctrl ou Cmd en combinaison avec une autre touche. Cela vous permet de créer des gestionnaires exécutés lorsque l'utilisateur appuie sur les touches spécifiées en combinaison avec la touche Cmd ou Ctrl.

Les raccourcis clavier utilisant la touche Ctrl ou Cmd qui sont associés aux menus auteur de Director ont la priorité durant la lecture de l'animation, sauf si vous avez installé des menus personnalisés en syntaxe Lingo ou JavaScript ou que vous êtes en train de lire une projection.

Exemple

Les instructions suivantes mettent une projection sur pause lorsque la tête de lecture passe sur une image et que l'utilisateur appuie sur Contrôle+A (Windows) ou sur Commande+A (Mac).

```
-- Lingo syntax
on enterFrame
    if (_key.commandDown and _key.key = "a") then
        _movie.go(_movie.frame)
```

```
    end if
end

// JavaScript syntax
function enterFrame() {
    if (_key.commandDown && _key.key == "a") {
        _movie.go(_movie.frame);
    }
}
```

Voir aussi

Touche, key

comments

Syntaxe

```
-- Lingo syntax
memberObjRef.comments

// JavaScript syntax
memberObjRef.comments;
```

Description

Propriété d'acteur ; fournit un emplacement de stockage des commentaires que vous souhaitez conserver à propos d'un acteur ou de toute autre chaîne à associer à cet acteur. Lecture/écriture.

Cette propriété est également définissable dans l'onglet Acteur de l'Inspecteur des propriétés.

Exemple

L'instruction suivante définit les commentaires de l'acteur Fond comme La permission d'utiliser ce graphique doit être obtenue.

```
-- Lingo syntax
member("Backdrop").comments = "Still need to license this artwork"

// JavaScript syntax
member("Backdrop").comments = "Still need to license this artwork";
```

Voir aussi

Acteur

compressed

Syntaxe

```
member(whichCastmember).texture(whichTexture).compressed
```

Description

Propriété 3D de texture ; indique si l'acteur source de la texture est compressé (TRUE) ou non (FALSE). La valeur de la propriété compressed passe automatiquement de TRUE à FALSE lorsque la texture est nécessaire pour le rendu. Cette propriété peut être définie sur FALSE pour décompresser la texture en avance. Elle peut recevoir la valeur TRUE pour supprimer la représentation décompressée de la mémoire. Les acteurs utilisés pour les textures ne sont pas compressés si la valeur est TRUE (indépendamment de la compression standard utilisée pour les acteurs bitmap lors de l'enregistrement d'une animation Director). La valeur par défaut de cette propriété est TRUE.

Exemple

L'instruction suivante définit la propriété compressed de la texture placagePluton sur la valeur TRUE.

```
-- Lingo syntax
member("scene").texture("Plutomap").compressed = TRUE

// Java Script
member("scene").getProp("texture",1).compressed = true;
```

Voir aussi

texture

constraint

Syntaxe

```
-- Lingo syntax
spriteObjRef.constraint

// JavaScript syntax
spriteObjRef.constraint;
```

Description

Propriété d'image-objet ; détermine si le point d'alignement d'une image-objet est limité au rectangle de délimitation d'une autre image-objet (1 ou TRUE) ou non (FALSE, valeur par défaut). Lecture/écriture.

La propriété constraint est utile pour limiter le déplacement d'une image-objet mobile au rectangle de délimitation d'une autre image-objet. Elle permet de simuler une piste pour un curseur ou de limiter l'endroit de l'écran où l'utilisateur peut faire glisser un objet dans un jeu.

La propriété d'image-objet constraint s'applique aux images-objets mobiles, ainsi qu'aux propriétés locH et locV. Le point de contrôle d'une image-objet mobile ne peut pas être déplacé en dehors du rectangle de délimitation de l'image-objet associée. (Dans le cas d'une image-objet bitmap, le point de contrôle est le point d'alignement. Dans le cas d'une image-objet forme, le point de contrôle est l'angle supérieur gauche délimitant la forme.) Lorsqu'une contrainte est définie pour une image-objet, les limites définies ont priorité sur les valeurs des propriétés locH et locV.

Exemple

L'instruction suivante supprime une propriété d'image-objet constraint :

```
-- Lingo syntax
sprite(5).constraint = 0

// JavaScript syntax
sprite(5).constraint = 0;
```

L'instruction suivante contraint l'image-objet (i + 1) à la limite de l'image-objet 14 :

```
-- Lingo syntax
sprite(i + 1).constraint = 14

// JavaScript syntax
sprite(i + 1).constraint = 14;
```

L'instruction suivante vérifie si le déplacement de l'image-objet 3 est limité et, le cas échéant, active le gestionnaire `showConstraint` :

```
-- Lingo syntax
if (sprite(3).constraint <> 0) then
    showConstraint
end if

// JavaScript syntax
if (sprite(3).constraint != 0) {
    showConstraint();
}
```

Voir aussi

locH, locV, Image-objet

controlDown

Syntaxe

```
-- Lingo syntax
_key.controlDown

// JavaScript syntax
_key.controlDown;
```

Description

Propriété de touche ; détermine si l'utilisateur appuie sur la touche Ctrl. Lecture seule.

Cette propriété renvoie la valeur `TRUE` si la touche Ctrl est enfoncée ; dans le cas contraire, elle renvoie la valeur `FALSE`.

Vous pouvez utiliser la propriété `controlDown` avec la propriété `key` pour déterminer si l'utilisateur appuie sur la touche Ctrl en combinaison avec une autre touche. Cette opération vous permet de créer des gestionnaires exécutés lorsque l'utilisateur appuie sur des combinaisons de touches impliquant la touche Ctrl.

Les raccourcis clavier utilisant la touche Ctrl qui sont associés aux menus auteur de Director ont la priorité durant la lecture de l'animation, sauf si vous avez installé des menus personnalisés en syntaxe Lingo ou JavaScript ou que vous êtes en train de lire une projection.

Exemple

Le gestionnaire `on keyDown` suivant vérifie si la touche enfoncée est la touche Ctrl et, le cas échéant, active le gestionnaire `on doControlKey`. L'argument `(_key.key)` identifie la touche enfoncée en même temps que la touche Ctrl.

```
-- Lingo syntax
on keyDown
    if (_key.controlDown) then
        doControlKey(_key.key)
    end if
```

```
end

on doControlKey(theKey)
    trace("The " & theKey & " key is down")
end

// JavaScript syntax
function keyDown() {
    if (_key.controlDown) {
        doControlKey(_key.key);
    }
}

function doControlKey(theKey) {
    trace("The " & theKey & " key is down");
}
```

Voir aussi

Touche, key

controller

Syntaxe

```
member(whichCastMember).controller
the controller of member whichCastMember
```

Description

Propriété d'acteur vidéo numérique ; détermine si un acteur animation vidéo numérique affiche ou masque son contrôleur. Le paramétrage de cette propriété sur 1 affiche le contrôleur, alors que 0 le masque.

La propriété d'acteur `controller` s'applique uniquement à une vidéo numérique QuickTime®.

- La définition de la propriété d'acteur `controller` pour une vidéo numérique Vidéo pour Windows ne donne aucun résultat et n'entraîne aucun message d'erreur.
- Le test de la propriété d'acteur `controller` pour une vidéo numérique Vidéo pour Windows renvoie toujours la valeur `FALSE`.

La vidéo numérique doit être en mode de lecture au premier plan pour afficher le contrôleur.

Exemple

L'instruction suivante entraîne l'affichage du contrôleur de l'acteur QuickTime Démo :

```
--Lingo Dot syntax:
member("Demo").controller = 1

--Lingo Verbose syntax:
set the controller of member "Demo" to 1

// JavaScript syntax
member("Demo").controller = 1;
```

Voir aussi

directToStage

copyrightInfo (animation)

Syntaxe

```
-- Lingo syntax
_movie.copyrightInfo

// JavaScript syntax
_movie.copyrightInfo;
```

Description

Propriété d'animation ; permet de saisir une chaîne dans la boîte de dialogue Propriétés de l'animation en phase de création. Cette propriété permettra d'effectuer des améliorations dans les futures versions de Shockwave Player. Lecture seule.

Voir aussi

aboutInfo, Animation

copyrightInfo (SWA)

Syntaxe

```
-- Lingo syntax
memberObjRef.copyrightInfo

// JavaScript syntax
memberObjRef.copyrightInfo;
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; affiche le texte de copyright d'un fichier SWA. Cette propriété n'est disponible qu'après le début de la lecture du son SWA ou après le préchargement du fichier à l'aide de la commande `preLoadBuffer`.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante entraîne l'affichage par Director des informations de copyright du fichier Shockwave Audio SWA dans un acteur champ nommé Infos.

```
-- Lingo syntax
whatState = member("SWAfile").state
if whatState > 1 AND whatState < 9 then
    member("Info Display").text = member("SWAfile").copyrightInfo
end if

// JavaScript syntax
var whatState = member("SWAfile").state;
if (whatState > 1 && whatState < 9) {
    member("Info Display").text = member("SWAfile").copyrightInfo;
}
```

count

Syntaxe

```
list.count
count (list)
count(theObject)
object.count
textExpression.count
```

Description

Propriété (Lingo uniquement) ; renvoie le nombre d'entrées d'une liste linéaire ou de propriétés, le nombre de propriétés d'un script parent sans compter les propriétés d'un script ancêtre ou les sous-chaînes d'une expression texte telles que caractères, lignes ou mots.

La commande `count` fonctionne avec les listes linéaires et de propriétés, les objets créés avec des scripts parents et la propriété `globals`.

Vous pouvez voir un exemple d'utilisation de `count()` dans une animation en consultant l'animation Text du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante affiche la valeur 3, qui correspond au nombre d'entrées :

```
put [10,20,30].count
-- 3
```

Voir aussi

`globals`

count (3D)

Syntaxe

```
member(whichCastmember).light.count
member(whichCastmember).camera.count
member(whichCastmember).modelResource(whichModelResource).bone.count
member(whichCastmember).model.count
member(whichCastmember).group.count
member(whichCastmember).shader.count
member(whichCastmember).texture.count
member(whichCastmember).modelResource.count
member(whichCastmember).motion.count
member(whichCastmember).light.child.count
member(whichCastmember).camera.child.count
member(whichCastmember).model.child.count
member(whichCastmember).group.child.count
sprite(whichSprite).camera{(index)}.backdrop.count
member(whichCastmember).camera(whichCamera).backdrop.count
sprite(whichSprite).camera{(index)}.overlay.count
member(whichCastmember).camera(whichCamera).overlay.count
member(whichCastmember).model(whichModel).modifier.count
member(whichCastmember).model(whichModel).keyframePlayer.playlist.count
member(whichCastmember).model(whichModel).bonesPlayer.playlist.count
member(whichCastmember).modelResource(whichModelResource).face.count
member(whichCastmember).model(whichModel).meshDeform.mesh[index].textureLayer.count
```

```
member(whichCastmember).model(whichModel).meshDeform.mesh.count
member(whichCastmember).model(whichModel).meshDeform.mesh[index].face.count
```

Description

Propriété 3D ; renvoie le nombre d'éléments de la liste spécifiée associée à l'objet 3D indiqué. Utilisable avec n'importe quel type d'objet.

La propriété `face.count` permet d'obtenir le nombre de triangles dans la maille pour une ressource de modèle de type `#mesh`.

Cette propriété peut être testée, mais pas définie.

Exemple

Les exemples suivants déterminent le nombre de types d'objets au sein d'un acteur 3D appelé Univers 3D.

```
numberOfCameras = member("3D World").camera.count
put member("3D World").light.count
-- 3
numberOfModels = member("3D World").model.count
numberOfTextures = member("3D World").texture.count
put member("3D World").modelResource("mesh2").face.count
-- 4
```

L'instruction suivante indique que la première maille du modèle Oreille se compose de 58 faces.

```
put member("Scene").model("Ear").meshdeform.mesh[1].face.count
-- 58
```

L'instruction suivante indique que le modèle Oreille se compose de trois mailles.

```
put member("Scene").model("Ear").meshdeform.mesh.count
-- 3
```

L'instruction suivante indique que la première maille du modèle Oreille compte deux couches de texture.

```
put member("Scene").model("Ear").meshdeform.mesh[1].textureLayer.count
-- 2
```

Voir aussi

`cameraCount()`

cpuHogTicks

Syntaxe

```
the cpuHogTicks
```

Description

Propriété système ; détermine la fréquence avec laquelle Director libère le contrôle du processeur afin de permettre à l'ordinateur de traiter des événements d'arrière-plan, tels que ceux qui se produisent dans d'autres applications, des événements réseau, des mises à jour de l'horloge et d'autres événements de clavier.

La valeur par défaut est de 20 battements. Pour accorder un délai supplémentaire à Director avant de libérer le processeur pour les événements d'arrière-plan ou pour contrôler les opérations réseau, augmentez la valeur de `cpuHogTicks`.

Vous pouvez accélérer la répétition automatique des touches en diminuant la valeur de `cpuHogTicks` ; toutefois, cette opération ralentit l'animation. Dans une animation, lorsque l'utilisateur maintient une touche enfoncée pour créer une suite rapide de pressions de touches, Director vérifie généralement moins fréquemment les répétitions automatiques de pression de touches que la fréquence définie dans le tableau de bord de l'ordinateur.

La propriété `cpuHogTicks` ne fonctionne que sur Mac.

Exemple

L'instruction suivante entraîne Director à libérer le processeur tous les 6 battements, ce qui représente un dixième de seconde :

```
the cpuHogTicks = 6
```

Voir aussi

milliseconds

creaseAngle

Syntaxe

```
member(whichCastmember).model(whichModel).inker.creaseAngle
member(whichCastmember).model(whichModel).toon.creaseAngle
```

Description

Propriété 3D de modificateur `inker` et `toon` ; indique la sensibilité de la fonction de traçage de ligne du modificateur à la présence de plis dans la géométrie du modèle. Des paramètres plus élevés entraînent plus de lignes (détails) dessinées aux plis.

La propriété `creases` du modificateur doit être définie sur la valeur `TRUE` pour que la propriété `creaseAngle` soit effective.

La plage de valeurs de `CreaseAngle` est comprise entre -1,0 et +1,0. Le paramètre par défaut est 0,01.

Exemple

L'instruction suivante attribue à la propriété `creaseAngle` du modificateur `inker` appliqué au modèle Théière la valeur 0,10. Une ligne est dessinée pour tous les plis du modèle dépassant ce seuil. Ce paramètre n'est effectif que si la propriété `creases` du modificateur `inker` présente la valeur `TRUE`.

```
-- Lingo syntax
member("shapes").model("Teapot").addModifier(#inker);
member("shapes").model("Teapot").inker.creaseAngle = 0.10

// JavaScript syntax
member("shapes").getProp("model",1).addModifier(symbol("inker"));
member("shapes").getProp("model",1).getPropRef("inker").creaseAngle = 0.10;
```

Voir aussi

creases, lineColor, lineOffset, useLineOffset

creases

Syntaxe

```
member(whichCastmember).model(whichModel).inker.creases
member(whichCastmember).model(whichModel).toon.creases
```

Description

Propriété 3D de modificateur `toon` et `inker` ; détermine si des lignes sont dessinées au niveau des plis de la surface du modèle.

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante attribue à la propriété `creases` du modificateur `inker` du modèle Théière la valeur `TRUE`. Une ligne est dessinée sur tous les plis du modèle dépassant le seuil défini par la propriété `creaseAngle` du modificateur `inker`.

```
-- Lingo syntax
member("shapes").model("Teapot").addModifier(#inker);
member("shapes").model("Teapot").inker.creases = TRUE

// JavaScript syntax
member("shapes").getProp("model",1).addModifier(symbol("inker"));
member("shapes").getProp("model",1).getPropRef("inker").creases = true;
```

Voir aussi

`creaseAngle`, `lineColor`, `lineOffset`, `useLineOffset`

creationDate

Syntaxe

```
-- Lingo syntax
memberObjRef.creationDate

// JavaScript syntax
memberObjRef.creationDate;
```

Description

Propriété d'acteur ; enregistre la date de création initiale de l'acteur à l'aide de la date système de l'ordinateur. Lecture seule.

Vous pouvez utiliser cette propriété pour planifier un projet ; Director n'en fait pas usage.

Exemple

Bien que la propriété `creationDate` soit généralement vérifiée à l'aide de l'Inspecteur des propriétés ou de la fenêtre Distribution en mode d'affichage sous forme de liste, vous pouvez également la vérifier dans la fenêtre Messages :

```
-- Lingo syntax
put(member(1).creationDate)

// JavaScript syntax
put(member(1).creationDate);
```

Voir aussi

Acteur

crop

Syntaxe

```
member(whichCastMember).crop
the crop of member whichCastMember
```

Description

Propriété d'acteur ; met à l'échelle un acteur vidéo numérique pour le faire tenir exactement dans le rectangle de l'image-objet dans lequel il apparaît (FALSE) ou le recadre, sans en modifier la taille, pour le faire tenir dans le rectangle de l'image-objet (TRUE).

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante indique à Lingo de recadrer toute image-objet faisant référence à l'acteur vidéo numérique Entrevue.

```
-- Lingo Dot syntax:
member("Interview").crop = TRUE

-- Lingo Verbose syntax:
set the crop of member "Interview" to TRUE

// JavaScript syntax
member("Interview").crop = true;
```

Voir aussi

center

cuePointNames

Syntaxe

```
-- Lingo syntax
memberObjRef.cuePointNames

// JavaScript syntax
memberObjRef.cuePointNames;
```

Description

Propriété d'acteur ; crée une liste des noms de point de repère ou, si un point de repère n'a pas de nom, insère une chaîne vide ("") en tant que repère dans la liste. Les noms de points de repère sont pratiques pour la synchronisation du son, des acteurs QuickTime et de l'animation.

Cette propriété est supportée par les acteurs SoundEdit, les acteurs vidéo numérique QuickTime et les acteurs Xtras contenant des points de repère. La liste des points de repère peut ne pas être disponible pour les Xtras créant des points de repère au moment de l'exécution de l'animation.

Exemple

L'instruction suivante obtient le nom du troisième point de repère d'un acteur.

```
-- Lingo syntax
put member("symphony").cuePointNames[3]

// JavaScript syntax
put(member("symphony").cuePointNames[3]);
```

Voir aussi

cuePointTimes, mostRecentCuePoint

cuePointTimes

Syntaxe

```
-- Lingo syntax
memberObjRef.cuePointTimes

// JavaScript syntax
memberObjRef.cuePointTimes;
```

Description

Propriété d'acteur ; fournit une liste des positions des points de repère d'un acteur donné, en millisecondes. Les positions des points de repère sont pratiques pour la synchronisation du son, des acteurs QuickTime et de l'animation.

Cette propriété est supportée par les acteurs SoundEdit, les acteurs vidéo numérique QuickTime et les acteurs Xtras supportant des points de repère. La liste des points de repère peut ne pas être disponible pour les Xtras créant des points de repère au moment de l'exécution de l'animation.

Exemple

L'instruction suivante obtient la position du troisième point de repère d'un acteur son.

```
-- Lingo syntax
put member("symphony").cuePointTimes[3]

// JavaScript syntax
put(member("symphony").cuePointTimes[3]);
```

Voir aussi

cuePointNames, mostRecentCuePoint

currentLoopState

Syntaxe

```
member(whichCastmember).model(whichModel).keyframePlayer.currentLoopState
member(whichCastmember).model(whichModel).bonesPlayer.currentLoopState
```

Description

Propriété 3D de modificateur `keyframePlayer` et `bonesPlayer` ; indique si le mouvement exécuté par le modèle se répète continuellement (`TRUE`) ou s'il est exécuté jusqu'à la fin puis remplacé par le mouvement suivant dans la liste de lecture du modificateur (`FALSE`).

Le paramétrage par défaut de cette propriété correspond à la valeur du paramètre de boucle de la commande `play()` ayant démarré la lecture du mouvement ou la valeur de la commande `queue()` ayant ajouté le mouvement à la liste de lecture du modificateur. La modification de la propriété `currentLoopState` change également la valeur de la propriété `#looped` de l'entrée du mouvement dans la liste de lecture du modificateur.

Exemple

L'instruction suivante entraîne la lecture continue du mouvement exécuté par le modèle Monstre.

```
-- Lingo syntax
member("NewAlien").model("Monster").addModifier(#keyframeplayer)
member("NewAlien").model("Monster").keyframePlayer.currentLoopState = TRUE

// JavaScript syntax
member("NewAlien").getPropRef("model",1).addModifier(symbol("keyframeplayer"));
member("NewAlien").getProp("model",1).getPropRef("keyframePlayer").currentLoopState =
true;
```

Voir aussi

`loop (3D)`, `play() (3D)`, `queue() (3D)`, `playlist`

currentSpriteNum

Syntaxe

```
-- Lingo syntax
_player.currentSpriteNum

// JavaScript syntax
_player.currentSpriteNum;
```

Description

Propriété de lecteur ; indique le numéro de piste de l'image-objet dont le script est en cours d'exécution. Lecture seule.

Cette propriété est utilisable dans les scripts d'acteur et dans les comportements. Lorsqu'elle est utilisée dans les scripts d'image ou d'animation, la propriété `currentSpriteNum` présente la valeur 0.

La propriété `currentSpriteNum` est comparable à la propriété `spriteNum` de l'objet image-objet.

Remarque : *cette propriété était plus utile pour le passage des anciennes animations à Director 6, à l'apparition des comportements. Elle permettait une fonction de type comportement sans nécessiter la réécriture complète du script. Elle n'est pas nécessaire pour la création avec des comportements et se révèle donc moins utile que par le passé.*

Exemple

Le gestionnaire suivant dans un script d'acteur ou d'animation remplace l'acteur affecté à l'image-objet impliquée dans l'événement `mouseDown` :

```
-- Lingo syntax
on mouseDown
    sprite(_player.currentSpriteNum).member = member("DownPict")
```

```
end

// JavaScript syntax
function mouseDown() {
    sprite(_player.currentSpriteNum).member = member("DownPict");
}
```

Voir aussi

Lecteur, spriteNum

currentTime (3D)

Syntaxe

```
member(whichCastmember).model(whichModel).keyframePlayer.currentTime
member(whichCastmember).model(whichModel).bonesPlayer.currentTime
```

Description

Propriété 3D de modificateur `keyframePlayer` et `bonesPlayer` ; indique la position locale du mouvement exécuté par le modèle. La propriété `currentTime` est mesurée en millisecondes, mais ne correspond à la position réelle que lorsque le mouvement est lu à sa cadence d'origine.

La lecture d'un mouvement par un modèle est le résultat d'une commande `play()` ou `queue()`. Le paramètre `scale` de la commande `play()` ou `queue()` est multiplié par la propriété `playRate` du modificateur, et la valeur qui en résulte est multipliée par la cadence d'origine du mouvement pour déterminer la cadence à laquelle le modèle exécute le mouvement. Ainsi, si le paramètre `scale` a pour valeur 2 et la propriété `playRate` du modificateur pour valeur 3, le modèle exécute le mouvement six fois plus vite qu'à la cadence d'origine.

La propriété `currentTime` réinitialise la valeur sur celle du paramètre `cropStart` de la commande `play()` ou `queue()` au début de chaque itération d'un mouvement en boucle.

Exemple

L'instruction suivante indique la position locale du mouvement exécuté par le modèle Martien3.

```
-- Lingo syntax
member("NewAlien").model("Alien3").addModifier(#keyframeplayer)
put member("NewAlien").model("Alien3").keyframePlayer.play()
put member("NewAlien").model("Alien3").keyframePlayer.currentTime
-- 1393.8599

// JavaScript syntax
member("NewAlien").getPropRef("model",1).addModifier(symbol("keyframeplayer"));
member("NewAlien").getProp("model",1).getPropRef("keyframePlayer").play();
put(member("NewAlien").getProp("model",1).getPropRef("keyframePlayer").currentTime );
// 1393.8599
```

Voir aussi

play() (3D), queue() (3D), playlist

currentTime (DVD)

Syntaxe

```
-- Lingo syntax
dvdObjRef.currentTime

// JavaScript syntax
dvdObjRef.currentTime;
```

Description

Propriété de DVD ; renvoie la durée écoulée en millisecondes. Lecture/écriture.

Exemple

L'instruction suivante renvoie la durée écoulée :

```
-- Lingo syntax
trace (member(1).currentTime)-- 11500

// JavaScript syntax
trace (member(1).currentTime);// 11500
```

L'instruction suivante définit la propriété `currentTime` sur un point spécifique du titre en cours :

```
-- Lingo syntax
member(1).currentTime = 22000

// JavaScript syntax
member(1).currentTime = 22000
```

Voir aussi

DVD

currentTime (QuickTime, AVI)

Syntaxe

```
-- Lingo syntax
spriteObjRef.currentTime

// JavaScript syntax
spriteObjRef.currentTime;
```

Description

Propriété d'image-objet vidéo numérique ; détermine la position temporelle en cours de l'animation vidéo numérique exécutée dans la piste spécifiée par *`quelleImageObjet`*. La valeur de `movieTime` est exprimée en battements.

Cette propriété peut être testée et définie.

Vous pouvez voir un exemple d'utilisation de `currentTime` dans une animation en consultant l'animation QT and Flash du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante affiche dans la fenêtre Messages la durée actuelle de l'animation QuickTime de la piste 9 :

```
-- Lingo syntax
put(sprite(9).currentTime)

// JavaScript syntax
put(sprite(9).currentTime);
```

L'instruction suivante affecte à la variable `Poster` la valeur de position temporelle en cours de l'animation QuickTime de la piste 9 :

```
-- Lingo syntax
sprite(9).currentTime = Poster

// JavaScript syntax
sprite(9).currentTime = Poster;
```

Voir aussi

duration (acteur)

currentTime (RealMedia)

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.currentTime

// JavaScript syntax
memberOrSpriteObjRef.currentTime;
```

Description

Propriété d'image-objet ou d'acteur RealMedia ; permet d'obtenir ou de définir la position du flux RealMedia, en millisecondes. Si l'acteur RealMedia n'est pas en cours de lecture, la valeur de cette propriété est 0, qui constitue la valeur par défaut. Il s'agit d'une propriété de lecture qui n'est pas enregistrée.

Si le flux est en cours de lecture lors de la définition ou de la modification de la propriété `currentTime`, une recherche est lancée, le flux est remis en mémoire tampon, puis la lecture reprend à la nouvelle position définie. Si le flux est interrompu (valeur `#paused mediaStatus`) lors de la définition ou de la modification de la propriété `currentTime`, le flux redessine l'image à la nouvelle position, puis la lecture reprend si la propriété `pausedAtStart` présente la valeur `FALSE`. Lorsque le flux est interrompu ou arrêté dans la fenêtre RealMedia, la propriété `mediaStatus` présente la valeur `#paused`. Lorsque le flux est arrêté par la commande Lingo `stop`, `mediaStatus` a pour valeur `#closed`. Cette propriété n'a aucun effet lorsque la propriété `mediaStatus` du flux présente la valeur `#closed`. Lorsque vous définissez des nombres entiers, ils sont ajoutés à la plage à partir de 0 pour la durée du flux.

La définition de la propriété `currentTime` équivaut à invoquer la commande `seek` : `x.seek(n)` est identique à `x.currentTime = n`. La modification de `currentTime` ou l'appel de `seek` nécessite la remise en mémoire tampon du flux.

Exemple

Les exemples suivants indiquent que la valeur de position temporelle de l'image-objet 2 et de l'acteur Real est 15 534 millisecondes (15,534 secondes) depuis le début du flux.

```
-- Lingo syntax
put(sprite(2).currentTime) -- 15534
put(member("Real").currentTime) -- 15534
```

```
// JavaScript syntax
put(sprite(2).currentTime) // 15534
put(member("Real").currentTime) // 15534
```

Les exemples suivants entraînent un saut de lecture de 20 000 millisecondes (20 secondes) dans le flux de l'image-objet 2 et de l'acteur Real.

```
-- Lingo syntax
sprite(2).currentTime = 20000
member("Real").currentTime = 20000

// JavaScript syntax
sprite(2).currentTime = 20000
member("Real").currentTime = 20000
```

Voir aussi

duration (RealMedia, SWA), seek(), mediaStatus (RealMedia, Windows Media)

currentTime (Sprite)

Syntaxe

```
-- Lingo syntax
spriteObjRef.currentTime

// JavaScript syntax
spriteObjRef.currentTime;
```

Description

Propriété d'image-objet et de piste audio ; renvoie la position temporelle de lecture, en millisecondes, d'une image-objet audio ou vidéo numérique QuickTime ou de n'importe quel Xtra prenant en charge les points de repère. Pour une piste audio, renvoie la position temporelle de lecture de l'acteur son actuellement en cours de lecture dans la piste audio concernée.

Cette propriété ne peut être définie que pour les acteurs son traditionnels (*.wav, *.aif, *.snd). Lorsque cette propriété est définie, la plage de valeurs autorisée est comprise entre zéro et la valeur `duration` de l'acteur.

Les sons Shockwave Audio (SWA) peuvent apparaître sous forme d'images-objets dans les pistes d'images-objets, mais sont lus dans les pistes audio. Il est conseillé de faire référence aux images-objets audio SWA par le numéro de leur piste d'image-objet plutôt que par celui de leur piste audio.

Exemple

L'instruction suivante indique la position temporelle actuelle, en secondes, de l'image-objet dans la piste 10.

```
-- Lingo syntax
member("time").text = string(sprite(10).currentTime/ 1000)

// JavaScript syntax
member("time").text = (sprite(10).currentTime / 1000).toString();
```

L'instruction suivante entraîne le passage du son de la piste audio 2 au point 2,7 secondes à partir du début de l'acteur son :

```
-- Lingo syntax
sound(2).currentTime = 2700
```

```
// JavaScript syntax
sound(2).currentTime = 2700;
```

Voir aussi

duration (acteur)

cursor

Syntaxe

```
-- Lingo syntax
spriteObjRef.cursor

// JavaScript syntax
spriteObjRef.cursor;
```

Description

Propriété d'image-objet ; détermine le curseur utilisé lorsque le pointeur est positionné sur une image-objet. Lecture/écriture.

Cette propriété reste en vigueur jusqu'à ce que vous la désactiviez en lui attribuant la valeur 0. Utilisez la propriété `cursor` pour modifier le pointeur de la souris lorsque ce dernier est positionné sur des zones spécifiques de l'écran, ainsi que pour indiquer les zones où certaines actions sont possibles lorsque l'utilisateur clique.

Lorsque vous définissez la propriété `cursor` pour une image spécifique, Director examine le rectangle de l'image-objet avant de décider de modifier ou non le curseur. Ce rectangle ne change pas lorsque l'animation passe à l'image suivante, sauf si vous attribuez une valeur nulle à la propriété `cursor` de cette piste.

- Utilisez la syntaxe suivante pour spécifier le numéro d'acteur à utiliser comme curseur et son masque facultatif.

```
-- Lingo syntax
spriteObjRef.cursor = [castMemberObjRef, maskCastMemberObjRef]

// JavaScript syntax
spriteObjRef.cursor = [castMemberObjRef, maskCastMemberObjRef];
```

- Utilisez la syntaxe suivante pour spécifier les curseurs par défaut fournis par le système.

```
-- Lingo syntax
spriteObjRef.cursor = castMemberObjRef

// JavaScript syntax
spriteObjRef.cursor = castMemberObjRef;
```

La propriété `cursor` peut être définie sur l'un des nombres entiers suivants :

Valeur	Description
-1, 0	Flèche
1	I
2	Croix
3	Croix épaisse
4	Montre (Mac) ou sablier (Windows)
5	Nord Sud Est Ouest (NSEO)

Valeur (Suite)	**Description (Suite)**
6	Nord Sud (NS)
200	Vide (masque le curseur)
254	Aide
256	Crayon
257	Gomme
258	Sélection
259	Pot de peinture
260	Main
261	Outil Rectangle
262	Outil Rectangle arrondi
263	Outil Cercle
264	Outil Ligne
265	Outil Texte enrichi
266	Outil Champ de texte
267	Outil Bouton
268	Outil Case à cocher
269	Outil Bouton radio
270	Outil Placement
271	Outil Point d'alignement
272	Lasso
280	Doigt
281	Pipette
282	Attente souris appuyée 1
283	Attente souris appuyée 2
284	Taille verticale
285	Taille horizontale
286	Taille diagonale
290	Main fermée
291	Interdiction
292	Copie (main fermée)
293	Flèche inversée
294	Rotation
295	Inclinaison

Valeur (Suite)	Description (Suite)
296	Double flèche horizontale
297	Double flèche verticale
298	Double flèche Sud-ouest Nord-est
299	Double flèche Nord-ouest Sud-est
300	Pinceau Enduire/Lisser
301	Aérographe
302	Zoom avant
303	Zoom arrière
304	Annulation du zoom
305	Début de forme
306	Ajout de point
307	Fermeture de forme
308	Zoom de caméra
309	Déplacement de caméra
310	Rotation de caméra
457	Personnalisé

Pour utiliser des curseurs personnalisés, définissez la propriété `cursor` sur une liste contenant l'acteur à utiliser en tant que curseur ou sur le numéro spécifiant un curseur système. Sous Windows, un curseur doit être un acteur, et non une ressource ; si aucun curseur n'est disponible puisqu'il s'agit d'une ressource, Director affiche le curseur en forme de flèche standard. Pour garantir des résultats optimaux, veillez à ne pas utiliser de curseurs personnalisés lorsque vous créez des animations multiplates-formes.

Les acteurs curseur personnalisé ne doivent pas dépasser 16 par 16 pixels et doivent être codés sur 1 bit.

Si l'image-objet est un bitmap avec une encre Dessin seul, le curseur ne change que lorsqu'il se trouve sur la portion Dessin seul de l'image-objet.

Lorsque le pointeur se trouve à l'emplacement d'une image-objet qui a été supprimée, le survol se produit quand même. Vous pouvez éviter ce problème en ne faisant pas de survol à ces endroits ou en déplaçant l'image-objet au-dessus de la barre des menus avant de la supprimer.

Sur le Mac, vous pouvez utiliser une ressource de curseur numérotée dans l'animation en cours à la place du curseur en remplaçant `cursor` par le numéro de la ressource de curseur.

Exemple

L'instruction suivante change en montre (Mac) ou en sablier (Windows) le curseur qui apparaît sur l'image-objet 20.

```
-- Lingo syntax
sprite(20).cursor = 4

// JavaScript syntax
sprite(20).cursor = 4;
```

Voir aussi

Image-objet

cursorSize

Syntaxe

```
-- Lingo syntax
memberObjRef.cursorSize

// JavaScript syntax
memberObjRef.cursorSize;
```

Description

Propriété d'acteur curseur ; spécifie la taille de l'acteur curseur couleur animé `quelActeurCurseur`.

Spécifiez la taille :	Pour des curseurs pouvant atteindre :
`16`	16 x 16 pixels
`32`	32 x 32 pixels

Les acteurs bitmap inférieurs à la taille spécifiée sont affichés en taille normale, tandis que les acteurs plus grands sont ramenés à la taille spécifiée.

La valeur par défaut est de 32 pour Windows et de 16 pour Mac. Si vous choisissez une valeur incorrecte, un message d'erreur s'affiche pendant la lecture de l'animation (mais pas à la compilation).

Cette propriété peut être testée et définie.

Exemple

La commande suivante redimensionne le curseur couleur animé stocké dans l'acteur curseur couleur 20 à 32 x 32 pixels.

```
-- Lingo syntax
member(20).cursorSize = 32

// JavaScript syntax
member(20).cursorSize = 32;
```

curve

Syntaxe

```
-- Lingo syntax
memberObjRef.curve[curveListIndex]

// JavaScript syntax
memberObjRef.curve[curveListIndex];
```

Description

Cette propriété contient la liste `vertexList` d'une courbe (forme) individuelle provenant d'un acteur forme vectorielle. Vous pouvez utiliser la propriété curve en même temps que la propriété `vertex` pour obtenir les sommets individuels d'une courbe spécifique dans une forme vectorielle.

La propriété `vertexList` définit une liste de sommets, chaque sommet étant une liste de propriétés comprenant un maximum de trois propriétés : une propriété #vertex, avec l'emplacement du sommet, une propriété #handle1 avec l'emplacement du premier point de contrôle de ce sommet et une propriété #handle2, avec l'emplacement du second point de contrôle de ce sommet. Reportez-vous à l'entrée `vertexList`.

Exemple

L'instruction suivante affiche un exemple de liste des sommets de la troisième courbe de l'acteur forme vectorielle CourbesSimples :

```
-- Lingo syntax
put(member("SimpleCurves").curve[3])
-- [[#vertex: point(113.0000, 40.0000), #handle1: point(32.0000, 10.0000), #handle2: point(-
32.0000, -10.0000)], [#vertex: point(164.0000, 56.0000)]]

// JavaScript syntax
put(member("SimpleCurves").curve[3]);
// [[#vertex: point(113.0000, 40.0000), #handle1: point(32.0000, 10.0000),#handle2: point(-
32.0000, -10.0000)], [#vertex: point(164.0000, 56.0000)]]
```

L'instruction suivante déplace de 10 pixels vers le bas et à droite le premier sommet de la première courbe dans une forme vectorielle :

```
-- Lingo syntax
member(1).curve[1].vertex[1] = member(1).curve[1].vertex[1] + point(10, 10)

// JavaScript syntax
member(1).curve[1].vertex[1] = member(1).curve[1].vertex[1] + point(10, 10);
```

Le code suivant déplace une image-objet vers l'emplacement du premier sommet de la première courbe d'une forme vectorielle. La propriété `originMode` de la forme vectorielle doit être définie sur `#topLeft` pour cette opération.

```
-- Lingo syntax
vertexLoc = member(1).curve[1].vertex[1]
spriteLoc = mapMemberToStage(sprite(3), vertexLoc)
sprite(7).loc = spriteLoc

// JavaScript syntax
var vertexLoc = member(1).curve[1].vertex[1];
var spriteLoc = mapMemberToStage(sprite(3), vertexLoc);
sprite(7).loc = spriteLoc;
```

debug

Syntaxe

```
member(whichCastmember).model(whichModel).debug
```

Description

Propriété 3D de modèle ; indique si la sphère de délimitation et les axes locaux du modèle sont affichés.

Exemple

L'instruction suivante attribue à la propriété `debug` du modèle Chien la valeur `TRUE`.

```
-- Lingo syntax
member("ParkScene").model("Dog").debug = TRUE

// JavaScript syntax
member("ParkScene").getProp("model", 1).debug = true;
```

Voir aussi

`boundingSphere`

debugPlaybackEnabled

Syntaxe

```
-- Lingo syntax
_player.debugPlaybackEnabled

// JavaScript syntax
_player.debugPlaybackEnabled;
```

Description

Propriété de lecteur ; sous Windows, ouvre une fenêtre Messages à des fins de débogage dans Shockwave et les projections. Sur Mac, cette propriété génère un fichier journal pour permettre à des instructions `put` d'afficher des données à des fins de débogage. Lecture/écriture.

Sous Windows, cette propriété n'a aucun effet lorsqu'elle est utilisée dans l'application Director. Une fois la fenêtre Messages fermée, elle ne peut pas être rouverte dans une session Shockwave Player ou dans une projection. Si plusieurs animations avec du contenu Shockwave utilisent ce script dans un seul navigateur, seule la première animation ouvrira une fenêtre Messages qui est liée exclusivement à cette animation.

Pour visualiser les entrées de la fenêtre Messages sur Macintosh, ouvrez le fichier 1msgdump.txt à l'emplacement \Library\Logs\.

Pour ouvrir la fenêtre Messages, définissez la propriété `debugPlaybackEnabled` sur la valeur `TRUE`. Pour fermer la fenêtre Messages, attribuez à la propriété `debugPlaybackEnabled` la valeur `FALSE`.

Exemple

Cette instruction entraîne l'ouverture de la fenêtre Messages dans Shockwave Player ou dans une projection :

```
-- Lingo syntax
_player.debugPlaybackEnabled = TRUE

// JavaScript syntax
_player.debugPlaybackEnabled = true;
```

Voir aussi

`Lecteur`, `put()`

decayMode

Syntaxe

```
member(whichCastmember).camera(whichCamera).fog.decayMode
sprite(whichSprite).camera{(index)}.fog.decayMode
```

Description

Propriété 3D ; indique la façon dont la densité du brouillard évolue d'une densité minimale à une densité maximale lorsque la propriété `fog.enabled` de la caméra présente la valeur `TRUE`.

Les valeurs possibles de cette propriété sont les suivantes :

- `#linear` : la densité du brouillard est interpolée de façon linéaire entre `fog.near` et `fog.far`.
- `#exponential` : `fog.far` est le point de saturation ; `fog.near` est ignoré.
- `#exponential2` : `fog.near` est le point de saturation ; `fog.far` est ignoré.

La valeur par défaut de cette propriété est `#exponential`.

Exemple

L'instruction suivante attribue à la propriété `decayMode` du brouillard de la caméra vueParDéfaut la valeur `#linear`. Si la propriété fog.enabled présente la valeur `TRUE`, la densité du brouillard augmente de façon constante entre les distances définies par les propriétés `near` et `far` du brouillard. Si la propriété `near` présente la valeur 100 et que la propriété `far` présente la valeur 1 000, le brouillard commence à 100 unités d'univers devant la caméra, puis gagne régulièrement en densité jusqu'à une distance de 1 000 unités d'univers devant la caméra.

```
-- Lingo syntax
member("3d world").camera("Defaultview").fog.decayMode = #linear

// Java Script
member("3d world").getProp("camera",1).getPropRef("fog").decayMode = symbol("linear");
```

Voir aussi

fog, near (brouillard), far (brouillard), enabled (brouillard)

defaultRect

Syntaxe

```
-- Lingo syntax
memberObjRef.defaultRect

// JavaScript syntax
memberObjRef.defaultRect;
```

Description

Propriété d'acteur ; contrôle la taille par défaut utilisée pour toutes les nouvelles images-objets créées depuis un acteur animation Flash ou forme vectorielle. Le paramètre `defaultRect` s'applique également à toutes les images-objets existantes qui n'ont pas été étirées sur la scène. Vous spécifiez les valeurs de propriété comme un rectangle Director ; par exemple, rect(0, 0, 32, 32).

La propriété d'acteur `defaultRect` est affectée par la propriété d'acteur `defaultRectMode` de l'acteur. La propriété `defaultRectMode` présente toujours la valeur `#Flash` lorsqu'une animation est insérée dans une distribution, ce que signifie que le paramètre `defaultRect` initial présente toujours la taille de l'animation d'origine créée dans Flash. La définition de `defaultRect` après ce stade affecte implicitement la valeur `#fixed` à la propriété `defaultRectMode` de l'acteur.

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant accepte une référence de distribution et un rectangle comme paramètres. Il recherche ensuite les acteurs Flash dans la distribution spécifiée et définit leur propriété `defaultRect` sur le rectangle spécifié.

```
-- Lingo syntax
on setDefaultFlashRect(whichCast, whichRect)
    repeat with i = 1 to castLib(whichCast).member.count
        if member(i, whichCast).type = #flash then
            member(i, whichCast).defaultRect = whichRect
        end if
    end repeat
end

// JavaScript syntax
function setDefaultFlashRect(whichCast, whichRect) {
    var i = 1;
    while( i < (castLib(whichCast).member.count) + 1)
        var tp = member(i, whichCast).type;
        if (tp == "flash") {
            member(i, whichCast).defaultRect = whichRect;
            i++;
        }
    }
}
```

Voir aussi

`defaultRectMode`, `flashRect`

defaultRectMode

Syntaxe

```
-- Lingo syntax
memberObjRef.defaultRectMode

// JavaScript syntax
memberObjRef.defaultRectMode;
```

Description

Propriété d'acteur ; contrôle la méthode par laquelle la taille par défaut est définie pour toutes les nouvelles images-objets créées depuis les acteurs animation Flash ou forme vectorielle. Vous spécifiez les valeurs de propriété comme un rectangle Director ; par exemple, rect(0, 0, 32, 32).

La propriété `defaultRectMode` ne définit pas la taille réelle du rectangle par défaut d'une animation Flash, mais détermine uniquement le réglage du rectangle par défaut. La propriété d'acteur `defaultRectMode` peut prendre l'une des valeurs suivantes :

- `#flash` (valeur par défaut) : affecte au rectangle par défaut la taille de l'animation d'origine créée dans Flash.
- `#fixed` : affecte au rectangle par défaut la taille fixe spécifiée par la propriété d'acteur `defaultRect`.

La propriété d'acteur `defaultRect` est affectée par la propriété d'acteur `defaultRectMode` de l'acteur. La propriété `defaultRectMode` présente toujours la valeur `#flash` lorsqu'une animation est insérée dans une distribution, ce que signifie que le paramètre `defaultRect` initial présente toujours la taille de l'animation d'origine créée dans Flash. La définition de `defaultRect` après ce stade affecte implicitement la valeur `#fixed` à la propriété `defaultRectMode` de l'acteur.

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant accepte une référence de distribution et un rectangle comme paramètres. Il recherche ensuite les acteurs Flash dans la distribution spécifiée, définit leur propriété `defaultRectMode` sur `#fixed`, puis attribue à leur propriété `defaultRect` la valeur rect(0, 0, 320, 240).

```
-- Lingo syntax
on setDefaultRectSize(whichCast)
    repeat with i = 1 to castLib(whichCast).member.count
        if member(i, whichCast).type = #flash then
            member(i, whichCast).defaultRectMode = #fixed
            member(i, whichCast).defaultRect = rect(0,0,320,240)
        end if
    end repeat
end

// JavaScript syntax
function setDefaultRectSize(whichCast) {
    var i = 1;
    while( i < (castLib(whichCast).member.count) + 1)
        var tp = member(i, whichCast).type;
        if (tp == "flash") {
            member(i, whichCast).defaultRectMode = symbol("fixed");
            member(i, whichCast).defaultRect = rect(0,0,320,240);
            i++;
        }
    }
}
```

Voir aussi

`flashRect`, `defaultRect`

density

Syntaxe

```
member(whichCastmember).shader(whichShader).density
member(whichCastmember).model(whichModel).shader.density
member(whichCastmember).model(whichModel).shaderList{[index]}.density
```

Description

Propriété 3D de matériau `#engraver` et `#newsprint` ; ajuste le nombre de lignes ou de points utilisés pour créer les effets de ces types de matériaux spécialisés. Plus les valeurs sont élevées plus le nombre de lignes ou de points est élevé.

Pour les matériaux `#engraver`, cette propriété ajuste le nombre de lignes utilisées pour créer l'image. La valeur peut être comprise entre 0 et 100, la valeur par défaut étant 40.

Pour les matériaux `#newsprint`, cette propriété ajuste le nombre de points utilisés pour créer l'image. La valeur peut être comprise entre 0 et 100, la valeur par défaut étant 45.

Exemple

L'instruction suivante attribue à la propriété `density` du matériau matériauMoteur la valeur 10. Les lignes utilisées par ce matériau `#engraver` pour créer son image stylisée sont grossières et éloignées les unes des autres.

```
-- Lingo syntax
member("scene").shader("EngShader").density = 10
// JavaScript syntax
member("scene").getProp("shader",1).density = 10;
```

L'instruction suivante attribue à la propriété `density` du matériau gbMatériau la valeur 100. Les points utilisés par ce matériau `#newsprint` pour créer son image stylisée sont très fins et rapprochés.

```
-- Lingo syntax
member("scene").shader("gbShader").density = 100
// JavaScript syntax
member("scene").getProp("shader", 2).density = 100;
```

Voir aussi

`newShader`

depth (3D)

Syntaxe

```
member(whichCastmember).model(whichModel).sds.depth
```

Description

Propriété 3D de modificateur de fractionnement de surface (`sds`) ; spécifie le nombre maximal de niveaux de résolution que le modèle peut afficher lorsque le modificateur `sds` est utilisé.

Si les valeurs des paramètres `error` et `tension` du modificateur `sds` sont faibles, l'augmentation de la valeur de la propriété `depth` produira un effet plus prononcé sur la géométrie du modèle.

Le modificateur `sds` ne peut pas être utilisé avec les modificateurs `inker` ou `toon` ; vous devrez également faire preuve de prudence lors de l'utilisation du modificateur `sds` avec le modificateur `lod`.

Exemple

L'instruction suivante définit la propriété `depth` du modificateur sds du modèle Bébé sur la valeur 3. Si les valeurs des paramètres `error` et `tension` du modificateur sds sont faibles, cette instruction produira un effet très prononcé sur la géométrie de Bébé.

```
-- Lingo syntax
member("Scene").model("Baby").addModifier(#sds)
member("Scene").model("Baby").sds.depth = 3

// JavaScript syntax
member("Scene").getProp("model",2).addModifier(symbol("sds"));
member("Scene").getProp("model", 2).getPropRef("sds").depth = 3;
```

Voir aussi

sds (modificateur), error, tension

depth (bitmap)

Syntaxe

```
imageObject.depth
member(whichCastMember).depth
the depth of member whichCastMember
```

Description

Propriété d'objet image ou d'acteur bitmap ; affiche le codage de couleur de l'objet image ou de l'acteur bitmap donné.

Depth	Nombre de couleurs
1	Noir et blanc
2	4 couleurs
4, 8	Couleurs de palettes 16 ou 256 couleurs ou niveaux de gris
16	Milliers de couleurs
32	Millions de couleurs

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affiche le codage des couleurs de l'objet image stocké dans la variable `newImage`. Le résultat apparaît dans la fenêtre Messages.

```
-- Lingo syntax
put(newImage.depth)

// JavaScript syntax
trace(newImage.depth);
```

L'instruction suivante affiche le codage des couleurs de l'acteur Temple dans la fenêtre Messages :

```
-- Lingo syntax
put(member("Shrine").depth)

// JavaScript syntax
put(member("Shrine").depth);
```

depthBufferDepth

Syntaxe

```
getRendererServices().depthBufferDepth
```

Description

Propriété 3D `rendererServices` ; indique la précision du tampon de codage du matériel du système de l'utilisateur. Sa valeur est de 16 ou de 24, selon le matériel de l'utilisateur.

Exemple

L'instruction suivante indique que la valeur `depthBufferDepth` de la carte vidéo de l'utilisateur est de 16 :

```
-- Lingo syntax
put getRendererServices().depthBufferDepth
-- 16

// JavaScript syntax
put (getRendererServices().depthBufferDepth);
// 16
```

Voir aussi

`getRendererServices()`, `getHardwareInfo()`, `colorBufferDepth`

deskTopRectList

Syntaxe

```
-- Lingo syntax
_system.deskTopRectList

// JavaScript syntax
_system.deskTopRectList;
```

Description

Propriété système ; affiche la taille, et la position sur le bureau, des moniteurs connectés à l'ordinateur. Lecture seule.

Cette propriété est pratique pour vérifier si des objets tels que des fenêtres, des images-objets et des fenêtres contextuelles apparaissent entièrement sur un écran.

Le résultat est une liste de rectangles, dans laquelle chaque rectangle indique la limite physique d'un moniteur. Les coordonnées de chaque moniteur sont relatives à l'angle supérieur gauche du moniteur 1, qui a la valeur (0, 0). Le premier ensemble de coordonnées de rectangle correspond à la taille du premier moniteur. Si un second moniteur est présent, un second ensemble de coordonnées indique où les coins du second moniteur sont placés par rapport au premier moniteur.

Exemple

L'instruction suivante teste la taille des moniteurs connectés à l'ordinateur et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(_system.deskTopRectList)

// JavaScript syntax
put(_system.deskTopRectList);
```

Le gestionnaire suivant indique le nombre de moniteurs du système en cours :

```
-- Lingo syntax
on countMonitors
    return _system.deskTopRectList
end

// JavaScript syntax
function countMonitors() {
    return _system.deskTopRectList;
}
```

Voir aussi

Système

diffuse

Syntaxe

```
member(whichCastmember).shader(whichShader).diffuse
member(whichCastmember).model(whichModel).shader.diffuse
member(whichCastmember).model(whichModel).shaderList{[index]}.diffuse
```

Description

Propriété 3D de matériau `#standard` ; indique la couleur fusionnée avec la première texture du matériau lorsque les conditions suivantes sont réunies :

- la propriété `useDiffuseWithTexture` du matériau présente la valeur `TRUE` et, soit
- la propriété `blendFunction` du matériau présente la valeur `#add` ou `#multiply`, soit
- la propriété `blendFunction` du matériau présente la valeur `#blend`, la propriété `blendSource` du matériau a pour valeur `#constant` et la valeur de la propriété `blendConstant` du matériau est inférieure à 100.

La valeur par défaut de cette propriété est `rgb( 255, 255, 255 )`.

Exemple

L'instruction suivante attribue à la propriété `diffuse` du matériau Globe la valeur `rgb(255, 0, 0)`.

```
-- Lingo syntax
member("MysteryWorld").shader("Globe").diffuse = rgb(255, 0, 0)

// JavaScript syntax
member("MysteryWorld").getProp("shader", 1).diffuse = color(255, 0, 0);
```

Voir aussi

diffuseColor, useDiffuseWithTexture, blendFunction, blendSource, blendConstant

diffuseColor

Syntaxe

```
member(whichCastmember).diffuseColor
```

Description

Propriété 3D d'acteur ; indique la couleur qui est fusionnée à la première texture du premier matériau de l'acteur lorsque les conditions suivantes sont réunies :

- la propriété `useDiffuseWithTexture` du matériau présente la valeur `TRUE` et, soit
- la propriété `blendFunction` du matériau présente la valeur `#add` ou `#multiply`, soit
- la propriété `blendFunction` du matériau présente la valeur `#blend`, la propriété `blendSource` du matériau a pour valeur `#constant` et la valeur de la propriété `blendConstant` du matériau est inférieure à 100.

La valeur par défaut de la propriété `diffuseColor` est `rgb( 255, 255, 255 )`.

Exemple

L'instruction suivante définit la propriété `diffuseColor` de l'acteur Pièce sur la valeur `rgb(255, 0, 0)`.

```
-- Lingo syntax
member("Room").diffuseColor = rgb(255, 0, 0)

// JavaScript syntax
member("Room").diffuseColor = color(255, 0, 0);
```

Voir aussi

diffuse, useDiffuseWithTexture, blendFunction, blendSource, blendConstant

diffuseLightMap

Syntaxe

```
member(whichCastmember).shader(whichShader).diffuseLightMap
member(whichCastmember).model(whichModel).shader.diffuseLightMap
member(whichCastmember).model(whichModel).shaderList{[index]}.diffuseLightMap
```

Description

Propriété 3D de matériau `#standard` ; spécifie la texture à utiliser pour la lumière diffuse.

Les propriétés suivantes sont automatiquement définies avec cette propriété :

- La seconde couche de texture du matériau reçoit la texture que vous spécifiez.
- La valeur de `textureModeList[2]` est définie sur `#diffuse`.
- La valeur de `blendFunctionList[2]` est définie sur `#multiply`.
- La valeur de `blendFunctionList[1]` est définie sur `#replace`.

Exemple

L'instruction suivante attribue la texture Ovale à la propriété `diffuseLightMap` du matériau utilisé par le modèle boîteEnVerre.

```
-- Lingo syntax
member("3DPlanet").model("GlassBox").shader.diffuseLightMap =
member("3DPlanet").texture("Oval")

// JavaScript syntax
member("3DPlanet").getProp("model", 1).shaderList[1].diffuseLightMap =
member("3DPlanet").getprop("texture", 1);
```

Voir aussi

blendFunctionList, textureModeList, glossMap, region, specularLightMap

digitalVideoTimeScale

Syntaxe

```
-- Lingo syntax
_player.digitalVideoTimeScale
```

```
// JavaScript syntax
_player.digitalVideoTimeScale;
```

Description

Propriété de lecteur ; détermine l'échelle temporelle, en unités par seconde, que le système utilise pour mesurer la durée des acteurs vidéo numérique. Lecture/écriture.

La propriété `digitalVideoTimeScale` peut recevoir n'importe quelle valeur.

La valeur de cette propriété détermine la fraction de seconde utilisée pour suivre la progression de la vidéo, comme dans les exemples suivants :

- 100 : l'échelle temporelle est de 1/100ème de seconde (et la séquence est mesurée à 100 unités par seconde).
- 500 : l'échelle temporelle est de 1/500ème de seconde (et la séquence est mesurée à 500 unités par seconde).
- 0 : Director utilise l'échelle temporelle de l'animation en cours de lecture.

Définissez `digitalVideoTimeScale` pour accéder précisément aux pistes en vous assurant que l'unité temporelle du système pour la vidéo est un multiple de l'unité temporelle de la vidéo numérique. Définissez la propriété `digitalVideoTimeScale` sur une valeur supérieure pour permettre un contrôle plus précis de la lecture de la vidéo.

Exemple

L'instruction suivante définit l'échelle temporelle utilisée par le système pour mesurer la vidéo numérique sur 600 unités par seconde :

```
-- Lingo syntax
_player.digitalVideoTimeScale = 600

// JavaScript syntax
_player.digitalVideoTimeScale = 600;
```

Voir aussi

Lecteur

digitalVideoType

Syntaxe

```
member(whichCastMember).digitalVideoType
the digitalVideoType of member whichCastMember
```

Description

Propriété d'acteur ; indique le format de la vidéo numérique spécifiée. Les valeurs possibles sont `#quickTime` ou `#videoForWindows`.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante teste si l'acteur Evénements du jour est une vidéo numérique QuickTime ou AVI (Audio-Video Interleaved) et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put member("Today's Events").digitalVideoType
```

```
// JavaScript syntax
put ( member("Today's Events").digitalVideoType );
```

Voir aussi

QuickTimeVersion()

direction

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).emitter.direction
```

Description

Propriété 3D d'émetteur ; vecteur indiquant la direction dans laquelle les particules d'un système sont émises. Un système de particules est une ressource de modèle de type `#particle`.

La principale direction de l'émission des particules est le vecteur défini par la propriété `direction` de l'émetteur. Toutefois, la direction de l'émission d'une particule donnée dévie de ce vecteur selon un angle aléatoire compris entre 0 et la valeur de la propriété `angle (3D)` de l'émetteur.

La définition de la propriété `direction` sur la valeur `vector(0,0,0)` entraîne l'émission des particules dans toutes les directions.

La valeur par défaut de cette propriété est `vector(1,0,0)`.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. L'instruction suivante attribue à la propriété `direction` de l'émetteur de systèmeThermique la valeur `vector(1, 0, 0)`, ce qui entraîne l'émission des particules de systèmeThermique dans une région conique dont l'axe est l'axe des x de l'univers 3D.

```
-- Lingo syntax
member("Fires").modelResource("ThermoSystem").emitter.direction = vector(1,0,0)

// JavaScript syntax
member("Fires").getProp("modelResource", 1).getPropRef("emitter").direction =
vector(1,0,0);
```

Voir aussi

emitter, angle (3D)

directionalColor

Syntaxe

```
member(whichCastmember).directionalColor
```

Description

Propriété 3D d'acteur ; indique la couleur RVB de la lumière directionnelle par défaut de l'acteur.

La valeur par défaut de cette propriété est rgb(255, 255, 255).

Exemple

L'instruction suivante définit la propriété `directionalColor` de l'acteur Pièce sur la valeur `rgb(0, 255, 0)`. La lumière directionnelle par défaut de l'acteur est verte. Cette propriété peut également être définie dans l'Inspecteur des propriétés.

```
-- Lingo syntax
member("Room").directionalcolor = rgb(0, 255, 0)

// JavaScript syntax
member("Room").directionalcolor = color(0, 255, 0);
```

Voir aussi

`directionalPreset`

directionalPreset

Syntaxe

```
member(whichCastmember).directionalPreset
```

Description

Propriété 3D d'acteur ; indique la direction depuis laquelle provient la lumière directionnelle par défaut, par rapport à la caméra de l'image-objet.

La modification de la valeur de cette propriété entraîne la modification des propriétés `position` et `rotation` de la transformation de la lumière.

Les valeurs possibles de `directionalPreset` sont les suivantes :

- `#topLeft`
- `#topCenter`
- `#topRight`
- `#middleLeft`
- `#middleCenter`
- `#middleRight`
- `#bottomLeft`
- `#bottomCenter`
- `#bottomRight`
- `#None`

La valeur par défaut de cette propriété est `#topCenter`.

Exemple

L'instruction suivante attribue à la propriété `directionalPreset` de l'acteur Pièce la valeur `#middleCenter`. La lumière par défaut de Pièce est pointée vers le centre de la vue actuelle de la caméra de l'image-objet. Cette propriété peut également être définie dans l'Inspecteur des propriétés.

```
-- Lingo syntax
member("Room").directionalpreset = #middleCenter
```

```
// JavaScript syntax
member("Room").directionalpreset = symbol("middleCenter");
```

Voir aussi

directionalColor

directToStage

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.directToStage

// JavaScript syntax
memberOrSpriteObjRef.directToStage;
```

Description

Propriété d'acteur et d'image-objet ; détermine la couche sur laquelle un acteur vidéo numérique, GIF animé, forme vectorielle, 3D, Windows Media ou Flash Asset est lu.

Si cette propriété présente la valeur `TRUE` (1), l'acteur est lu devant toutes les autres couches de la scène et les effets d'encre n'ont aucun effet.

Si cette propriété reçoit la valeur `FALSE` (0), l'acteur peut apparaître dans n'importe quelle couche des plans d'animation de la scène et les effets d'encre affectent l'aspect de l'image-objet.

- Utilisez la syntaxe `member(`*`quelActeur`*`).directToStage` pour les vidéos numériques ou les GIF animés.
- Utilisez la syntaxe `sprite(`*`quelleImageObjet`*`.directToStage` pour Flash ou les formes vectorielles.
- Utilisez n'importe quelle syntaxe pour les acteurs ou les images-objets 3D.

L'utilisation de cette propriété améliore les performances de lecture des acteurs ou images-objets.

Aucun autre acteur ne peut apparaître devant une image-objet `directToStage`. Les effets d'encre n'ont également aucun effet sur l'aspect d'une image-objet `directToStage`.

Lorsque la propriété `directToStage` d'une image-objet présente la valeur `TRUE`, Director dessine l'image-objet directement à l'écran sans la composer préalablement dans sa mémoire tampon hors écran. Le résultat peut être semblable à un effet de traces sur la scène.

Vous pouvez réactualiser une zone contenant des traces en activant/désactivant la propriété `directToStage`, à l'aide d'une transition plein écran ou du passage d'une autre image-objet sur cette zone. Sous Windows, si vous ne le faites pas, vous pouvez passer à un écran semblable sans que la vidéo disparaisse complètement.

Vous pouvez voir un exemple d'utilisation de `directToStage` dans une animation en consultant l'animation QT and Flash du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante entraîne la lecture de l'animation QuickTime Résidents sur la couche supérieure de la scène :

```
-- Lingo syntax
member("The Residents").directToStage = 1

// JavaScript syntax
member("The Residents").directToStage = 1;
```

disableImagingTransformation

Syntaxe

```
-- Lingo syntax
_player.disableImagingTransformation

// JavaScript syntax
_player.disableImagingTransformation;
```

Description

Propriété de lecteur ; détermine si Director tient automatiquement compte du défilement ou du zoom de la scène lors de la capture de l'image de la scène. Lecture/écriture.

Lorsque cette propriété présente la valeur `TRUE`, elle empêche Director de prendre automatiquement en compte le défilement ou le zoom de la scène lorsque la propriété `image` est utilisée pour obtenir l'image de la scène. Le zoom et le défilement de la scène affectent l'aspect de l'image capturée à l'aide de la propriété `image`.

Lorsque cette propriété présente la valeur `FALSE`, Director capture toujours l'image de la scène comme si la fenêtre de la scène était à 100 % et n'était pas éloignée du centre de la fenêtre de la scène. `FALSE` est la valeur par défaut.

Exemple

L'instruction suivante définit la propriété `disableImagingTransformation` sur la valeur `TRUE` :

```
-- Lingo syntax
_player.disableImagingTransformation = TRUE

// JavaScript syntax
_player.digitalVideoTimeScale = true;
```

Voir aussi

`image (image)`, `Lecteur`

displayFace

Syntaxe

```
member(whichTextCastmember).displayFace
member(which3DCastmember).modelResource(whichModelResource).displayFace
```

Description

Propriété 3D de texte ; liste linéaire indiquant la ou les faces du texte 3D à afficher. Les valeurs possibles sont `#front`, `#tunnel` et `#back`. Vous pouvez afficher n'importe quelle combinaison de faces et la liste peut être dans n'importe quel ordre.

La valeur par défaut de cette propriété est `[#front, #back, #tunnel]`.

Pour les acteurs texte, il s'agit d'une propriété d'acteur. Pour le texte extrudé d'un acteur 3D, il s'agit d'une propriété de ressource de modèle.

Exemple

Dans l'exemple suivant, l'acteur Panneau est un acteur texte. L'instruction suivante définit la propriété `displayFace` de Panneau sur la valeur `[#tunnel]`. Lorsque Panneau est affiché en mode 3D, ses faces avant et arrière n'apparaissent pas.

```
-- Lingo syntax
member("Rugsign").displayFace = [#tunnel]

// JavaScript syntax
member("Rugsign").displayFace = list( symbol("tunnel") );
```

Dans l'exemple suivant, la ressource du modèle Slogan correspond à du texte extrudé. L'instruction suivante donne à la propriété `displayFace` de la ressource de modèle de Slogan la valeur `[#back, #tunnel]`. La face avant de Slogan n'est pas tracée.

```
-- Lingo syntax
member("scene").model("Slogan").resource.displayFace = [#back, #tunnel]
```

Voir aussi

`extrude3D`, `displayMode`

displayMode

Syntaxe

```
member(whichTextCastmember).displayMode
```

Description

Propriété d'acteur texte ; spécifie si le texte est rendu en 2D ou en 3D.

Lorsque cette propriété présente la valeur `#Mode3D`, le texte s'affiche en 3D. Vous pouvez définir les propriétés 3D du texte (telles que `displayFace` et `bevelDepth`), ainsi que les propriétés de texte habituelles (telles que `text` et `font`). L'image-objet contenant cet acteur devient une image-objet 3D.

Lorsque cette propriété présente la valeur `#ModeNormal`, le texte s'affiche en 2D.

La valeur par défaut de cette propriété est `#ModeNormal`.

Exemple

Dans l'exemple suivant, l'acteur Logo est un acteur texte. L'instruction suivante entraîne l'affichage de Logo en 3D.

```
-- Lingo syntax
member("Logo").displayMode = #mode3D

// JavaScript syntax
member("Logo").displayMode = symbol("mode3D");
```

Voir aussi

`extrude3D`

displayRealLogo

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.displayRealLogo

// JavaScript syntax
memberOrSpriteObjRef.displayRealLogo;
```

Description

Propriété d'acteur ou d'image-objet RealMedia ; permet de savoir ou de définir si le logo RealNetworks® est affiché (TRUE) ou non (FALSE). Lorsque cette propriété présente la valeur TRUE, le logo RealNetworks apparaît dans la fenêtre RealMedia au début du flux et lorsque la vidéo est arrêtée ou rembobinée.

La valeur par défaut de cette propriété est TRUE (1). Les valeurs entières autres que 1 ou 0 sont considérées comme TRUE.

Exemple

Les exemples suivants indiquent que la propriété displayRealLogo de l'image-objet 2 et de l'acteur Real a pour valeur TRUE, ce qui signifie que le logo RealNetworks s'affiche au début de la lecture de l'animation et lorsque cette dernière est arrêtée ou rembobinée.

```
-- Lingo syntax
put(sprite(2).displayRealLogo) -- 1
put(member("Real").displayRealLogo) -- 1

// JavaScript syntax
trace(sprite(2).displayRealLogo); // 1
put(member("Real").displayRealLogo); // 1
```

Les exemples suivants attribuent à la propriété displayRealLogo de l'image-objet 2 et de l'acteur Real la valeur FALSE, ce qui signifie que le logo RealNetworks ne s'affiche pas.

```
-- Lingo syntax
sprite(2).displayRealLogo = FALSE
member("Real").displayRealLogo = FALSE

// JavaScript syntax
sprite(2).displayRealLogo = 0;
member("Real").displayRealLogo = 0;
```

displayTemplate

Syntaxe

```
-- Lingo syntax
_movie.displayTemplate

// JavaScript syntax
_movie.displayTemplate;
```

Description

Propriété d'animation ; permet d'accéder à la liste des propriétés qui s'appliquent à la fenêtre dans laquelle une animation est en cours de lecture. Lecture/écriture.

La propriété displayTemplate donne accès aux propriétés de l'objet fenêtre qui servent à définir les paramètres de fenêtre par défaut. Par conséquent, la propriété displayTemplate est utilisée sur l'objet animation pour renvoyer ou définir les paramètres de fenêtre par défaut, tout comme les propriétés appearanceOptions et titlebarOptions sont utilisées sur l'objet fenêtre.

La propriété displayTemplate permet d'accéder aux propriétés suivantes.

Propriété	Description
appearanceOptions	Liste de propriétés stockant les options d'aspect relatives à une fenêtre. Les options d'aspect sont `mask`, `border`, `metal`, `dragRegionMask`, `shadow` et `liveresize`. Pour plus d'informations, reportez-vous à l'entrée `appearanceOptions`.
dockingEnabled	Détermine si une animation dans une fenêtre est ancrable lors de son ouverture en phase de création. Si cette propriété présente la valeur `TRUE`, la fenêtre est ancrable. Si elle présente la valeur `FALSE`, la fenêtre n'est pas ancrable. La valeur par défaut est `FALSE`. Pour plus d'informations, reportez-vous à l'entrée `dockingEnabled`.
resizable	Détermine si une fenêtre est redimensionnable. Si cette propriété présente la valeur `TRUE`, la fenêtre est redimensionnable. Si elle présente la valeur `FALSE`, la fenêtre n'est pas redimensionnable. La valeur par défaut est `TRUE`. Pour plus d'informations, reportez-vous à l'entrée `resizable`.
title	Renvoie ou définit le titre du modèle d'affichage. Pour plus d'informations, reportez-vous à l'entrée `title`.
titlebarOptions	Liste de propriétés stockant les options de barre de titre relatives à une fenêtre. Les options de barre de titre sont `icon`, `visible`, `closebox`, `minimizebox`, `maximizebox` et `sideTitlebar`. Pour plus d'informations, reportez-vous à l'entrée `titlebarOptions`.
systemTrayIcon	(Microsoft Windows uniquement) Détermine si une icône est associée à la fenêtre dans la barre d'état système du Bureau d'un utilisateur.
systemTrayTooltip	(Microsoft Windows uniquement) Détermine la chaîne qui apparaît dans l'info-bulle associée à l'icône de la barre d'état système.
type	Renvoie ou définit le type d'une fenêtre. Si le type d'une fenêtre est défini, toutes les propriétés relatives à ce type de fenêtre sont également définies. Les types de fenêtres sont `tool`, `document` et `dialog`. Pour plus d'informations, reportez-vous à l'entrée `type`.

Exemple

Les instructions suivantes affichent les propriétés `displayTemplate` et leurs valeurs correspondantes dans la fenêtre Messages.

```
-- Lingo syntax
trace(_movie.displayTemplate)

// JavaScript syntax
trace(_movie.displayTemplate);
```

Les instructions suivantes définissent différentes propriétés `displayTemplate`.

```
-- Lingo syntax
_movie.displayTemplate.dockingEnabled = TRUE
_movie.displayTemplate.resizable = FALSE
_movie.displayTemplate.appearanceOptions.mask = member("mask")
_movie.displayTemplate.titlebarOptions.sideTitlebar = TRUE

// JavaScript syntax
_movie.displayTemplate.dockingEnabled = true;
_movie.displayTemplate.resizable = false;
_movie.displayTemplate.appearanceOptions.mask = member("mask");
_movie.displayTemplate.titlebarOptions.sideTitlebar = true;
```

Voir aussi

`appearanceOptions`, `dockingEnabled`, `Animation`, `resizable`, `systemTrayIcon`, `title (fenêtre)`, `titlebarOptions`, `type (fenêtre)`, `Fenêtre`

distribution

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).emitter.distribution
```

Description

Propriété 3D d'émetteur ; indique la façon dont les particules d'un système de particules sont réparties dans la région de l'émetteur lors de leur création. Les valeurs possibles de cette propriété sont `#gaussian` ou `#linear`. La valeur par défaut est `#linear`.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. L'instruction suivante attribue à la propriété `distribution` de l'émetteur de systèmeThermique la valeur `#linear`, ce qui entraîne la répartition uniforme des particules de systèmeThermique dans leur région d'origine lorsqu'elles sont créées.

```
-- Lingo syntax
member("Fires").modelResource("ThermoSystem").emitter.distribution = #linear

// JavaScript syntax
member("Fires").getProp("modelResource", 1).getPropRef("emitter").distribution =
symbol("linear");
```

Voir aussi

`emitter`, `region`

dither

Syntaxe

```
-- Lingo syntax
memberObjRef.dither

// JavaScript syntax
memberObjRef.dither;
```

Description

Propriété d'acteur bitmap ; trame l'acteur lorsque ce dernier s'affiche avec un codage de couleur sur 8 bits ou moins (256 couleurs) si l'écran doit afficher une nuance de couleurs n'existant pas dans l'acteur (`TRUE`) ou indique à Director de choisir la couleur la plus proche parmi celles de la palette en cours (`FALSE`).

Pour des raisons de performances et de qualité, n'attribuez à `dither` la valeur `TRUE` que lorsqu'une qualité d'affichage supérieure est nécessaire. Le tramage est plus lent qu'une conversion des couleurs et des détails ennuyeux peuvent être plus apparents lors de l'animation sur une image tramée.

Si le codage des couleurs est supérieur à 8 bits, cette propriété n'a aucun effet.

Voir aussi

`depth (bitmap)`

dockingEnabled

Syntaxe

```
-- Lingo syntax
_movie.displayTemplate.dockingEnabled
windowObjRef.dockingEnabled

// JavaScript syntax
_movie.displayTemplate.dockingEnabled;
windowObjRef.dockingEnabled;
```

Description

Propriété d'animation et de fenêtre ; indique si une animation dans une fenêtre constitue une fenêtre ancrable lors de son ouverture en phase de création. Lecture/écriture.

Cette propriété n'est pas directement accessible à partir d'un objet animation ; vous y accédez par l'intermédiaire de la propriété `displayTemplate` de l'objet animation.

La valeur par défaut de cette propriété est `FALSE`, ce qui signifie qu'une animation dans une fenêtre n'est pas ancrable lors de son ouverture en phase de création. Si cette propriété présente la valeur `TRUE`, la valeur de la propriété `type` de l'objet fenêtre détermine le mode d'affichage de la fenêtre en phase de création.

- Si la propriété `dockingEnabled` reçoit la valeur `TRUE` et que la propriété `type` est définie sur `#document`, l'animation dans une fenêtre présente l'aspect et le comportement d'une fenêtre de type Document dans Director. La fenêtre s'affiche dans la zone appelée « espace principal » et pourra s'ancrer avec les fenêtres Scène, Scénario et Distribution, les éditeurs de médias, ainsi que les fenêtres de messages. Toutefois, il vous est impossible de regrouper la fenêtre avec l'une de ces autres fenêtres.

- Si la propriété `dockingEnabled` reçoit la valeur `TRUE` et que la propriété `type` est définie sur `#tool`, l'animation dans une fenêtre présente l'aspect et le comportement d'une fenêtre de type Outil dans Director. Vous pouvez regrouper cette fenêtre avec toutes les fenêtres de type Outil, à l'exception de l'Inspecteur des propriétés et de la Palette des outils.

- Si la propriété `dockingEnabled` reçoit la valeur `TRUE` et que la propriété `type` est définie sur `#dialog`, le type est ignoré et la fenêtre constitue une fenêtre de création.

Cette propriété est ignorée dans les projections.

Exemple

Les instructions suivantes définissent la propriété `dockingEnabled` sur `TRUE`.

```
-- Lingo syntax
_movie.displayTemplate.dockingEnabled = TRUE -- from the Movie object
window("Instructions").dockingEnabled = TRUE -- from the Window object

// JavaScript syntax
_movie.displayTemplate.dockingEnabled = true; // from the Movie object
window("Instructions").dockingEnabled = true; // from the Window object
```

Voir aussi

appearanceOptions, displayTemplate, Animation, titlebarOptions, type (fenêtre), Fenêtre

domain

Syntaxe

```
-- Lingo syntax
dvdObjRef.domain

// JavaScript syntax
dvdObjRef.domain;
```

Description

Propriété de DVD ; renvoie un symbole indiquant le domaine en cours. Lecture seule.

Exemple

L'instruction suivante renvoie le domaine en cours :

```
-- Lingo syntax
trace (member(1).domain)-- #title

// JavaScript syntax
trace (member(1).domain);// #title
```

Voir aussi

DVD

doubleClick

Syntaxe

```
-- Lingo syntax
_mouse.doubleClick

// JavaScript syntax
_mouse.doubleClick;
```

Description

Propriété de souris ; indique si deux clics de souris effectués dans un laps de temps prédéfini pour un double-clic correspondent à un double-clic plutôt qu'à deux clics simples (TRUE) ou, s'ils n'ont pas eu lieu dans le temps déterminé, les traite comme des clics simples (FALSE). Lecture seule.

Exemple

L'instruction suivante envoie la tête de lecture sur l'image Entrée de l'offre lorsque l'utilisateur double-clique sur le bouton de la souris :

```
-- Lingo syntax
if (_mouse.doubleClick) then
    _movie.go("Enter Bid")
end if

// JavaScript syntax
if (_mouse.doubleClick) {
    _movie.go("Enter Bid");
}
```

Voir aussi

clickLoc, clickOn, Souris

drag

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).drag
```

Description

Propriété 3D de ressource de modèle `#particle` ; indique le pourcentage de vélocité de chaque particule qui est perdu à chaque étape de simulation. La valeur de cette propriété peut être comprise entre 0 (aucune perte de vélocité) et 100 (toute la vélocité est perdue et la particule arrête de bouger). La valeur par défaut est 0.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. L'instruction suivante attribue à la propriété `drag` de systèmeThermique la valeur 5, ce qui applique une forte résistance au mouvement des particules de systèmeThermique et les empêche de se déplacer très loin.

```
-- Lingo syntax
member("Fires").modelResource("ThermoSystem").drag = 5

// JavaScript syntax
member("Fires").getProp("modelResource", 1).drag = 5;
```

Voir aussi

wind, gravity

drawRect

Syntaxe

```
-- Lingo syntax
windowObjRef.drawRect

// JavaScript syntax
windowObjRef.drawRect;
```

Description

Propriété de fenêtre ; identifie les coordonnées rectangulaires de la scène de l'animation qui apparaît dans une fenêtre. Lecture/écriture.

Les coordonnées sont données sous la forme d'un rectangle, avec les entrées dans l'ordre gauche, haut, droite et bas.

Cette propriété est pratique pour mettre les animations à l'échelle ou faire un panorama, mais ne redimensionne pas les acteurs texte et champ. La mise à l'échelle des bitmaps peut affecter la performance.

Exemple

L'instruction suivante affiche les coordonnées actuelles de la fenêtre d'animation intitulée Tableau de commande :

```
-- Lingo syntax
put(window("Control Panel").drawRect)
```

```
// JavaScript syntax
put(window("Control Panel").drawRect);
```

L'instruction suivante attribue au rectangle de l'animation les valeurs du rectangle intitulé rectangleDanimation. La partie de l'animation contenue dans le rectangle correspond à la zone affichée dans la fenêtre.

```
-- Lingo syntax
movieRectangle = rect(10, 20, 200, 300)
window("Control Panel").drawRect = movieRectangle

// JavaScript syntax
var movieRectangle = rect(10, 20, 200, 300);
window("Control Panel").drawRect = movieRectangle;
```

Les lignes suivantes entraînent le remplissage de la zone principale du moniteur par la scène :

```
-- Lingo syntax
_movie.stage.drawRect = _system.deskTopRectList[1]
_movie.stage.rect = _system.deskTopRectList[1]

// JavaScript syntax
_movie.stage.drawRect = _system.deskTopRectList[1];
_movie.stage.rect = _system.deskTopRectList[1];
```

Voir aussi

rect(), Fenêtre

dropShadow

Syntaxe

```
-- Lingo syntax
memberObjRef.dropShadow

// JavaScript syntax
memberObjRef.dropShadow;
```

Description

Propriété d'acteur ; détermine la taille de l'ombre, en pixels, du texte d'un acteur champ.

Exemple

L'instruction suivante attribue à l'ombre portée de l'acteur champ Commentaire une taille de 5 pixels :

```
--Lingo syntax
member("Comment").dropShadow = 5

// JavaScript syntax
member("Comment").dropShadow = 5;
```

duration (3D)

Syntaxe

```
member(whichCastmember).motion(whichMotion).duration
motionObjectReference.duration
```

Description

Propriété 3D ; permet d'obtenir le temps, en millisecondes, nécessaire à la lecture complète du mouvement spécifié dans le paramètre *quelMouvement*. Cette propriété est toujours supérieure ou égale à 0.

Exemple

L'instruction suivante indique la durée, en millisecondes, du mouvement Coup.

```
-- Lingo syntax
put member("GbMember").motion("Kick").duration
-- 5100.0000

// JavaScript syntax
put( member("GbMember").getProp("motion", 1).duration );
// 5100.0000
```

Voir aussi

motion, currentTime (3D), play() (3D), queue() (3D)

duration (DVD)

Syntaxe

```
-- Lingo syntax
dvdObjRef.duration

// JavaScript syntax
dvdObjRef.duration;
```

Description

Propriété de DVD ; renvoie la durée totale du titre en millisecondes. Lecture seule.

Exemple

L'instruction suivante renvoie la durée du titre en cours :

```
--Lingo syntax
trace (member(1).duration)-- 1329566

// JavaScript syntax
trace (member(1).duration);// 1329566
```

Voir aussi

DVD

duration (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.duration

// JavaScript syntax
memberObjRef.duration;
```

Description

Propriété d'acteur ; détermine la durée des acteurs Shockwave Audio (SWA), transition, Windows Media et QuickTime spécifiés.

• Si *quelActeur* est un fichier audio lu en flux continu, cette propriété indique la durée du son. La propriété `duration` renvoie 0 jusqu'au démarrage de la lecture en flux continu. La définition de la propriété `preLoadTime` sur une valeur d'une seconde autorise le renvoi de la durée réelle.

• Si *quelActeur* est un acteur vidéo numérique, cette propriété indique la durée de la vidéo numérique. La valeur est exprimée en battements.

• Si *quelActeur* est un acteur transition, cette propriété indique la durée de la transition. La valeur de la transition est exprimée en millisecondes. Au cours de la lecture, ce paramètre a le même effet que le paramètre Durée de la boîte de dialogue Préférences de l'image : Transition.

Cette propriété peut être testée pour tous les acteurs qui la supportent, mais n'est définissable que pour les transitions.

Vous pouvez voir un exemple d'utilisation de `duration` dans une animation en consultant l'animation QT and Flash du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

Si l'acteur SWA Louie Prima a été préchargé, l'instruction suivante affiche la durée du son dans l'acteur champ Affichage de la durée :

```
-- Lingo syntax
on exitFrame
    if member("Louie Prima").state = 2 then
        member("Duration Displayer").text = string(member("Louie Prima").duration)
    end if
end

// JavaScript syntax
function exitFrame() {
    if (member("Louie Prima").state == 2) {
        member("Duration Displayer").text = member("Louie Prima").duration.toString()
    }
}
```

duration (RealMedia, SWA)

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.duration

// JavaScript syntax
memberOrSpriteObjRef.duration;
```

Description

Propriété d'image-objet audio ou d'acteur RealMedia ou Shockwave ; renvoie la durée d'un flux RealMedia ou Shockwave Audio en millisecondes. La durée du flux reste inconnue jusqu'au démarrage de la lecture de l'acteur. Si le flux provient d'un flux en direct ou n'a pas été lu, la valeur de cette propriété est 0. Cette propriété peut être testée, mais non définie.

Exemple

Les exemples suivants indiquent que l'état du flux RealMedia de l'image-objet 2 et de l'acteur Real est 100 500 millisecondes (100,50 secondes).

```
-- Lingo syntax
put(sprite(2).duration) -- 100500
put(member("Real").duration) -- 100500

// JavaScript syntax
put(sprite(2).duration); // 100500
put(member("Real").duration); // 100500
```

Voir aussi

play() (RealMedia, SWA, Windows Media), seek(), currentTime (RealMedia)

editable

Syntaxe

```
-- Lingo syntax
spriteObjRef.editable

// JavaScript syntax
spriteObjRef.editable;
```

Description

Propriété d'image-objet ; détermine si une image-objet spécifiée est modifiable sur la scène (TRUE) ou non (FALSE). Lecture/écriture.

Lorsque la propriété d'acteur est définie, le paramètre est appliqué à toutes les images-objets contenant le champ.

Lorsque cette propriété est définie, seule l'image-objet spécifiée est affectée.

Vous pouvez également rendre une image-objet champ modifiable à l'aide de l'option Modifiable dans la boîte de dialogue Propriétés de l'acteur champ.

Vous pouvez rendre une image-objet champ modifiable à l'aide de l'option Modifiable dans le scénario.

Pour que la valeur définie par un script dure au-delà de l'image-objet en cours, l'image-objet doit être contrôlée par un script.

Exemple

Le gestionnaire suivant fait de la piste d'image-objet un esclave et rend l'image-objet champ modifiable :

```
-- Lingo syntax
on myNotes
    _movie.puppetSprite(5, TRUE)
    sprite(5).editable = TRUE
end

// JavaScript syntax
function myNotes() {
    _movie.puppetSprite(5, true);
    sprite(5).editable = true;
}
```

L'instruction suivante vérifie si une image-objet champ est modifiable et affiche un message le cas échéant :

```
-- Lingo syntax
if (sprite(13).editable = TRUE) then
    member("Notice").text = "Please enter your answer below."
end if

// JavaScript syntax
if (sprite(13).editable == true) {
    member("Notice").text = "Please enter your answer below.";
}
```

Voir aussi

Image-objet

editShortCutsEnabled

Syntaxe

```
-- Lingo syntax
_movie.editShortCutsEnabled

// JavaScript syntax
_movie.editShortCutsEnabled;
```

Description

Propriété d'animation ; détermine si les opérations Couper, Copier et Coller, ainsi que leurs raccourcis clavier, fonctionnent dans l'animation actuelle. Lecture/écriture.

Ces opérations de texte fonctionnent si elles sont définies sur `TRUE`. Ces opérations ne sont pas autorisées si elles sont définies sur `FALSE`. La valeur par défaut est `TRUE` pour les animations créées dans Director 8 et ses versions ultérieures, et `FALSE` pour les animations créées dans les versions précédentes de Director.

Exemple

L'instruction suivante désactive les opérations Couper, Copier et Coller :

```
-- Lingo syntax
_movie.editShortCutsEnabled = 0

// JavaScript syntax
_movie.editShortCutsEnabled = 0;
```

Voir aussi

Animation

elapsedTime

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.elapsedTime

// JavaScript syntax
soundChannelObjRef.elapsedTime;
```

Description

Propriété de piste audio ; indique la durée écoulée, en millisecondes, depuis le début de la lecture de l'acteur son en cours dans une piste audio. Lecture seule.

La durée écoulée commence à 0 au début de la lecture du son et augmente à mesure de la lecture, indépendamment de la lecture en boucle, du paramètre `currentTime` ou de toute autre manipulation. Utilisez `currentTime` pour tester la durée absolue en cours au sein du son.

La valeur de cette propriété est un nombre à virgule flottante, ce qui permet de mesurer la lecture du son en fraction de millisecondes.

Exemple

Ce gestionnaire `idle` affiche le temps écoulé pour la piste audio 4 dans un champ de la scène en période d'inactivité :

```
-- Lingo syntax
on idle
    member("time").text = string(sound(4).elapsedTime)
end idle

// JavaScript syntax
function idle() {
    member("time").text = sound(4).elapsedTime.toString();
}
```

Voir aussi

currentTime (Sprite), Piste audio

emissive

Syntaxe

```
member(whichCastmember).shader(whichShader).emissive
member(whichCastmember).model(whichModel).shader.emissive
member(whichCastmember).model(whichModel).shaderList{[index]}.emissive
```

Description

Propriété 3D de matériau `#standard` ; ajoute de la lumière au matériau indépendamment de l'éclairage de la scène. Par exemple, un modèle utilisant un matériau dont la propriété `emissive` a pour valeur `rgb(255, 255, 255)` apparaîtra comme étant illuminé par une lumière blanche, même si la scène ne contient aucune lumière. Cependant, le modèle n'éclaire par les autres modèles et n'apporte aucune lumière à la scène.

La valeur par défaut de cette propriété est `rgb(0, 0, 0)`.

Exemple

L'instruction suivante attribue à la propriété `emissive` du matériau Globe la valeur `rgb(255, 0, 0)`. Les modèles utilisant ce matériau apparaissent comme étant éclairés par une lumière rouge :

```
-- Lingo syntax
member("MysteryWorld").shader("Globe").emissive = rgb(255, 0, 0)

// JavaScript syntax
member("MysteryWorld").getProp("shader", 1).emissive = color(255, 0, 0);
```

Voir aussi

silhouettes

emitter

Syntaxe

```
--Lingo Usage
member(whichCastmember).modelResource(whichModelResource).emitter.numParticles
member(whichCastmember).modelResource(whichModelResource).emitter.mode
member(whichCastmember).modelResource(whichModelResource).emitter.loop
member(whichCastmember).modelResource(whichModelResource).emitter.direction
member(whichCastmember).modelResource(whichModelResource).emitter.region
member(whichCastmember).modelResource(whichModelResource).emitter.distribution
member(whichCastmember).modelResource(whichModelResource).emitter.angle
member(whichCastmember).modelResource(whichModelResource).emitter.path
member(whichCastmember).modelResource(whichModelResource).emitter.pathStrength
member(whichCastmember).modelResource(whichModelResource).emitter.minSpeed
member(whichCastmember).modelResource(whichModelResource).emitter.maxSpeed

// JavaScript usage
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("em
itter").numParticles
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("em
itter").mode
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("em
itter").loop
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("em
itter").direction
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("em
itter").region
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("em
itter").distribution
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("em
itter").angle
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("em
itter").path
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("em
itter").pathStrength
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("em
itter").minSpeed
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("em
itter").maxSpeed
```

Description

Elément 3D de système de particules ; contrôle la propulsion initiale de particules à partir d'une ressource de modèle de type `#particle`.

La section « Voir aussi » de cette entrée contient une liste complète des propriétés d'émetteur. Pour plus d'informations, consultez les entrées des différentes propriétés.

Voir aussi

`numParticles`, `loop (émetteur)`, `direction`, `distribution`, `region`, `angle (3D)`, `path (3D)`, `pathStrength`, `minSpeed`, `maxSpeed`

emulateMultibuttonMouse

Syntaxe

```
-- Lingo syntax
_player.emulateMultibuttonMouse

// JavaScript syntax
_player.emulateMultibuttonMouse;
```

Description

Propriété de lecteur ; détermine si une animation considère un clic de la souris combiné à l'activation de la touche Ctrl d'un Mac comme un clic avec le bouton droit de la souris sous Windows (`TRUE`) ou non (`FALSE`, valeur par défaut). Lecture/écriture.

L'exécution d'un clic avec le bouton droit n'a aucun équivalent direct sur le Mac.

La définition de cette propriété sur la valeur `TRUE` vous permet d'apporter des réponses cohérentes aux clics de la souris dans les animations lues sur différentes plates-formes.

Exemple

L'instruction suivante définit la propriété `emulateMultibuttonMouse` sur `TRUE` :

```
-- Lingo syntax
_player.emulateMultibuttonMouse = TRUE

// JavaScript syntax
_player.emulateMultibuttonMouse = true;
```

Voir aussi

Lecteur

enabled

Syntaxe

```
the enabled of menuItem whichItem of menu whichMenu
```

Description

Propriété d'élément de menu ; détermine si l'élément de menu spécifié par *quelElément* s'affiche en texte normal et peut être sélectionné (`TRUE`, valeur par défaut) ou apparaît en texte grisé et ne peut pas être sélectionné (`FALSE`).

L'expression *quelElément* peut être le nom d'un élément de menu ou son numéro. L'expression *quelMenu* peut être le nom d'un menu ou son numéro.

Cette propriété peut être testée et définie.

Remarque : *les menus ne sont pas disponibles dans Shockwave Player.*

Exemple

Le gestionnaire suivant active ou désactive tous les éléments du menu spécifié. L'argument leMenu spécifie le menu ; l'argument `Setting` spécifie `TRUE` ou `FALSE`. Par exemple, l'instruction d'appel `ableMenu("Spécial", FALSE)` désactive tous les éléments du menu Spécial.

```
-- Lingo syntax
on ableMenu theMenu, vSetting
    set n = the number of menuItems of menu theMenu
    repeat with i = 1 to n
        set the enabled of menuItem i of menu theMenu to vSetting
    end repeat
end ableMenu
```

```
// JavaScript syntax
function ableMenu (theMenu, vSetting){
    n = _menuBar.menu[theMenu].item.count;
    for( i = 1 ; i <= n ; i++)
        _menuBar.menu[theMenu].item[i].enabled = vSetting;
}
```

Voir aussi

name (propriété de menu), number (menus), checkMark, script, number (éléments de menu)

enabled (collision)

Syntaxe

```
member(whichCastmember).model(whichModel).collision.enabled
```

Description

Propriété 3D de collision ; permet de savoir ou de définir si les collisions sont détectées au niveau des modèles (TRUE) ou non (FALSE). La valeur FALSE désactive temporairement le modificateur collision sans le supprimer du modèle.

La valeur par défaut de cette propriété est TRUE.

Exemple

L'instruction suivante active le modificateur collision pour le modèle Boîte :

```
-- Lingo syntax
member("3d world").model("box").collision.enabled = TRUE

// JavaScript syntax
member("3d world").getProp("model",1).getPropRef("collision").enabled = true;
```

Voir aussi

addModifier, collision (modificateur), modifier

enabled (brouillard)

Syntaxe

```
member(whichCastmember).camera(whichCamera).fog.enabled
sprite(whichSprite).camera{(index)}.fog.enabled
```

Description

Propriété 3D de caméra ; indique si la caméra ajoute du brouillard à la vue. La valeur par défaut de cette propriété est FALSE.

Exemple

L'instruction suivante crée du brouillard dans la vue de la caméra vueDeLaBaie :

```
member("MyYard").camera("BayView").fog.enabled = TRUE
```

Voir aussi

fog

enabled (sds)

Syntaxe

```
member(whichCastmember).model(whichModel).sds.enabled
```

Description

Propriété 3D de modificateur `sds` ; indique si le modificateur `sds` associé à un modèle est utilisé par ce dernier.

La valeur par défaut de cette propriété est `TRUE`.

Une tentative d'ajout du modificateur `sds` à un modèle déjà associé au modificateur `inker` ou `toon` échoue sans message d'erreur. De même, une tentative d'ajout du modificateur `inker` ou `toon` à un modèle déjà associé au modificateur `sds` échoue sans message d'erreur. Soyez vigilant lorsque vous utilisez le modificateur `sds` avec le modificateur `lod`. Pour plus d'informations, reportez-vous à l'entrée `sds (modificateur)`.

Exemple

L'instruction suivante active le modificateur sds associé au modèle Bébé :

```
member("Scene").model("Baby").sds.enabled = TRUE
```

Voir aussi

sds (modificateur), modifier, addModifier

enableFlashLingo

Syntaxe

```
-- Lingo syntax
_movie.enableFlashLingo

// JavaScript syntax
_movie.enableFlashLingo;
```

Description

Propriété d'animation ; déterminer si une image-objet comportant du contenu Flash peut effectuer des rappels de scripts directs lors de l'utilisation de la méthode Flash `getURL()`. Lecture/écriture.

La méthode Flash `getURL()` charge une nouvelle URL dans une fenêtre de navigateur vide.

Si la propriété `enableFlashLingo` présente la valeur `TRUE`, une image-objet comportant du contenu Flash peut exécuter n'importe quelle commande de script valide (soumise aux règles de sécurité standard définies pour Shockwave Player) lors de l'appel de la méthode `getURL()`.

Si la propriété `enableFlashLingo` est définie sur `FALSE`, une image-objet comportant du contenu Flash n'est pas autorisée à exécuter des commandes de script lors de l'appel de la méthode `getURL()`. La valeur par défaut de cette propriété est `FALSE`.

Cette propriété se révèle utile lors de la création d'une animation affichant un contenu Flash d'origine inconnue, par exemple dans une projection recherchant des fichiers SWF dans un dossier système ou d'une animation intégrant du contenu Shockwave qui accepte une URL relative à un fichier SWF de la part d'un utilisateur final.

Exemple

L'instruction suivante définit la propriété `enableFlashLingo` sur `TRUE` :

```
-- Lingo syntax
_movie.enableFlashLingo = TRUE

// JavaScript syntax
_movie.enableFlashLingo = true;
```

Voir aussi

`Animation`

endAngle

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).endAngle
```

Description

Propriété 3D de ressource de modèle `#cylinder` ou `#sphere` ; indique la quantité dessinée de la sphère ou du cylindre.

La surface d'une sphère est générée en balayant un arc de demi-cercle 2D autour de l'axe des y de la sphère, de `startAngle` à `endAngle`. Si `startAngle` a pour valeur 0 et `endAngle` a pour valeur 360, le résultat est une sphère complète. Pour dessiner une section de sphère, attribuez à `endAngle` une valeur inférieure à 360.

La surface d'un cylindre est générée en balayant une ligne 2D autour de l'axe des y du cylindre, de `startAngle` à `endAngle`. Si `startAngle` a pour valeur 0 et `endAngle` a pour valeur 360, le résultat est un cylindre complet. Pour dessiner une section de cylindre, attribuez à `endAngle` une valeur inférieure à 360.

Le paramètre par défaut de cette propriété est 360.

Exemple

Pour l'exemple suivant, l'acteur monActeur contient un modèle qui utilise la ressource de modèle sphère4, dont la valeur `endAngle` est 310, ce qui laisse une ouverture de 50 degrés. Le gestionnaire `closeSphere` ferme cette ouverture de façon à donner l'impression d'une porte coulissante. La boucle de répétition change la valeur `endAngle` de la sphère d'un degré à la fois. La commande `updateStage` de la boucle de répétition entraîne la mise à jour de la scène après chaque incrément d'un degré.

```
-- Lingo syntax
on closeSphere
    MyAngle = member("MyMember").modelresource("Sphere4").endAngle
    repeat with r = 1 to 50
        MyAngle = MyAngle + 1
        member("MyMember").modelresource("Sphere4").endAngle = MyAngle
        updateStage
```

```
    end repeat
end

// JavaScript syntax
function closeSphere(){
    var MyAngle = member("MyMember").getProp("modelresource", 1).endAngle;
    for( r = 1; r <= 50; r++)
    {
        MyAngle++;
        member("MyMember").getProp("modelresource", 1).endAngle = MyAngle;
        _movie.updateStage();
    }
}
```

Voir aussi

state (3D)

endColor

Syntaxe

```
-- Lingo syntax
memberObjRef.endColor

// JavaScript syntax
memberObjRef.endColor;
```

Description

Propriété d'acteur forme vectorielle ; couleur finale d'un remplissage en dégradé spécifiée sous la forme d'une valeur rvb.

`endColor` est uniquement valide lorsque `fillMode` prend la valeur `#gradient` et que la couleur de départ est définie avec `fillColor`.

Cette propriété peut être testée et définie.

Vous pouvez voir un exemple d'utilisation de la propriété `endColor` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Voir aussi

color(), fillColor, fillMode

endFrame

Syntaxe

```
-- Lingo syntax
spriteObjRef.endFrame

// JavaScript syntax
spriteObjRef.endFrame;
```

Description

Propriété d'image-objet ; renvoie le numéro de la dernière image de l'étendue de l'image-objet. Lecture seule.

Cette propriété est utile pour déterminer l'étendue d'une image-objet spécifique dans le scénario.

Cette propriété est uniquement disponible dans une image contenant l'image-objet. Elle n'est pas applicable à des images-objets situées dans d'autres images de l'animation.

Exemple

L'instruction suivante indique l'image finale de l'image-objet de la piste 5 dans la fenêtre Messages :

```
-- Lingo syntax
put(sprite(5).endFrame)

// JavaScript syntax
put(sprite(5).endFrame);
```

Voir aussi

Image-objet, startFrame

endTime

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.endTime

// JavaScript syntax
soundChannelObjRef.endTime;
```

Description

Propriété de piste audio ; spécifie la position temporelle de fin du son en cours de lecture, en pause ou en file d'attente. Lecture/écriture.

La position temporelle de fin correspond à la position au sein de l'acteur son à laquelle la lecture du son s'arrêtera. Il s'agit d'une valeur à virgule flottante, permettant de mesurer et de contrôler la lecture du son en fractions de millisecondes. La valeur par défaut est la fin normale du son.

Cette propriété n'est définissable sur une valeur différente de la fin normale du son que si elle est transmise en tant que paramètre à l'aide des méthodes `queue()` ou `setPlayList()`.

Exemple

Les instructions suivantes vérifient si l'acteur son Annonce est défini pour une lecture sans interruption dans la piste audio 1 :

```
-- Lingo syntax
if (sound(1).startTime > 0 and sound(1).endTime < member("Jingle").duration) then
_player.alert("Not playing the whole sound.")
end if

// JavaScript syntax
if (sound(1).startTime > 0 && sound(1).endTime < member("Jingle").duration) {
    _player.alert("Not playing the whole sound.");
}
```

Voir aussi

queue(), setPlayList(), Piste audio

environmentPropList

Syntaxe

```
-- Lingo syntax
_system.environmentPropList

// JavaScript syntax
_system.environmentPropList;
```

Description

Propriété système ; contient une liste d'informations concernant l'environnement dans lequel le contenu Director est en cours d'exécution. Lecture seule.

Cette structure permet à Adobe d'ajouter des informations à la propriété `environmentPropList` par la suite, sans affecter les animations existantes.

Les informations sont présentées sous la forme de paires de valeurs et de propriétés pour cette zone.

`#shockMachine`	Valeur `TRUE` ou `FALSE` indiquant si l'animation est lue dans ShockMachine.
`#shockMachineVersion`	Chaîne indiquant le numéro de la version de ShockMachine installée.
`#platform`	Chaîne contenant « Mac,PowerPC » ou « Windows,32 ». Ceci est déterminé d'après le système d'exploitation et le matériel sur lesquels l'animation est exécutée.
`#runMode`	Chaîne contenant « Author » (environnement auteur), « Projector » (projection) ou « Plugin » (module de navigateur Web). Ceci est déterminé en fonction de l'application actuelle dans laquelle l'animation est exécutée.
`#colorDepth`	Nombre entier représentant le codage des couleurs du moniteur sur lequel la scène apparaît. Les valeurs possibles sont 1, 2, 4, 8, 16 ou 32.
`#internetConnected`	Symbole indiquant si l'ordinateur sur lequel l'animation est lue est connecté à Internet. Les valeurs possibles sont `#online` et `#offline`.
`#uiLanguage`	Chaîne indiquant la langue qu'utilise le lecteur pour l'affichage de son interface utilisateur.

`#osLanguage`	Chaîne indiquant la langue native du système d'exploitation de l'ordinateur.
#osVersion	Chaîne. Sous Windows, la propriété #osVersion est renseignée avec les informations obtenues à l'aide de l'appel système GetVersionEx(). Les valeurs de la chaîne proviennent de la structure OSVERSIONINFO : " Windows CE " ou " Windows NT " ou " Windows 2000 " ou " Windows XP " ou " Windows 95 " ou " Windows 98 "ou " Windows ME " dwMajorVersion dwMinorVersion dwOSVersionInfoSize dwPlatformId szCSDVersion Sur Mac, les valeurs de la chaîne proviennent de l'appel Gestalt(gestaltSystemVersion) : " Mac OS " version principale version secondaire sous-version
`#productBuildVersion`	Chaîne indiquant le numéro de série interne de l'application de lecture.

Les propriétés contiennent exactement les mêmes informations que les propriétés et fonctions du même nom.

Exemple

L'instruction suivante affiche la liste d'environnement dans la fenêtre Messages :

```
-- Lingo syntax
put(_system.environmentPropList)

// JavaScript syntax
put(_system.environmentPropList);
```

Voir aussi

`Système`

error

Syntaxe

```
member(whichCastmember).model(whichModel).sds.error
```

Description

Propriété 3D de modificateur `#sds` ; indique le pourcentage d'erreurs toléré par le modificateur lors de la synthèse des détails géométriques dans les modèles.

Cette propriété ne fonctionne que lorsque la propriété `subdivision` du modificateur a pour valeur `#adaptive`. Les propriétés `tension` et `depth` (3D) du modificateur se combinent à la propriété `error` pour contrôler l'importance du fractionnement effectué par le modificateur.

Exemple

L'instruction suivante définit la propriété `error` du modificateur `#sds` du modèle Bébé sur 0. Si la valeur de la propriété `tension` du modificateur est faible, que la valeur de sa propriété depth est élevée et que sa propriété `subdivision` présente la valeur `#adaptive`, cette instruction produira un effet très prononcé sur la géométrie de Bébé.

```
-- Lingo syntax
member("Scene").model("Baby").addModifier(#sds)
member("Scene").model("Baby").sds.subdivision = #adaptive
member("Scene").model("Baby").sds.error = 0

// JavaScript syntax
member("Scene").getProp("model",2).addModifier(symbol("sds"));
member("Scene").getProp("model",2).getPropRef("sds").subdivision = symbol("adaptive");
member("Scene").getProp("model",2).getPropRef("sds").error = 0;
```

Voir aussi

sds (modificateur), subdivision, depth (3D), tension

eventPassMode

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.eventPassMode

// JavaScript syntax
memberOrSpriteObjRef.eventPassMode;
```

Description

Propriété d'acteur et d'image-objet Flash ; contrôle le moment auquel une animation Flash passe les événements de souris aux comportements associés aux images-objets placées sous l'image-objet Flash. La propriété `eventPassMode` peut prendre les valeurs suivantes :

- `#passAlways` (valeur par défaut) : transmet toujours les événements de souris.
- `#passButton` : ne transmet les événements de souris que lorsque l'utilisateur clique sur un bouton dans l'animation Flash.
- `#passNotButton` : ne transmet les événements de souris que lorsque l'utilisateur clique sur un objet autre qu'un bouton.
- `#passNever` : ne transmet jamais les événements de souris.

Cette propriété peut être testée et définie.

Exemple

Le script d'image suivant vérifie si les boutons d'une image-objet d'animation Flash sont activés et, le cas échéant, définit la propriété `eventPassMode` sur `#passNotButton` ; si les boutons sont désactivés, le script attribue à `eventPassMode` la valeur `#passAlways`. Ce script produit l'effet suivant :

- Les événements de souris sur des objets autres que des boutons sont toujours transmis aux scripts d'images-objets.
- Les événements de souris sur des objets boutons sont passés aux scripts d'images-objets lorsque les boutons sont désactivés. Lorsque les boutons sont activés, les événements de souris sur les boutons sont arrêtés.

```
-- Lingo syntax
on enterFrame
    if sprite(5).buttonsEnabled = TRUE then
        sprite(5).eventPassMode= #passNotButton
    else
        sprite(5).eventPassMode = #passAlways
    end if
end

// JavaScript syntax
function enterFrame() {
    var btEn = sprite(5).buttonsEnabled;
    if (btEn == 1) {
        sprite(5).eventPassMode= symbol("passNotButton");
    } else {
        sprite(5).eventPassMode = symbol("passAlways");
    }
}
```

exitLock

Syntaxe

```
-- Lingo syntax
_movie.exitLock

// JavaScript syntax
_movie.exitLock;
```

Description

Propriété d'animation ; détermine si un utilisateur peut quitter le programme et revenir au Bureau Windows ou au Finder Mac à partir de projections (`FALSE`, valeur par défaut) ou non (`TRUE`). Lecture/écriture.

L'utilisateur peut quitter le programme et revenir au Bureau en appuyant sur Ctrl+point (Windows) ou Cmd+point (Mac), Ctrl+Q (Windows) ou Cmd+Q (Mac) ou Ctrl+W (Windows) ou Cmd+W (Mac) (la touche d'échappement est également prise en charge sous Windows).

Exemple

L'instruction suivante définit la propriété `exitLock` sur `TRUE` :

```
-- Lingo syntax
_movie.exitLock = TRUE

// JavaScript syntax
_movie.exitLock = true;
```

Si la propriété `exitLock` est définie sur `TRUE`, rien ne se produira automatiquement lors de l'utilisation des raccourcis clavier Ctrl+point/Q/W, Echap ou Cmd+point/Q/W. Ce gestionnaire vérifie les informations saisies au clavier pour quitter le programme et présente une séquence personnalisée à l'utilisateur :

```
-- Lingo syntax
on checkExit
    if ((_key.commandDown) and (_key.key = "." or _key.key = "q") and (_movie.exitLock =
TRUE)) then _movie.go("quit sequence")
end checkExit
```

```
// JavaScript syntax
function checkExit() {
    if ((_key.commandDown) && (_key.key == "." || _key.key == "q") && (_movie.exitLock ==
true)) {
        _movie.go("quit sequence");
    }
}
```

Voir aussi

Animation

externalParamCount

Syntaxe

```
-- Lingo syntax
_player.externalParamCount

// JavaScript syntax
_player.externalParamCount;
```

Description

Propriété de lecteur ; renvoie le nombre de paramètres transmis par une balise HTML <EMBED> ou <OBJECT> à une animation comportant du contenu Shockwave. Lecture seule.

Cette propriété n'est valide que pour les animations avec contenu Shockwave exécutées dans un navigateur. Elle ne fonctionne pas pour les animations en cours de création, ni pour les projections.

Pour plus d'informations sur les paramètres externes valides, reportez-vous aux entrées `externalParamName()` et `externalParamValue()`.

Exemple

Le gestionnaire suivant détermine si une balise <OBJECT> ou <EMBED> transmet des paramètres externes à une animation intégrant du contenu Shockwave et exécute des instructions Lingo si les paramètres sont transmis :

```
-- Lingo syntax
if (_player.externalParamCount > 0) then
    -- perform some action
end if

// JavaScript syntax
if (_player.externalParamCount > 0) {
    // perform some action;
}
```

Voir aussi

externalParamName(), externalParamValue(), Lecteur

face

Syntaxe

```
\-- Lingo Usage
member(whichCastmember).modelResource(whichModelResource).face.count
```

```
member(whichCastmember).modelResource(whichModelResource).face[index].colors
member(whichCastmember).modelResource(whichModelResource).face[index].normals
member(whichCastmember).modelResource(whichModelResource).face[index].shader
member(whichCastmember).modelResource(whichModelResource).face[index].textureCoordinates
member(whichCastmember).modelResource(whichModelResource).face[index].vertices
member(whichCastmember).model(whichModel).meshdeform.face.count
member(whichCastmember).model(whichModel).meshdeform.mesh[index].face.count
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].face[faceIndex]
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].face[faceIndex].neigh
bor{[neighborIndex]}

// JavaScript Usage
member(whichCastmember).getProp("modelResource",
whichModelResourceIndex).getPropRef("face", index).colors
member(whichCastmember).getProp("modelResource",
whichModelResourceIndex).getPropRef("face", index).normals
member(whichCastmember).getProp("modelResource",
whichModelResourceIndex).getPropRef("face", index).shader
member(whichCastmember).getProp("modelResource",
whichModelResourceIndex).getPropRef("face", index).textureCoordinates
member(whichCastmember).getProp("modelResource",
whichModelResourceIndex).getPropRef("face", index).vertices
member(whichCastmember).getProp("model",
whichModelIndex).getPropRef("meshdeform").getPropRef("mesh", meshIndex).face[faceIndex]
member(whichCastmember).getProp("model",
whichModelIndex).getPropRef("meshdeform").getPropRef("mesh",
meshIndex).face[faceIndex].neighbor{[neighborIndex]}
```

Description

Propriété 3D de ressource de modèle `#mesh` et de modificateur `meshdeform`. Toutes les ressources de modèle sont des mailles composées de triangles. Chaque triangle est une face.

Vous pouvez accéder aux propriétés des faces des ressources de modèle de type `#mesh`. Les modifications apportées à ces propriétés n'entrent pas en vigueur jusqu'à l'appel de la commande `build()`.

Remarque : *pour plus d'informations, consultez les entrées des différentes propriétés.*

- `count` indique le nombre de triangles de la maille.
- `colors` indique les valeurs d'index de la liste des couleurs de la ressource de modèle à utiliser pour chacun des sommets de la face.
- `normals` indique les valeurs d'index de la liste des normales de la ressource de modèle à utiliser pour chacun des sommets de la face.
- `shadowPercentage` identifie le matériau utilisé lorsque la face est rendue.
- `textureCoordinates` indique les valeurs d'index de la liste des coordonnées de texture de la ressource de modèle à utiliser pour chacun des sommets de la face.
- `vertices` indique les valeurs d'index de la liste des sommets de la ressource de modèle à utiliser pour définir la face.

Pour plus d'informations sur ces propriétés de face, reportez-vous à l'entrée `meshDeform`.

Voir aussi

`build()`, `newMesh`, `meshDeform (modificateur)`

face[ ]

Syntaxe

```
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].face[faceIndex]
```

Description

Propriété 3D de modificateur `meshdeform` ; indique les valeurs d'index de la liste des sommets de la ressource de modèle utilisées pour définir la face.

Cette propriété peut être testée, mais pas définie. Vous pouvez spécifier les sommets d'une face de la ressource de modèle `#mesh` en définissant ses propriétés `vertexList` et `vertices` et en appelant la commande `build`.

Exemple

L'instruction suivante indique que la première face de la première maille du modèle Sol est définie par les trois premiers vecteurs de la liste des sommets de la ressource de modèle utilisée par Sol :

```
put member("Scene").model("Floor").meshdeform.mesh[1].face[1]
-- [1, 2, 3]
```

Voir aussi

meshDeform (modificateur), face, vertexList (déformation de maille), vertices

far (brouillard)

Syntaxe

```
member(whichCastmember).camera(whichCamera).fog.far
sprite(whichSprite).camera{(index)}.fog.far
```

Description

Propriété 3D de caméra ; indique la distance à partir de la caméra, en unités de l'univers, à laquelle le brouillard atteint sa densité maximale lorsque la propriété `fog.enabled` de la caméra présente la valeur `TRUE`.

La valeur par défaut de cette propriété est 1 000.

Exemple

L'instruction suivante attribue à la propriété de brouillard `far` de la caméra vueDeLaBaie la valeur 5 000. Si la propriété `enabled` du brouillard présente la valeur `TRUE`, le brouillard atteindra sa densité maximale à 5 000 unités (de l'univers) en face de la caméra.

```
-- Lingo syntax
member("MyYard").camera("BayView").fog.far = 5000

// JavaScript syntax
member("MyYard").getProp("camera",1).getPropRef("fog").far = 5000;
```

Voir aussi

fog, near (brouillard)

fieldOfView

Syntaxe

```
-- Lingo syntax
spriteObjRef.fieldOfView

// JavaScript syntax
spriteObjRef.fieldOfView;
```

Description

Propriété d'image-objet QTVR ; indique le champ de vue, en degrés, de l'image-objet spécifiée.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante attribue à la propriété `fieldOfView` de la caméra 1 la valeur 90 :

```
-- Lingo syntax
member("3d world").camera[1].fieldOfView = 90

// JavaScript syntax
member("3d world").getProp("camera",1).fieldOfView = 90;
```

fieldOfView (3D)

Syntaxe

```
member(whichCastmember).camera(whichCamera).fieldOfView
sprite(whichSprite).camera{(index)}.fieldOfView
```

Description

Propriété 3D de caméra ; indique l'angle formé par deux rayons : un tracé de la caméra vers le haut du plan de projection et l'autre de la caméra vers le bas du plan de projection.

Les images des modèles de l'univers 3D sont plaquées sur le plan de projection, qui est positionné en face de la caméra, tel un écran en face d'un projecteur. Le plan de projection est ce que vous voyez dans l'image-objet 3D. Le haut et le bas du plan de projection sont définis par la propriété `fieldOfView`. Remarquez cependant que l'image-objet n'est pas redimensionnée en fonction des changements de la propriété `fieldOfView`. L'image du plan de projection est en fait redimensionnée pour tenir dans le rect de l'image-objet.

La valeur de cette propriété n'a d'importance que lorsque la valeur de la propriété `projection` de la caméra est `#perspective`. Lorsque la propriété `projection` présente la valeur `#orthographic`, utilisez la propriété `orthoHeight` de la caméra pour définir le haut et le bas du plan de projection.

Le paramètre par défaut de cette propriété est 30,0.

Exemple

L'instruction suivante attribue à la propriété `fieldOfView` de la caméra 1 la valeur 90 :

```
member("3d world").camera[1].fieldOfView = 90
```

Voir aussi

`orthoHeight`

fileFreeSize

Syntaxe

```
-- Lingo syntax
_movie.fileFreeSize

// JavaScript syntax
_movie.fileFreeSize;
```

Description

Propriété d'animation ; renvoie le nombre d'octets inutilisés dans l'animation en cours par suite de modifications apportées aux acteurs et bibliothèques de distribution d'une animation. Lecture seule.

Les commandes `Enregistrer` et `Compresser` et `Enregistrer sous` suppriment automatiquement l'espace inutilisé du fichier de l'animation.

Si l'animation ne contient aucun espace inutilisé, la propriété `fileFreeSize` renvoie la valeur 0.

Exemple

L'instruction suivante affiche le nombre d'octets inutilisés dans le fichier de l'animation actuelle :

```
-- Lingo syntax
put(_movie.fileFreeSize)

// JavaScript syntax
put(_movie.fileFreeSize);
```

Voir aussi

`Animation`

fileName (distribution)

Syntaxe

```
-- Lingo syntax
castObjRef.fileName

// JavaScript syntax
castObjRef.fileName;
```

Description

Propriété de bibliothèque de distribution ; renvoie ou définit le nom de fichier d'une bibliothèque de distribution. Lecture seule pour les bibliothèques de distribution internes, lecture/écriture pour les bibliothèques de distribution externes.

Dans le cas des bibliothèques de distribution externes, `fileName` renvoie le chemin d'accès complet et le nom de fichier de la distribution.

Pour les bibliothèques de distribution internes, `fileName` renvoie une valeur dépendant de la bibliothèque de distribution interne spécifiée.

- Si la première bibliothèque de distribution interne est spécifiée, `fileName` renvoie le nom de l'animation.
- Si une autre bibliothèque de distribution interne est spécifiée, `fileName` renvoie une chaîne vide.

Cette propriété accepte des URL en tant que références. Toutefois, pour utiliser une bibliothèque de distribution depuis Internet et minimiser la durée de téléchargement, utilisez la méthode `downloadNetThing()` ou `preloadNetThing()` pour télécharger le fichier de la distribution sur un disque local, puis définissez `fileName` sur le fichier situé sur le disque.

Si une animation définit le nom de fichier d'une distribution externe, n'utilisez pas l'option permettant de dupliquer les acteurs pour un chargement plus rapide dans la boîte de dialogue Options de projection.

Exemple

L'instruction suivante affiche le chemin d'accès et le nom de fichier de la distribution externe Boutons dans la fenêtre Messages :

```
-- Lingo syntax
trace(castLib("Buttons").fileName)

// JavaScript syntax
trace(castLib("Buttons").fileName);
```

L'instruction suivante affecte le nom de fichier Contenu.cst à la distribution externe Boutons :

```
-- Lingo syntax
castLib("Buttons").fileName = _movie.path & "Content.cst"

// JavaScript syntax
castLib("Buttons").fileName = _movie.path + "Content.cst";
```

L'animation utilise ensuite le fichier de distribution externe Contenu.cst en tant que distribution Boutons.

Les instructions suivantes téléchargent une distribution externe depuis une URL dans le dossier de Director et définissent ce fichier en tant que distribution externe intitulée Distribution :

```
-- Lingo syntax
downloadNetThing("http://wwwcbDeMille.com/Thousands.cst", _player.applicationPath &
"Thousands.cst")
castLib("Cast of Thousands").fileName = _player.applicationPath & "Thousands.cst"

// JavaScript syntax
downloadNetThing("http://wwwcbDeMille.com/Thousands.cst", _player.applicationPath +
"Thousands.cst");
castLib("Cast of Thousands").fileName = _player.applicationPath + "Thousands.cst";
```

Voir aussi

Bibliothèque de distribution, downloadNetThing, preloadNetThing()

fileName (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.fileName

// JavaScript syntax
memberObjRef.fileName;
```

Description

Propriété d'acteur ; fait référence au nom du fichier affecté à un acteur lié. Lecture/écriture.

Cette propriété est pratique pour changer le fichier lié externe affecté à un acteur pendant la lecture d'une animation, d'une façon semblable au changement d'acteurs. Lorsque le fichier lié se trouve dans un dossier différent de celui de l'animation, vous devez inclure son chemin d'accès.

Vous pouvez également faire des médias non liés des médias liés en définissant le nom de fichier des types d'acteurs supportant le média lié.

Cette propriété accepte également les URL en tant que références. Toutefois, pour utiliser un fichier depuis une URL et minimiser la durée de téléchargement, utilisez la méthode `downloadNetThing()` ou `preloadNetThing()` pour télécharger le fichier sur un disque local, puis définissez la propriété `fileName` sur le fichier situé sur le disque local.

Une fois le nom de fichier défini, Director utilise ce fichier à la prochaine utilisation de l'acteur.

Exemple

L'instruction suivante lie l'acteur séquence QuickTime ChaiseAnimée à l'acteur 40 :

```
-- Lingo syntax
member(40).fileName = "ChairAnimation"

// JavaScript syntax
member(40).fileName = "ChairAnimation";
```

Les instructions suivantes téléchargent un fichier externe depuis une URL dans le dossier de Director et définissent ce fichier en tant que média pour l'acteur son Norma :

```
-- Lingo syntax
downloadNetThing("http://wwwcbDeMille.com/Talkies.AIF", _player.applicationPath &
"Talkies.AIF")
member("Norma Desmond Speaks").fileName = _player.applicationPath & "Talkies.AIF"

// JavaScript syntax
downloadNetThing("http://wwwcbDeMille.com/Talkies.AIF", _player.applicationPath +
"Talkies.AIF");
member("Norma Desmond Speaks").fileName = _player.applicationPath + "Talkies.AIF";
```

Voir aussi

`downloadNetThing`, `Acteur`, `preloadNetThing()`

fileName (fenêtre)

Syntaxe

```
-- Lingo syntax
windowObjRef.fileName

// JavaScript syntax
windowObjRef.fileName;
```

Description

Propriété de fenêtre ; fait référence au nom de fichier de l'animation affectée à une fenêtre. Lecture/écriture.

Lorsque le fichier lié se trouve dans un dossier différent de celui de l'animation, vous devez inclure son chemin d'accès.

Pour pouvoir lire l'animation dans une fenêtre, vous devez définir la propriété `fileName` sur le nom de fichier de l'animation.

La propriété `fileName` accepte les URL en tant que références. Toutefois, pour utiliser un fichier d'animation depuis une URL et minimiser la durée de téléchargement, utilisez la méthode `downloadNetThing()` ou `preloadNetThing()` pour télécharger le fichier sur un disque local, puis définissez la propriété `fileName` sur le fichier situé sur le disque local.

Exemple

L'instruction suivante affecte le fichier appelé Tableau de commande à la fenêtre Boîte à outils :

```
-- Lingo syntax
window("Tool Box").fileName = "Control Panel"

// JavaScript syntax
window("Tool Box").fileName = "Control Panel";
```

L'instruction suivante affiche le nom du fichier affecté à la fenêtre intitulée Navigateur :

```
-- Lingo syntax
trace(window("Navigator").fileName)

// JavaScript syntax
trace(window("Navigator").fileName);
```

Les instructions suivantes téléchargent un fichier d'animation depuis une URL dans le dossier de Director, puis affectent ce fichier à la fenêtre Mon gros plan :

```
-- Lingo syntax
downLoadNetThing("http://www.cbDeMille.com/Finale.DIR", _player.applicationPath &
"Finale.DIR")
window("My Close Up").fileName = _player.applicationPath & "Finale.DIR"

// JavaScript syntax
downLoadNetThing("http://www.cbDeMille.com/Finale.DIR", _player.applicationPath +
"Finale.DIR");
window("My Close Up").fileName = _player.applicationPath + "Finale.DIR";
```

Voir aussi

`downloadNetThing`, `preloadNetThing()`, Fenêtre

fileSize

Syntaxe

```
-- Lingo syntax
_movie.fileSize

// JavaScript syntax
_movie.fileSize;
```

Description

Propriété d'animation ; renvoie le nombre d'octets du fichier de l'animation actuelle enregistré sur le disque dur. Lecture seule.

Il s'agit du même nombre que celui qui s'affiche lorsque vous utilisez Propriétés du fichier (Windows) ou Lire les informations (dans le Finder du Mac).

Exemple

L'instruction suivante affiche le nombre d'octets de l'animation actuelle :

```
-- Lingo syntax
put(_movie.fileSize)

// JavaScript syntax
put(_movie.fileSize);
```

Voir aussi

Animation

fileVersion

Syntaxe

```
-- Lingo syntax
_movie.fileVersion

// JavaScript syntax
_movie.fileVersion;
```

Description

Propriété d'animation ; indique, sous forme de chaîne, la version de Director dans laquelle l'animation a été enregistrée pour la dernière fois. Lecture seule.

Exemple

L'instruction suivante affiche la version de Director dans laquelle l'animation a été enregistrée pour la dernière fois :

```
-- Lingo syntax
put(_movie.fileVersion)

// JavaScript syntax
put(_movie.fileVersion);
```

Voir aussi

Animation

fillColor

Syntaxe

```
-- Lingo syntax
memberObjRef.fillColor

// JavaScript syntax
memberObjRef.fillColor;
```

Description

Propriété d'acteur forme vectorielle ; la couleur de remplissage est spécifiée en valeur rvb.

Il est possible d'utiliser `fillColor` lorsque la propriété `fillMode` de la forme est définie sur `#solid` ou sur `#gradient`, mais non si elle est définie sur `#none`. Si `fillMode` est défini sur `#gradient`, `fillColor` spécifie la couleur de départ du dégradé. La couleur finale est spécifiée par la propriété `endColor`.

Cette propriété peut être testée et définie.

Vous pouvez voir un exemple d'utilisation de `fillColor` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante donne à la couleur de remplissage de l'acteur Archie une nouvelle valeur rvb :

```
-- Lingo syntax
member("Archie").fillColor = color( 24, 15, 153)

// JavaScript syntax
member("Archie").fillColor = color( 24, 15, 153);
```

Voir aussi

`endColor`, `fillMode`

fillCycles

Syntaxe

```
-- Lingo syntax
memberObjRef.fillCycles

// JavaScript syntax
memberObjRef.fillCycles;
```

Description

Propriété d'acteur forme vectorielle ; nombre de cycles de remplissage dégradé d'une forme vectorielle, spécifié par un nombre entier compris entre 1 et 7.

Cette propriété n'est valide que si la propriété `fillMode` de la forme présente la valeur `#gradient`.

Cette propriété peut être testée et définie.

Vous pouvez voir un exemple d'utilisation de `fillCycles` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante attribue à la propriété fillCycles de l'acteur Archie la valeur 3 :

```
-- Lingo syntax
member("Archie").fillCycles = 3

// JavaScript syntax
member("Archie").fillCycles = 3;
```

Voir aussi

`endColor`, `fillColor`, `fillMode`

fillDirection

Syntaxe

```
-- Lingo syntax
memberObjRef.fillDirection

// JavaScript syntax
memberObjRef.fillDirection;
```

Description

Propriété d'acteur forme vectorielle ; spécifie le degré de rotation du remplissage de la forme.

Cette propriété n'est valide que si la propriété `fillMode` de la forme présente la valeur `#gradient`.

Cette propriété peut être testée et définie.

Vous pouvez voir un exemple d'utilisation de `fillDirection` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

Le comportement suivant redéfinit la propriété fillMode de la forme vectorielle sur « gradient » et génère une animation simple en modifiant continuellement la propriété fillDirection de la forme vectorielle.

```
-- Lingo syntax
on beginSprite(me)
    member("VectorShape").fillMode = #gradient
end

on exitFrame(me)
    member("VectorShape").fillDirection = (member("VectorShape").fillDirection + 10 ) mod \
360
end

// JavaScript syntax
function beginSprite(me)
{
    member("VectorShape").fillMode = symbol("gradient");
}

function exitFrame(me)
{
    member("VectorShape").fillDirection = (member("VectorShape").fillDirection + 10 ) % 360;
}
```

Voir aussi

`fillMode`

filled

Syntaxe

```
member(whichCastMember).filled
the filled of member whichCastMember
```

Description

Propriété d'acteur forme ; indique si l'acteur spécifié contient un motif de remplissage (TRUE) ou non (FALSE).

Exemple

Les instructions suivantes donnent à l'acteur forme Cible une forme pleine et lui affectent le motif numéro 1, qui est une couleur unie :

```
-- Lingo syntax
member("Target Area").filled = TRUE
member("Target Area").pattern = 1

// Java Script
member("Target Area").filled = true;
member("Target Area").pattern = 1;
```

Voir aussi

fillColor, fillMode

fillMode

Syntaxe

```
-- Lingo syntax
memberObjRef.fillMode

// JavaScript syntax
memberObjRef.fillMode;
```

Description

Propriété d'acteur forme vectorielle ; indique la méthode de remplissage pour la forme, à l'aide des valeurs suivantes :

- `#none` : la forme est transparente.
- `#solid` : la forme utilise une seule couleur de remplissage.
- `#gradient` : la forme utilise un dégradé entre deux couleurs.

Cette propriété peut être testée et définie lorsque la forme est fermée ; les formes ouvertes n'ont pas de remplissage.

Vous pouvez voir un exemple d'utilisation de `fillMode` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante définit la propriété `fillMode` de l'acteur Archie sur la valeur de dégradé :

```
-- Lingo syntax
member("Archie").fillMode = #gradient

// JavaScript syntax
member("Archie").fillMode = symbol("gradient");
```

Voir aussi

endColor, fillColor

fillOffset

Syntaxe

```
-- Lingo syntax
memberObjRef.fillOffset

// JavaScript syntax
memberObjRef.fillOffset;
```

Description

Propriété d'acteur forme vectorielle ; spécifie le nombre de pixels horizontaux et verticaux (dans l'espace `defaultRect`) utilisés pour décaler le remplissage de la forme.

Cette propriété n'est valide que si la propriété `fillMode` de la forme présente la valeur `#gradient`, mais peut être testée et définie.

Vous pouvez voir un exemple d'utilisation de `fillOffset` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante change le décalage de remplissage de l'acteur forme vectorielle Miette à un décalage horizontal de 33 pixels et un décalage vertical de 27 pixels :

```
-- Lingo syntax
member("miette").fillOffset = point(33, 27)

// JavaScript syntax
member("miette").fillOffset = point(33, 27);
```

Voir aussi

`defaultRect`, `fillMode`

fillScale

Syntaxe

```
-- Lingo syntax
memberObjRef.fillScale

// JavaScript syntax
memberObjRef.fillScale;
```

Description

Propriété d'acteur forme vectorielle ; spécifie le degré de mise à l'échelle du remplissage de la forme. Cette propriété porte également le nom Echelle dans la fenêtre Forme vectorielle.

Cette propriété n'est valide que si la propriété `fillMode` de la forme présente la valeur `#gradient`, mais peut être testée et définie.

Vous pouvez voir un exemple d'utilisation de `fillScale` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante définit la propriété `fillScale` de l'acteur Archie sur la valeur 33 :

```
-- Lingo syntax
member("Archie").fillScale = 33.00

// JavaScript syntax
member("Archie").fillScale = 33.00;
```

Voir aussi

fillMode

filterlist

Syntaxe

```
-- Lingo syntax
spriteObjRef.filterlist

// JavaScript syntax
spriteObjRef.filterlist;
```

Description

Propriété d'image-objet ; permet de définir si un filtre de bitmap est appliqué à l'image-objet. Etant donné qu'il s'agit d'une liste, vous pouvez appliquer des filtres de bitmaps à cette propriété en les ajoutant à la liste.

Remarque : *Vous ne pouvez pas copier une propriété filterlist en utilisant la méthode duplicate().*

Exemple

La première instruction définit la variable dénommée monFiltre sur le filtre de flou. La ligne suivante définit le filtre de flou sur sprite(1).

```
--Lingo syntax
MyFilter=filter(#BlurFilter)
sprite(1).filterlist.append(MyFilter)

// JavaScript syntax
var MyFilter = filter(symbol("BlurFilter"));
sprite(1).filterlist.append(MyFilter);
```

Voir aussi

Filtres de bitmaps dans le mode d'emploi de Director.

firstIndent

Syntaxe

```
-- Lingo syntax
chunkExpression.firstIndent

// JavaScript syntax
chunkExpression.firstIndent;
```

Description

Propriété d'acteur texte ; contient le nombre de pixels de décalage correspondant au premier retrait de *expressionSousChaîne* à partir de la marge gauche de *expressionSousChaîne*.

La valeur est un nombre entier : un nombre inférieur à 0 indique un retrait négatif, 0 indique l'absence de retrait et un nombre supérieur à 0 indique un retrait normal.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante définit le retrait de la première ligne de l'acteur Bureau sur 0 pixel :

```
--Lingo syntax
member("Desk").firstIndent = 0

// JavaScript syntax
member("Desk").firstIndent = 0;
```

Voir aussi

`leftIndent`, `rightIndent`

fixedLineSpace

Syntaxe

```
-- Lingo syntax
chunkExpression.fixedLineSpace

// JavaScript syntax
chunkExpression.fixedLineSpace;
```

Description

Propriété d'acteur texte ; contrôle la hauteur de chaque ligne dans la partie de *expressionSousChaîne* de l'acteur texte.

La valeur elle-même est un nombre entier, indiquant la hauteur en pixels absolus de chaque ligne.

La valeur par défaut est 0, qui a pour résultat une hauteur naturelle.

Exemple

L'instruction suivante définit la hauteur, en pixels, de chaque ligne de l'acteur Bureau sur 24 :

```
--Lingo syntax
member("Desk").fixedLineSpace = 24

// JavaScript syntax
member("Desk").fixedLineSpace = 24;
```

fixedRate

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.fixedRate

// JavaScript syntax
memberOrSpriteObjRef.fixedRate;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la cadence d'image d'une animation Flash ou d'un GIF animé. La propriété `fixedRate` peut prendre des valeurs entières. La valeur par défaut est 15.

Cette propriété est ignorée si la propriété `playbackMode` de l'image-objet présente une valeur différente de `#fixed`.

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant règle la cadence d'image d'une image-objet d'animation Flash. Comme paramètres, le gestionnaire accepte une référence d'image-objet, une indication d'accélération ou de ralentissement de l'animation Flash et l'importance du réglage de la cadence.

```
-- Lingo syntax
on adjustFixedRate(whichSprite, adjustType, howMuch)
    case adjustType of
        #faster:
            sprite(whichSprite).fixedRate = sprite(whichSprite).fixedRate + howMuch
        #slower:
            sprite(whichSprite).fixedRate = sprite(whichSprite).fixedRate - howMuch
    end case
end

// JavaScript syntax
function adjustFixedRate(whichSprite, adjustType, howMuch) {
    switch(adjustType) {
        case "faster":
            sprite(whichSprite).fixedRate = sprite(whichSprite).fixedRate + howMuch;
            break;
        case "slower":
            sprite(whichSprite).fixedRate = sprite(whichSprite).fixedRate - howMuch;
            break;
    }
}
```

Voir aussi

playBackMode

fixStageSize

Syntaxe

```
-- Lingo syntax
_movie.fixStageSize

// JavaScript syntax
_movie.fixStageSize;
```

Description

Propriété d'animation ; détermine si la taille de la scène reste la même lorsque vous chargez une nouvelle animation (`TRUE`, valeur par défaut) ou non (`FALSE`), quelles que soient la valeur de la propriété `centerStage` et la taille de la scène enregistrée avec cette animation. Lecture/écriture.

La propriété `fixStageSize` ne peut pas changer la taille de la scène d'une animation en cours de lecture.

Exemple

L'instruction suivante détermine si la propriété `fixStageSize` est activée. Si la propriété `fixStageSize` présente la valeur `FALSE`, elle envoie la tête de lecture vers une image spécifiée.

```
-- Lingo syntax
if (_movie.fixStageSize = FALSE) then
    _movie.go("proper size")
end if

// JavaScript syntax
if (_movie.fixStageSize == false) {
    _movie.go("proper size");
}
```

L'instruction suivante inverse la valeur en cours de la propriété `fixStageSize` :

```
-- Lingo syntax
_movie.fixStageSize = not(_movie.fixStageSize)

// JavaScript syntax
_movie.fixStageSize = !(_movie.fixStageSize);
```

Voir aussi

`centerStage`, `Animation`

flashRect

Syntaxe

```
-- Lingo syntax
memberObjRef.flashRect

// JavaScript syntax
memberObjRef.flashRect;
```

Description

Propriété d'acteur ; indique la taille initiale d'un acteur animation Flash ou forme vectorielle. Les valeurs sont indiquées sous la forme de rectangle Director : par exemple, rect(0, 0, 32, 32).

Pour les acteurs Flash liés, la propriété d'acteur `FlashRect` ne renvoie une valeur valide que lorsque l'en-tête de l'acteur a été complètement chargé en mémoire.

Cette propriété peut être testée, mais pas définie.

Exemple

Ce script d'image-objet redimensionne une image-objet d'animation Flash pour qu'elle ait une taille identique à la taille d'origine de son acteur animation Flash :

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).rect = sprite(spriteNum).member.FlashRect
end
```

```
// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).rect = sprite(this.spriteNum).member.FlashRect;
}
```

Voir aussi

defaultRect, defaultRectMode, state (Flash, SWA)

flat

Syntaxe

```
member(whichCastmember).shader(whichShader).flat
member(whichCastmember).model(whichModel).shader.flat
member(whichCastmember).model(whichModel).shaderList{[index]}.flat
```

Description

Propriété 3D de matériau `#standard` ; indique si la maille doit être rendue avec un matériau plat (`TRUE`) ou un matériau de Gouraud (`FALSE`).

Le matériau plat utilise une couleur par face de la maille. La couleur utilisée pour la face est celle du premier sommet. Le matériau plat est plus rapide que le matériau de Gouraud.

Le matériau de Gouraud affecte une couleur à chaque sommet d'une face, puis interpole les couleurs sur la face dans un dégradé. Le matériau de Gouraud nécessite un plus grand effort de calcul mais produit une surface plus lisse.

La valeur par défaut de cette propriété est `FALSE`.

Exemple

L'instruction suivante attribue à la propriété `flat` du matériau Mur la valeur `TRUE`. La maille d'un modèle qui utilise ce matériau est rendue avec une couleur par face.

```
-- Lingo syntax
member("MysteryWorld").shader("Wall").flat = TRUE

// JavaScript syntax
member("MysteryWorld").getProp("shader", 1).flat = true;
```

Voir aussi

mesh (propriété), colors, vertices, generateNormals()

flipH

Syntaxe

```
-- Lingo syntax
spriteObjRef.flipH

// JavaScript syntax
spriteObjRef.flipH;
```

Description

Propriété d'image-objet ; indique si l'image d'une image-objet a été renversée horizontalement sur la scène (TRUE) ou non (FALSE). Lecture seule.

L'image même est renversée autour de son point d'alignement.

Cela signifie que les rotations ou inclinaisons restent constantes, seules les données de l'image étant renversées.

Exemple

L'instruction suivante affiche la valeur `flipH` de l'image-objet 5 :

```
-- Lingo syntax
put(sprite(5).flipH)

// JavaScript syntax
put(sprite(5).flipH);
```

Voir aussi

flipV, rotation, skew, Image-objet

flipV

Syntaxe

```
-- Lingo syntax
spriteObjRef.flipV

// JavaScript syntax
spriteObjRef.flipV;
```

Description

Propriété d'image-objet ; indique si l'image d'une image-objet a été renversée verticalement sur la scène (TRUE) ou non (FALSE). Lecture seule.

L'image même est renversée autour de son point d'alignement.

Cela signifie que les rotations ou inclinaisons restent constantes, seules les données de l'image étant renversées.

Exemple

L'instruction suivante affiche la valeur `flipV` de l'image-objet 5 :

```
-- Lingo syntax
put(sprite(5).flipV)

// JavaScript syntax
put(sprite(5).flipV);
```

Voir aussi

flipH, rotation, skew, Image-objet

floatPrecision

Syntaxe

```
the floatPrecision
```

Description

Propriété d'animation ; arrondit l'affichage des nombres à virgule flottante au nombre de chiffres après la virgule spécifié. La valeur de `floatPrecision` doit être un nombre entier. La valeur maximum est 15 chiffres utiles ; la valeur par défaut étant 4.

La propriété `floatPrecision` détermine uniquement le nombre de chiffres utilisés pour afficher les nombres à virgule flottante et n'affecte pas le nombre de chiffres utilisés pour les calculs.

- Si `floatPrecision` est compris entre 1 et 15, les nombres à virgule flottante s'affichent avec cette quantité de chiffres après le signe décimal. Les zéros ne sont pas tronqués.
- Si `floatPrecision` est égal à zéro, les nombres à virgule flottante sont arrondis à l'entier le plus proche. Aucun chiffre n'apparaît après la virgule.
- Si `floatPrecision` est un chiffre négatif, les nombres à virgule flottante sont arrondis à la valeur absolue du nombre de décimales. Les zéros sont alors tronqués.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante arrondit la racine carrée de 3,0 à trois chiffres après la virgule :

```
the floatPrecision = 3
x = sqrt(3.0)
put x
-- 1.732
```

L'instruction suivante arrondit la racine carrée de 3,0 à huit décimales :

```
the floatPrecision = 8
put x
-- 1.73205081
```

fog

Syntaxe

```
\-- Lingo Usage
member(whichCastmember).camera(whichCamera).fog.color
sprite(whichSprite).camera{(index)}.fog.color
member(whichCastmember).camera(whichCamera).fog.decayMode
sprite(whichSprite).camera{(index)}.fog.decayMode
member(whichCastmember).camera(whichCamera).fog.enabled
sprite(whichSprite).camera{(index)}.fog.enabled
member(whichCastmember).camera(whichCamera).fog.far
sprite(whichSprite).camera{(index)}.fog.far
member(whichCastmember).camera(whichCamera).fog.near
sprite(whichSprite).camera{(index)}.fog.near

// JavaScript Usage
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").color
sprite(whichSprite).camera.getPropRef("fog").color
```

```
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").decayMode
sprite(whichSprite).camera.getPropRef("fog").decayMode
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").enabled
sprite(whichSprite).camera.getPropRef("fog").enabled
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").far
sprite(whichSprite).camera.getPropRef("fog").far
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").near
sprite(whichSprite).camera.getPropRef("fog").near
```

Description

Propriété 3D de caméra ; le brouillard crée un flou qui augmente avec la distance. L'effet est semblable à celui de la réalité, à l'exception que le brouillard peut être de n'importe quelle couleur.

Voir aussi

color (brouillard), decayMode, enabled (brouillard), far (brouillard), near (brouillard)

folder

Syntaxe

```
-- Lingo syntax
dvdObjRef.folder

// JavaScript syntax
dvdObjRef.folder;
```

Description

Propriété de DVD. Détermine le chemin d'accès du dossier à partir duquel un DVD est en cours de lecture. Lecture/écriture.

Le chemin d'accès doit être une chaîne.

La propriété folder est définissable dans l'Inspecteur des propriétés ou par l'intermédiaire d'un script. L'actuelle mise en œuvre impose les règles suivantes :

Windows :

- Vous devez indiquer video_ts à la fin du chemin d'accès du média DVD local ciblé. Exemple : C:\video_ts or C:\myLocalDVDContent\video_ts

Mac :

- La valeur de chemin d'accès de la propriété folder doit commencer par /Volumes/
- L'ajout de video_ts à la valeur de chemin d'accès spécifiée pour la propriété folder est actuellement facultatif. Par exemple, si le dossier DVD video_ts est situé sur la racine du lecteur de démarrage, la valeur de la propriété peut être indiquée de l'une des deux façons suivantes :
    - /Volumes/Mac HD/myLocalDVDContent/video_ts

    ou

    - /Volumes/Mac HD/myLocalDVDContent

Pour modifier la valeur de la propriété folder dans l'Inspecteur des propriétés :

1 Sélectionnez l'acteur DVD, puis activez l'onglet DVD de l'Inspecteur des propriétés en mode d'affichage sous forme de liste.

2 Sous la section Propriétés de lecture, sélectionnez le champ modifiable de la propriété folder, puis entrez le chemin d'accès de l'emplacement du média DVD ciblé.

Utilisez les exemples suivants pour vous aider à établir la propriété folder grâce à l'utilisation de scripts. Ces instructions définissent le chemin d'accès de la propriété de DVD folder :

Exemple

Windows :

```
-- Lingo syntax
member(2).folder = "C:\myLocalDVDContent\video_ts"

// JavaScript syntax
member(2).folder = "C:\\myLocalDVDContent\\video_ts";
```

Mac :

```
-- Lingo syntax
member(2).folder = "/Volumes/Mac HD/myLocalDVDContent"

// JavaScript syntax
member(2).folder = "/Volumes/Mac HD/myLocalDVDContent";
```

Remarque : *si un dossier video_ts est introuvable lors de la création du premier acteur DVD, le message d'erreur suivant s'affiche : « Impossible de localiser le Volume DVD ». Cette alerte ne s'affiche qu'une fois par session. A ce stade, vous pouvez toujours nommer l'acteur DVD nouvellement créé, puis définir sa propriété folder sur un emplacement contenant un dossier video_ts valide.*

Problèmes relatifs aux chemins d'accès des dossiers DVD sur Mac

Sur les ordinateurs Mac, le chemin d'accès défini par la propriété folder doit comporter une barre oblique (/) comme séparateur du chemin d'accès, au lieu du séparateur Mac standard deux-points (:). En outre, /volumes/ doit être concaténé au début du chemin d'accès du dossier DVD. Par exemple, si le dossier DVD est situé à la racine du lecteur de démarrage, le chemin d'accès prend la forme suivante :

```
member (2).folder = "/Volumes/Mac HD/Test_DVD/video_ts"
```

Lorsque la commande _movie.path est utilisée pour récupérer le chemin d'accès de la projection ou de l'animation sur un Mac, le chemin d'accès contient un signe deux-points (:) au lieu de la barre oblique (/). L'utilisation des deux-points dans le chemin d'accès du dossier DVD provoque une erreur. Pour remédier à ce problème, les développeurs peuvent utiliser un script permettant de remplacer les deux-points dans le chemin d'accès par des barres obliques.

Voir aussi

DVD

font

Syntaxe

```
-- Lingo syntax
memberObjRef.font
```

```
// JavaScript syntax
memberObjRef.font;
```

Description

Propriété d'acteur texte et champ ; détermine la police utilisée pour afficher l'acteur spécifié et requiert la présence de caractères dans l'acteur, ne serait-ce qu'un espace. Le paramètre *quelActeur* peut être un nom ou un numéro d'acteur.

La propriété d'acteur `font` peut être testée et définie.

Vous pouvez voir un exemple d'utilisation de `font` dans une animation en consultant l'animation Text du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante définit la variable `oldFont` sur la valeur `font` en cours pour l'acteur champ Pierre :

```
-- Lingo syntax
oldFont = member("Rokujo Speaks").font

// JavaScript syntax
var oldFont = member("Rokujo Speaks").font;
```

Voir aussi

`text`, `alignment`, `fontSize`, `fontStyle`, `lineHeight`

fontSize

Syntaxe

```
-- Lingo syntax
memberObjRef.fontSize

// JavaScript syntax
memberObjRef.fontSize;
```

Description

Propriété d'acteur champ ; détermine la taille de la police utilisée pour afficher l'acteur champ spécifié et requiert la présence de caractères dans l'acteur, ne serait-ce qu'un espace. Le paramètre *quelActeur* peut être un nom ou un numéro d'acteur.

Cette propriété peut être testée et définie. Lorsqu'elle est testée, elle renvoie la hauteur de la première ligne du champ. Lorsqu'elle est définie, elle affecte chaque ligne du champ.

Vous pouvez voir un exemple d'utilisation de `fontSize` dans une animation en consultant l'animation Text du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante attribue à la variable `oldSize` la valeur `fontSize` en cours pour l'acteur champ Pierre :

```
--Lingo syntax
oldSize = member("Rokujo Speaks").fontSize

// JavaScript syntax
var oldSize = member("Rokujo Speaks").fontSize;
```

L'instruction suivante définit la troisième ligne de l'acteur texte monMenu sur 12 points :

```
member("myMenu").fontSize = 12

// JavaScript syntax
member("myMenu").fontSize = 12;
```

Voir aussi

`text`, `alignment`, `font`, `fontStyle`, `lineHeight`

fontStyle

Syntaxe

```
-- Lingo syntax
memberObjRef.fontStyle
memberObjRef.char[whichChar].fontStyle
memberObjRef.line[whichLine].fontStyle
memberObjRef.word[whichWord].fontStyle

// JavaScript syntax
memberObjRef.fontStyle;
memberObjRef.getPropRef("char", whichChar).fontStyle
memberObjRef.getPropRef("line", whichLine).fontStyle;
memberObjRef.getPropRef("word", whichWord).fontStyle;
```

Description

Propriété d'acteur champ ; détermine les styles appliqués à la police utilisée pour l'affichage de l'acteur champ, du caractère, de la ligne, du mot ou de toute autre expression de sous-chaîne et requiert la présence de caractères dans l'acteur, ne serait-ce qu'un espace.

Cette propriété a pour valeur une chaîne de styles délimités par des virgules. Lingo utilise une police combinant les styles de cette chaîne. Les styles disponibles sont normal, gras, italique, souligné, ombré, relief et étendu, le style condensé étant également disponible sur Mac.

Utilisez le style normal pour supprimer tous les styles déjà appliqués. Le paramètre *`quelActeur`* peut être un nom ou un numéro d'acteur.

Cette propriété peut être testée et définie.

Vous pouvez voir un exemple d'utilisation de `fontStyle` dans une animation en consultant l'animation Text du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante définit la variable `oldStyle` sur la valeur `fontStyle` en cours pour l'acteur champ Pierre :

```
--Lingo syntax
oldStyle = member("Rokujo Speaks").fontStyle

// JavaScript syntax
var oldStyle = member("Rokujo Speaks").fontStyle;
```

L'instruction suivante affecte les valeurs gras et italique à la propriété d'acteur `fontStyle` pour l'acteur champ Poème :

```
--Lingo syntax
member("Poem").fontStyle = "bold, italic"
```

```
// JavaScript syntax
member("Poem").fontStyle = "bold, italic";
```

L'instruction suivante affecte la valeur italique à la propriété `fontStyle` du troisième mot de l'acteur champ Noms :

```
--Lingo syntax
member("Son's Names").word[3].fontStyle = "italic"

// JavaScript syntax
member("Son's Names").getPropRef("word", 3).fontStyle = "italic";
```

L'instruction suivante définit la propriété `fontStyle` de l'acteur texte sur la valeur gras ou gras et italique :

```
--Lingo syntax
member("text").fontStyle=[#bold]
or
member("text").fontStyle=[#bold, #italic]

// JavaScript syntax
member("text").fontStyle = list(symbol("bold"));
```

Voir aussi

`text`, `alignment`, `fontSize`, `font`, `lineHeight`

foreColor

Syntaxe

```
-- Lingo syntax
spriteObjRef.foreColor

// JavaScript syntax
spriteObjRef.foreColor;
```

Description

Propriété d'image-objet ; renvoie ou définit la couleur du premier plan d'une image-objet. Lecture/écriture.

Il n'est pas recommandé d'appliquer cette propriété à des acteurs bitmap supérieurs à 1 bit, les résultats pouvant être difficiles à prévoir.

Il est recommandé d'utiliser la nouvelle propriété `color` plutôt que la propriété `foreColor`.

Exemple

L'instruction suivante définit la variable `ancienneCouleur` sur la couleur d'arrière-plan de l'image-objet 5 :

```
-- Lingo syntax
oldColor = sprite(5).foreColor

// JavaScript syntax
var oldColor = sprite(5).foreColor;
```

L'instruction suivante définit 36 comme couleur du premier plan d'une image-objet aléatoire entre les images-objets 11 et 13 :

```
-- Lingo syntax
sprite(10 + random(3)).foreColor = 36

// JavaScript syntax
sprite(10 + random(3)).foreColor = 36;
```

Voir aussi

backColor, color(), Image-objet

frame

Syntaxe

```
-- Lingo syntax
_movie.frame

// JavaScript syntax
_movie.frame;
```

Description

Propriété d'animation ; renvoie le numéro de l'image en cours de l'animation. Lecture seule.

Exemple

L'instruction suivante envoie la tête de lecture à l'image qui précède l'image en cours :

```
-- Lingo syntax
_movie.go(_movie.frame - 1)

// JavaScript syntax
_movie.go(_movie.frame - 1);
```

Voir aussi

go(), Animation

frameCount

Syntaxe

```
-- Lingo syntax
memberObjRef.frameCount

// JavaScript syntax
memberObjRef.frameCount;
```

Description

Propriété d'acteur Flash ; indique le nombre d'images dans l'acteur animation Flash. La propriété d'acteur `frameCount` peut prendre des valeurs entières.

Cette propriété peut être testée, mais pas définie.

Exemple

Ce script d'image-objet affiche, dans la fenêtre Messages, le numéro de piste et le nombre d'images d'une animation Flash :

```
-- Lingo syntax
property spriteNum

on beginSprite me
```

```
    put(""The Flash movie in channel"" && spriteNum && has"" &&
/sprite(spriteNum).member.frameCount && ""frames.""
end

// JavaScript syntax
function beginSprite() {
    trace("The Flash movie in channel " + (this.spriteNum) + " has " +
sprite(this.spriteNum).member.frameCount + " frames.");
}
```

frameLabel

Syntaxe

```
-- Lingo syntax
_movie.frameLabel

// JavaScript syntax
_movie.frameLabel;
```

Description

Propriété d'animation ; identifie le libellé affecté à l'image en cours. Lecture/écriture uniquement lors d'une session d'enregistrement de scénario.

Lorsque l'image en cours ne comporte aucun libellé, la propriété `frameLabel` présente la valeur 0.

Exemple

L'instruction suivante vérifie le libellé de l'image actuelle. Dans ce cas, la valeur `frameLabel` en cours est Démarrer :

```
-- Lingo syntax
put(_movie.frameLabel)

// JavaScript syntax
put(_movie.frameLabel);
```

Voir aussi

`labelList`, `Animation`

framePalette

Syntaxe

```
-- Lingo syntax
_movie.framePalette

// JavaScript syntax
_movie.framePalette;
```

Description

Propriété d'animation ; identifie le numéro d'acteur de la palette utilisée dans l'image en cours, correspondant soit à la palette en cours, soit à la palette définie dans l'image en cours. Lecture/écriture uniquement lors d'une session d'enregistrement de scénario.

Lorsqu'un contrôle exact des couleurs est nécessaire, utilisez Shockwave Player.

Exemple

L'instruction suivante vérifie la palette utilisée dans l'image actuelle. Dans ce cas, la palette est l'acteur 45.

```
-- Lingo syntax
put(_movie.framePalette)

// JavaScript syntax
put(_movie.framePalette);
```

L'instruction suivante définit l'acteur palette 45 en tant que palette pour l'image en cours :

```
-- Lingo syntax
_movie.framePalette = 45

// JavaScript syntax
_movie.framePalette = 45;
```

Voir aussi

Animation

frameRate

Syntaxe

```
-- Lingo syntax
memberObjRef.frameRate

// JavaScript syntax
memberObjRef.frameRate;
```

Description

Propriété d'acteur ; spécifie la cadence de lecture de l'acteur vidéo numérique ou animation Flash indiqué.

Les valeurs possibles de cadence d'image d'un acteur vidéo numérique correspondent aux boutons radio de sélection des options de lecture vidéo numérique.

- Lorsque la propriété d'acteur `frameRate` est comprise entre 1 et 255, l'animation vidéo numérique lit chaque image à cette cadence d'image. La propriété d'acteur `frameRate` ne peut pas être supérieure à 255.
- Lorsque la propriété d'acteur `frameRate` présente la valeur -1 ou 0, l'animation vidéo numérique lit chaque image à sa cadence normale. Ceci permet la synchronisation de la vidéo avec sa piste audio. Lorsque la propriété frameRate est définie sur une valeur autre que -1 ou 0, la piste audio de la vidéo numérique n'est pas lue.
- Lorsque la propriété d'acteur `frameRate` présente la valeur -2, l'animation vidéo numérique lit chaque image aussi vite que possible.

Pour les acteurs animation Flash, la propriété indique la cadence d'image de l'animation créée dans Flash.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante fixe la cadence d'image de l'acteur vidéo numérique QuickTime Chaise pivotante à 30 images par seconde :

```
-- Lingo syntax
member("Rotating Chair").frameRate = 30

// JavaScript syntax
```

```
member("Rotating Chair").frameRate = 30;
```

L'instruction suivante indique à l'acteur vidéo numérique QuickTime Chaise pivotante de lire chaque image aussi vite que possible :

```
-- Lingo syntax
member("Rotating Chair").frameRate = -2

// JavaScript syntax
member("Rotating Chair").frameRate = -2;
```

Le script d'image-objet suivant vérifie si l'acteur d'image-objet a été créé dans Flash avec une cadence inférieure à 15 images par seconde. Si la cadence de l'animation est inférieure à 15 images par seconde, le script définit la propriété playBackMode de l'image-objet de façon à pouvoir lui affecter une autre cadence. Le script attribue ensuite à la propriété fixedRate de l'image-objet la valeur de 15 images par seconde.

```
-- Lingo syntax
property spriteNum

on beginSprite me
    if sprite(spriteNum).member.frameRate < 15 then
        sprite(spriteNum).playBackMode = #fixed
        sprite(spriteNum).fixedRate = 15
    end if
end

// JavaScript syntax
function beginSprite () {
    var fr = sprite(this.spriteNum).member.frameRate;
    if (fr < 15) {
        sprite(this.spriteNum).playBackMode = symbol("fixed");
        sprite(this.spriteNum).fixedRate = 15;
    }
}
```

Voir aussi

fixedRate, playRate, currentTime (QuickTime, AVI), playBackMode

frameRate (DVD)

Syntaxe

```
-- Lingo syntax
dvdObjRef.frameRate

// JavaScript syntax
dvdObjRef.frameRate;
```

Description

Propriété de DVD. Renvoie la valeurframeRate du DVD. Lecture seule.

La valeur frameRate est renvoyée sous la forme de l'un des nombres à virgule flottante suivants :

Valeur à virgule flottante	Description
0,0	La valeur frameRate est impossible à déterminer parce qu'elle ne se trouve pas dans le domaine du titre ou que le titre n'est pas un titre vidéo séquentiel.
25,0	Le DVD a été créé pour être lu à 25 images par seconde.
30,0	Le DVD a été créé pour être lu à 30 images par seconde.
29,97	Le DVD a été créé pour être lu à 29,97 images par seconde.

Voir aussi

DVD

frameScript

Syntaxe

```
-- Lingo syntax
_movie.frameScript

// JavaScript syntax
_movie.frameScript;
```

Description

Propriété d'animation ; contient le numéro d'acteur unique du script d'image affecté à l'image en cours. Lecture/écriture uniquement lors d'une session d'enregistrement de scénario.

Au cours d'une session de création de scénario, vous pouvez également affecter un script à l'image en cours en définissant la propriété `frameScript`.

Si aucun script d'image n'est affecté à l'image en cours, cette propriété renvoie la valeur 0.

Exemple

L'instruction suivante affiche le numéro du script affecté à l'image actuelle. Dans ce cas, le numéro de script est 25.

```
-- Lingo syntax
put(_movie.frameScript)

// JavaScript syntax
put(_movie.frameScript);
```

L'instruction suivante définit l'acteur script Réponse des boutons en tant que script d'image pour l'image en cours :

```
-- Lingo syntax
_movie.frameScript = member("Button responses")

// JavaScript syntax
_movie.frameScript = member("Button responses");
```

Voir aussi

Animation

frameSound1

Syntaxe

```
-- Lingo syntax
_movie.frameSound1

// JavaScript syntax
_movie.frameSound1;
```

Description

Propriété d'animation ; détermine le numéro de l'acteur affecté à la première piste audio de l'image en cours. Lecture/écriture.

Cette propriété peut également être définie au cours d'une session d'enregistrement du scénario.

Exemple

Lors de la session d'enregistrement du scénario, l'instruction suivante affecte l'acteur son Jazz à la première piste audio :

```
-- Lingo syntax
_movie.frameSound1 = member("Jazz").number

// JavaScript syntax
_movie.frameSound1 = member("Jazz").number;
```

Voir aussi

frameSound2, Animation

frameSound2

Syntaxe

```
-- Lingo syntax
_movie.frameSound2

// JavaScript syntax
_movie.frameSound2;
```

Description

Propriété d'animation ; détermine le numéro de l'acteur affecté à la seconde piste audio de l'image en cours. Lecture/écriture.

Cette propriété peut également être définie au cours d'une session d'enregistrement du scénario.

Exemple

Lors de la session d'enregistrement du scénario, l'instruction suivante affecte l'acteur son Jazz à la seconde piste audio :

```
-- Lingo syntax
_movie.frameSound2 = member("Jazz").number

// JavaScript syntax
_movie.frameSound2 = member("Jazz").number;
```

Voir aussi

frameSound1, Animation

frameTempo

Syntaxe

```
-- Lingo syntax
_movie.frameTempo

// JavaScript syntax
_movie.frameTempo;
```

Description

Propriété d'animation ; indique la cadence affectée à l'image en cours. Lecture/écriture uniquement lors d'une session d'enregistrement de scénario.

Exemple

L'instruction suivante vérifie la cadence utilisée dans l'image actuelle. Dans ce cas, la cadence est de 15 images par seconde.

```
-- Lingo syntax
put(_movie.frameTempo)

// JavaScript syntax
put(_movie.frameTempo);
```

Voir aussi

Animation, puppetTempo()

frameTransition

Syntaxe

```
-- Lingo syntax
_movie.frameTransition

// JavaScript syntax
_movie.frameTransition;
```

Description

Propriété d'animation ; spécifie le numéro de l'acteur de transition affecté à l'image en cours. Lecture/écriture uniquement lors d'une session d'enregistrement de scénario afin de spécifier les transitions.

Exemple

Lorsque l'instruction suivante est utilisée au cours d'une session d'enregistrement de scénario, elle définit l'acteur Brouillard en tant que transition pour l'image en cours d'enregistrement par Lingo :

```
-- Lingo syntax
_movie.frameTransition = member("Fog")

// JavaScript syntax
_movie.frameTransition = member("Fog");
```

Voir aussi

Animation

front

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).front
```

Description

Propriété 3D de ressource de modèle #box ; indique si le côté de la boîte coupé par son axe des z négatif est fermé (TRUE) ou ouvert (FALSE).

La valeur par défaut de cette propriété est TRUE.

Exemple

L'instruction suivante définit la propriété front de la ressource de modèle Caisse sur la valeur FALSE, ce qui signifie que l'avant de la caisse est ouvert :

```
member("3D World").modelResource("Crate").front = FALSE
```

Voir aussi

back, bottom (3D), top (3D), left (3D), right (3D)

frontWindow

Syntaxe

```
-- Lingo syntax
_player.frontWindow

// JavaScript syntax
_player.frontWindow;
```

Description

Propriété de lecteur ; indique l'animation dans une fenêtre située au premier plan de l'écran. Lecture seule.

Lorsque la scène est au premier plan, la propriété frontWindow correspond à la scène. Lorsqu'un éditeur de média ou une palette flottante se trouve au premier plan, frontWindow renvoie la valeur VOID (Lingo) ou null (syntaxe JavaScript).

Exemple

L'instruction suivante détermine si la fenêtre Musique est la fenêtre qui se trouve au premier plan et, le cas échéant, amène la fenêtre Ecoutez ça à l'avant :

```
-- Lingo syntax
if (_player.frontWindow = "Music") then
    window("Try This").moveToFront()
end if

// JavaScript syntax
if (_player.frontWindow == "Music") {
    window("Try This").moveToFront();
}
```

Voir aussi

Lecteur

fullScreen

Syntaxe

```
-- Lingo syntax
dvdObjRef.fullScreen

// JavaScript syntax
dvdObjRef.fullScreen;
```

Description

Propriété de DVD ; indique si le DVD doit être lu en mode plein écran. Lecture/écriture.

La touche Echap désactive le mode d'affichage plein écran et définit cette propriété sur la valeur False.

Propriété non prise en charge sur Mac à ce jour.

Exemple

L'instruction suivante demande la lecture du DVD en mode plein écran :

```
-- Lingo syntax
member("DVDMember").fullScreen = TRUE

// Java Script
member("DVDMember").fullScreen = true;
```

Voir aussi

DVD

getBoneID

Syntaxe

```
memberReference.modelResource.getBoneID("boneName")
```

Description

Propriété 3D de ressource de modèle ; renvoie le numéro d'index du segment *nomDeSegment* de la ressource de modèle. Cette propriété renvoie 0 en l'absence d'un segment de ce nom.

Exemple

L'instruction suivante renvoie le numéro du segment tibiaG :

```
put member("ParkScene").modelResource("LittleKid").getBoneId("ShinL")
-- 40
```

Voir aussi

bone

globals

Syntaxe

```
the globals
```

Description

Propriété système ; cette propriété contient une liste de propriétés spéciale constituée de toutes les variables globales en cours présentant une valeur autre que `VOID`. Chaque variable globale est une propriété dans la liste, avec une valeur associée.

Vous pouvez utiliser les opérations de liste suivantes sur `globals` :

- `count()` : renvoie le nombre d'entrées de la liste.
- `getPropAt(n)` : renvoie le nom de la *n*ième entrée.
- `getProp(x)` : renvoie la valeur d'une entrée avec le nom spécifié.
- `getAProp(x)` : renvoie la valeur d'une entrée avec le nom spécifié.

Remarque : *la propriété `globals` contient automatiquement la propriété #version, qui correspond à la version de Director en cours d'exécution. Cela signifie qu'il y a toujours au moins une entrée dans la liste, même si aucune globale variable n'a encore été déclarée.*

Cette propriété diffère de `showGlobals` par le fait que `globals` est utilisable dans d'autres contextes que la fenêtre Messages. Pour afficher la liste `globals` dans la fenêtre Messages, utilisez la commande `showGlobals`.

Voir aussi

`showGlobals()`, `clearGlobals()`

glossMap

Syntaxe

```
member(whichCastmember).shader(whichShader).glossMap
member(whichCastmember).model(whichModel).shader.glossMap
member(whichCastmember).model(whichModel).shaderList{[index]}.glossMap
```

Description

Propriété 3D de matériau `#standard` ; spécifie la texture à utiliser pour le placage brillant.

Les propriétés suivantes sont automatiquement définies avec cette propriété :

- La quatrième couche de texture du matériau reçoit la texture que vous spécifiez.
- La valeur de `textureModeList[4]` est définie sur `#none`.
- La valeur de `blendFunctionList[4]` est définie sur `#multiply`.

Exemple

L'instruction suivante définit la texture Ovale en tant que valeur `glossMap` pour le matériau utilisé par le modèle boîteEnVerre :

```
-- Lingo syntax
member("3DPlanet").model("GlassBox").shader.glossMap = member("3DPlanet").texture("Oval")
```

```
// Java Script
member("House").getPropRef("model", 1).shaderList[1].glossMap =
member("House").getPropRef("texture",1);
```

Voir aussi

blendFunctionList, textureModeList, region, specularLightMap, diffuseLightMap

gravity

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).gravity
```

Description

Propriété 3D de ressource de modèle de particules ; lorsqu'elle est utilisée avec une ressource de modèle de type #particle, cette propriété vous permet d'obtenir ou de définir la propriété gravity de la ressource sous la forme d'un vecteur.

Cette propriété définit la force de gravité appliquée à toutes les particules de chaque palier de la simulation.

La valeur par défaut de cette propriété est vector(0,0,0).

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type #particle. L'instruction suivante définit la propriété gravity de systèmeThermique sur la valeur vector(0, -0,1, 0), ce qui a pour effet de tirer lentement les particules de systèmeThermique le long de l'axe des y.

```
-- Lingo syntax
member("Fires").modelResource("ThermoSystem").gravity = vector(0, -.1, 0)

// JavaScript syntax
member("Fires").getProp("modelResource", 1).gravity = vector(0, -.1, 0);
```

Voir aussi

drag, wind

gradientType

Syntaxe

```
-- Lingo syntax
memberObjRef.gradientType

// JavaScript syntax
memberObjRef.gradientType;
```

Description

Propriété d'acteur forme vectorielle ; spécifie le dégradé utilisé dans le remplissage de l'acteur.

Les valeurs possibles sont #linear ou #radial. La propriété gradientType n'est valide que lorsque fillMode présente la valeur #gradient.

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant bascule entre les dégradés linéaires ou radiaux dans l'acteur Fond.

```
-- Lingo syntax
on mouseUp me
    if member("backdrop").gradientType = #radial then
        member("backdrop").gradientType = #linear
    else
        member("backdrop").gradientType = #radial
    end if
end

// JavaScript syntax
function mouseUp() {
    var gt = member("backdrop").gradientType;
    if (gt == "radial") {
        member("backdrop").gradientType = symbol("linear");
    } else {
        member("backdrop").gradientType = symbol("radial");
    }
}
```

Voir aussi

fillMode

group

Syntaxe

```
member(whichCastmember).group(whichGroup)
member(whichCastmember).group[index]
```

Description

Elément 3D ; nœud de l'univers 3D qui a un nom, une transformation, un parent et des enfants, mais aucune autre propriété.

Chaque acteur 3D est associé à un groupe par défaut nommé Univers et qui ne peut pas être supprimé. La hiérarchie parente de tous les modèles, lumières, caméras et groupes qui existent dans l'univers 3D s'achève dans `group("world")`.

Exemple

La première ligne de cet exemple affiche le second groupe de l'acteur objets3D. La seconde ligne affiche le groupe collectionRB01 de l'acteur objets3D.

```
-- Lingo syntax
put member("3Dobjects").group("RBCollection01")
put member("3Dobjects").group[2]

// Javascript
put(member("3Dobjects").getPropRef("group",2));
```

Voir aussi

newGroup, deleteGroup, child (3D), parent

height

Syntaxe

```
-- Lingo syntax
imageObjRef.height
memberObjRef.height
spriteObjRef.height

// JavaScript syntax
imageObjRef.height;
memberObjRef.height;
spriteObjRef.height;
```

Description

Propriété d'image, d'acteur et d'image-objet ; pour les acteurs forme vectorielle, Flash, GIF animé, RealMedia, Windows Media, bitmap et forme, détermine la hauteur, en pixels, de l'acteur affiché sur la scène. Lecture seule pour les acteurs et les objets images, lecture/écriture pour les images-objets.

Exemple

L'instruction suivante affecte la hauteur de l'acteur Titre à la variable `vHeight` :

```
-- Lingo syntax
vHeight = member("Headline").height

// JavaScript syntax
var vHeight = member("Headline").height;
```

L'instruction suivante définit la hauteur de l'image-objet 10 sur 26 pixels :

```
-- Lingo syntax
sprite(10).height = 26

// JavaScript syntax
sprite(10).height = 26;
```

Voir aussi

Acteur, Image-objet, width

height (3D)

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).height
member(whichCastmember).texture(whichTexture).height
```

Description

Propriété 3D de ressource de modèle `#box` ou `#cylinder` et de texture ; indique la hauteur de l'objet.

La hauteur d'une ressource de modèle `#box` ou `#cylinder` est mesurée en unités de l'univers et peut être testée et définie. La valeur par défaut de cette propriété est 50.

La hauteur d'une texture est mesurée en pixels et peut être testée mais pas définie. La hauteur de la texture est arrondie à partir de la hauteur de la source de la texture à la puissance de 2 la plus proche.

Exemple

L'instruction suivante définit la hauteur de la ressource de modèle Tour sur 225,0 unités de l'univers :

```
member("3D World").modelResource("Tower").height = 225.0
```

L'instruction suivante indique que la hauteur de la texture placageMars est de 512 pixels.

```
put member("scene").texture("Marsmap").height
-- 512
```

Voir aussi

`length (3D)`, `width (3D)`

heightVertices

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).\heightVertices
```

Description

Propriété 3D de ressource de modèle `#box` ; indique le nombre de sommets de la maille le long de la hauteur de la boîte. L'augmentation de cette valeur augmente le nombre de faces et donc la précision de la maille.

La hauteur d'une boîte est mesurée sur son axe des y.

Définissez la propriété `renderStyle` du matériau d'un modèle sur la valeur `#wire` pour afficher toutes les faces de la maille de la ressource du modèle. Définissez la propriété `renderStyle` sur la valeur `#point` pour n'afficher que les sommets de la maille.

La valeur de cette propriété doit être supérieure ou égale à 2. La valeur par défaut est 4.

Exemple

L'instruction suivante attribue à la propriété `heightVertices` de la ressource de modèle Tour la valeur 10. Neuf polygones sont utilisés pour définir la géométrie de la ressource de modèle le long de son axe des z ; il y a donc dix sommets.

```
member("3D World").modelResource("Tower").heightVertices = 10
```

Voir aussi

`height (3D)`

highlightPercentage

Syntaxe

```
member(whichCastmember).model(whichModel).toon.highlightPercentage
member(whichCastmember).model(whichModel).shader.highlightPercentage
member(whichCastmember).shader(whichShader).highlightPercentage
```

Description

Propriété 3D de modificateur `toon` et de matériau `#painter` ; indique le pourcentage de couleurs disponibles utilisé dans la zone éclairée de la surface du modèle.

La plage de cette propriété s'étend de 0 à 100, la valeur par défaut étant 50.

Le nombre de couleurs utilisées par le modificateur `toon` et le matériau `painter` pour un modèle est déterminé par la propriété `colorSteps` du modificateur `toon` ou du matériau `#painter` du modèle.

Exemple

L'exemple suivant définit la propriété `highlightPercentage` du modificateur `toon` dans le modèle Sphère.

```
-- Lingo syntax
member("3Dobjects").model("Sphere01").toon.highlightPercentage = 25

// Javascript
member("3Dobjects").getPropRef("model",2).toon.highlightPercentage = 25;
```

Voir aussi

`highlightStrength`, `brightness`

highlightStrength

Syntaxe

```
member(whichCastmember).model(whichModel).toon.highlightStrength
member(whichCastmember).model(whichModel).shader.highlightStrength
member(whichCastmember).shader(whichShader).highlightStrength
```

Description

Propriété 3D de modificateur `toon` et de matériau `#painter` ; indique la luminosité de la zone éclairée de la surface du modèle.

La valeur par défaut de cette propriété est 1,0.

Exemple

L'exemple suivant définit la propriété `highlightStrength` du modificateur `toon` dans le modèle Sphère.

```
-- Lingo syntax
member("3Dobjects").model("Sphere01").toon.highlightStrength = 0.25

// Javascript
member("3Dobjects").getPropRef("model",2).toon.highlightStrength = 0.25;
```

Voir aussi

`highlightPercentage`, `brightness`

hilite

Syntaxe

```
-- Lingo syntax
memberObjRef.hilite

// JavaScript syntax
memberObjRef.hilite;
```

Description

Propriété d'acteur ; détermine si une case à cocher ou un bouton radio créé avec l'outil bouton est sélectionné (TRUE) ou non (FALSE, valeur par défaut). Lecture/écriture.

Exemple

L'instruction suivante vérifie si le bouton Son activé est sélectionné et, le cas échéant, augmente le volume de la piste audio 1 au maximum :

```
-- Lingo syntax
if (member("Sound On").hilite = TRUE) then
    sound(1).volume = 255
end if

// JavaScript syntax
if (member("Sound On").hilite == true) {
    sound(1).volume = 255;
}
```

L'instruction suivante sélectionne l'acteur bouton Interrupteur en définissant la propriété d'acteur `hilite` sur la valeur `TRUE` :

```
-- Lingo syntax
member("powerSwitch").hilite = TRUE

// JavaScript syntax
member("powerSwitch").hilite = true;
```

Voir aussi

Acteur

hither

Syntaxe

```
member(whichCastmember).camera(whichCamera).hither
sprite(whichSprite).camera{(index)}.hither
```

Description

Propriété 3D de caméra ; indique la distance, en unités de l'univers et à partir de la caméra, à partir de laquelle les modèles sont tracés. Les objets plus proches de la caméra que le point `hither` ne sont pas dessinés.

La valeur de cette propriété doit être supérieure ou égale à 1,0 et a une valeur par défaut de 5,0.

Exemple

L'instruction suivante attribue à la propriété `hither` de la caméra 1 la valeur 1 000. Les modèles plus proches de la caméra que 1 000 unités de l'univers ne sont pas visibles.

```
member("SolarSystem").camera[1].hither = 1000
```

Voir aussi

yon

hotSpot

Syntaxe

```
-- Lingo syntax
memberObjRef.hotSpot

// JavaScript syntax
memberObjRef.hotSpot;
```

Description

Propriété d'acteur curseur ; spécifie l'emplacement horizontal et vertical du pixel représentant la zone référencée dans l'acteur curseur couleur animé `quelActeurCurseur`. Director utilise ce point pour suivre la position du curseur à l'écran (par exemple, lorsqu'il renvoie les valeurs des fonctions Lingo `mouseH` et `mouseV`) et pour déterminer l'endroit d'un survol (signalé par le message Lingo `mouseEnter`).

L'angle supérieur gauche du curseur est le point(0, 0), qui correspond à la valeur par défaut de `hotSpot`. La définition d'un point en dehors des limites du curseur produit une erreur. Par exemple, le réglage de la zone référencée d'un curseur 16 x 16 pixels sur point(16, 16) produit une erreur (le point de départ étant 0, 0 et non 1, 1).

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant définit la zone référencée d'un curseur de 32x32 pixels (dont le numéro d'acteur est enregistré dans la variable `cursorNum`) au milieu du curseur :

```
-- Lingo syntax
on startMovie
    member(cursorNum).hotSpot = point(16,16)
end

// JavaScript syntax
function startMovie() {
    member(cursorNum).hotSpot = point(16,16);
}
```

hotSpotEnterCallback

Syntaxe

```
-- Lingo syntax
spriteObjRef.hotSpotEnterCallback

// JavaScript syntax
spriteObjRef.hotSpotEnterCallback;
```

Description

Propriété d'image-objet QuickTime VR ; contient le nom du gestionnaire exécuté lorsque le curseur entre dans une zone référencée QuickTime VR visible sur la scène. L'image-objet QuickTime VR reçoit le message en premier. Ce message a deux arguments : le paramètre `me` et l'identifiant de la zone référencée dans laquelle le curseur est entré.

Pour annuler l'appel, donnez à cette propriété une valeur de 0.

Pour des performances optimales, ne définissez de propriété d'appel que lorsque absolument nécessaire.

Cette propriété peut être testée et définie.

Exemple

L'exemple suivant indique le nom du gestionnaire exécuté lorsque le curseur entre dans une zone référencée QuickTime VR visible sur la scène.

```
-- Lingo syntax
put sprite("multinode").hotSpotEnterCallback

// Javascript
put(sprite("multinode").hotSpotEnterCallback);
```

Voir aussi

hotSpotExitCallback, nodeEnterCallback, nodeExitCallback, triggerCallback

hotSpotExitCallback

Syntaxe

```
-- Lingo syntax
spriteObjRef.hotSpotExitCallback

// JavaScript syntax
spriteObjRef.hotSpotExitCallback;
```

Description

Propriété d'image-objet QuickTime VR ; contient le nom du gestionnaire exécuté lorsque le curseur quitte une zone référencée QuickTime VR visible sur la scène. L'image-objet QuickTime VR reçoit le message en premier. Ce message a deux arguments : le paramètre `me` et l'identifiant de la zone référencée dans laquelle le curseur est entré.

Pour annuler l'appel, donnez à cette propriété une valeur de 0.

Pour des performances optimales, ne définissez de propriété d'appel uniquement si cela est absolument nécessaire.

Cette propriété peut être testée et définie.

Exemple

L'exemple suivant indique le nom du gestionnaire exécuté lorsque le curseur quitte une zone référencée QuickTime VR visible sur la scène.

```
-- Lingo syntax
put sprite("multinode").hotSpotExitCallback

// Javascript
put(sprite("multinode").hotSpotExitCallback);
```

Voir aussi

hotSpotEnterCallback, nodeEnterCallback, nodeExitCallback, triggerCallback

HTML

Syntaxe

```
-- Lingo syntax
memberObjRef.HTML
```

```
// JavaScript syntax
memberObjRef.HTML;
```

Description

Propriété d'acteur ; accède au texte et aux balises contrôlant la disposition du texte dans un acteur texte au format HTML.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche dans la fenêtre Messages l'information de format HTML intégrée à l'acteur texte Page d'accueil :

```
--Lingo syntax
put(member("Home Page").HTML)

// JavaScript syntax
trace(member("Home Page").HTML);
```

Voir aussi

importFileInto(), RTF

hyperlink

Syntaxe

```
-- Lingo syntax
chunkExpression.hyperlink

// JavaScript syntax
chunkExpression.hyperlink;
```

Description

Propriété d'acteur texte ; renvoie la chaîne de lien hypertexte pour l'expression de sous-chaîne spécifiée dans l'acteur texte.

Cette propriété peut être testée et définie.

Lors de la récupération de cette propriété, le lien contenant le premier caractère de *expressionSousChaîne* est utilisé.

Les liens hypertexte ne peuvent pas se chevaucher. La définition d'un lien hypertexte sur un lien existant (même partiel) remplace le lien initial par le nouveau.

La définition d'un lien hypertexte en chaîne vide supprime ce lien.

Exemple

Le gestionnaire suivant crée un lien hypertexte dans le premier mot de l'acteur texte LienMacromedia. Ce texte est lié au site Web d'Adobe.

```
--Lingo syntax
on startMovie
    member("MacroLink").word[1].hyperlink = "http://www.adobe.com"
end
```

```
// JavaScript syntax
function startMovie() {
    member("MacroLink").getPropRef("word", 1).hyperlink = "http://www.adobe.com";
}
```

Voir aussi

hyperlinkRange, hyperlinkState

hyperlinkRange

Syntaxe

```
-- Lingo syntax
chunkExpression.hyperlinkRange

// JavaScript syntax
chunkExpression.hyperlinkRange;
```

Description

Propriété d'acteur texte ; renvoie la plage du lien hypertexte contenant le premier caractère de l'expression de sous-chaîne.

Cette propriété peut être testée, mais pas définie.

De même que pour les propriétés `hyperLink` et `hyperLinkState`, la plage du lien renvoyée contient le premier caractère de *`expressionSousChaîne`*.

Exemple

L'exemple suivant affiche la plage du lien hypertexte contenant le premier caractère de l'expression de sous-chaîne.

```
-- Lingo syntax
put member("MyText").hyperlinkRange

// Javascript
put(member("MyText").hyperlinkRange);
```

Voir aussi

hyperlink, hyperlinkState

hyperlinks

Syntaxe

```
-- Lingo syntax
chunkExpression.hyperlinks

// JavaScript syntax
chunkExpression.hyperlinks;
```

Description

Propriété d'acteur texte ; renvoie une liste linéaire contenant toutes les plages de liens hypertexte pour l'expression de sous-chaîne spécifiée de l'acteur texte. Chaque plage est donnée sous la forme d'une liste linéaire avec deux éléments, un pour le caractère de début du lien et un pour le caractère de fin.

Exemple

L'instruction suivante renvoie tous les liens de l'acteur texte Glossaire dans la fenêtre Messages :

```
--Lingo syntax
put(member("Glossary").hyperlinks) -- [[3, 8], [10, 16], [41, 54]]

// JavaScript syntax
trace(member("Glossary").hyperlinks); // [[3, 8], [10, 16], [41, 54]]
```

hyperlinkState

Syntaxe

```
-- Lingo syntax
chuckExpression.hyperlinkState

// JavaScript syntax
chuckExpression.hyperlinkState;
```

Description

Propriété d'acteur texte ; contient l'état actuel du lien hypertexte. Les valeurs possibles de l'état sont : `#normal`, `#active` et `#visited`.

Cette propriété peut être testée et définie.

De même que pour les propriétés `hyperLink` et `hyperLinkRange`, la plage du lien renvoyée contient le premier caractère de *`expressionSousChaîne`*.

Exemple

Le gestionnaire suivant vérifie si le lien hypertexte sélectionné est une adresse web. Le cas échéant, l'état du lien hypertexte prend la valeur `#visited` et l'animation est orientée vers l'adresse Web.

```
--Lingo syntax
property spriteNum

on hyperlinkClicked me, data, range
    if data starts "http://" then
        currentMember = sprite(spriteNum).member
        currentMember.word[4].hyperlinkState = #visited
        gotoNetPage(data)
    end if
end

// JavaScript syntax
function hyperlinkClicked(data, range) {
    var st = data.slice(0,7);
    var ht = "http://";
    if (st == ht) {
        currentMember = sprite(spriteNum).member;
        currentMember.getPropRef("word", 4).hyperlinkState = symbol("visited");
        gotoNetPage(data);
    }
}
```

Voir aussi

`hyperlink`, `hyperlinkRange`

idleHandlerPeriod

Syntaxe

```
-- Lingo syntax
_movie.idleHandlerPeriod

// JavaScript syntax
_movie.idleHandlerPeriod;
```

Description

Propriété d'animation ; détermine le nombre de battements maximal avant l'envoi d'un message `idle` par l'animation. Lecture/écriture.

La valeur par défaut est de 1, ce qui indique à l'animation de ne pas envoyer de messages de gestionnaires `idle` plus de 60 fois par seconde.

Lorsque la tête de lecture entre dans une image, Director démarre un compteur, redessine les images-objets appropriées sur la scène et émet un événement `enterFrame`. Ensuite, si le temps défini pour la cadence s'est écoulé, Director génère un événement `exitFrame` et passe à l'image suivante spécifiée ; si le temps défini pour cette image ne s'est pas écoulé, Director attend que ce temps se soit écoulé et génère périodiquement un message `idle`. Le temps écoulé entre les événements `idle` est déterminé par la propriété `idleHandlerPeriod`.

Les valeurs possibles pour `idleHandlerPeriod` sont les suivantes :

- 0 : autant d'événements idle que possible.
- 1 : jusqu'à 60 par seconde.
- 2 : jusqu'à 30 par seconde.
- 3 : jusqu'à 20 par seconde.
- *n* : jusqu'à 60/*n* par seconde.

Le nombre d'événements `idle` par image dépend également de la cadence d'image de l'animation et d'autres activités, notamment si des scripts sont ou non en cours d'exécution. Si la cadence est de 60 images par seconde (ips) et que la valeur de `idleHandlerPeriod` est de 1, un seul événement `idle` se produit par image. Si la cadence est de 20 ips, trois événements `idle` surviennent par image. Un temps mort résulte du fait que Director n'a pas de tâches à exécuter et ne peut pas générer d'événements.

A l'inverse, si la propriété `idleHandlerPeriod` présente la valeur 0 et que la cadence est très basse, des milliers d'événements `idle` peuvent être générés.

La valeur par défaut de cette propriété est 1.

Exemple

L'instruction suivante demande à l'animation d'envoyer un message idle une fois par seconde au maximum :

```
-- Lingo syntax
_movie.idleHandlerPeriod = 60

// JavaScript syntax
_movie.idleHandlerPeriod = 60;
```

Voir aussi

`on idle`, `idleLoadMode`, `idleLoadPeriod`, `idleLoadTag`, `idleReadChunkSize`, `Animation`

idleLoadMode

Syntaxe

```
-- Lingo syntax
_movie.idleLoadMode

// JavaScript syntax
_movie.idleLoadMode;
```

Description

Propriété d'animation ; détermine le moment où les méthodes `preLoad()` et `preLoadMember()` tentent de charger des acteurs pendant les périodes d'inactivité. Lecture/écriture.

Les périodes d'inactivité peuvent prendre l'une des valeurs suivantes :

- 0 : aucun chargement en période d'inactivité.
- 1 : chargement pendant les périodes d'inactivité entre les images.
- 2 : chargement pendant les événements `idle`.
- 3 : chargement aussi fréquemment que possible.

La propriété système `idleLoadMode` n'effectue aucune fonction et ne fonctionne qu'avec les méthodes `preLoad()` et `preLoadMember()`.

Les acteurs chargés pendant les périodes d'inactivité restent compressés jusqu'à ce que l'animation les utilise. Lors de la lecture de l'animation, des pauses importantes peuvent se produire pendant la décompression des acteurs.

Exemple

L'instruction suivante ordonne à l'animation d'essayer de charger aussi fréquemment que possible les acteurs à précharger par le biais des commandes `preLoad` et `preLoadMember` :

```
-- Lingo syntax
_movie.idleLoadMode = 3

// JavaScript syntax
_movie.idleLoadMode = 3;
```

Voir aussi

`on idle`, `Animation`, `preLoad() (animation)`, `preLoadMember()`

idleLoadPeriod

Syntaxe

```
-- Lingo syntax
_movie.idleLoadPeriod

// JavaScript syntax
_movie.idleLoadPeriod;
```

Description

Propriété d'animation ; détermine le nombre de battements pendant lequel Director attend avant de tenter de charger les acteurs en attente de chargement. Lecture/écriture.

La valeur par défaut de `idleLoadPeriod` est de 0, ce qui indique à Director de traiter la file d'attente de chargement aussi fréquemment que possible.

Exemple

L'instruction suivante indique à Director d'essayer de charger toutes les demi-secondes (30 battements) les acteurs en attente de chargement :

```
-- Lingo syntax
_movie.idleLoadPeriod = 30

// JavaScript syntax
_movie.idleLoadPeriod = 30;
```

Voir aussi

`on idle`, `Animation`

idleLoadTag

Syntaxe

```
-- Lingo syntax
_movie.idleLoadTag

// JavaScript syntax
_movie.idleLoadTag;
```

Description

Propriété d'animation ; identifie les acteurs en attente de chargement pendant les périodes d'inactivité de l'ordinateur ou leur affecte un numéro. Lecture/écriture.

La propriété `idleLoadTag` se révèle utile pour identifier les acteurs d'un groupe à précharger et peut prendre n'importe quelle valeur.

Exemple

L'instruction suivante fait du numéro 10 la balise de chargement en période d'inactivité :

```
-- Lingo syntax
_movie.idleLoadTag = 10

// JavaScript syntax
_movie.idleLoadTag = 10;
```

Voir aussi

`on idle`, `Animation`

idleReadChunkSize

Syntaxe

```
-- Lingo syntax
_movie.idleReadChunkSize
```

```
// JavaScript syntax
_movie.idleReadChunkSize;
```

Description

Propriété d'animation ; détermine le nombre maximal d'octets que Director peut charger lorsqu'il tente de charger les acteurs à partir de la file d'attente de chargement. Lecture/écriture.

La valeur par défaut de `idleReadChunkSize` est de 32 Ko.

Exemple

L'instruction suivante spécifie que 500 Ko est le nombre maximal d'octets que Director peut charger lors d'un essai de chargement des acteurs situés dans la file d'attente :

```
-- Lingo syntax
_movie.idleReadChunkSize = (500 * 1024)

// JavaScript syntax
_movie.idleReadChunkSize = (500 * 1024);
```

Voir aussi

`on idle`, `Animation`

image (image)

Syntaxe

```
-- Lingo syntax
imageObjRef.image

// JavaScript syntax
imageObjRef.image;
```

Description

Propriété d'image. Fait référence à l'objet image d'un acteur bitmap ou texte, de la scène ou d'une fenêtre. Lecture/écriture pour l'image d'un acteur, lecture seule pour une image de la scène ou d'une fenêtre.

La définition de la propriété `image` d'un acteur modifie immédiatement le contenu de l'acteur. Toutefois, lorsque vous obtenez l'image d'un acteur ou d'une fenêtre, Director crée une référence à l'image de l'acteur ou de la fenêtre en question. Si vous apportez des modifications aux fenêtres, le contenu de l'acteur ou de la fenêtre change immédiatement.

Si vous envisagez d'apporter de nombreuses modifications à la propriété `image` d'un élément, il est plus rapide de copier sa propriété image dans un nouvel objet image à l'aide de la méthode `duplicate()`, d'apporter vos modifications au nouvel objet image, puis d'appliquer l'image initiale de l'élément au nouvel objet image. Pour les acteurs autres que bitmap, il est toujours plus rapide d'utiliser la méthode `duplicate()`.

Exemple

L'instruction suivante place l'image de l'acteur fleurDorigine dans l'acteur nouvelleFleur :

```
-- Lingo syntax
member("newFlower").image = member("originalFlower").image

// JavaScript syntax
member("newFlower").image = member("originalFlower").image;
```

Les instructions suivantes placent une référence à l'image de la scène dans la variable `myImage`, puis placent cette image dans l'acteur Fleur :

```
-- Lingo syntax
myImage = _movie.stage.image
member("flower").image = myImage

// JavaScript syntax
var myImage = _movie.stage.image;
member("flower").image = myImage;
```

Voir aussi

copyPixels(), draw(), duplicate() (image), fill(), image(), setPixel()

image (RealMedia)

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.image

// JavaScript syntax
memberOrSpriteObjRef.image;
```

Description

Propriété d'acteur ou d'image-objet RealMedia ; renvoie un objet image Lingo contenant l'image en cours du flux vidéo RealMedia. Vous pouvez utiliser cette propriété pour placer du contenu RealVideo® sur un modèle 3D (voir l'exemple ci-après).

Exemple

Cette instruction copie l'image actuelle de l'acteur RealMedia Real sur l'acteur bitmap Insta :

```
-- Lingo syntax
member("Still").image = member("Real").image

// JavaScript syntax
member("Still").image = member("Real").image;
```

image (fenêtre)

Syntaxe

```
-- Lingo syntax
windowObjRef.image

// JavaScript syntax
windowObjRef.image;
```

Description

Propriété de fenêtre ; fait référence à l'objet image d'une fenêtre. Lecture seule.

Lorsque vous obtenez l'image d'une fenêtre, Director crée une référence à l'image de la fenêtre en question. Si vous apportez des modifications à l'image, le contenu de la fenêtre change immédiatement.

Si vous envisagez d'apporter de nombreuses modifications à la propriété `image` d'un élément, il est plus rapide de copier sa propriété `image`dans un nouvel objet image à l'aide de la méthode `duplicate()` de l'objet acteur, d'apporter vos modifications au nouvel objet image, puis d'appliquer l'image initiale de l'élément au nouvel objet image. Pour les acteurs autres que bitmap, il est toujours plus rapide d'utiliser la méthode `duplicate()`.

Exemple

Les instructions suivantes placent une référence à l'image de la scène dans la variable `myImage`, puis placent cette image dans la fenêtre Fleur :

```
-- Lingo syntax
myImage = _movie.stage.image
window("Flower").image = myImage

// JavaScript syntax
var myImage = _movie.stage.image;
window("Flower").image = myImage;
```

Voir aussi

`duplicate() (acteur)`, `Fenêtre`

imageCompression

Syntaxe

```
-- Lingo syntax
_movie.imageCompression
memberObjRef.imageCompression

// JavaScript syntax
_movie.imageCompression;
memberObjRef.imageCompression;
```

Description

Propriété d'animation et d'acteur bitmap ; indique le type de compression que Director applique aux acteurs bitmap internes (non liés) lors de l'enregistrement d'une animation au format Shockwave Player. Lecture/écriture.

Les valeurs possibles de la propriété `imageCompression` sont les suivantes :

Valeur	Signification
#standard	Utilise le format de compression interne standard de Director.
#movieSetting	Utilise les paramètres de compression de l'animation stockés dans la propriété `movie.imageCompression`. Il s'agit de la valeur par défaut pour les formats d'image non limités à la compression standard.
#jpeg	Utilise la compression JPEG. Reportez-vous à l'entrée `imageQuality`.

Vous devez normalement définir cette propriété dans la boîte de dialogue Paramètres de publication de Director.

Exemple

L'instruction suivante affiche dans la fenêtre Messages la valeur imageCompression qui s'applique à l'animation en cours de lecture :

```
-- Lingo syntax
put(_movie.imageCompression)
```

```
// JavaScript syntax
put(_movie.imageCompression);
```

Voir aussi

imageQuality, Animation

imageEnabled

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.imageEnabled

// JavaScript syntax
memberOrSpriteObjRef.imageEnabled;
```

Description

Propriété d'acteur et d'image-objet ; contrôle si les graphiques d'une animation Flash ou d'une forme vectorielle sont visibles (TRUE, valeur par défaut) ou invisibles (FALSE).

Cette propriété peut être testée et définie.

Exemple

Le script beginSprite suivant masque les graphiques d'une image-objet d'animation Flash liée lorsqu'elle apparaît pour la première fois sur la scène et commence à arriver en mémoire, puis enregistre son numéro d'image-objet dans une variable globale intitulée gStreamingSprite pour l'utiliser ultérieurement dans un script d'image du scénario :

```
-- Lingo syntax
global gStreamingSprite

on beginSprite me
    gStreamingSprite = me.spriteNum
    sprite(gStreamingSprite).imageEnabled = FALSE
end

// JavaScript syntax
function beginSprite() {
    _global.gStreamingSprite = this.spriteNum;
    sprite(_global.gStreamingSprite).imageEnabled = 0;
}
```

Dans une image suivante de l'animation, ce script d'image vérifie si l'image-objet de l'animation Flash spécifiée par la variable globale gStreamingSprite a été transférée en mémoire. Si ce n'est pas le cas, le script maintient la tête de lecture en boucle dans l'image actuelle jusqu'à ce que l'animation ait été intégralement transférée en mémoire. Il définit ensuite la propriété imageEnabled sur la valeur TRUE de façon à ce que le graphique apparaisse et laisse la tête de lecture passer à l'image suivante du scénario.

```
-- Lingo syntax
global gStreamingSprite

on exitFrame me
    if sprite(gStreamingSprite).member.percentStreamed < 100 then
        _movie.go(_movie.frame)
    else
```

```
        sprite(gStreamingSprite).imageEnabled = TRUE
        _movie.updatestage()
    end if
end

// JavaScript syntax
function exitFrame() {
    var stmSp = sprite(_global.gStreamingSprite).member.percentStreamed;
    if (stmSp < 100) {
        _movie.go(_movie.frame);
    } else {
        sprite(_global.gStreamingSprite).imageEnabled = 1;
        _movie.updatestage();
    }
}
```

imageQuality

Syntaxe

```
-- Lingo syntax
_movie.imageQuality
memberObjRef.imageQuality

// JavaScript syntax
_movie.imageQuality;
memberObjRef.imageQuality;
```

Description

Propriété d'animation et d'acteur bitmap ; indique le niveau de compression à utiliser lorsque la propriété `imageCompression` d'une animation est définie sur `#jpeg`. Lecture/écriture uniquement en phase de création.

La plage de valeurs admises s'étend de 0 à 100. Zéro produit la qualité d'image la moins bonne et le plus haut degré de compression, tandis que 100 produit la meilleure qualité d'image et le degré de compression le plus bas.

Cette propriété n'est définissable qu'en phase de création et n'a aucun effet tant que l'animation n'est pas enregistrée au format Shockwave Player.

Exemple

L'instruction suivante affiche dans la fenêtre Messages la valeur `imageQuality` qui s'applique à l'animation en cours de lecture :

```
-- Lingo syntax
put(_movie.imageQuality)

// JavaScript syntax
put(_movie.imageQuality);
```

Voir aussi

`imageCompression`, `Animation`

immovable

Syntaxe

```
member(whichCastmember).model(whichModel).collision.immovable
```

Description

Propriété 3D de modificateur `#collision` ; indique si un modèle peut être déplacé à la suite de collisions durant les animations. La valeur `TRUE` rend le modèle immuable ; la valeur `FALSE` autorise le déplacement du modèle. Cette propriété est un moyen pratique d'améliorer les performances au cours de l'animation étant donné que Lingo n'a pas à vérifier les collisions pour les modèles immobiles.

La valeur par défaut de cette propriété est `FALSE`.

Exemple

L'instruction suivante attribue à la propriété `immovable` du modificateur `collision` associé au second modèle de l'acteur objets3D la valeur `TRUE`.

```
-- Lingo syntax
member("3Dobjects").model[2].collision.immovable = TRUE

// Javascript
member("3Dobjects").getPropRef("model",2).collision.immovable = 1;
```

Voir aussi

collision (modificateur)

ink

Syntaxe

```
-- Lingo syntax
spriteObjRef.ink

// JavaScript syntax
spriteObjRef.ink;
```

Description

Propriété d'image-objet ; détermine l'effet d'encre appliqué à une image-objet. Lecture/écriture.

Les valeurs possibles de la propriété `ink` sont les suivantes :

0 - Copie	32 - Opacité
1 - Transparente	33 - Somme limitée
2 - Inverse	34 - Somme
3 - Spectre	35 - Différence limitée
4 - Copie nég.	36 - Fond transparent
5 - Transp. nég.	37 - Plus claire
6 - Inverse nég.	38 - Différence

7 - Spectre nég.	39 - Plus foncée
8 - Dessin seul	40 - Eclaircir
9 - Masque	41 - Assombrir

Dans le cas de la valeur 36 (Fond transparent), vous sélectionnez une image-objet dans le scénario et une couleur de transparence dans la zone de couleur d'arrière-plan de la fenêtre Outils. Vous pouvez également effectuer cette opération en définissant la propriété `backColor`.

Si vous définissez cette propriété dans un script alors que la tête de lecture ne bouge pas, prenez soin d'utiliser la méthode `updateStage()` de l'objet animation pour redessiner la scène. Si vous modifiez plusieurs propriétés d'image-objet ou plusieurs images-objets, utilisez une seule méthode `updateStage()` à l'issue de toutes les modifications.

Exemple

L'instruction suivante attribue à la variable `encreactuelle` la valeur de l'effet d'encre de l'image-objet (3) :

```
-- Lingo syntax
currentInk = sprite(3).ink

// JavaScript syntax
var currentInk = sprite(3).ink;
```

L'instruction suivante attribue à l'image-objet (i + 1) un effet d'encre Dessin seul en définissant l'effet d'encre de la propriété `sprite` sur la valeur 8, qui spécifie une encre Dessin seul :

```
-- Lingo syntax
sprite(i + 1).ink = 8

// JavaScript syntax
sprite(i + 1).ink = 8;
```

Voir aussi

`backColor`, `Image-objet`, `updateStage()`

inker (modificateur)

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).inker.inkerModifierProperty
modelResourceObjectReference.inker.inkerModifierProperty
```

Description

Modificateur 3D ; après avoir ajouté le modificateur `#inker` à une ressource de modèle (avec `addModifier`), vous pouvez en obtenir et en définir les propriétés.

Le modificateur `#inker` ajoute des silhouettes, des plis et des bords de délimitation à un modèle existant ; les propriétés `#inker` vous permettent d'en contrôler la définition et l'ampleur.

Lorsque le modificateur `#inker` est utilisé en conjonction avec le modificateur `#toon`, le rendu est cumulatif et varie en fonction du premier modificateur appliqué. La liste des modificateurs renvoyée par la propriété `modifier` indique `#inker` ou `#toon` (en fonction de celui qui a été ajouté en premier), mais non les deux. Le modificateur `#inker` n'est pas utilisable en conjonction avec le modificateur `#sds`.

Le modificateur `#inker` comporte les propriétés suivantes :

- `lineColor` permet d'obtenir ou de définir la couleur des lignes dessinées par le modificateur.
- `silhouettes` permet de savoir ou de définir si les lignes sont dessinées de façon à définir les bords le long de la bordure d'un modèle, en soulignant la forme.
- `creases` permet de savoir ou de définir si les lignes sont dessinées avec des plis.
- `creaseAngle` permet de vérifier ou de définir la sensibilité de détection des angles des plis pour le modificateur.
- `boundary` permet de savoir ou de définir si les lignes sont dessinées autour de la limite de la surface.
- `lineOffset` permet de savoir ou de définir si les lignes sont tracées en fonction de la surface et de la caméra.
- `useLineOffset` permet de savoir ou de définir si `lineOffset` est activé ou désactivé.

Remarque : *pour plus d'informations, consultez les entrées des différentes propriétés.*

Exemple

L'instruction suivante ajoute le modificateur `inker` au second modèle de l'acteur objets3D. Après avoir ajouté le modificateur `#inker` à une ressource de modèle (avec addModifier), vous pouvez en obtenir et en définir les propriétés.

```
-- Lingo syntax
member("3Dobjects").model[2].addModifier(#inker)

// JavaScript syntax
member("3Dobjects").getPropRef("model",2).addModifier(symbol("inker"));
```

Voir aussi

`addModifier`, `modifiers`, `toon (modificateur)`, `shadowPercentage`

inlineImeEnabled

Syntaxe

```
-- Lingo syntax
_player.inlineImeEnabled

// JavaScript syntax
_player.inlineImeEnabled;
```

Description

Dans Director 11, la propriété inlineImeEnabled présente toujours la valeur TRUE, qu'elle soit définie sur TRUE ou sur FALSE par l'intermédiaire d'un script. Cette propriété est uniquement conservée pour garantir une compatibilité en amont.

Exemple

L'instruction suivante définit la propriété `inlineImeEnabled` du lecteur sur `TRUE`.

```
-- Lingo syntax
_player.inlineImeEnabled=TRUE

// JavaScript syntax
_player.inlineImeEnabled=1;
```

Voir aussi

Lecteur

interval

Syntaxe

```
-- Lingo syntax
memberObjRef.interval

// JavaScript syntax
memberObjRef.interval;
```

Description

Propriété d'acteur curseur ; spécifie l'intervalle, en millisecondes (ms), entre chaque image de l'acteur curseur couleur animé *quelActeurCurseur*. L'intervalle par défaut est de 100 ms.

L'intervalle du curseur est indépendant de la cadence d'images définie pour l'animation dans la piste des cadences ou avec la commande Lingo `puppetTempo`.

Cette propriété peut être testée et définie.

Exemple

Dans ce script d'image-objet, lorsque le curseur animé en couleur stocké dans l'acteur Papillon pénètre dans l'image-objet, l'intervalle est réglé sur 50 ms pour accélérer l'animation. Lorsque le curseur quitte l'image-objet, l'intervalle est réglé sur 100 ms pour ralentir l'animation.

```
-- Lingo syntax
on mouseEnter
    member("Butterfly").interval = 50
end

on mouseLeave
    member("Butterfly").interval = 100
end

// JavaScript syntax
function mouseEnter() {
    member("Butterfly").interval = 50;
}

function mouseLeave() {
    member("Butterfly").interval = 100;
}
```

invertMask

Syntaxe

```
-- Lingo syntax
memberObjRef.invertMask

// JavaScript syntax
memberObjRef.invertMask;
```

Description

Propriété d'acteur QuickTime ; détermine si Director dessine les animations QuickTime dans les pixels blancs du masque de l'animation (TRUE) ou dans ses pixels noirs (FALSE, valeur par défaut).

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant inverse la valeur en cours de la propriété `invertMask` d'une animation QuickTime intitulée Etoile :

```
-- Lingo syntax
on toggleMask
    member("Starburst").invertMask = not(member("Starburst").invertMask)
end

// JavaScript syntax
function toggleMask() {
    member("Starburst").invertMask = !(member("Starburst").invertMask);
}
```

Voir aussi

mask

isVRMovie

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.isVRMovie

// JavaScript syntax
memberOrSpriteObjRef.isVRMovie;
```

Description

Propriété d'image-objet et d'acteur QuickTime ; indique si un acteur ou une image-objet est une animation QuickTime VR non encore téléchargée (TRUE) ou si l'acteur ou l'image-objet n'est pas une animation QuickTime VR (FALSE).

Le test de cette propriété dans n'importe quel élément dont le type est différent de #quickTimeMedia produira un message d'erreur.

Cette propriété peut être testée, mais pas définie.

Exemple

Le gestionnaire suivant vérifie si l'acteur d'une image-objet est une animation QuickTime. Le cas échéant, le gestionnaire poursuit sa recherche pour vérifier s'il s'agit d'une animation QuickTime VR. Dans les deux cas, un message d'alerte est généré.

```
-- Lingo syntax
on checkForVR(theSprite)
    if sprite(theSprite).member.type = #quickTimeMedia then
        if sprite(theSprite).isVRMovie then
            _player.alert("This is a QTVR asset.")
        else
            _player.alert("This is not a QTVR asset.")
```

```
        end if
    else
        _player.alert("This is not a QuickTime asset.")
    end if
end

// JavaScript syntax
function checkForVR(theSprite) {
    var memType = sprite(theSprite).member.type;
    if (memType == "quickTimeMedia") {
        var isType = sprite(theSprite).isVRMovie;
        if (isType == 1) {
            _player.alert("This is a QTVR asset.");
        } else {
            _player.alert("This is not a QTVR asset.");
        } else {
            _player.alert("This is not a QuickTime asset.");
        }
    }
}
```

itemDelimiter

Syntaxe

```
the itemDelimiter
```

Description

Propriété de lecteur ; indique le caractère spécial utilisé pour délimiter les éléments.

Vous pouvez utiliser la propriété `itemDelimiter` pour analyser des noms de fichier en lui attribuant la valeur d'une barre oblique inversée (\) sous Windows ou d'un deux-points (:) sur le Mac. Pour revenir à un fonctionnement normal, redéfinissez la propriété `itemDelimiter` sur une virgule (,).

Cette fonction peut être testée et définie.

Exemple

Le gestionnaire suivant recherche le dernier élément d'un chemin d'accès Mac. Il enregistre d'abord le séparateur actuel, puis lui donne la valeur de deux-points (:). Lorsque le séparateur est un deux-points, Lingo peut utiliser `the last item of` pour déterminer le dernier élément de la sous-chaîne constituant le chemin d'accès Mac. Avant de quitter le gestionnaire, le séparateur reprend sa valeur d'origine.

```
on getLastComponent pathName
    save = the itemDelimiter
    the itemDelimiter = ":"
    f = the last item of pathName
    the itemDelimiter = save
    return f
end
```

Voir aussi

Lecteur

kerning

Syntaxe

```
-- Lingo syntax
memberObjRef.kerning

// JavaScript syntax
memberObjRef.kerning;
```

Description

Propriété d'acteur texte ; spécifie si le crénage doit être automatiquement appliqué au texte lorsque le contenu de l'acteur texte est modifié.

Lorsque cette propriété présente la valeur `TRUE`, le crénage est automatique. Lorsqu'elle prend la valeur `FALSE`, le crénage n'est pas appliqué.

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante définit la propriété `kerning` de l'acteur Texte sur la valeur `TRUE`.

```
-- Lingo syntax
member("MyText").kerning=TRUE

// JavaScript syntax
member("MyText").kerning=1;
```

Voir aussi

kerningThreshold

kerningThreshold

Syntaxe

```
-- Lingo syntax
memberObjRef.kerningThreshold

// JavaScript syntax
memberObjRef.kerningThreshold;
```

Description

Propriété d'acteur texte ; permet de contrôler la taille à partir de laquelle le crénage est automatiquement appliqué à un acteur texte. Cette propriété n'a d'effet que lorsque la propriété kerning de l'acteur présente la valeur TRUE.

Sa valeur est un entier servant à indiquer la taille en points à partir de laquelle le crénage prend effet.

Cette propriété a une valeur par défaut de 14 points.

Exemple

L'instruction suivante définit la taille de police de la propriété `kerningThreshold` de l'acteur Texte sur la valeur 14.

```
-- Lingo syntax
member("MyText").kerningThreshold=14
```

```
// JavaScript syntax
member("MyText").kerningThreshold=14;
```

Voir aussi

kerning

key

Syntaxe

```
-- Lingo syntax
_key.key

// JavaScript syntax
_key.key;
```

Description

Propriété de touche ; renvoie la valeur de la dernière touche sur laquelle l'utilisateur a appuyé. Lecture seule.

La valeur renvoyée correspond à la valeur ANSI (American National Standards Institute) de la touche et non à sa valeur numérique.

Vous pouvez utiliser la propriété `key` dans les gestionnaires qui exécutent certaines actions lorsque l'utilisateur appuie sur des touches de raccourcis spécifiques ou d'autres formes d'interactivité. Dans le cas d'une utilisation dans un gestionnaire d'événement principal, les actions spécifiées sont les premières à être exécutées.

Remarque : *la valeur de la propriété `key` n'est pas mise à jour si l'utilisateur appuie sur une touche alors que la syntaxe Lingo ou JavaScript se trouve dans une boucle.*

Utilisez l'animation Clavier et Lingo pour tester la correspondance des caractères des différentes touches sur différents claviers.

Exemple

Les instructions suivantes font revenir l'animation au repère du menu principal lorsque l'utilisateur appuie sur la touche q. Puisque la propriété `keyDownScript` est définie sur `on checkKey`, le gestionnaire `on prepareMovie` demande que le gestionnaire `on checkKey` soit exécuté le premier lorsque l'utilisateur appuie sur une touche. Le gestionnaire `on checkKey` vérifie si la touche q a été enfoncée et, le cas échéant, revient au repère du menu principal.

```
-- Lingo syntax
on prepareMovie
    keyDownScript = "checkKey"
end

on checkKey
    if (_key.key = "q") then _movie.go("Main Menu")
    end if
end

// JavaScript syntax
function prepareMovie() {
    keyDownScript = checkKey();
}

function checkKey() {
    if (_key.key == "q") {
         _movie.go("Main Menu");
```

```
    }
}
```

Le gestionnaire `on keyDown` suivant vérifie si la dernière touche enfoncée est la touche z et, le cas échéant, appelle le gestionnaire additionnerLesNombres :

```
-- Lingo syntax
on keyDown
    if (_key.key = "z") then addNumbers
end

// JavaScript syntax
function keyDown() {
    if (_key.key == "z") {
        addNumbers();
    }
}
```

Voir aussi

`commandDown`, `Touche`

keyboardFocusSprite

Syntaxe

```
-- Lingo syntax
_movie.keyboardFocusSprite

// JavaScript syntax
_movie.keyboardFocusSprite;
```

Description

Propriété d'animation ; permet de concentrer les entrées au clavier (sans contrôler le point d'insertion du curseur) sur une image-objet texte spécifique à l'écran. Lecture/écriture.

Cette propriété équivaut à utiliser la touche de tabulation lorsque la propriété `autoTab` de l'acteur est sélectionnée.

L'attribution de la valeur -1 à la propriété `keyboardFocusSprite` redonne le contrôle des saisies clavier au scénario, tandis que la définition de la valeur 0 désactive toute entrée au clavier dans une image-objet modifiable.

Exemple

L'instruction suivante désactive les entrées au clavier dans les images-objets modifiables.

```
-- Lingo syntax
_movie.keyboardFocusSprite=0

// JavaScript syntax
_movie.keyboardFocusSprite=0;
```

Voir aussi

`Animation`

keyCode

Syntaxe

```
-- Lingo syntax
_key.keyCode

// JavaScript syntax
_key.keyCode;
```

Description

Propriété de touche ; renvoie le code numérique de la dernière touche sur laquelle l'utilisateur a appuyé. Lecture seule.

La valeur renvoyée correspond à la valeur numérique de la touche et non à sa valeur ANSI.

Vous pouvez utiliser la propriété `keyCode` pour détecter si l'utilisateur a appuyé sur une touche fléchée ou une touche de fonction, lesquelles ne peuvent pas être spécifiées par la propriété `key`.

Utilisez l'animation Clavier et Lingo pour tester la correspondance des caractères des différentes touches sur différents claviers.

Exemple

Le gestionnaire suivant utilise la fenêtre Messages pour afficher le code approprié chaque fois que l'utilisateur appuie sur une touche :

```
-- Lingo syntax
on enterFrame
    keyDownScript = put(_key.keyCode)
end

// JavaScript syntax
function enterFrame() {
    keyDownScript = put(_key.keyCode);
}
```

L'instruction suivante vérifie si la touche fléchée Haut (dont le code est 126) a été enfoncée et, le cas échéant, passe au repère précédent :

```
-- Lingo syntax
if (_key.keyCode = 126) then
    _movie.goPrevious()
end if

// JavaScript syntax
if (_key.keyCode == 126) {
    _movie.goPrevious();
}
```

Le gestionnaire suivant vérifie si l'une des touches fléchées a été enfoncée et, le cas échéant, répond en conséquence :

```
-- Lingo syntax
on keyDown
    case (_key.keyCode) of
        123: TurnLeft
        126: GoForward
        125: BackUp
        124: TurnRight
    end case
end keyDown
```

```
// JavaScript syntax
function keyDown() {
    switch (_key.keyCode) {
        case 123: TurnLeft();
            break;
        case 126: GoForward();
            break;
        case 125: BackUp();
            break;
        case 124: TurnRight();
            break;
    }
}
```

Voir aussi

Touche, key

keyDownScript

Syntaxe

```
the keyDownScript
```

Description

Propriété système ; spécifie le code Lingo exécuté lorsque l'utilisateur appuie sur une touche. Le code Lingo est rédigé sous forme d'une chaîne entourée de guillemets droits et peut être une instruction simple ou le script d'appel d'un gestionnaire.

Lorsque l'utilisateur appuie sur une touche et que la propriété `keyDownScript` est définie, Lingo commence par exécuter les instructions spécifiées dans la propriété `keyDownScript`. A moins que ces instructions contiennent la commande `pass` autorisant la transmission du message `keyDown` à d'autres objets de l'animation, aucun autre gestionnaire `on keyDown` n'est exécuté.

La propriété `keyDownScript` produit le même résultat que la commande `when keyDown then` utilisée dans les versions précédentes de Director.

Lorsque les instructions spécifiées pour la propriété `keyDownScript` ne sont plus appropriées, désactivez-les à l'aide de l'instruction `set the keyDownScript to EMPTY`.

Exemple

L'instruction suivante spécifie le script exécuté lorsque l'utilisateur appuie sur une touche.

```
-- Lingo syntax
 the keyDownScript = "go to the frame"

// JavaScript syntax
 _system.keyDownScript = "_movie.go(_movie.frame)";
```

Voir aussi

on keyDown, keyUpScript, mouseDownScript, mouseUpScript

keyframePlayer (modificateur)

Syntaxe

```
member(whichCastmember).model(whichModel).keyframePlayer.keyframePlayerModifierProperty
```

Description

Modificateur 3D ; gère l'utilisation des mouvements par les modèles. Les mouvements gérés par le modificateur `keyframePlayer` animent la totalité du modèle, contrairement aux mouvements bonesPlayer qui animent les segments du modèle.

Les mouvements et les modèles qui les utilisent doivent être créés dans un programme de modélisation 3D, exportés au format *.w3d, puis importés dans une animation. Les mouvements ne peuvent pas être appliqués aux primitives de modèle créées dans Director.

L'ajout du modificateur `keyframePlayer` à un modèle à l'aide de la commande `addModifier` permet d'accéder aux propriétés suivantes du modificateur `keyframePlayer` :

- `playing` indique si un modèle exécute un mouvement.
- `playList` est une liste linéaire de listes de propriétés contenant les paramètres de lecture des mouvements d'un modèle placés en file d'attente.
- `currentTime` indique la position locale, en millisecondes, du mouvement en cours de lecture ou en pause.
- `playRate` est un nombre multiplié par le paramètre *`échelle`* de la commande `play()` ou `queue()` pour déterminer la cadence de lecture du mouvement.
- `playlist.count` renvoie le nombre de mouvements en file d'attente dans la liste de lecture.
- `rootLock` indique si le composant de translation du mouvement est utilisé ou ignoré.
- `currentLoopState` indique si le mouvement est lu une seule fois ou continuellement répété.
- `blendTime` indique la durée de la transition créée par le modificateur entre les mouvements lorsque la propriété `autoblend` du modificateur présente la valeur `TRUE`.
- `autoBlend` indique si le modificateur crée une transition linéaire entre le mouvement en cours de lecture et le mouvement précédent.
- `blendFactor` indique le degré de fusion entre les mouvements lorsque la propriété `autoBlend` du modificateur présente la valeur `FALSE`.
- `lockTranslation` indique si le modèle peut être déplacé à partir des plans spécifiés.
- `positionReset` indique si le modèle revient à sa position de départ à la fin d'un mouvement ou de chaque itération d'une boucle.
- `rotationReset` indique l'élément de rotation d'une transition d'un mouvement à un autre ou de la boucle d'un seul mouvement.

Remarque : *pour plus d'informations, consultez les entrées des différentes propriétés.*

Le modificateur `keyframePlayer` utilise les commandes suivantes :

- `pause` stoppe le mouvement du modèle en cours d'exécution.
- `play()` entraîne ou reprend l'exécution d'un mouvement.
- `playNext()` entraîne la lecture du mouvement suivant de la liste de lecture.
- `queue()` ajoute un mouvement à la fin de la liste de lecture.

Le modificateur `keyframePlayer` génère les événements suivants, qui sont utilisés par les gestionnaires déclarés dans les commandes `registerForEvent()` et `registerScript()`. L'appel au gestionnaire déclaré contient trois arguments : le type d'événement (`#animationStarted` ou `#animationEnded`), le nom du mouvement, ainsi que sa position en cours. Pour plus d'informations sur les événements de notification, reportez-vous à l'entrée `registerForEvent()`.

`#animationStarted` est envoyé au début de la lecture d'un mouvement. Si la fusion est utilisée entre des mouvements, l'événement est envoyé au début de la transition.

`#animationEnded` est envoyé à la fin d'un mouvement. Si la fusion est utilisée entre des mouvements, l'événement est envoyé à la fin de la transition.

Voir aussi

`addModifier`, `modifiers`, `bonesPlayer (modificateur)`, `motion`

keyUpScript

Syntaxe

```
the keyUpScript
```

Description

Propriété système ; spécifie le code Lingo exécuté lorsque l'utilisateur relâche une touche. Le code Lingo est rédigé sous forme d'une chaîne entourée de guillemets droits et peut être une instruction simple ou le script d'appel d'un gestionnaire.

Lorsque l'utilisateur relâche une touche et que la propriété `keyUpScript` est définie, Lingo commence par exécuter les instructions spécifiées dans la propriété `keyUpScript`. A moins que ces instructions contiennent la commande `pass` autorisant la transmission du message `keyDown` à d'autres objets de l'animation, aucun autre gestionnaire `on keyUp` n'est exécuté.

Lorsque les instructions spécifiées dans la propriété `keyUpScript` ne sont plus appropriées, désactivez-les à l'aide de l'instruction `set the keyUpScript to empty`.

Exemple

L'instruction suivante spécifie le script exécuté lorsque l'utilisateur appuie sur une touche.

```
-- Lingo syntax
 the keyUpScript = "go to the frame +1 "

// JavaScript syntax
 _system.keyUpScript = "_movie.go(_movie.frame+1)";
```

Voir aussi

`on keyUp`

labelList

Syntaxe

```
the labelList
```

Description

Propriété système ; crée une liste des libellés d'images de l'animation actuelle sous forme de chaîne délimitée par des retours de chariot (et non de liste) contenant un libellé par ligne. Les libellés sont répertoriés en fonction de leur ordre dans le scénario. Les entrées de la liste étant délimitées par des retours chariot, la dernière ligne de la liste est une ligne vide. Assurez-vous de supprimer cette ligne vide si nécessaire.

Exemple

Cette instruction répertorie les libellés d'image de l'animation en cours sous la forme d'une chaîne délimitée par des retours chariot (et non sous la forme d'une liste) contenant un libellé par ligne.

```
put the labelList
```

Voir aussi

`frameLabel`, `label()`, `marker()`

lastChannel

Syntaxe

```
-- Lingo syntax
_movie.lastChannel

// JavaScript syntax
_movie.lastChannel;
```

Description

Propriété d'animation ; affiche le numéro de la dernière piste de l'animation, tel qu'il a été saisi dans la boîte de dialogue Propriétés de l'animation. Lecture seule.

Vous pouvez voir un exemple d'utilisation de `lastChannel` dans une animation en consultant l'animation QT and Flash du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante affiche le numéro de la dernière piste de l'animation dans la fenêtre Messages :

```
-- Lingo syntax
put(_movie.lastChannel)

// JavaScript syntax
put(_movie.lastChannel);
```

Voir aussi

`Animation`

lastClick

Syntaxe

```
-- Lingo syntax
_player.lastClick

// JavaScript syntax
_player.lastClick;
```

Description

Propriété de lecteur ; renvoie le nombre de battements (1 battement = 1/60ème de seconde) écoulés depuis le dernier clic effectué à l'aide de la souris. Lecture seule.

Exemple

L'instruction suivante vérifie si 10 secondes se sont écoulées depuis le dernier clic de la souris et, le cas échéant, fait passer la tête de lecture sur le repère Pas de clic :

```
-- Lingo syntax
if (_player.lastClick > (10 * 60)) then
    _movie.go("No Click")
end if

// JavaScript syntax
if (_player.lastClick > (10 * 60)) {
    _movie.go("No Click");
}
```

Voir aussi

lastEvent, lastKey, lastRoll, Lecteur

lastError

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.lastError

// JavaScript syntax
memberOrSpriteObjRef.lastError;
```

Description

Propriété d'acteur ou d'image-objet RealMedia ; permet d'obtenir le dernier symbole d'erreur renvoyé par RealPlayer® sous la forme d'un symbole Lingo. Les symboles d'erreur renvoyés par RealPlayer sont des chaînes simples, en anglais, qui fournissent le point de départ du processus de dépannage. Cette propriété est dynamique en cours de lecture et peut être testée, mais pas définie.

La valeur `#PNR_OK` indique que tout fonctionne correctement.

Exemple

Les exemples suivants indiquent que la dernière erreur renvoyée par RealPlayer pour l'image-objet 2 et l'acteur Real était `#PNR_OUTOFMEMORY` :

```
-- Lingo syntax
put(sprite(2).lastError) -- #PNR_OUTOFMEMORY
put(member("Real").lastError) -- #PNR_OUTOFMEMORY

// JavaScript syntax
trace(sprite(2).lastError); // #PNR_OUTOFMEMORY
put(member("Real").lastError); // #PNR_OUTOFMEMORY
```

lastEvent

Syntaxe

```
-- Lingo syntax
_player.lastEvent

// JavaScript syntax
_player.lastEvent;
```

Description

Propriété de lecteur ; renvoie le nombre de battements (1 battement = 1/60ème de seconde) écoulés depuis la dernière fois où l'utilisateur a appuyé sur le bouton de la souris, a survolé un élément avec la souris ou a appuyé sur une touche. Lecture seule.

Exemple

L'instruction suivante vérifie si 10 secondes se sont écoulées depuis la dernière fois où l'utilisateur a appuyé sur le bouton de la souris, a survolé un élément avec la souris ou a appuyé sur une touche. Si c'est le cas, la tête de lecture est placée sur le repère Aide :

```
-- Lingo syntax
if (_player.lastEvent > (10 * 60)) then
    _movie.go("Help")
end if

// JavaScript syntax
if (_player.lastEvent > (10 * 60)) {
    _movie.go("Help");
}
```

Voir aussi

lastClick, lastKey, lastRoll, Lecteur

lastFrame

Syntaxe

```
-- Lingo syntax
_movie.lastFrame

// JavaScript syntax
_movie.lastFrame;
```

Description

Propriété d'animation ; affiche le numéro de la dernière image de l'animation. Lecture seule.

Exemple

L'instruction suivante affiche le numéro de la dernière image de l'animation dans la fenêtre Messages :

```
-- Lingo syntax
put(_movie.lastFrame)

// JavaScript syntax
put(_movie.lastFrame);
```

Voir aussi

Animation

lastKey

Syntaxe

```
-- Lingo syntax
_player.lastKey

// JavaScript syntax
_player.lastKey;
```

Description

Propriété de lecteur ; indique le nombre de battements (1 battement = 1/60ème de seconde) écoulés depuis que l'utilisateur a appuyé sur une touche. Lecture seule.

Exemple

L'instruction suivante vérifie si 10 secondes se sont écoulées depuis que l'utilisateur a appuyé sur une touche et, le cas échéant, fait passer la tête de lecture sur le repère Pas de touche :

```
-- Lingo syntax
if (_player.lastKey > (10 * 60)) then
    _movie.go("No Key")
end if

// JavaScript syntax
if (_player.lastKey > (10 * 60)) {
    _movie.go("No Key");
}
```

Voir aussi

lastClick, lastEvent, lastRoll, Lecteur

lastRoll

Syntaxe

```
-- Lingo syntax
_player.lastRoll

// JavaScript syntax
_player.lastRoll;
```

Description

Propriété de lecteur ; indique le nombre de battements (1 battement = 1/60ème de seconde) écoulés depuis que l'utilisateur a déplacé la souris. Lecture seule.

Exemple

L'instruction suivante vérifie si 45 secondes se sont écoulées depuis le dernier déplacement de la souris et, le cas échéant, fait passer la tête de lecture au repère Boucle :

```
-- Lingo syntax
if (_player.lastRoll > (45 * 60)) then
    _movie.go("Attract Loop")
end if

// JavaScript syntax
if (_player.lastRoll > (45 * 60)) {
    _movie.go("Attract Loop");
}
```

Voir aussi

lastClick, lastEvent, lastKey, Lecteur

left

Syntaxe

```
-- Lingo syntax
spriteObjRef.left

// JavaScript syntax
spriteObjRef.left;
```

Description

Propriété d'image-objet ; indique la coordonnée horizontale du bord gauche du rectangle de délimitation d'une image-objet. Lecture/écriture.

Les coordonnées d'images-objets sont exprimées en pixels, (0, 0) correspond au coin supérieur gauche de la scène.

Exemple

L'instruction suivante détermine si le bord gauche de l'image-objet se trouve à gauche du bord gauche de la scène. Le cas échéant, le script exécute le gestionnaire `offLeftEdge` :

```
-- Lingo syntax
if (sprite(3).left < 0) then
    offLeftEdge()
end if

// JavaScript syntax
if (sprite(3).left < 0) {
    offLeftEdge();
}
```

L'instruction suivante mesure la coordonnée horizontale gauche de l'image-objet portant le numéro indiqué (i + 1) et attribue cette valeur à la variable `vLowest` :

```
-- Lingo syntax
vLowest = sprite(i + 1).left
```

```
// JavaScript syntax
var vLowest = sprite(i + 1).left
```

Voir aussi

bottom, height, locH, locV, right, Image-objet, top, width

left (3D)

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).left
```

Description

Propriété 3D de ressource de modèle `#box` ; indique si le côté de la boîte coupé par son axe des x négatif est fermé (`TRUE`) ou ouvert (`FALSE`).

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante attribue à la propriété `left` de la ressource de modèle Caisse la valeur `FALSE`, ce qui signifie que le côté gauche de la caisse est ouvert :

```
member("3D World").modelResource("crate").left = FALSE
```

Voir aussi

back, front, bottom (3D), top (3D), right (3D)

leftIndent

Syntaxe

```
chunkExpression.leftIndent
```

Description

Propriété d'acteur texte ; contient le décalage (exprimé en pixels) entre la marge gauche de la sous-chaîne spécifiée par *`expressionSousChaîne`* et le côté gauche de l'acteur texte.

La valeur est un nombre entier supérieur ou égal à 0.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante définit un retrait à gauche de 10 pixels pour l'acteur texte monTexte.

```
-- Lingo syntax
member("myText").leftIndent=10

// JavaScript syntax
member("myText").leftIndent=10;
```

Voir aussi

firstIndent, rightIndent

length (3D)

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).length
vectorReference.length
```

Description

Propriété 3D de ressource de modèle `#box` et `#plane` et de vecteur ; indique la longueur, en unités d'univers, de la boîte ou du plan.

La longueur d'une boîte est mesurée sur son axe des z. La longueur par défaut de la boîte est 50.

La longueur d'un plan est mesurée sur son axe des y. La longueur par défaut du plan est 1.

La longueur d'un vecteur définit, en unités d'univers, la distance qui sépare ce vecteur de `vector(0, 0, 0)`. Il s'agit de la magnitude du vecteur.

Exemple

L'instruction suivante attribue à la variable `myBoxLenght` la longueur de la ressource de modèle boîteACadeau.

```
myBoxLength = member("3D World").modelResource("GiftBox").length
```

Voir aussi

`height (3D)`, `width (3D)`, `magnitude`

lengthVertices

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).lengthVertices
```

Description

Proprieté 3D de ressource de modèle `#box` et `#plane` ; indique le nombre de sommets de la maille le long de la longueur de la boîte ou du plan. L'augmentation de cette valeur augmente le nombre de faces et donc la précision de la maille.

La longueur d'une boîte est mesurée sur son axe des z. La longueur d'un plan est mesurée le long de son axe des y.

Définissez la propriété `renderStyle` du matériau d'un modèle sur la valeur `#wire` pour afficher toutes les faces de la maille de la ressource du modèle. Définissez la propriété `renderStyle` sur la valeur `#point` pour n'afficher que les sommets de la maille.

La valeur de cette propriété doit être supérieure ou égale à 2. La valeur par défaut est 4.

Exemple

L'instruction suivante attribue à la propriété `lengthVertices` de la ressource de modèle Tour la valeur 10. Neuf triangles sont utilisés pour définir la géométrie de la ressource de modèle le long de son axe des y ; il y a donc dix sommets.

```
member("3D World").modelResource("Tower").lengthVertices = 10
```

Voir aussi

`length (3D)`

level

Syntaxe

```
member(whichCastmember).model(whichModel).lod.level
```

Description

Propriété 3D de modificateur `lod` ; indique la quantité de détails supprimés par le modificateur lorsque sa propriété `auto` présente la valeur `FALSE`. La plage de valeurs de cette propriété est comprise entre 0,0 et 100,00.

Lorsque la propriété `auto` du modificateur présente la valeur `TRUE`, la valeur de la propriété `level` est mise à jour de façon dynamique, mais n'est pas définissable.

Le modificateur `#lod` ne peut être ajouté qu'aux modèles créés dans un programme de modélisation 3D autre que Director. La valeur de la propriété `type` des ressources de modèle utilisées par ces modèles est `#fromFile`. Le modificateur ne peut pas être ajouté aux primitives créées dans Director.

Exemple

L'instruction suivante définit la propriété `level` du modificateur lod du modèle vaisseauSpatial sur 50. Si la propriété `auto` de ce modificateur est définie sur `FALSE`, vaisseauSpatial est tracé avec un niveau de détail moyen. Si la propriété `auto` du modificateur lod est définie sur `TRUE`, ce code n'a aucun effet.

```
member("3D World").model("Spaceship").lod.level = 50
```

Voir aussi

`lod (modificateur)`, `auto`, `bias`

lifetime

Syntaxe

```
member(whichCastmember).modelResource(modelResource).lifetime
```

Description

Propriété 3D de ressource de modèle `#particle` ; pour toutes les particules d'un système de particules, cette propriété indique le nombre de millisecondes entre la création d'une particule et la fin de son existence.

La valeur par défaut de cette propriété est 10 000.

Exemple

L'exemple suivant définit la propriété `lifetime` d'une particule sur 100 millisecondes dans l'acteur objets3D.

```
-- Lingo syntax
member("3Dobjects").modelResource("Particle01").lifetime = 100.0

// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",10).lifetime = 100.0;
```

Voir aussi

`emitter`

light

Syntaxe

```
member(whichCastmember).light(whichLight)
member(whichCastmember).light[index]
member(whichCastmember).light(whichLight).whichLightProperty
member(whichCastmember).light[index].whichLightProperty
```

Description

Elément 3D ; objet à une position de vecteur à partir de laquelle la lumière émane.

Exemple

L'exemple suivant affiche la première lumière de l'acteur objets3D.

```
-- Lingo syntax
putmember("3Dobjects").light(1)

// JavaScript syntax
put( member("3Dobjects").getPropRef("light",1));
```

Voir aussi

`newLight`, `deleteLight`

lineColor

Syntaxe

```
member(whichCastmember).model(whichModel).inker.lineColor
member(whichCastmember).model(whichModel).toon.lineColor
```

Description

Propriété 3D de modificateur `toon` et `inker` ; indique la couleur des lignes tracées sur le modèle par le modificateur. Pour que cette propriété prenne effet, la propriété `creases`, `silhouettes` ou `boundary` du modificateur doit présenter la valeur `TRUE`.

La valeur par défaut de cette propriété est `rgb(0, 0, 0)`.

Exemple

L'instruction suivante attribue à la couleur de toutes les lignes tracées par le modificateur `toon` sur le second modèle la valeur rgb(255, 0, 0), ce qui correspond à la couleur rouge :

```
-- Lingo syntax
member("3Dobjects").model[2].toon.lineColor = rgb(255, 0, 0)

// JavaScript syntax
member("3Dobjects").getPropRef("model",2).toon.lineColor =color(100,0,0);
```

Voir aussi

`creases`, `silhouettes`, `boundary`, `lineOffset`

lineCount

Syntaxe

```
-- Lingo syntax
memberObjRef.lineCount

// JavaScript syntax
memberObjRef.lineCount;
```

Description

Propriété d'acteur ; indique le nombre de lignes qui apparaissent dans l'acteur champ sur la scène en fonction des retours à la ligne automatiques et non du nombre de retours de chariot que contient la chaîne.

Exemple

L'instruction suivante détermine le nombre de lignes de l'acteur champ Nouvelles du jour lorsque ce dernier apparaît sur la scène et affecte cette valeur à la variable `numberOfLines` :

```
--Lingo syntax
numberOfLines = member("Today's News").lineCount

// JavaScript syntax
var numberOfLines = member("Today's News").lineCount;
```

lineDirection

Syntaxe

```
member(whichCastMember).lineDirection
```

Description

Propriété d'acteur forme ; utilise 0 ou 1 pour indiquer l'inclinaison de la ligne dessinée.

Si la ligne s'élève de gauche à droite, la valeur de cette propriété est 1. Si la ligne s'abaisse de gauche à droite, sa valeur est 0.

Cette propriété peut être testée et définie.

Exemple

L'exemple suivant inverse l'inclinaison de la ligne de l'acteur laLigne, produisant un effet de balancier :

```
-- Lingo syntax
member("theLine").lineDirection = not member("theLine").lineDirection

// JavaScript syntax
member("theLine").lineDirection = ! (member("theLine").lineDirection);
```

lineHeight

Syntaxe

```
member(whichCastMember).lineHeight
the lineHeight of member whichCastMember
```

Description

Propriété d'acteur ; détermine l'interligne utilisé pour afficher l'acteur champ spécifié. Le paramètre *quelActeur* peut être un nom ou un numéro d'acteur.

La définition de la propriété d'acteur `lineHeight` remplace temporairement le paramétrage du système jusqu'à la fermeture de l'animation. Pour utiliser l'interligne souhaité dans la totalité d'une animation, définissez la propriété `lineHeight` dans un gestionnaire `on prepareMovie`.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante définit la hauteur de ligne de l'acteur monTexte.

```
-- Lingo syntax
member("myText").lineHeight=20

// JavaScript syntax
member("myText").lineHeight=20;
```

Voir aussi

`text`, `alignment`, `font`, `fontSize`, `fontStyle`

lineOffset

Syntaxe

```
member(whichCastmember).model(whichModel).toon.lineOffset
member(whichCastmember).model(whichModel).inker.lineOffset
```

Description

Propriété 3D de modificateur `toon` et `inker` ; indique la distance apparente à laquelle les lignes sont tracées par le modificateur à partir de la surface du modèle. Pour que cette propriété prenne effet, la propriété `useLineOffset` du modificateur doit être définie sur `TRUE` et l'une au moins de ses propriétés `creases`, `silhouettes` ou `boundary` doit également présenter la valeur `TRUE`.

La plage de cette propriété s'étend de -100,00 à +100,00. Son paramètre par défaut est -2,0.

Exemple

L'instruction suivante attribue à la propriété `lineOffset` du modificateur toon du modèle Théière la valeur 10. Les lignes tracées par le modificateur toon à la surface du modèle sont plus apparentes qu'avec la valeur par défaut de -2.

```
member("shapes").model("Teapot").toon.lineOffset = 10
```

Voir aussi

`creases`, `silhouettes`, `boundary`, `useLineOffset`, `lineColor`

lineSize

Syntaxe

```
member(whichCastMember).lineSize
the lineSize of member whichCastMember
```

```
sprite whichSprite.lineSize
the lineSize of sprite whichSprite
```

Description

Propriété d'acteur forme ; détermine l'épaisseur, en pixels, de la bordure de l'acteur forme spécifié sur la scène. Pour les formes non rectangulaires, la bordure correspond au bord de la forme même et non à son rectangle de délimitation.

La propriété `lineSize` de l'image-objet prévaut sur la valeur `lineSize` de l'acteur. Si Lingo modifie la valeur de la propriété `lineSize` de l'acteur alors qu'une image-objet se trouve sur la scène, la valeur de la propriété `lineSize` de cette dernière reste inchangée tant que l'image-objet est présente.

Pour que la valeur définie par Lingo dure au-delà de l'image-objet en cours, l'image-objet doit être contrôlée par un script.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante attribue à l'épaisseur de l'acteur forme maForme une valeur de 5 pixels :

```
--Lingo syntax
member("myShape").lineSize=5

// JavaScript syntax
member("myShape").lineSize=5;
```

linked

Syntaxe

```
-- Lingo syntax
memberObjRef.linked

// JavaScript syntax
memberObjRef.linked;
```

Description

Propriété d'acteur ; contrôle si un script, une animation Flash ou un fichier GIF animé est stocké dans un fichier externe (`TRUE`, valeur par défaut) ou dans la bibliothèque de distribution Director (`FALSE`). Lecture/écriture pour les acteurs script, Flash et GIF animé ; lecture seule pour tous les autres types d'acteurs.

Lorsque les données sont stockées de façon externe dans un fichier lié, la propriété `pathName` de l'acteur doit indiquer l'emplacement du fichier de l'animation.

Exemple

L'instruction suivante convertit l'acteur Flash amis provenant d'un acteur lié en un acteur stocké de façon interne.

```
-- Lingo syntax
member("homeBodies").linked = 0

// JavaScript syntax
member("homeBodies").linked = 0;
```

Voir aussi

Acteur

loaded

Syntaxe

```
-- Lingo syntax
memberObjRef.loaded

// JavaScript syntax
memberObjRef.loaded;
```

Description

Propriété d'acteur ; indique si un acteur spécifié est chargé en mémoire (TRUE) ou non (FALSE). Lecture seule.

Les types d'acteurs existants se comportent de manière légèrement différente lors de leur chargement :

- Les acteurs forme et script sont toujours chargés en mémoire.
- Les acteurs animation ne sont jamais purgés de la mémoire.
- Les acteurs vidéo numérique peuvent être préchargés en mémoire et purgés de la mémoire indépendamment de leur utilisation. La lecture d'un acteur vidéo numérique chargé en mémoire est plus rapide que lorsqu'il est chargé à partir du disque dur.

Exemple

L'instruction suivante vérifie si l'acteur Démonstration est chargé en mémoire et passe à une autre animation si ce n'est pas le cas :

```
-- Lingo syntax
if member("Demo Movie").loaded = FALSE then
    _movie.go(1, "Waiting.dir")
end if

// JavaScript syntax
if (member("Demo Movie").loaded == false) {
    _movie.go(1, "Waiting.dir")
}
```

Voir aussi

Acteur

loc (fond et recouvrement)

Syntaxe

```
sprite(whichSprite).camera{(index)}.backdrop[index].loc
member(whichCastmember).camera(whichCamera).backdrop[index].loc
sprite(whichSprite).camera{(index)}.overlay[index].loc
member(whichCastmember).camera(whichCamera).overlay[index].loc
```

Description

Propriété 3D de fond et de recouvrement ; indique l'emplacement 2D du fond ou recouvrement, mesuré à partir du coin supérieur gauche de l'image-objet.

Cette propriété est initialement définie comme paramètre de la commande addBackdrop, addOverlay, insertBackdrop ou insertOverlay qui crée le fond ou le recouvrement.

Exemple

L'instruction suivante positionne le premier fond de la caméra de l'image-objet 2.

```
sprite(2).camera.backdrop[1].loc = point(120, 120)
```

Voir aussi

bevelDepth, overlay, regPoint (3D)

locH

Syntaxe

```
-- Lingo syntax
spriteObjRef.locH

// JavaScript syntax
spriteObjRef.locH;
```

Description

Propriété d'image-objet ; indique la position horizontale du point d'alignement d'une image-objet. Lecture/écriture.

Les coordonnées des images-objets sont calculées par rapport au coin supérieur gauche de la scène.

Pour que la valeur dure au-delà de l'image-objet en cours, cette image-objet doit être contrôlée par un script.

Exemple

L'instruction suivante place l'image-objet 15 à l'emplacement horizontal auquel l'utilisateur a cliqué :

```
-- Lingo syntax
sprite(15).locH = _mouse.mouseH

// JavaScript syntax
sprite(15).locH = _mouse.mouseH;
```

Voir aussi

bottom, height, left, locV, point(), right, Image-objet, top, updateStage()

lockTranslation

Syntaxe

```
member(whichCastmember).model(whichModel).bonesPlayer.lockTranslation
member(whichCastmember).model(whichModel).keyframePlayer.lockTranslation
```

Description

Propriété 3D de modificateur `#bonesPlayer` et `#keyframePlayer` ; empêche le déplacement du ou des plans spécifiés, sauf dans le cas d'une translation absolue des données de mouvement. Toute translation supplémentaire ajoutée manuellement ou suite à une accumulation d'erreurs est supprimée. Les valeurs possibles de `#none`, `#x`, `#y`, `#z`, `#xy`, `#yz`, `#xz` et `#all` contrôlent, parmi les trois composants de translation, ceux qui sont contrôlés pour chaque image. Lorsqu'un verrou est activé sur un axe, le déplacement actuel le long de cet axe est enregistré et utilisé ensuite comme déplacement fixé pour l'animation. Ce déplacement peut être réinitialisé en désactivant ce verrou d'axe, en déplaçant l'objet, puis en réactivant le verrou.

En d'autres termes, il définit l'axe de translation à ignorer à la lecture d'un mouvement. Pour conserver un modèle verrouillé sur un plan horizontal, avec le sommet pointant le long de l'axe des *z*, attribuez à `lockTranslation` la valeur `#z`. La valeur par défaut de cette propriété est `#none`.

Exemple

L'instruction suivante attribue à la propriété `lockTranslation` du modèle Marcheur la valeur `#z`.

```
member("ParkScene").model("Walker").bonesPlayer.lockTranslation = #z
```

Voir aussi

immovable

locV

Syntaxe

```
-- Lingo syntax
spriteObjRef.locV

// JavaScript syntax
spriteObjRef.locV;
```

Description

Propriété d'image-objet ; indique la position verticale du point d'alignement d'une image-objet. Lecture/écriture.

Les coordonnées des images-objets sont calculées par rapport au coin supérieur gauche de la scène.

Pour que la valeur dure au-delà de l'image-objet en cours, cette image-objet doit être contrôlée par un script.

Exemple

L'instruction suivante place l'image-objet 15 à la position verticale où l'utilisateur a cliqué :

```
-- Lingo syntax
sprite(15).locV = _mouse.mouseV

// JavaScript syntax
sprite(15).locV = _mouse.mouseV;
```

Voir aussi

bottom, height, left, locH, point(), right, Image-objet, top, updateStage()

locZ

Syntaxe

```
-- Lingo syntax
spriteObjRef.locZ

// JavaScript syntax
spriteObjRef.locZ;
```

Description

Propriété d'image-objet ; spécifie l'ordre z dynamique d'une image-objet, permettant de contrôler les différentes couches d'images-objets sans avoir à manipuler les pistes ou les propriétés de ces images-objets. Lecture/écriture.

Cette propriété peut avoir pour valeur un nombre entier allant de -2 milliards à +2 milliards. Les nombres élevés font apparaître l'image-objet devant les images-objets dont la valeur est inférieure. Si deux images-objets présentent la même valeur `locZ`, le numéro de piste détermine l'ordre d'affichage final de ces deux images-objets. Cela signifie que les images-objets des pistes portant les numéros les plus faibles apparaissent derrière les images-objets dont le numéro de piste est plus élevé, même lorsque leurs valeurs `locZ` sont identiques.

Par défaut, la valeur `locZ` des images-objets est égale à leur numéro de piste.

Les opérations dépendantes des couches, telles que la détection d'un clic ou les événements de souris, sont régies par la valeur `locZ` des images-objets ; par conséquent, la modification de la valeur `locZ` d'une image-objet risque de rendre cette dernière partiellement ou totalement masquée par d'autres images-objets, empêchant ainsi l'utilisateur de cliquer dessus.

D'autres fonctions de Director ne suivent pas l'ordre des images-objets défini par la valeur `locZ`. Les événements générés démarrent toujours dans la piste 1 et passent par les pistes suivantes, quel que soit l'ordre Z des images-objets.

Exemple

Le gestionnaire suivant utilise une variable globale appelée `gHighestSprite` qui a été initialisée dans le gestionnaire startMovie en fonction du nombre d'images-objets utilisées. Lorsque vous cliquez sur l'image-objet, sa valeur locZ est définie sur gImageObjetPlusElevée +1, qui déplace l'image-objet au premier plan de la scène. La valeur gImageObjetPlusElevée est ensuite incrémentée de 1 pour préparer au prochain appel de mouseUp.

```
-- Lingo syntax
on mouseUp me
    global gHighestSprite
    sprite(me.spriteNum).locZ = gHighestSprite + 1
    gHighestSprite = gHighestSprite + 1
end

// JavaScript syntax
function mouseUp() {
    _global.gHighestSprite;
    sprite(this.spriteNum).locZ = _global.gHighestSprite + 1
    _global.gHighestSprite = _global.gHighestSprite + 1
}
```

Voir aussi

locH, locV, Image-objet

lod (modificateur)

Syntaxe

```
member(whichCastmember).model(whichModel).lod.lodModifierProperty
```

Description

Modificateur 3D ; supprime de façon dynamique les détails des modèles, au fur et à mesure que ces derniers s'éloignent de la caméra.

Ce modificateur ne peut être ajouté qu'aux modèles créés dans un programme de modélisation 3D autre que

Director. La valeur de la propriété `type` des ressources de modèle utilisées par ces modèles est `#fromFile`. De tels modèles utilisent tous la réduction des détails, que le modificateur `lod` y soit ou non associé. L'association du modificateur vous permet de contrôler les propriétés de réduction des détails. Le modificateur ne peut pas être ajouté aux primitives créées dans Director.

Les données du modificateur `lod` sont générées par les programmes de modélisation 3D pour tous les modèles. La définition de la propriété de `userData` sur `"sw3d_no_lod = true"` vous permet de demander que les données du modificateur `lod` et la mémoire soit libérées à l'issue de la lecture en flux continu.

Faites attention lorsque vous utilisez les modificateurs `sds` et `lod` conjointement, car ils exécutent des opérations opposées (le modificateur `sds` ajoute des détails géométriques alors que le modificateur `lod` les supprime). Avant d'ajouter le modificateur `sds`, Il est recommandé de désactiver la propriété `lod.auto` et de définir la propriété `lod.level` sur la résolution maximale, comme suit :

```
member("myMember").model("myModel").lod.auto = 0
member("myMember").model("myModel").lod.level = 100
member("myMember").model("myModel").addmodifier(#sds)
```

Le modificateur `lod` comporte les propriétés suivantes :

- `auto` permet au modificateur de définir le niveau de réduction des détails à mesure que la distance entre le modèle et la caméra change. La valeur de la propriété `level` du modificateur est mise à jour, mais la définition de la propriété `level` n'a aucun effet lorsque la propriété `auto` présente la valeur `TRUE`.
- `bias` indique la mesure dans laquelle le modificateur supprime les détails du modèle lorsque la propriété `auto` de ce modificateur présente la valeur `TRUE`. La plage de cette propriété s'étend de 0,0 (qui supprime tous les polygones) à 100,0 (qui ne supprime aucun polygone). Le paramètre par défaut de cette propriété est 100,0.
- `level` indique le niveau de réduction des détails lorsque la propriété `auto` du modificateur présente la valeur `FALSE`. La plage de cette propriété s'étend de 0,0 à 100,00.

Remarque : *pour plus d'informations, consultez les entrées des différentes propriétés.*

Exemple

L'exemple suivant supprime de façon dynamique le modèle Sphère01 de l'acteur objets3D.

```
--Lingo
member("3Dobjects").model("Sphere01").lod.auto=1
member("3Dobjects").model("Sphere01").lod.bias=0

// Javascript
member("3Dobjects").getPropRef("model",2).lod.auto=1;
member("3Dobjects").getPropRef("model",2).lod.bias=0;
```

Voir aussi

`sds (modificateur)`, `auto`, `bias`, `level`, `addModifier`

loop (3D)

Syntaxe

```
member(whichCastmember).loop
```

Description

Propriété 3D d'acteur ; indique si les mouvements appliqués au premier modèle de l'acteur se répètent continuellement (`TRUE`) ou s'ils ne sont lus qu'une seule fois (`FALSE`).

La valeur par défaut de cette propriété est TRUE.

Exemple

L'instruction suivante attribue à la propriété loop de l'acteur Marcheurs la valeur TRUE. Les mouvements exécutés par le premier modèle de Marcheurs sont lus de façon répétée.

```
member("Walkers").loop = TRUE
```

Voir aussi

motion, play() (3D), queue() (3D), animationEnabled

loop (émetteur)

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).emitter.loop
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type #particle, cette propriété permet d'obtenir et de définir ce qu'il advient des particules à la fin de leur vie. La valeur de boucle TRUE entraîne la « résurrection » des particules à l'emplacement défini par la propriété region de l'émetteur. La valeur FALSE entraîne la mort des particules à la fin de la durée prévue. La valeur par défaut de cette propriété est TRUE.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type #particle. L'instruction suivante attribue à la propriété emitter.loop de systèmeThermique la valeur 1, ce qui entraîne l'émission continue des particules de systèmeThermique.

```
member("Fires").modelResource("ThermoSystem").emitter.loop = 1
```

Voir aussi

emitter

loop (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.loop

// JavaScript syntax
memberObjRef.loop;
```

Description

Propriété d'acteur ; détermine si l'acteur vidéo numérique, audio ou animation Flash spécifié doit être exécuté en boucle (TRUE) ou non (FALSE).

Exemple

L'instruction suivante fait boucler l'acteur séquence QuickTime Démo :

```
-- Lingo syntax
member("Demo").loop = 1
```

```
// JavaScript syntax
member("Demo").loop = 1;
```

loop (Flash)

Syntaxe

```
sprite(whichFlashSprite).loop
the loop of sprite whichFlashSprite
member (whichFlashMember).loop
the loop of member whichFlashMember
```

Description

Propriété d'image-objet et d'acteur Flash ; contrôle si une animation Flash est lue en boucle continue (`TRUE`) ou n'est lue qu'une fois avant de s'arrêter (`FALSE`).

Cette propriété peut être testée et définie.

Exemple

Le script d'image suivant vérifie la progression de la lecture en continu d'un acteur Flash lié appelé objFlash à l'aide de la propriété `percentStreamed`. Pendant le téléchargement de l'acteur objFlash, l'animation lit en boucle l'image en cours. A l'issue du téléchargement de l'acteur objFlash, l'animation passe à l'image suivante et la propriété `loop` de l'animation Flash de la piste 1 reçoit la valeur `FALSE` pour permettre la lecture de l'animation jusqu'à la fin avant son arrêt (l'image-objet est lue en boucle pendant le téléchargement d'objFlash).

```
-- Lingo syntax
on exitFrame me
if member("flashObj").percentStreamed = 100 then
    member(1).loop = FALSE
    go the frame + 1
else
    go to the frame
end if
end

// JavaScript syntax
function exitFrame(me) {
if(member("flashObj").percentStreamed == 100)
    {
        member(1).loop = 0;
        _movie.go(_movie.frame+1);
    }
    else
    {
        _movie.go(_movie.frame);
    }
}
```

loop (Windows Media)

Syntaxe

```
-- Lingo syntax
windowsMediaObjRef.loop
```

```
// JavaScript syntax
windowsMediaObjRef.loop;
```

Description

Propriété Windows Media. Détermine si une animation est lue en boucle (`TRUE`, valeur par défaut) ou non (`FALSE`) une fois qu'elle est terminée. Lecture/écriture.

Exemple

L'instruction suivante indique que l'acteur Classique doit être lu en boucle une fois terminé :

```
-- Lingo syntax
member("Classical").loop = TRUE

// JavaScript syntax
member("Classical").loop = true;
```

Voir aussi

Windows Media

loopBounds

Syntaxe

```
-- Lingo syntax
spriteObjRef.loopBounds

// JavaScript syntax
spriteObjRef.loopBounds;
```

Description

Propriété d'image-objet QuickTime ; définit les points de boucle internes d'un acteur ou d'une image-objet QuickTime. Les points de la boucle sont spécifiés sous forme de liste Director : [*positionDeDépart*, *positionDeFin*].

Les paramètres *positionDeDépart* et *positionDeFin* doivent répondre aux exigences suivantes :

- Ils doivent tous deux être des nombres entiers spécifiant le temps en nombre de battements Director.
- Leurs valeurs doivent être comprises entre 0 et la durée de l'acteur QuickTime.
- La position de départ doit être inférieure à la position de fin.

Si les exigences ci-dessus ne sont pas satisfaites, l'animation QuickTime est lue en boucle pendant sa durée complète.

La propriété `loopBounds` n'a aucun effet lorsque la propriété `loop` de l'animation présente la valeur `FALSE`. Si la propriété `loop` est définie sur `TRUE` pendant la lecture de l'animation, la lecture se poursuit. Director utilise les règles suivantes pour décider de la lecture en boucle de l'animation :

- Lorsque la position de fin spécifiée par `loopBounds` est atteinte, l'animation est relue en boucle à partir de la position de départ.
- Lorsque la fin de l'animation est atteinte, l'animation est relue en boucle à partir de son début.

Si la propriété `loop` est désactivée pendant la lecture de l'animation, la lecture se poursuit. Director arrête la lecture lorsqu'il atteint la fin de l'animation.

Cette propriété peut être testée et définie. La valeur par défaut est [0, 0].

Exemple

Le script d'image-objet suivant définit les positions de début et de fin de la boucle à l'intérieur d'une image-objet QuickTime. Ces positions sont exprimées en secondes, puis converties en battements (en étant multipliées par 60).

```
-- Lingo syntax
on beginSprite me
    sprite(me.spriteNum).loopBounds = [(16 * 60),(32 * 60)]
end

// JavaScript syntax
function beginSprite() {
    sprite(me.spriteNum).loopBounds = list((16 * 60),(32 * 60));
}
```

loopCount

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.loopCount

// JavaScript syntax
soundChannelObjRef.loopCount;
```

Description

Propriété de piste audio ; définit le nombre total d'occurrences de lecture en boucle du son en cours dans une piste audio. Lecture seule.

La valeur par défaut de cette propriété est de 1 pour les sons simplement placés en file d'attente sans boucle interne.

Vous pouvez définir la lecture en boucle d'un son en transmettant les paramètres `loopStartTime`, `loopEndTime` et `loopCount` avec une méthode `queue()` ou `setPlayList()`. Il s'agit là des seules méthodes permettant de définir cette propriété.

Si la valeur `loopCount` est définie sur 0, la boucle se répète à l'infini. Si la propriété `loop` de l'acteur son est définie sur `TRUE`, la propriété `loopCount` reprend la valeur 0.

Exemple

Le gestionnaire suivant place en file d'attente et lit deux sons dans la piste audio 2. Le premier son, l'acteur intro, est exécuté cinq fois sur une durée comprise entre 8 et 8,9 secondes. Le second son, l'acteur Crédits, est exécuté trois fois en boucle. Toutefois, aucune valeur `#loopStartTime` ni `#loopEndTime` n'étant spécifiée, ces valeurs sont respectivement définies par défaut sur `#startTime` et `#endTime`.

```
-- Lingo syntax
on playMusic
    sound(2).queue([#member:member("introMusic"), #startTime:3000, #loopCount:5,
#loopStartTime:8000, #loopEndTime:8900])
    sound(2).queue([#member:member("creditsMusic"), #startTime:3000, #endTime:3000,
#loopCount:3])
    sound(2).play()
end playMusic

// JavaScript syntax
function playMusic() {
```

```
    sound(2).queue(propList("member",member("introMusic"), "startTime",3000,"loopCount",5,
"loopStartTime",8000, "loopEndTime",8900));
    sound(2).queue(propList("member",member("creditsMusic"),
"startTime",3000,"endTime",3000, "loopCount",3]);
    sound(2).play();
}
```

Voir aussi

loopEndTime, loopStartTime, queue(), setPlayList(), Piste audio

loopEndTime

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.loopEndTime

// JavaScript syntax
soundChannelObjRef.loopEndTime;
```

Description

Propriété de piste audio ; indique la position temporelle de fin, en millisecondes, de la boucle définie pour le son en cours de lecture dans une piste audio. Lecture seule.

La valeur de cette propriété est un nombre à virgule flottante, permettant de mesurer et de contrôler la lecture du son en fractions de millisecondes.

Cette propriété n'est définissable que si elle est transmise comme propriété dans une commande `queue()` ou `setPlaylist()`.

Exemple

Le gestionnaire suivant lit l'acteur son Intro sur la piste audio 2. La lecture s'exécute cinq fois en boucle entre le point 8 secondes et le point 8,9 secondes du son.

```
-- Lingo syntax
on playMusic
    sound(2).play([#member:member("introMusic"), #startTime:3000, #loopCount:5 \
#loopStartTime:8000, #loopEndTime:8900])
end playMusic

// JavaScript syntax
function playMusic() {
    sound(2).play(propList("member",member("introMusic"), "startTime",3000,"loopCount",5,
"loopStartTime",8000, "loopEndTime",8900));
}
```

Voir aussi

queue(), setPlayList(), Piste audio

loopsRemaining

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.loopsRemaining

// JavaScript syntax
soundChannelObjRef.loopsRemaining;
```

Description

Propriété de piste audio ; indique le nombre d'occurrences restantes d'une lecture en boucle du son en cours dans une piste audio. Lecture seule.

Si aucune lecture en boucle n'a été définie pour le son lors de son placement en file d'attente, cette propriété présente la valeur 0. Si cette propriété est testée immédiatement après le début de la lecture d'un son, elle renvoie un nombre décrémenté d'une unité par rapport au nombre de boucles défini à l'aide de la propriété `#loopCount` dans les méthodes `queue()` ou `setPlayList()`.

Exceptions

L'exemple suivant affiche le nombre de boucles restantes dans la piste audio 2.

```
-- Lingo syntax
put sound(2).loopsRemaining

// JavaScript syntax
put(sound(2).loopsRemaining);
```

Voir aussi

`loopCount`, `queue()`, `setPlayList()`, `Piste audio`

loopStartTime

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.loopStartTime

// JavaScript syntax
soundChannelObjRef.loopStartTime;
```

Description

Propriété de piste audio ; indique la position temporelle de début, en millisecondes, de la boucle définie pour le son en cours de lecture dans une piste audio. Lecture seule.

Il s'agit d'une valeur à virgule flottante, permettant de mesurer et de contrôler la lecture du son en fraction de millisecondes. La valeur par défaut est la position de départ `startTime` du son si aucune boucle n'a été définie.

Cette propriété n'est définissable que si elle est transmise comme propriété dans une méthode `queue()` ou `setPlaylist()`.

Exemple

Le gestionnaire suivant lit l'acteur son Intro sur la piste audio 2. La lecture est exécutée cinq fois en boucle entre le point 8 secondes et le point 8,9 secondes du son.

```
-- Lingo syntax
on playMusic
    sound(2).play([#member:member("introMusic"), #startTime:3000, #loopCount:5
\#loopStartTime:8000, #loopEndTime:8900])
end playMusic

// JavaScript syntax
function playMusic() {
    sound(2).play(propList("member",member("introMusic"), "startTime",3000,"loopCount",5,
"loopStartTime",8000, "loopEndTime",8900));
}
```

Voir aussi

queue(), setPlayList(), Piste audio, startTime

magnitude

Syntaxe

```
whichVector.magnitude
```

Description

Propriété 3D ; renvoie la magnitude d'un vecteur. La valeur est un nombre à virgule flottante. La magnitude correspond à la longueur d'un vecteur et est toujours supérieure ou égale à 0,0 (`vector (0, 0, 0)` est égal à 0.)

Exemple

L'instruction suivante indique que la magnitude de MonVecteur1 est de 100.

```
-- Lingo syntax
MyVec1 = vector(100, 0, 0)
put MyVec1.magnitude

// JavaScript syntax
MyVec1 = vector(100, 0, 0);
put(MyVec1.magnitude);
```

Voir aussi

length (3D), identity()

margin

Syntaxe

```
-- Lingo syntax
memberObjRef.margin

// JavaScript syntax
memberObjRef.margin;
```

Description

Propriété d'acteur champ : détermine la taille, en pixels, de la marge à l'intérieur de la case du champ.

Exemple

L'instruction suivante définit la marge de la case de l'acteur champ Nouvelles du Jour sur 15 pixels :

```
--Lingo syntax
member("Today's News").margin = 15

// JavaScript syntax
member("Today's News").margin = 15;
```

markerList

Syntaxe

```
-- Lingo syntax
_movie.markerList

// JavaScript syntax
_movie.markerList;
```

Description

Propriété d'animation ; contient une liste de propriétés de script des repères du scénario. Lecture seule.

La liste est au format :

```
frameNumber: "markerName"
```

Exemple

L'instruction suivante affiche la liste des repères dans la fenêtre Messages :

```
-- Lingo syntax
put(_movie.markerList)

// JavaScript syntax
put(_movie.markerList);
```

Voir aussi

Animation

mask

Syntaxe

```
-- Lingo syntax
memberObjRef.mask

// JavaScript syntax
memberObjRef.mask;
```

Description

Propriété d'acteur ; spécifie l'acteur noir et blanc (1 bit) qui servira à masquer des médias rendus au premier plan avec les médias apparaissant dans des zones dans lesquelles les pixels du masque sont noirs. La propriété `mask` vous permet de bénéficier des performances d'une vidéo numérique Premier plan pendant la lecture d'une séquence QuickTime dans une zone non rectangulaire. La propriété `mask` n'a aucun effet sur les acteurs non rendus au premier plan.

Director aligne toujours le point d'alignement de l'acteur masque avec le coin supérieur gauche de l'image-objet séquence QuickTime. N'oubliez pas de définir le point d'alignement d'un bitmap sur le coin supérieur gauche, car il est défini sur le centre par défaut. Le point d'alignement de l'acteur QuickTime ne peut pas être défini ailleurs que sur le coin supérieur gauche. L'acteur masque ne peut pas être déplacé et n'est pas affecté par les propriétés `center` et `crop` de l'acteur qui lui est associé.

Pour optimiser les résultats, définissez la propriété mask d'un acteur QuickTime avant l'apparition de ses images-objets sur la scène dans le gestionnaire d'événement `on beginSprite`. En effet, la définition ou la modification de la propriété `mask` d'un acteur alors que celui-ci se trouve déjà sur la scène peut produire des résultats inattendus (par exemple, le masque peut apparaître sous la forme d'une image figée de l'animation numérique au moment où la propriété `mask` prend effet).

L'utilisation des masques est une fonction avancée, qui exige vraisemblablement plusieurs essais avant d'être maîtrisée.

Cette propriété peut être testée et définie. Pour supprimer un masque, définissez la propriété `mask` sur la valeur 0.

Exemple

Le script d'image suivant définit le masque d'une image-objet QuickTime avant que Director ne commence à dessiner l'image :

```
-- Lingo syntax
on prepareFrame
    member("Peeping Tom").mask = member("Keyhole")
end

// JavaScript syntax
function prepareFrame() {
    member("Peeping Tom").mask = member("Keyhole");
}
```

Voir aussi

`invertMask`

maxInteger

Syntaxe

```
the maxInteger
```

Description

Propriété système ; renvoie le nombre entier le plus élevé supporté par le système. Sur la plupart des ordinateurs personnels, ce nombre est 2 147 483 647 (2 à la puissance 31, moins 1).

Cette propriété peut servir à initialiser des variables utilisées dans des boucles ou pour limiter certains tests.

Pour utiliser des nombres supérieurs à la plage d'entiers utilisables, utilisez des nombres à virgule flottante. Ces nombres ne sont pas traités aussi rapidement que les nombres entiers, mais permettent d'utiliser une plage de valeurs plus étendue.

Exemple

L'instruction suivante crée dans la fenêtre Messages un tableau contenant les valeurs décimales maximales pouvant être représentées par un certain nombre de chiffres binaires :

```
on showMaxValues
    b = 31
    v = the maxInteger
    repeat while v > 0
        put b && "-" && v
        b = b-1
        v = v/2
    end repeat
end
```

maxSpeed

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).emitter.maxSpeed
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir et de définir la vitesse maximale à laquelle les particules sont émises. La vitesse initiale de chaque particule est sélectionnée de façon aléatoire et est comprise entre les propriétés `minSpeed` et `maxSpeed` de l'émetteur.

Cette valeur est un nombre à virgule flottante et doit être supérieure à 0,0.

Exemple

L'exemple suivant définit la vitesse maximale de la particule sur la valeur 15 dans l'acteur objets3D.

```
-- Lingo syntax
member("3Dobjects").modelResource("Particle01").emitter.maxSpeed=15

// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",10).emitter.maxSpeed=15;
```

Voir aussi

`minSpeed`, `emitter`

media

Syntaxe

```
-- Lingo syntax
memberObjRef.media

// JavaScript syntax
memberObjRef.media;
```

Description

Propriété d'acteur ; identifie l'acteur spécifié sous la forme d'une série de numéros. Lecture/écriture.

Cette propriété utilisant une grande quantité de mémoire, il est préférable de ne l'utiliser qu'en phase de création.

Vous pouvez utiliser la propriété `media` pour copier le contenu d'un acteur dans un autre en attribuant à la valeur `media` du deuxième acteur une valeur identique à la valeur `media` du premier acteur.

Pour un acteur boucle, la propriété `media` spécifie une sélection d'images et de pistes du scénario.

Pour permuter des médias dans une projection, il est plus efficace de définir la propriété d'image-objet `member`.

Exemple

L'instruction suivante copie le contenu de l'acteur leverDeSoleil dans l'acteur Aube en définissant la valeur de la propriété d'acteur `media` de l'acteur Aube sur la même valeur que la propriété d'acteur `media` de l'acteur leverDeSoleil :

```
-- Lingo syntax
member("Dawn").media = member("Sunrise").media

// JavaScript syntax
member("Dawn").media = member("Sunrise").media;
```

Voir aussi

Acteur

mediaReady

Syntaxe

```
-- Lingo syntax
memberObjRef.mediaReady

// JavaScript syntax
memberObjRef.mediaReady;
```

Description

Propriété d'acteur ; détermine si le contenu d'un acteur, d'un fichier d'animation ou de bibliothèque de distribution ou d'un acteur lié a été entièrement téléchargé à partir d'Internet et est disponible sur le disque local (`TRUE`) ou non (`FALSE`). Lecture seule.

Cette propriété n'est utile qu'en cas de lecture en flux continu d'un fichier d'animation ou de bibliothèque de distribution. Pour activer le transfert d'une animation, choisissez l'option Lire pendant le téléchargement (option par défaut) dans la boîte de dialogue Propriétés de lecture de l'animation du menu Modification.

Pour voir une démonstration de la propriété `mediaReady`, consultez l'animation Shockwave en flux continu dans l'Aide de Director.

Exemple

L'instruction suivante change les acteurs lorsque l'acteur souhaité est récupéré et disponible localement :

```
-- Lingo syntax
if member("background").mediaReady = TRUE then sprite(2).member =
member("background").number
end if

// JavaScript syntax
if (member("background").mediaReady == true) {
    sprite(2).member = member("background").number;
}
```

Voir aussi

Acteur

mediaStatus (DVD)

Syntaxe

```
-- Lingo syntax
dvdObjRef.mediaStatus

// JavaScript syntax
dvdObjRef.mediaStatus;
```

Description

Propriété de DVD ; renvoie un symbole indiquant l'état en cours du lecteur de DVD. Lecture seule.

Les symboles possibles sont les suivants :

Symbole	Description
#stopped	Le DVD est arrêté.
#playing	Le DVD est en cours de lecture.
#paused	Le DVD a été mis sur pause.
#scanning	Le DVD est en cours de balayage.
#uninitialized	Le DVD n'est pas initialisé.
#volumeInvalid	Le DVD spécifié n'est pas valide.
#volumeUknown	Le DVD n'existe pas ou le lecteur ne contient aucun disque.
#systemSoftwareMissing	Les décodeurs DVD ne sont pas installés.
#systemSoftwareBusy	Le logiciel système requis pour la lecture du DVD est en cours d'utilisation par une autre application.

Voir aussi

DVD

mediaStatus (RealMedia, Windows Media)

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.mediaStatus

// JavaScript syntax
memberOrSpriteObjRef.mediaStatus;
```

Description

Propriété d'image-objet ou d'acteur RealMedia et Windows Media ; permet d'obtenir un symbole représentant l'état du flux RealMedia ou Windows Media. Lecture seule.

La valeur de cette propriété peut changer pendant la lecture.

Les valeurs possibles de cette propriété sont les suivantes :

- `#closed` indique que l'acteur RealMedia ou Windows Media n'est pas actif. La valeur de `mediaStatus` reste `#closed` jusqu'au début de la lecture.

- `#connecting` indique qu'une connexion au flux RealMedia ou Windows Media est en cours d'établissement.
- `#opened` indique qu'une connexion au flux RealMedia ou Windows Media a été établie et est ouverte. Il s'agit d'un état transitoire, très rapidement suivi par `#buffering`.
- `#buffering` indique que le flux RealMedia ou Windows Media est en cours de téléchargement dans le tampon de lecture. Une fois la mise en tampon terminée (`percentBuffered` est égal à 100), la lecture du flux démarre si la propriété `pausedAtStart` présente la valeur `FALSE`. Pour plus d'informations, reportez-vous à l'entrée `percentBuffered`.
- `#playing` indique que le flux RealMedia ou Windows Media est en cours de lecture.
- `#seeking` indique que la lecture a été interrompue par la commande `seek`.
- `#paused` indique que la lecture a été interrompue, soit par la sélection du bouton Arrêter de la fenêtre RealMedia ou Windows Media, soit par l'invocation de la méthode `pause()` par un script.
- `#error` indique que le flux n'a pas pu être connecté, mis en tampon ou lu pour une raison quelconque. La propriété `lastError` indique l'erreur proprement dite.

Selon la valeur `state (RealMedia)` de l'acteur, une autre valeur de la propriété `mediaStatus` est renvoyée. Chaque valeur `mediaStatus` correspond à une seule valeur `state`.

Exemple

Les exemples suivants indiquent que l'élément RealMedia de l'image-objet 1 et l'acteur Real sont en cours de lecture.

```
-- Lingo syntax
put(sprite(1).mediaStatus)
put(member("RealDemo").mediaStatus)

// JavaScript syntax
put(sprite(1).mediaStatus);
put(member("RealDemo").mediaStatus);
```

Voir aussi

`state (RealMedia)`, `percentBuffered`, `lastError`

mediaXtraList

Syntaxe

```
-- Lingo syntax
_player.mediaXtraList

// JavaScript syntax
_player.mediaXtraList;
```

Description

Propriété de lecteur ; renvoie une liste linéaire de tous les Xtras de type Media disponibles dans le lecteur Director. Lecture seule.

Exemple

L'instruction suivante affiche dans la fenêtre Messages tous les Xtras de type Media disponibles dans le lecteur Director.

```
-- Lingo syntax
put(_player.mediaXtraList)
```

```
// JavaScript syntax
put(_player.mediaXtraList);
```

Voir aussi

Types de médias, Lecteur, scriptingXtraList, toolXtraList, transitionXtraList, xtraList (lecteur)

member

Syntaxe

```
member(whichCastmember).texture(whichTexture).member
member(whichCastmember).model(whichModel).shader.texture.member
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].textureList[textureL
istIndex].member
```

Description

Propriété 3D de texture ; si la texture est de type `#fromCastMember`, cette propriété indique l'acteur utilisé comme source de texture.

Cette propriété peut être testée et définie.

Si la texture est de type `#importedFromFile`, cette propriété présente la valeur void et n'est pas définissable. Si la texture est de type `#fromImageObject`, cette propriété présente la valeur `void`, mais peut néanmoins être définie.

Exemple

Le code Lingo suivant ajoute une nouvelle texture. La seconde instruction indique que l'acteur utilisé pour créer la texture gbTexture est l'acteur 16 de la distribution 1.

```
member("scene").newTexture("gbTexture", #fromCastmember, member(16, 1))
put member("scene").texture("gbTexture").member
-- (member 16 of castLib 1)
```

member (distribution)

Syntaxe

```
-- Lingo syntax
castObjRef.member[memberNameOrNum]

// JavaScript syntax
castObjRef.member[memberNameOrNum]
```

Description

Propriété de bibliothèque de distribution ; permet d'accéder par index ou par nom aux acteurs d'une bibliothèque de distribution. Lecture seule.

L'argument *`nomOuNumActeur`* peut être une chaîne spécifiant l'acteur par son nom ou un nombre entier désignant l'acteur par son numéro.

Exemple

L'exemple suivant accède au second acteur de la bibliothèque de distribution Interne.

```
-- Lingo syntax
myMember = castLib("Internal").member[2]

// JavaScript syntax
var myMember = castLib("Internal").member[2];
```

Voir aussi

Bibliothèque de distribution

member (animation)

Syntaxe

```
-- Lingo syntax
_movie.member[memberNameOrNum]

// JavaScript syntax
_movie.member[memberNameOrNum];
```

Description

Propriété d'animation ; permet d'accéder par index ou par nom aux acteurs d'une bibliothèque de distribution d'une animation. Lecture seule.

L'argument *nomOuNumActeur* peut être une chaîne spécifiant l'acteur par son nom ou un nombre entier désignant l'acteur par son numéro.

Exemple

L'instruction suivante accède à un acteur par son nom et par son numéro, puis affecte le résultat à la variable `myMember`.

```
-- Lingo syntax
myMember = _movie.member[2] -- using numbered access
myMember = _movie.member["Athlete"] -- using named access

// JavaScript syntax
var myMember = _movie.member[2]; // using numbered access;
var myMember = _movie.member["Athlete"]; // using named access;
```

Voir aussi

Animation

member (piste audio)

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.member

// JavaScript syntax
soundChannelObjRef.member;
```

Description

Propriété de piste audio ; indique l'acteur son en cours de lecture dans une piste audio. Lecture seule.

Cette propriété renvoie la valeur `VOID` (Lingo) ou `null` (syntaxe JavaScript) si aucun son n'est en cours de lecture.

Exemple

L'instruction suivante affiche dans la fenêtre Messages le nom de l'acteur correspondant au son lu dans la piste audio 2 :

```
-- Lingo syntax
put(sound(2).member)

// JavaScript syntax
put(sound(2).member);
```

Voir aussi

`Piste audio`

member (image-objet)

Syntaxe

```
-- Lingo syntax
spriteObjRef.member

// JavaScript syntax
spriteObjRef.member;
```

Description

Propriété d'image-objet ; spécifie l'acteur et la bibliothèque de distribution d'une image-objet. Lecture/écriture.

La propriété d'image-objet `member` diffère de la propriété d'image-objet `spriteNum` qui ne spécifie que le numéro de l'image-objet pour identifier son emplacement dans la bibliothèque de distribution, mais ne spécifie pas la bibliothèque de distribution proprement dite. La propriété `member` diffère également de la propriété `mouseMember` de l'objet souris qui ne spécifie pas la bibliothèque de distribution d'une image-objet.

Lorsque vous affectez la propriété d'image-objet `member`, utilisez l'un des formats suivants :

- Spécifiez la description complète de l'acteur et de la bibliothèque de distribution (*`refObjImageObjet`*`.member = member(`*`numActeur`* `{, nomOuNumBibliothèqueDistribution}`)).
- Spécifiez le nom de l'acteur (*`refObjImageObjet`*`.member = member("`*`chaîneNomActeur`*").
- Spécifiez le nombre entier unique incluant toutes les bibliothèques de distribution et correspondant à la propriété `mouseMember` (*`refObjImageObjet`*`.member = 132`).

Si vous n'utilisez que le nom de l'acteur, Director recherche le premier acteur portant ce nom dans toutes les bibliothèques de distribution en cours. Si ce nom se répète dans deux bibliothèques de distribution, seul le premier nom est utilisé.

Pour spécifier un acteur par son numéro uniquement lorsqu'il existe plusieurs distributions, utilisez la propriété d'image-objet `memberNum`, qui change la position de l'acteur dans sa bibliothèque de distribution sans affecter la bibliothèque de distribution de l'image-objet (*`refObjImageObjet`*`.memberNum = 10`).

L'acteur affecté à une piste d'image-objet n'est que l'une des propriétés de cette image-objet. Celle-ci comporte d'autres propriétés qui varient selon le type d'élément de média de cette piste du scénario. Par exemple, si vous remplacez un bitmap par une forme vide en définissant la propriété d'image-objet `member`, la propriété d'image-objet `lineSize` de l'image-objet forme ne change pas automatiquement et vous ne voyez probablement pas la forme.

Des problèmes de correspondance semblables peuvent se produire si vous changez l'acteur d'une image-objet champ en acteur vidéo. Il est généralement plus utile et plus sûr de remplacer des acteurs par des acteurs similaires. Par exemple, remplacez des images-objets bitmap par des acteurs bitmap.

Exemple

L'instruction suivante affecte l'acteur 3 de la distribution 4 à l'image-objet 15 :

```
-- Lingo syntax
sprite(15).member = member(3, 4)

// JavaScript syntax
sprite(15).member = member(3, 4);
```

Le gestionnaire suivant utilise la fonction mouseMember avec la propriété `sprite.member` afin de découvrir si la souris est positionnée sur une image-objet spécifique :

```
-- Lingo syntax
on exitFrame
    mm = _mouse.mouseMember
    target = sprite(1).member
    if (target = mm) then
        put("Above the hotspot.")
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var mm = _mouse.mouseMember;
    var target = sprite(1).member;
    if (target == mm) {
        put("Above the hotspot.");
        _movie.go(_movie.frame);
    }
}
```

Voir aussi

lineSize, mouseMember, Image-objet, spriteNum

memorySize

Syntaxe

```
the memorySize
```

Description

Propriété système ; renvoie la quantité totale de mémoire affectée au programme, qu'elle soit disponible ou non. Elle est utile pour vérifier la quantité de mémoire minimale requise. La valeur est exprimée en octets.

Sous Windows, cette valeur correspond à la mémoire physique totale disponible ; sur le Mac, elle représente la partition complète allouée à l'application.

Exemple

L'instruction suivante vérifie si l'ordinateur alloue plus de 500 Ko de mémoire et, le cas échéant, affiche un message d'alerte.

```
-- Lingo syntax
if the memorySize > 500 * 1024 then alert "There is enough memory to run this movie."

// JavaScript syntax
if ( _system.memorySize > 500 * 1024)
{
    _player.alert( "There is enough memory to run this movie.");
}
```

Voir aussi

freeBlock(), freeBytes(), ramNeeded(), size

meshDeform (modificateur)

Syntaxe

```
member(whichCastmember).model(whichModel).meshDeform.propertyName
```

Description

Modificateur 3D ; permet de contrôler les différents aspects de la structure de maille du modèle référencé. Lorsque vous ajoutez à un modèle le modificateur `#meshDeform` (à l'aide de la commande `addModifier`), vous avez accès aux propriétés suivantes du modificateur `#meshDeform` :

Remarque : *pour plus d'informations sur ces propriétés, consultez les entrées correspondantes (et qui apparaissent sous la section Voir aussi de cette entrée).*

- `face.count` renvoie le nombre total de faces du modèle référencé.
- `mesh.count` renvoie le nombre de mailles du modèle référencé.
- `mesh[index]` permet d'accéder aux propriétés de la maille spécifiée.

Exemple

L'instruction suivante affiche le nombre de faces du modèle gbFace.

```
put member("3D World").model("gbFace").meshDeform.face.count
-- 432
```

L'instruction suivante affiche le nombre de mailles du modèle gbFace :

```
put member("3D World").model("gbFace").meshDeform.mesh.count
-- 2
```

L'instruction suivante affiche le nombre de faces de la deuxième maille du modèle gbFace :

```
put member("3D World").model("gbFace").meshDeform.mesh[2].face.count
-- 204
```

Voir aussi

mesh (propriété), addModifier

milliseconds

Syntaxe

```
-- Lingo syntax
_system.milliseconds

// JavaScript syntax
_system.milliseconds;
```

Description

Propriété système ; renvoie l'heure actuelle en millisecondes (1/1 000ème de seconde). Lecture seule.

Le compte commence à partir du démarrage de l'ordinateur.

Exemple

L'instruction suivante convertit des millisecondes en secondes et minutes en divisant le nombre de millisecondes par 1 000 et le résultat par 60, puis affecte ce résultat à la variable `currentMinutes` :

```
-- Lingo syntax
currentSeconds = _system.milliseconds/1000
currentMinutes = currentSeconds/60

// JavaScript syntax
var currentSeconds = _system.milliseconds/1000;
var currentMinutes = currentSeconds/60;
```

La précision du calcul dépend de l'ordinateur et du système d'exploitation.

Le gestionnaire suivant compte les millisecondes et affiche un message d'alerte si vous travaillez depuis trop longtemps :

```
-- Lingo syntax
on idle
    if (_system.milliseconds > (1000 * 60 * 60 * 4)) then
        _player.alert("Take a break")
    end if
end

// JavaScript syntax
function idle() {
    if (_system.milliseconds > (1000 * 60 * 60 * 4)) {
        _player.alert("Take a break");
    }
}
```

Voir aussi

Système

minSpeed

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).emitter.minSpeed
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir et de définir la vitesse minimale à laquelle les particules sont émises. La vitesse initiale de chaque particule est sélectionnée de façon aléatoire et est comprise entre les propriétés `minSpeed` et `maxSpeed` de l'émetteur.

Cette valeur est un nombre à virgule flottante et doit être supérieure à 0,0.

Exemple

L'exemple suivant définit la vitesse minimale de la particule sur la valeur 4 dans l'acteur objets3D.

```
-- Lingo syntax
member("3Dobjects").modelResource("Particle01").emitter.minSpeed=4

// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",10).emitter.minSpeed=4;
```

Voir aussi

`maxSpeed`, `emitter`

missingFonts

Syntaxe

```
member(textCastMember).missingFonts
```

Description

Propriété d'acteur texte ; contient une liste des noms de polices utilisées dans le texte mais non disponibles sur le système.

Cette propriété permet aux développeurs de vérifier si une police spécifique est disponible ou non pendant l'exécution.

Cette propriété peut être testée, mais pas définie.

Exemple

L'exemple suivant affiche la liste des noms des polices référencées dans le texte mais non disponibles sur le système.

```
-- Lingo syntax
put member(4).missingFonts

// JavaScript syntax
put(member(4).missingFonts);
```

Voir aussi

`substituteFont`

mode (émetteur)

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).emitter.mode
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir et de définir la propriété `mode` de l'émetteur de particules de la ressource.

Cette propriété peut prendre la valeur `#burst` ou `#stream` (valeur par défaut). Lorsque `mode` présente la valeur `#burst`, toutes les particules sont émises en même temps, alors qu'avec la valeur `#stream`, un groupe de particules est émis à chaque image. Le nombre de particules émises à chaque image est déterminé au moyen de l'équation suivante :

```
particlesPerFrame = resourceObject.emitter.numParticles (resourceObject.lifetime x
millisecondsPerRenderedFrame)
```

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. L'instruction suivante attribue à la propriété `emitter.mode` de systèmeThermique la valeur `#burst`, ce qui entraîne l'émission des particules de systèmeThermique sous forme d'éclats. Pour créer une seule explosion de particules, définissez `emitter.mode = #burst` et `emitter.loop = 0`.

```
member("Fires").modelResource("ThermoSystem").emitter.mode = #burst
```

Voir aussi

`emitter`

mode (collision)

Syntaxe

```
member(whichCastmember).model(whichModel).collision.mode
```

Description

Propriété 3D de modificateur de collision ; indique la géométrie à utiliser dans l'algorithme de détection de collision. L'utilisation d'une géométrie plus simple, telle qu'une sphère de délimitation, permet une meilleure performance. Les valeurs possibles de cette propriété sont :

- `#mesh` utilise la géométrie de maille réelle de la ressource de modèle. Cette valeur offre une précision unitriangulaire et se révèle généralement plus lente que `#box` ou `#sphere`.
- `#box` utilise le cadre de délimitation du modèle. Cette valeur est utile pour les objets, tels qu'un mur, qui logent plus facilement dans une boîte que dans une sphère.
- `#sphere` constitue le mode le plus rapide car il utilise la sphère de délimitation du modèle. Il s'agit de la valeur par défaut de cette propriété.

Exemple

Les instructions suivantes ajoutent le modificateur de collision au modèle `yourModel` et attribuent à la propriété `mode` la valeur `#mesh` :

```
member("3d").model("yourModel").addModifier(#collision)
member("3d").model("yourModel").collision.mode = #mesh
```

model

Syntaxe

```
member(whichCastmember).model(whichModel)
member(whichCastmember).model[index]
member(whichCastmember).model.count
member(whichCastmember).model(whichModel).propertyName
member(whichCastmember).model[index].propertyName
```

Description

Commande 3D ; renvoie le modèle trouvé dans l'acteur référencé dont le nom est spécifié par *quelModèle*ou trouvé à la position d'index spécifiée par *index*. Si aucun modèle n'existe pour le paramètre spécifié, la commande renvoie `void`. Tout comme `model.count`, la commande renvoie le nombre de modèles détectés dans l'acteur référencé. Cette commande permet également d'accéder aux propriétés du modèle spécifié.

Les comparaisons de noms de modèle ne sont pas sensibles à la hauteur de casse. La position d'index d'un modèle particulier peut changer lorsque des objets dans des positions inférieures sont supprimés.

Si aucun modèle n'utilise le nom spécifié ou n'est trouvé à la position d'index spécifiée, cette commande renvoie la valeur void.

Exemple

L'instruction suivante stocke une référence au modèle Lecteur dans la variable `thismodel` :

```
thismodel = member("3DWorld").model("Player Avatar")
```

L'instruction suivante stocke une référence au huitième modèle de l'acteur Univers3D dans la variable `thismodel`.

```
thismodel = member("3DWorld").model[8]
```

L'instruction suivante indique qu'il existe quatre modèles dans l'acteur de l'image-objet 1.

```
put sprite(1).member.model.count
-- 4
```

modelA

Syntaxe

```
collisionData.modelA
```

Description

Propriété 3D `collisionData` ; indique l'un des modèles impliqués dans une collision, l'autre modèle étant `modelB`.

L'objet `collisionData` est envoyé comme argument avec les événements `#collideWith` et `#collideAny` au gestionnaire spécifié dans les commandes `registerForEvent`, `registerScript` et `setCollisionCallback`.

Les événements `#collideWith` et `#collideAny` sont envoyés en cas de collision entre deux modèles associés à des modificateur de collision. La propriété `resolve` des modificateurs de modèles doit être définie sur `TRUE`.

Cette propriété peut être testée, mais pas définie.

Exemple

L'exemple suivant est constitué de trois parties. La première partie constitue la première ligne de code, qui enregistre le gestionnaire `#putDetails` pour l'événement `#collideAny`. La deuxième partie correspond au gestionnaire `#placerDétails`. Lorsque deux modèles de l'acteur maSéquence entrent en collision, le gestionnaire `#putDetails` est appelé et l'argument `collisionData` lui est envoyé. Ce gestionnaire affiche les propriétés `modelA` et `modelB` de l'objet `collisionData` dans la fenêtre Messages. La troisième partie de l'exemple affiche les résultats de la fenêtre Messages. Cet exemple indique que le modèle balleVerte était modelA et que le modèle balleJaune était `modelB` dans la collision.

```
member("MyScene").registerForEvent(#collideAny, #putDetails, 0)
on putDetails me, collisionData
    put collisionData.modelA
    put collisionData.modelB
end
-- model("GreenBall")
-- model("YellowBall")
```

Voir aussi

`registerScript()`, `registerForEvent()`, `sendEvent`, `modelB`, `setCollisionCallback()`

modelB

Syntaxe

```
collisionData.modelB
```

Description

Propriété 3D `collisionData` ; indique l'un des modèles impliqués dans une collision, l'autre modèle étant `modelA`.

L'objet `collisionData` est envoyé comme argument avec les événements `#collideWith` et `#collideAny` au gestionnaire spécifié dans les commandes `registerForEvent`, `registerScript` et `setCollisionCallback`.

Les événements `#collideWith` et `#collideAny` sont envoyés en cas de collision entre deux modèles associés à des modificateur de collision. La propriété `resolve` des modificateurs des modèles doit présenter la valeur `TRUE`.

Cette propriété peut être testée, mais pas définie.

Exemple

L'exemple suivant est constitué de trois parties. La première partie constitue la première ligne de code, qui enregistre le gestionnaire `#putDetails` pour l'événement `#collideAny`. La deuxième partie correspond au gestionnaire `#putDetails`. Lorsque deux modèles de l'acteur maSéquence entrent en collision, le gestionnaire `#putDetails` est appelé et l'argument `collisionData` lui est envoyé. Ce gestionnaire affiche les propriétés `modelA` et `modelB` de l'objet `collisionData` dans la fenêtre Messages. La troisième partie de l'exemple affiche les résultats de la fenêtre Messages. Cet exemple indique que le modèle balleVerte était `modelA` et que le modèle balleJaune était `modelB` dans la collision.

```
member("MyScene").registerForEvent(#collideAny, #putDetails, 0)
on putDetails me, collisionData
    put collisionData.modelA
    put collisionData.modelB
end
-- model("GreenBall")
-- model("YellowBall")
```

Voir aussi

registerScript(), registerForEvent(), sendEvent, modelA, collisionNormal, setCollisionCallback()

modelResource

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource)
member(whichCastmember).modelResource[index]
member(whichCastmember).modelResource.count
member(whichCastmember).modelResource(whichModelResource).propertyName
member(whichCastmember).modelResource[index].propertyName
```

Description

Commande 3D ; renvoie la ressource de modèle trouvée dans l'acteur référencé dont le nom est spécifié par *quelleRessDeMod*ou trouvée à la position d'index spécifiée par le paramètre *index*. Si aucune ressource de modèle n'existe pour le paramètre spécifié, la commande renvoie `void`. Tout comme `modelResource.count`, la commande renvoie le nombre de ressources de modèle détectées dans l'acteur référencé. Cette commande permet également d'accéder aux propriétés du modèle spécifié.

Les comparaisons de chaînes de noms de ressources de modèle ne sont pas sensibles à la hauteur de casse. La position d'index d'une ressource de modèle particulière peut changer lorsque des objets dans des positions inférieures sont supprimés.

Exemple

L'instruction suivante affiche la première ressource de modèle de l'acteur objets3D.

```
-- Lingo syntax
putmember("3Dobjects").modelResource[1]

// JavaScript syntax
put( member("3Dobjects").getPropRef("modelResource",1));
```

modified

Syntaxe

```
-- Lingo syntax
memberObjRef.modified

// JavaScript syntax
memberObjRef.modified;
```

Description

Propriété d'acteur ; indique si un acteur a été modifié depuis sa lecture à partir d'un fichier d'animation. Lecture seule.

- Lorsque la propriété `modified` présente la valeur `TRUE` (`1`), l'acteur a été modifié depuis sa lecture à partir du fichier d'animation.
- Lorsque la propriété `modified` présente la valeur `FALSE` (`0`), l'acteur n'a pas été modifié depuis sa lecture à partir du fichier d'animation.

Exemple

L'instruction suivante vérifie si l'acteur Introduction a été modifié depuis sa lecture à partir du fichier de l'animation :

```
-- Lingo syntax
if (member("Introduction").modified) then
    _player.alert("Introduction has been modified")
else
    _player.alert("Introduction has not been modified")
end if

// JavaScript syntax
if (member("Introduction").modified) {
    _player.alert("Introduction has been modified");
}
else {
    _player.alert("Introduction has not been modified");
}
```

Voir aussi

Acteur

modifiedBy

Syntaxe

```
-- Lingo syntax
memberObjRef.modifiedBy

// JavaScript syntax
memberObjRef.modifiedBy;
```

Description

Propriété d'acteur ; enregistre le nom de l'utilisateur ayant modifié l'acteur en dernier. Lecture seule.

Cette information provient des données de nom d'utilisateur fournies au cours de l'installation de Director. Vous pouvez modifier cette information dans la boîte de dialogue Préférences générales de Director.

Cette propriété se révèle utile pour assurer le suivi et la coordination des projets Director impliquant plusieurs auteurs et peut également être affichée dans l'onglet Acteur de l'Inspecteur des propriétés.

Exemple

L'instruction suivante affiche le nom de la personne qui a effectué la modification la plus récente de l'acteur 1 :

```
-- Lingo syntax
put(member(1).modifiedBy)

// JavaScript syntax
put(member(1).modifiedBy);
```

Voir aussi

Acteur

modifiedDate

Syntaxe

```
-- Lingo syntax
memberObjRef.modifiedDate

// JavaScript syntax
memberObjRef.modifiedDate;
```

Description

Propriété d'acteur ; indique la date et l'heure de la dernière modification de l'acteur, à l'aide de l'heure système de l'ordinateur. Lecture seule.

Cette propriété s'avère utile pour assurer le suivi et la coordination des projets Director. Il est également possible de l'afficher dans l'onglet Acteur de l'Inspecteur des propriétés et dans la fenêtre Distribution en mode d'affichage sous forme de liste.

Exemple

L'instruction suivante affiche la date de la modification la plus récente de l'acteur 1 :

```
-- Lingo syntax
put(member(1).modifiedDate)

// JavaScript syntax
put(member(1).modifiedDate);
```

Voir aussi

Acteur

modifier

Syntaxe

```
member(whichCastmember).model(whichModel).modifier
member(whichCastmember).model(whichModel).modifier.count
```

Description

Propriété 3D de modèle ; renvoie une liste des modificateurs associés au modèle spécifié. Tout comme `modifier.count`, la commande renvoie le nombre de modificateurs associés au modèle.

Si les modificateurs `toon` et `inker` sont tous deux appliqués à un modèle, seul celui ayant été ajouté en premier est renvoyé.

Cette propriété peut être testée, mais pas définie. Utilisez les commandes `addModifier` et `removeModifier` pour ajouter des modificateurs aux modèles ou en supprimer.

Exemple

L'instruction suivante indique les modificateurs associés au modèle Sphère01 de l'acteur objets3D.

```
-- Lingo syntax
put member("3Dobjects").model("Sphere01").modifier

// JavaScript syntax
put( member("3Dobjects").getPropRef("model",2).modifier);
```

Voir aussi

modifier[], modifiers, addModifier, removeModifier

modifier[]

Syntaxe

```
member(whichCastmember).model(whichModel).modifier[index]
```

Description

Propriété 3D de modèle ; renvoie le type du modificateur situé à la position spécifiée par *index* dans la liste des modificateurs associés au modèle. La valeur renvoyée est un symbole.

Si aucun modificateur n'existe à la position spécifiée, cette propriété présente la valeur void.

Pour obtenir des informations sur la liste des modificateurs associés à un modèle, utilisez la propriété `modifier`.

L'accès direct aux propriétés d'un modificateur associé n'est pas supporté par cette commande.

Exemple

```
put member("3d world").model("box").modifier[1]
-- #lod
```

Voir aussi

modifier, modifiers, addModifier, removeModifier

modifiers

Syntaxe

```
getRendererServices().modifiers
```

Description

Propriété 3D globale ; renvoie une liste des modificateurs disponibles pour tous les modèles d'acteurs 3D.

Exemple

L'instruction suivante renvoie la liste de tous les modificateurs actuellement disponibles.

```
-- Lingo syntax
put getRendererServices().modifiers

// JavaScript syntax
put(getRendererServices().modifiers);
```

Voir aussi

getRendererServices(), addModifier

mostRecentCuePoint

Syntaxe

```
-- Lingo syntax
spriteObjRef.mostRecentCuePoint

// JavaScript syntax
spriteObjRef.mostRecentCuePoint;
```

Description

Propriété de piste audio et d'image-objet ; pour les images-objets audio, la vidéo numérique QuickTime et les Xtras prenant en charge les points de repère, cette propriété indique le numéro du dernier point de repère transmis dans l'image-objet ou dans le son. Sa valeur correspond au nombre ordinal du point de repère. Si aucun point de repère n'a été passé, sa valeur est de 0.

Les sons Shockwave Audio (SWA) peuvent apparaître sous forme d'images-objets dans les pistes d'images-objets, mais sont lus dans les pistes audio. Il est conseillé de faire référence aux images-objets audio SWA par le numéro de leur piste d'image-objet plutôt que par celui de leur piste audio.

Exemple

L'instruction suivante affiche dans la fenêtre Messages le numéro du point de repère passé en dernier dans l'image-objet de la piste d'image-objet 1 :

```
-- Lingo syntax
put sprite(1).mostRecentCuePoint

// JavaScript syntax
put(sprite(1).mostRecentCuePoint);
```

L'instruction suivante renvoie le nombre ordinal du point de repère transmis en dernier dans le son en cours de lecture dans la piste audio 2 :

```
-- Lingo syntax
put sound(2).mostRecentCuePoint

// JavaScript syntax
put(sound(2).mostRecentCuePoint);
```

Voir aussi

`cuePointNames`, `isPastCuePoint()`, `cuePointTimes`, `on cuePassed`

motion

Syntaxe

```
member(whichCastmember).motion(whichMotion)
member(whichCastmember).motion[index]
member(whichCastmember).motion.count
```

Description

Commande 3D ; renvoie le mouvement trouvé dans l'acteur référencé dont le nom est spécifié par *`quelMouvement`* ou trouvé à la position d'index spécifiée par *`index`*. Tout comme `motion.count`, cette propriété renvoie le nombre total de mouvements détectés dans l'acteur.

Les comparaisons de chaînes de nom d'objet ne sont pas sensibles à la hauteur de casse. La position d'index d'un mouvement particulier peut changer lorsque des objets dans des positions inférieures sont supprimés.

Si aucun mouvement n'utilise le nom spécifié ou n'est trouvé à la position d'index spécifiée, cette commande renvoie la valeur void.

Exemple

L'instruction suivante affiche le premier mouvement trouvé dans l'acteur référencé.

```
-- Lingo syntax
put member("3Dobjects").motion[1]

// JavaScript syntax
put(member("3Dobjects").getPropRef("motion",1));
```

Voir aussi

duration (3D), map (3D)

motionQuality

Syntaxe

```
-- Lingo syntax
spriteObjRef.motionQuality

// JavaScript syntax
spriteObjRef.motionQuality;
```

Description

Propriété d'image-objet QuickTime VR ; qualité de codec utilisée lorsque l'utilisateur clique sur une image-objet QuickTime VR et la fait glisser. Cette propriété peut prendre la valeur `#minQuality`, `#maxQuality` ou `#normalQuality`.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante définit la propriété motionQuality de l'image-objet 1 sur la valeur `#minQuality`.

```
-- Lingo syntax
sprite(1).motionQuality = #minQuality

// JavaScript syntax
sprite(1).motionQuality = symbol("minQuality");
```

mouseChar

Syntaxe

```
-- Lingo syntax
_mouse.mouseChar

// JavaScript syntax
_mouse.mouseChar;
```

Description

Propriété de souris ; pour les images-objets champ, cette propriété indique le numéro du caractère se trouvant sous le pointeur lorsqu'elle est appelée. Lecture seule.

Le compte démarre au début du champ. Si la souris n'est pas positionnée sur un champ ou se trouve sur la bordure d'un champ, le résultat est -1.

La valeur de la propriété `mouseChar` peut changer dans un gestionnaire ou dans une boucle. Si un gestionnaire ou une boucle utilise cette propriété à plusieurs reprises, il est recommandé de n'appeler cette propriété qu'une seule fois et d'affecter sa valeur à une variable locale.

Exemple

L'instruction suivante vérifie si le pointeur se trouve au-dessus d'une image-objet champ et, dans le cas contraire, remplace le contenu de l'acteur champ Instructions par « Veuillez pointer sur un caractère. »:

```
-- Lingo syntax
if (_mouse.mouseChar = -1) then
    member("Instructions").text = "Please point to a character."
end if

// JavaScript syntax
if (_mouse.mouseChar == -1) {
    member("Instructions").text = "Please point to a character.";
}
```

L'instruction suivante affecte à la variable `currentChar` le caractère du champ spécifié situé sous le pointeur :

```
-- Lingo syntax
currentChar = member(_mouse.mouseMember).char[_mouse.mouseChar]

// JavaScript syntax
var currentChar = member(_mouse.mouseMember).getProp("char",_mouse.mouseChar);
```

Voir aussi

Souris, mouseItem, mouseLine

mouseDown

Syntaxe

```
-- Lingo syntax
_mouse.mouseDown

// JavaScript syntax
_mouse.mouseDown;
```

Description

Propriété de souris ; indique si l'utilisateur est en train d'appuyer sur le bouton de la souris (`TRUE`) ou non (`FALSE`). Lecture seule.

Exemple

Le gestionnaire `mouseEnter` suivant associé à une image-objet appelle un gestionnaire si le bouton de la souris n'est pas enfoncé lorsque la souris entre dans l'image-objet, et appelle un autre gestionnaire si le bouton de la souris est enfoncé.

```
-- Lingo syntax
on mouseEnter
    if (_mouse.mouseDown) then
        runMouseDownScript
    else
        runMouseUpScript
    end if
end

// JavaScript syntax
function mouseEnter() {
    if (_mouse.mouseDown) {
        runMouseDownScript();
    }
    else {
        runMouseUpScript();
    }
}
```

Voir aussi

Souris, on mouseDown (gestionnaire d'événement), mouseH, mouseUp, on mouseUp (gestionnaire d'événement), mouseV

mouseDownScript

Syntaxe

```
the mouseDownScript
```

Description

Propriété système ; spécifie le code Lingo exécuté lorsque l'utilisateur appuie sur le bouton de la souris. Ce code est rédigé sous forme d'une chaîne encadrée de guillemets droits et peut être une simple instruction ou le script d'appel d'un gestionnaire. La valeur par défaut de la propriété `mouseDownScript` est `EMPTY`, ce qui signifie que cette propriété n'est associée à aucune expression Lingo.

Lorsque l'utilisateur appuie sur le bouton de la souris et que la propriété `mouseDownScript` est définie, Lingo commence par exécuter les instructions spécifiées dans la propriété `mouseDownScript`. A moins que les instructions contiennent la commande `pass` autorisant la transmission du message `mouseDown` vers d'autres objets de l'animation, aucun autre gestionnaire `on mouseDown` n'est exécuté.

La définition de la propriété `mouseDownScript` équivaut à utiliser la commande `when mouseDown then` des versions précédentes de Director.

Pour désactiver les instructions spécifiées pour la propriété `mouseDownScript`, utilisez l'instruction `set the mouseDownScript to empty`.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante demande l'exécution du script lorsque l'utilisateur appuie sur le bouton de la souris.

```
-- Lingo syntax
the mouseDownScript = "go to the frame"

// JavaScript syntax
_system.mouseDownScript = "_movie.go(_movie.frame)";
```

Voir aussi

stopEvent(), mouseUpScript, on mouseDown (gestionnaire d'événement), on mouseUp (gestionnaire d'événement)

mouseH

Syntaxe

```
-- Lingo syntax
_mouse.mouseH

// JavaScript syntax
_mouse.mouseH;
```

Description

Propriété de souris ; indique la position horizontale du pointeur de la souris. Lecture seule.

La valeur de `mouseH` correspond au nombre de pixels entre le curseur et le bord gauche de la scène.

La propriété `mouseH` se révèle utile pour déplacer des images-objets au même niveau horizontal que le pointeur de la souris et pour vérifier si celui-ci se trouve sur la scène. L'utilisation conjointe des propriétés `mouseH` et `mouseV` permet de déterminer la position exacte du curseur.

Exemple

Le gestionnaire suivant place l'image-objet 10 à la position du pointeur et met la scène à jour chaque fois que l'utilisateur appuie sur le bouton de la souris :

```
-- Lingo syntax
on mouseDown
    sprite(10).locH = _mouse.mouseH
    sprite(10).locV = _mouse.mouseV
end

// JavaScript syntax
function mouseDown() {
    sprite(10).locH = _mouse.mouseH;
    sprite(10).locV = _mouse.mouseV;
}
```

L'instruction suivante vérifie si le pointeur se trouve à plus de 10 pixels à droite ou à gauche d'un point de départ et, le cas échéant, attribue à la variable `Far` la valeur `TRUE` :

```
-- Lingo syntax
startH = 7
if (abs(_mouse.mouseH - startH) > 10) then
    Far = TRUE
end if

// JavaScript syntax
var startH = 7;
if (Math.abs(_mouse.mouseH - startH) > 10) {
    var Far = true;
}
```

Voir aussi

locH, locV, Souris, mouseLoc, mouseV

mouseItem

Syntaxe

```
-- Lingo syntax
_mouse.mouseItem

// JavaScript syntax
_mouse.mouseItem;
```

Description

Propriété de souris ; lorsqu'elle est appelée et que le pointeur se trouve au-dessus d'une image-objet champ, elle renvoie le numéro de l'élément situé sous le pointeur. Lecture seule.

Un élément est une séquence de caractères délimitée par le séparateur en cours défini par la propriété `itemDelimiter`. Le compte démarre au début du champ. Si la souris ne se trouve pas sur un champ, le résultat est -1.

La valeur de la propriété `mouseItem` peut changer dans un gestionnaire ou dans une boucle. Si un gestionnaire ou une boucle utilise la valeur initiale de `mouseItem` lorsqu'il ou elle démarre, n'appelez la propriété qu'une seule fois et affectez sa valeur à une variable locale.

Exemple

L'instruction suivante détermine si le pointeur se trouve au-dessus d'une image-objet champ. Dans le cas contraire, elle remplace le contenu de l'acteur champ Instructions par « Veuillez pointer sur un élément. » :

```
-- Lingo syntax
if (mouse.mouseItem = -1) then
    member("Instructions").text = "Please point to an item."
end if

// JavaScript syntax
if (_mouse.mouseItem == -1) {
    member("Instructions").text = "Please point to an item.";
}
```

L'instruction suivante affecte à la variable `currentItem` l'élément du champ spécifié situé sous le pointeur :

```
-- Lingo syntax
currentItem = member(_mouse.mouseMember).item[_mouse.mouseItem]

// JavaScript syntax
var currentItem = member(_mouse.mouseMember).getProp("item",_mouse.mouseItem);
```

Voir aussi

itemDelimiter, Souris, mouseChar, mouseLine, mouseWord

mouseLevel

Syntaxe

```
-- Lingo syntax
spriteObjRef.mouseLevel

// JavaScript syntax
spriteObjRef.mouseLevel;
```

Description

Propriété d'image-objet QuickTime ; contrôle la façon dont Director transmet à QuickTime les clics de la souris sur une image-objet QuickTime. La possibilité de transmettre les clics de souris à l'intérieur du rectangle de délimitation de l'image-objet peut se révéler utile avec les médias interactifs tels que QuickTime VR. La propriété d'image-objet `mouseLevel` peut prendre les valeurs suivantes :

- `#controller` : Ne transmet à QuickTime que les clics de la souris sur le contrôleur de l'animation. Director ne répond qu'aux clics de la souris qui se produisent à l'extérieur du contrôleur. Il s'agit du comportement normal des images-objets QuickTime autres que QuickTime VR.
- `#all` : Transmet à QuickTime tous les clics de la souris effectués à l'intérieur du rectangle de délimitation de l'image-objet. Aucun clic n'est passé aux autres gestionnaires Lingo.
- `#none` : Ne transmet aucun clic de souris à QuickTime. Director répond à tous les clics de la souris.
- `#shared` : Transmet à QuickTime tous les clics de la souris effectués dans le rectangle de délimitation d'une image-objet QuickTime VR, puis transmet ces événements aux gestionnaires Lingo. Il s'agit de la valeur par défaut pour QuickTime VR.

Cette propriété peut être testée et définie.

Exemple

Le script d'image suivant vérifie si le nom de l'image-objet QuickTime de la piste 5 contient la chaîne QTVR. Si ce n'est pas le cas, ce script définit la propriété `mouseLevel` sur `#all` ; dans l'autre cas, il définit `mouseLevel` sur `#none`.

```
on prepareFrame
    if sprite(5).member.name contains "QTVR" then
        sprite(5).mouseLevel = #all
    else
        sprite(5).mouseLevel = #none
    end if
end

// JavaScript syntax
function prepareFrame() {
    var nm = sprite(5).member.name;
    var nmStr = nm.indexOf("QTVR");
    if (nmStr != -1) {
        sprite(5).mouseLevel = symbol("all");
    } else {
        sprite(5).mouseLevel = symbol("none");
    }
}
```

mouseLine

Syntaxe

```
-- Lingo syntax
_mouse.mouseLine

// JavaScript syntax
_mouse.mouseLine;
```

Description

Propriété de souris ; lorsqu'elle est appelée et que le pointeur se trouve au-dessus d'une image-objet champ, elle renvoie le numéro de la ligne située sous le pointeur. Lecture seule.

Le compte commence au début du champ. Une ligne est définie par un retour de chariot et non par un retour à la ligne automatique. Si le pointeur de la souris ne se trouve pas sur une image-objet champ, le résultat est -1.

La valeur de la propriété `mouseLine` peut changer dans un gestionnaire ou dans une boucle. Si un gestionnaire ou une boucle utilise cette propriété à plusieurs reprises, il est recommandé de n'appeler cette propriété qu'une seule fois et d'affecter sa valeur à une variable locale.

Exemple

L'instruction suivante vérifie si le pointeur se trouve sur une image-objet champ et, dans le cas contraire, remplace le contenu de l'acteur champ Instructions par « Veuillez pointer sur une ligne. » :

```
-- Lingo syntax
if (_mouse.mouseLine = -1) then
    member("Instructions").text = "Please point to a line."
end if

// JavaScript syntax
if (_mouse.mouseLine == -1) {
    member("Instructions").text = "Please point to a line.";
}
```

L'instruction suivante affecte à la variable `ligneActuelle` le contenu de la ligne placée sous le pointeur dans le champ spécifié :

```
-- Lingo syntax
currentLine = member(_mouse.mouseMember).line[_mouse.mouseLine]

// JavaScript syntax
var currentLine = member(_mouse.mouseMember).getProp("line",_mouse.mouseLine);
```

Voir aussi

Souris, mouseChar, mouseItem, mouseWord

mouseLoc

Syntaxe

```
-- Lingo syntax
_mouse.mouseLoc

// JavaScript syntax
_mouse.mouseLoc;
```

Description

Propriété de souris ; renvoie la position en cours de la souris sous la forme d'un point(). Lecture seule.

L'emplacement du point est indiqué sous forme de deux coordonnées, d'abord l'emplacement horizontal, puis l'emplacement vertical.

Exemple

L'instruction suivante affiche la position en cours de la souris.

```
-- Lingo syntax
trace(_mouse.mouseLoc)

// JavaScript syntax
trace(_mouse.mouseLoc);
```

Voir aussi

Souris, mouseH, mouseV

mouseMember

Syntaxe

```
-- Lingo syntax
_mouse.mouseMember

// JavaScript syntax
_mouse.mouseMember;
```

Description

Propriété de souris ; lorsqu'elle est appelée, elle renvoie l'acteur affecté à l'image-objet qui se trouve sous le pointeur. Lecture seule.

Lorsque le pointeur ne se trouve pas sur une image-objet, cette propriété renvoie le résultat `VOID` (Lingo) ou `null` (syntaxe JavaScript).

Vous pouvez utiliser cette propriété pour que l'animation effectue des actions spécifiques lorsque le pointeur survole une image-objet et que celle-ci utilise un acteur spécifique.

La valeur de la propriété `mouseMember` peut changer fréquemment. Pour utiliser cette propriété à plusieurs reprises dans un gestionnaire avec une valeur cohérente, affectez la valeur de `mouseMember` à une variable locale lorsque le gestionnaire démarre et utilisez cette variable.

Exemple

L'instruction suivante vérifie si l'acteur Hors limites est l'acteur associé à l'image-objet qui se trouve sous le curseur et affiche un message d'alerte si c'est le cas. L'exemple suivant illustre la façon dont vous pouvez spécifier une action en fonction de l'acteur affecté à l'image-objet.

```
-- Lingo syntax
if (_mouse.mouseMember = member("Off Limits")) then
    _player.alert("Stay away from there!")
end if

// JavaScript syntax
if (_mouse.mouseMember == member("Off Limits")) {
    _player.alert("Stay away from there!");
}
```

L'instruction suivante affecte l'acteur de l'image-objet qui se trouve sous le pointeur à la variable `lastMember`:

```
-- Lingo syntax
lastMember = _mouse.mouseMember

// JavaScript syntax
var lastMember = _mouse.mouseMember;
```

Voir aussi

castLibNum, Souris

mouseOverButton

Syntaxe

```
-- Lingo syntax
spriteObjRef.mouseOverButton

// JavaScript syntax
spriteObjRef.mouseOverButton;
```

Description

Propriété d'image-objet Flash ; indique si le pointeur de la souris se trouve sur un bouton dans une image-objet d'animation Flash spécifiée par le paramètre *quelleImageObjetFlash* (TRUE) ou s'il se trouve en-dehors de l'image-objet ou encore à l'intérieur de l'image-objet, mais sur un élément autre qu'un bouton, tel qu'un arrière-plan (FALSE).

Cette propriété peut être testée, mais pas définie.

Exemple

Le script d'image suivant vérifie si le pointeur de la souris se trouve sur un bouton de navigation de l'animation Flash dans l'image-objet 3. Le cas échéant, le script met à jour un champ de texte en affichant un message approprié. Dans le cas contraire, le script efface le message.

```
-- Lingo syntax
on enterFrame
    case sprite(3).mouseOverButton of
        TRUE:
        member("Message Line").text = "Click here to go to the next page."
        FALSE:
        member("Message Line").text = " "
    end case
    _movie.updatestage()
end

// JavaScript syntax
function enterFrame() {
    switch(sprite(3).mouseOverButton)
    {
        case 1:
            member("Message Line").text = "Click here to go to the next page.";
            break;
        case 0:
            member("Message Line").text = " ";
            break;
    }
    _movie.updatestage();
}
```

mouseUp

Syntaxe

```
-- Lingo syntax
_mouse.mouseUp

// JavaScript syntax
_mouse.mouseUp;
```

Description

Propriété de souris ; indique si le bouton de la souris est relâché (TRUE) ou enfoncé (FALSE). Lecture seule.

Exemple

Le gestionnaire suivant entraîne la lecture de l'animation jusqu'à ce que l'utilisateur clique sur la souris. La tête de lecture s'arrête lorsque l'utilisateur relâche le bouton de la souris.

```
-- Lingo syntax
on exitFrame me
    if (_mouse.mouseUp) then
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
     if (_mouse.mouseUp) {
        _movie.go(_movie.frame);
    }
}
```

L'instruction suivante demande à Lingo de quitter la boucle de répétition ou le gestionnaire qu'il est en train d'exécuter lorsque l'utilisateur relâche le bouton de la souris :

```
-- Lingo syntax
if (_mouse.mouseUp) then exit

// JavaScript syntax
if (_mouse.mouseUp) {
    return;
}
```

Voir aussi

Souris, mouseDown, mouseH, mouseV

mouseUpScript

Syntaxe

```
the mouseUpScript
```

Description

Propriété système ; spécifie le code Lingo exécuté lorsque l'utilisateur relâche le bouton de la souris. Le code Lingo est rédigé sous forme d'une chaîne encadrée de guillemets droits et peut être une instruction simple ou un script d'appel d'un gestionnaire.

Lorsque l'utilisateur relâche le bouton de la souris et que la propriété `mouseUpScript` est définie, Lingo commence par exécuter les instructions spécifiées dans la propriété `mouseUpScript`. A moins que ces instructions contiennent la commande `pass` autorisant la transmission du message `mouseUp` à d'autres objets de l'animation, aucun autre gestionnaire `on mouseUp` n'est exécuté.

Lorsque les instructions spécifiées pour la propriété `mouseUpScript` ne sont plus appropriées, désactivez-les à l'aide de l'instruction `set the mouseUpScript to empty`.

La définition de la propriété `mouseUpScript` produit le même résultat que la commande `when mouseUp then` des versions précédentes de Director.

Cette propriété peut être testée et définie. La valeur par défaut est `EMPTY`.

Exemple

Lorsque l'instruction suivante est activée et que l'animation est sur pause, la lecture de l'animation reprend dès que l'utilisateur relâche le bouton de la souris :

```
the mouseUpScript = "go to the frame +1"
```

L'instruction suivante entraîne l'émission d'un signal sonore par l'animation chaque fois que l'utilisateur relâche le bouton de la souris après avoir cliqué sur un endroit quelconque de la scène :

```
the mouseUpScript = "if the clickOn = 0 then beep"
```

L'instruction suivante définit la propriété `mouseUpScript` sur le gestionnaire personnalisé *`myCustomHandler`*. Un gestionnaire personnalisé Lingo utilisé avec la propriété `mouseUpScript` doit être indiqué entre guillemets droits.

```
the mouseUpScript = "myCustomHandler"
```

Voir aussi

`stopEvent()`, `mouseDownScript`, `on mouseDown (gestionnaire d'événement)`, `on mouseUp (gestionnaire d'événement)`

mouseV

Syntaxe

```
-- Lingo syntax
_mouse.mouseV

// JavaScript syntax
_mouse.mouseV;
```

Description

Propriété de souris ; indique la position verticale du curseur de la souris, en pixels, à partir du haut de la scène. Lecture seule.

La valeur de cette propriété augmente lorsque le curseur se déplace vers le bas et diminue lorsqu'il se déplace vers le haut.

La propriété `mouseV` se révèle utile pour déplacer des images-objets au même niveau vertical que le curseur de la souris et pour vérifier si celui-ci se trouve dans une zone de la scène. L'utilisation conjointe des propriétés `mouseH` et `mouseV` permet d'identifier la position exacte du curseur.

Exemple

Le gestionnaire suivant place l'image-objet 1 à la position du pointeur et met la scène à jour chaque fois que l'utilisateur appuie sur le bouton de la souris :

```
-- Lingo syntax
on mouseDown
    sprite(1).locH = _mouse.mouseH
    sprite(1).locV = _mouse.mouseV
end

// JavaScript syntax
function mouseDown() {
    sprite(1).locH = _mouse.mouseH;
    sprite(1).locV = _mouse.mouseV;
}
```

L'instruction suivante vérifie si le pointeur se trouve à plus de 10 pixels au-dessus ou en dessous d'un point de départ et, le cas échéant, attribue à la variable `vFar` la valeur `TRUE` :

```
-- Lingo syntax
startV = 7
if (abs(_mouse.mouseV - startV) > 10) then
    vFar = TRUE
end if

// JavaScript syntax
var startV = 7
if (Math.abs(_mouse.mouseV - startV) > 10) {
    var vFar = true;
}
```

Voir aussi

locH, locV, Souris, mouseH, mouseLoc

mouseWord

Syntaxe

```
-- Lingo syntax
_mouse.mouseWord

// JavaScript syntax
_mouse.mouseWord;
```

Description

Propriété de souris ; lorsqu'elle est appelée et que le pointeur se trouve au-dessus d'une image-objet champ, elle renvoie le numéro du mot sur lequel se trouve le pointeur. Lecture seule.

Le compte commence au début du champ. Si la souris ne se trouve pas sur un champ, le résultat est 1.

La valeur de la propriété `mouseWord` peut changer dans un gestionnaire ou dans une boucle. Si un gestionnaire ou une boucle utilise cette propriété à plusieurs reprises, il est recommandé de n'appeler cette dernière qu'une seule fois et d'affecter sa valeur à une variable locale.

Exemple

L'instruction suivante vérifie si le pointeur se trouve au-dessus d'une image-objet champ et, si ce n'est pas le cas, remplace le contenu de l'acteur champ Instructions par Veuillez pointer sur un mot:

```
-- Lingo syntax
if (_mouse.mouseWord = -1) then
    member("Instructions").text = "Please point to a word."
else
    member("Instructions").text = "Thank you."
end if

// JavaScript syntax
if (_mouse.mouseWord == -1) {
    member("Instructions").text = "Please point to a word.";
}
else {
    member("Instructions").text = "Thank you.";
}
```

L'instruction suivante affecte à la variable `currentWord` le numéro du mot situé sous le pointeur dans le champ spécifié :

```
-- Lingo syntax
currentWord = member(_mouse.mouseMember).word[_mouse.mouseWord]

// JavaScript syntax
var currentWord = member(_mouse.mouseMember).getProp("word",_mouse.mouseWord);
```

Voir aussi

Souris, mouseChar, mouseItem

moveableSprite

Syntaxe

```
sprite(whichSprite).moveableSprite
the moveableSprite of sprite whichSprite
```

Description

Propriété d'image-objet ; indique si une image-objet peut être déplacée par l'utilisateur (`TRUE`) ou non (`FALSE`).

Vous pouvez rendre une image-objet déplaçable en sélectionnant l'option Déplaçable dans le scénario. Toutefois, pour contrôler si une image-objet est déplaçable et permettre ou non son déplacement, utilisez Lingo. Par exemple, pour permettre à l'utilisateur de faire glisser des images-objets une par une, puis d'empêcher le déplacement de ces dernières une fois qu'elles sont positionnées à l'endroit souhaité, vous pouvez activer et désactiver la propriété `moveableSprite` au moment approprié.

Remarque : *pour permettre un contrôle personnalisé tel que le retour automatique des images-objets à leur position d'origine ou leur animation pendant le glissement, créez un comportement permettant de gérer ces fonctionnalités supplémentaires.*

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant rend les images-objets de la piste 5 déplaçables :

```
on spriteMove
     sprite(5).moveableSprite = TRUE
end
```

L'instruction suivante vérifie si une image-objet est déplaçable et, dans le cas contraire, affiche un message :

```
if sprite(13).moveableSprite = FALSE thenmember("Notice").text = "You can't drag this item
by using the mouse."
```

Voir aussi

mouseLoc

movie

Syntaxe

```
-- Lingo syntax
windowObjRef.movie

// JavaScript syntax
windowObjRef.movie;
```

Description

Propriété de fenêtre ; renvoie une référence à l'objet animation en cours de lecture dans une fenêtre spécifiée. Lecture seule.

Exemple

L'instruction suivante affiche dans la fenêtre Messages l'objet animation en cours de lecture dans la fenêtre Empires :

```
-- Lingo syntax
trace(window("Empires").movie)

// JavaScript syntax
trace(window("Empires").movie);
```

Voir aussi

Fenêtre

multiSound

Syntaxe

```
the multiSound
```

Description

Propriété système ; indique si le système prend en charge plusieurs pistes audio et si une carte audio multipiste est requise sur un ordinateur Windows (TRUE) ou non (FALSE).

Exemple

L'instruction suivante lit le fichier audio Musique dans la piste audio 2 si l'ordinateur supporte plus d'une piste audio :

```
if the multiSound then sound playFile 2, "Music.wav"
```

name

Syntaxe

```
-- Lingo syntax
castObjRef.name
memberObjRef.name
_movie.name
windowObjRef.name

// JavaScript syntax
castObjRef.name;
memberObjRef.name;
_movie.name;
windowObjRef.name;
```

Description

Propriété de distribution, d'acteur, d'animation et de fenêtre ; renvoie ou définit le nom d'un objet. Lecture/écriture pour les objets Cast, Member et Window ; lecture seule pour les objets Movie.

Exemple

L'instruction suivante renomme l'acteur 1 en nouveauNom.

```
-- Lingo syntax
member(1).name="Newname"

// JavaScript syntax
member(1).name="Newname";
```

Voir aussi

Bibliothèque de distribution, Acteur, Animation, Fenêtre

name (3D)

Syntaxe

```
member(whichCastmember).texture(whichTexture).name
member(whichCastmember).shader(whichShader).name
member(whichCastmember).motion(whichMotion).name
member(whichCastmember).modelResource(whichModelResource).name
member(whichCastmember).model(whichModel).name
member(whichCastmember).light(whichLight).name
member(whichCastmember).camera(whichCamera).name
member(whichCastmember).group(whichGroup).name
node.name
```

Description

Propriété 3D ; utilisée avec une référence d'objet, cette propriété permet d'obtenir le nom de l'objet référencé. Vous ne pouvez qu'obtenir le nom ; ce dernier n'est pas modifiable.

Tous les noms doivent être uniques. S'il est créé avec du code Lingo, le nom renvoyé est celui donné par la fonction de constructeur. S'il est créé par une application de programmation 3D, il se peut que le nom renvoyé soit le nom du modèle.

Exemple

L'instruction suivante attribue à la cinquième caméra de l'acteur scèneDeTable le nom camOiseau :

```
member("TableScene").camera[5].name = "BirdCam"
```

name (propriété de menu)

Syntaxe

```
the name of menu(whichMenu)
the name of menu whichMenu
```

Description

Propriété de menu ; renvoie une chaîne contenant le nom du menu spécifié.

Cette propriété peut être testée, mais pas définie. Utilisez la commande `installMenu` pour créer une barre de menus personnalisée.

Remarque : *les menus ne sont pas disponibles dans Shockwave Player.*

Exemple

L'instruction suivante attribue le nom de menu numéro 1 à la variable `firstMenu` :

```
firstMenu = menu(1).name
```

Le gestionnaire suivant renvoie une liste de noms de menu, à raison d'un nom par ligne :

```
on menuList
    theList = []
    repeat with i = 1 to the number of menus
        theList[i] = the name of menu i
    end repeat
    return theList
end menuList
```

Voir aussi

number (menus), name (propriété d'élément de menu)

name (propriété d'élément de menu)

Syntaxe

```
the name of menuItem(whichItem) of menu(whichMenu)
the name of menuItem whichItem of menu whichMenu
```

Description

Propriété de menu ; détermine le texte qui apparaît dans l'élément de menu spécifié par *quelElément* dans le menu spécifié par *quelMenu*. L'argument *quelElément* est un nom ou un numéro d'élément de menu et *quelMenu* correspond à un nom ou à un numéro de menu.

Cette propriété peut être testée et définie.

Remarque : *les menus ne sont pas disponibles dans Shockwave Player.*

Exemple

L'instruction suivante attribue à la variable `itemName` le nom du huitième élément du menu Edition :

```
set itemName = the name of menuItem(8) of menu("Edit")
```

L'instruction suivante fait suivre la commande *Ouvrir* du menu Fichier par un nom de fichier spécifique :

```
the name of menuItem("Open") of menu("fileMenu") = "Open" && fileName
```

Voir aussi

`name (propriété de menu)`, `number (éléments de menu)`

name (image-objet)

Syntaxe

```
-- Lingo syntax
spriteObjRef.name

// JavaScript syntax
spriteObjRef.name;
```

Description

Propriété d'image-objet ; identifie le nom d'une image-objet. Lecture/écriture uniquement lors d'une session d'enregistrement de scénario.

Contrairement aux propriétés d'affichage d'image-objet telles que `backColor` et `blend`, la propriété d'image-objet `name` ne peut pas correspondre à une image-objet contrôlée par un script. Ceci signifie que la propriété `name` n'est définissable que lors d'une session d'enregistrement de scénario, c'est-à-dire entre les appels des méthodes `beginRecording()` et `endRecording()` de l'objet Animation. Vous ne pouvez définir la propriété `name` que si la méthode `beginRecording()` est appelée sur ou avant une image du scénario contenant l'image-objet.

Remarque : *le démarrage d'une session d'enregistrement de scénario à l'aide de la méthode* `beginRecording()` *réinitialise les propriétés de toutes les images-objets contrôlées par un script et de toutes les pistes d'image-objet.*

Si vous utilisez un script pour créer une image-objet lors d'une session d'enregistrement de scénario et que vous utilisez `updateFrame()` pour appliquer les données d'image-objet à la session, vous ne pouvez pas définir le nom de l'image-objet tant que vous n'êtes pas revenu à l'image dans laquelle l'image-objet a été créée. Pour revenir à une image spécifique, utilisez une méthode telle que `go()`.

Exemple

L'instruction suivante attribue à l'image-objet 5 le nom Fond Sonore :

```
-- Lingo syntax
sprite(5).name = "Background Sound"

// JavaScript syntax
sprite(5).name = "Background Sound";
```

Voir aussi

`beginRecording()`, `endRecording()`, `go()`, `Image-objet`, `updateFrame()`

name (piste d'image-objet)

Syntaxe

```
-- Lingo syntax
spriteChannelObjRef.name

// JavaScript syntax
spriteChannelObjRef.name;
```

Description

Propriété de piste d'image-objet ; identifie le nom d'une piste d'image-objet. Lecture/écriture uniquement lors d'une session d'enregistrement de scénario.

Définissez la propriété `name` d'une piste d'image-objet lors d'une session d'enregistrement de scénario, c'est-à-dire entre les appels des méthodes `beginRecording()` et `endRecording()` de l'objet Animation.

Remarque : *le démarrage d'une session d'enregistrement de scénario à l'aide de la méthode `beginRecording()` réinitialise les propriétés de toutes les images-objets contrôlées par un script et de toutes les pistes d'image-objet.*

Contrairement à la propriété `name` d'un objet image-objet, uniquement définissable sur ou après une image dans laquelle une image-objet apparaît dans le scénario, la propriété `name` d'un objet piste d'image-objet peut être définie sur une piste vide. Ceci signifie que vous n'avez pas besoin d'appeler la méthode `updateFrame()` avant de définir la propriété `name` de la piste d'image-objet.

Le changement d'un nom de piste d'image-objet n'est pas répercuté dans la fenêtre Scénario.

Exemple

L'instruction suivante définit le nom de la piste d'image-objet 6 sur Ficelle de cerf-volant lors d'une session d'enregistrement de scénario :

```
-- Lingo syntax
on mouseDown
    _movie.beginRecording()
    channel(6).name = "Kite string"
    _movie.endRecording()
end

// JavaScript syntax
function mouseDown() {
    _movie.beginRecording();
    channel(6).name = "Kite string";
    _movie.endRecording();
}
```

Voir aussi

`beginRecording()`, `endRecording()`, `Piste d'image-objet`

name (temporisation)

Syntaxe

```
timeoutObject.name
```

Description

Cette propriété de temporisation est le nom de l'objet de temporisation tel qu'il a été défini lors de la création de l'objet. La commande `new()` permet de créer des objets de temporisation.

Exemple

Le gestionnaire de temporisation suivant affiche une alerte contenant le nom de la temporisation ayant envoyé l'événement :

```
on handleTimeout timeoutObject
    alert "Timeout:" && timeoutObject.name
end
```

Voir aussi

`forget() (temporisation)`, `new()`, `period`, `persistent`, `target`, `time (objet de temporisation)`, `timeout()`, `timeoutHandler`, `timeoutList`

name (XML)

Syntaxe

```
XMLnode.name
```

Description

Propriété XML ; renvoie le nom du nœud XML spécifié.

Exemple

Avec le code XML suivant :

```
<?xml version="1.0"?>
 <e1>
<tagName attr1="val1" attr2="val2"/>
<e2> element 2</e2>
<e3>element 3</e3>
 </e1>
```

L'instruction Lingo suivante renvoie le nom de la seconde balise imbriquée dans la balise `<e1>` :

```
put gParserObject.child[1].child[2].name
-- "e2"
```

Voir aussi

`attributeName`

near (brouillard)

Syntaxe

```
member(whichCastmember).camera(whichCamera).fog.near
cameraReference.fog.near
member(whichCastmember).camera(whichCamera).fog.far
cameraReference.fog.far
```

Description

Propriété 3D ; permet d'obtenir ou de définir la distance séparant l'avant de la caméra du point où le brouillard commence si la propriété `fog.enabled` présente la valeur `TRUE`.

La valeur par défaut de cette propriété est 0,0.

Exemple

L'instruction suivante définit la propriété `near` du brouillard de la caméra vueParDéfaut sur la valeur 100.

```
-- Lingo syntax
member("3dobjects").camera("defaultview").fog.near = 100.0

// JavaScript syntax
member("3dobjects").getPropRef("camera",1).fog.near = 100.00;
```

Voir aussi

fog, far (brouillard), enabled (brouillard), decayMode

nearFiltering

Syntaxe

```
member(whichCastmember).texture(whichTexture).nearFiltering
member(whichCastmember).shader(whichShader).texture(whichTexture).nearFiltering
member(whichCastmember).model(whichModel).shader.texture(whichTexture).nearFiltering
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].texture(whichTexture
).nearFiltering
```

Description

Propriété 3D de texture ; permet de savoir ou de définir si le filtrage bilinéaire est utilisé pour le rendu d'une texture projetée qui couvre une plus grande partie de l'écran que l'original. Le filtrage bilinéaire estompe les erreurs de texture afin d'améliorer l'aspect de la texture. Le filtrage bilinéaire estompe les erreurs en deux dimensions. Le filtrage trilinéaire estompe les erreurs en trois dimensions. Le filtrage améliore l'aspect au détriment des performances, le filtrage bilinéaire se révélant plus rapide que le filtrage trilinéaire.

Lorsque cette propriété présente la valeur `TRUE`, le filtrage bilinéaire est utilisé. Lorsque cette propriété présente la valeur `FALSE`, le filtrage bilinéaire n'est pas utilisé. La valeur par défaut est `TRUE`.

Exemple

L'instruction suivante désactive le filtrage bilinéaire pour la première texture de l'acteur objets3D.

```
-- Lingo syntax
member("3dobjects").texture[1].nearFiltering = FALSE

// JavaScript syntax
member("3dobjects").getPropRef("texture",1).nearFiltering = 0;
```

netPresent

Syntaxe

```
-- Lingo syntax
_player.netPresent

// JavaScript syntax
_player.netPresent;
```

Description

Propriété de lecteur ; détermine si les Xtras requis pour accéder à Internet sont disponibles, mais n'indique pas si une connexion à Internet est actuellement active. Lecture seule.

Si les Xtras réseau ne sont pas disponibles, `netPresent` fonctionne correctement, mais `netPresent()` entraîne une erreur de script.

Exemple

L'instruction suivante envoie un message d'alerte si les Xtras ne sont pas disponibles :

```
-- Lingo syntax
if (not(_player.netPresent)) then
    _player.alert("Sorry, the Network Support Xtras could not be found.")
end if

// JavaScript syntax
if (!(_player.netPresent)) {
    _player.alert("Sorry, the Network Support Xtras could not be found.");
}
```

Voir aussi

Lecteur

netThrottleTicks

Syntaxe

```
-- Lingo syntax
_player.netThrottleTicks

// JavaScript syntax
_player.netThrottleTicks;
```

Description

Propriété de lecteur ; dans un environnement auteur Mac, permet de contrôler la fréquence de service d'une opération réseau. Lecture/écriture.

La valeur par défaut est de 15. Plus cette valeur est élevée, plus la lecture de l'animation et des effets animés est régulière, mais le temps passé à desservir les opérations réseau est diminué. Une valeur plus faible permet de passer plus de temps à desservir les opérations réseau, mais affecte négativement les performances de lecture des animations et des effets animés.

Cette propriété n'affecte que l'environnement auteur et les projections sur le Mac. Elle n'a aucun effet sous Windows ni dans Shockwave Player sur le Mac.

Exemple

L'instruction suivante indique la valeur de la propriété `netthrottleticks` dans le lecteur.

```
-- Lingo syntax
put _player.netThrottleTicks

// JavaScript syntax
 put( _player.netThrottleTicks);
```

Voir aussi

Lecteur

node

Syntaxe

```
-- Lingo syntax
spriteObjRef.node

// JavaScript syntax
spriteObjRef.node;
```

Description

Propriété d'image-objet QuickTime VR ; renvoie l'identifiant de nœud affiché par l'image-objet.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante indique l'identifiant de nœud en cours affiché par l'image-objet.

```
-- Lingo syntax
put sprite(3).node

// JavaScript syntax
put( sprite(3).node);
```

nodeEnterCallback

Syntaxe

```
-- Lingo syntax
spriteObjRef.nodeEnterCallback

// JavaScript syntax
spriteObjRef.nodeEnterCallback;
```

Description

Propriété d'image-objet QuickTime VR ; contient le nom du gestionnaire exécuté lorsque l'animation QuickTime VR est passée à un nouveau nœud actif de la scène. Ce message a deux arguments : le paramètre `me` et l'identifiant du nœud affiché.

L'image-objet QuickTime VR reçoit le message en premier.

Pour annuler l'appel, donnez à cette propriété une valeur de 0.

Pour des performances optimales, ne définissez de propriété d'appel uniquement lorsque cela est absolument nécessaire.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante indique le nom du gestionnaire exécuté une fois que l'animation QuickTime VR est passée à un nouveau nœud actif de la scène.

```
-- Lingo syntax
put sprite(3).nodeEnterCallback

// JavaScript syntax
put( sprite(3).nodeEnterCallback);
```

nodeExitCallback

Syntaxe

```
-- Lingo syntax
spriteObjRef.nodeExitCallback

// JavaScript syntax
spriteObjRef.nodeExitCallback;
```

Description

Propriété d'image-objet QuickTime VR ; contient le nom du gestionnaire exécuté lorsque l'animation QuickTime VR est sur le point de passer à un nouveau nœud actif de la scène. Le message contient trois arguments : le paramètre `me`, l'identifiant du nœud que l'animation est sur le point de quitter et celui du nœud auquel elle s'apprête à passer.

La valeur renvoyée par le gestionnaire détermine si l'animation passe ou non au nœud suivant. Si le gestionnaire renvoie la valeur `#continue`, l'image-objet QuickTime VR passe normalement au nœud suivant. S'il renvoie la valeur #cancel, la transition n'a pas lieu et l'animation reste sur le nœud d'origine.

Cette propriété doit être réglée sur 0 pour effacer l'instruction d'appel.

L'image-objet QuickTime VR reçoit le message en premier.

Pour des performances optimales, ne définissez de propriété d'appel uniquement lorsque cela est absolument nécessaire.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante indique le nom du gestionnaire exécuté lorsque l'animation QuickTime VR est sur le point de passer à un nouveau nœud actif de la scène.

```
-- Lingo syntax
put sprite(3).nodeExitCallback

// JavaScript syntax
put( sprite(3).nodeExitCallback);
```

nodeType

Syntaxe

```
-- Lingo syntax
spriteObjRef.nodeType

// JavaScript syntax
spriteObjRef.nodeType;
```

Description

Propriété d'image-objet QuickTime VR ; renvoie le type du nœud de l'image-objet spécifiée actuellement sur la scène. Les valeurs possibles sont `#object`, `#panorama` ou `#unknown`. (La valeur `#unknown` correspond à la valeur d'une image-objet autre qu'une image-objet QuickTime VR.)

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante indique le type du nœud de l'image-objet spécifiée actuellement sur la scène.

```
-- Lingo syntax
put sprite(3).nodeType

// JavaScript syntax
put( sprite(3).nodeType);
```

normalList

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).normalList
model.meshDeform.mesh[index].normalList
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#mesh`, cette propriété permet d'obtenir ou de définir la propriété `normalList` de la ressource de modèle.

La propriété `normalList` est une liste de vecteurs linéaire à partir de laquelle vous pouvez spécifier des normales de sommets pour la construction des faces de votre maille.

Cette propriété doit être définie à l'aide d'une liste comprenant exactement le même nombre de vecteurs que dans l'appel `newMesh()`.

La propriété `normalList` peut également être générée par la méthode `generateNormals()` des ressources de modèle de maille.

De la même façon, dans le cas du modificateur `meshDeform`, la propriété `normalList` est une liste de vecteurs linéaire à partir de laquelle vous pouvez spécifier les normales des sommets pour déformer votre maille.

Pour plus d'informations sur les normales de faces et de sommets, reportez-vous à l'entrée `normals`.

Exemple

L'instruction suivante affiche la propriété `normalList` de la ressource de modèle pyramide de l'acteur objets3D.

```
-- Lingo syntax
put member("3Dobjects").modelResource("pyramid").normalList[2]
```

```
// JavaScript syntax
put(member("3Dobjects").getPropRef("modelResource",10).normalList[2]);
```

Voir aussi

face, meshDeform (modificateur)

normals

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).face[index].normals
```

Syntaxe

Propriété 3D de face ; pour les ressources de modèle de type `#mesh` (créées à l'aide de la commande `newMesh`), cette propriété permet d'obtenir et de définir la liste des vecteurs de normales utilisés par la face spécifiée par le paramètre *index*.

Définissez cette propriété à l'aide d'une liste linéaire de nombres entiers correspondant à la position d'index de chaque normale de sommet dans la propriété `normalList` de la ressource de modèle.

Cette propriété doit être définie à la même longueur que la liste `face[index].vertices` ou peut être une liste vide [ ].

Ne définissez aucune valeur pour cette propriété si vous prévoyez de générer des vecteurs de normales à l'aide de la commande `generateNormals()`.

Si vous apportez des modifications à cette propriété ou utilisez la commande `generateNormals()`, vous devrez appeler la commande `build()` pour reconstruire la maille.

Exemple

L'instruction suivante affiche la propriété `normals` de la cinquième face de la ressource de modèle pyramide de l'acteur objets3D.

```
-- Lingo syntax
put member("3Dobjects").modelResource("pyramid").face[5].normals

// JavaScript syntax
put(member("3Dobjects").getPropRef("modelResource",10).face[5].normals);
```

Voir aussi

face, normalList, vertices

number (distribution)

Syntaxe

```
-- Lingo syntax
castObjRef.number

// JavaScript syntax
castObjRef.number;
```

Description

Propriété de bibliothèque de distribution ; renvoie le numéro d'une bibliothèque de distribution spécifiée. Lecture seule.

Exemple

La boucle de répétition suivante utilise la fenêtre Messages pour afficher le nombre d'acteurs contenus dans chaque distribution de l'animation :

```
-- Lingo syntax
repeat with n = 1 to _movie.castLib.count
    put(castLib(n).name && "contains" && castLib(n).member.count && "cast members.")
end repeat

// JavaScript syntax
for (var n=1; n<=_movie.castLib.count; n++) {
    put(castLib(n).name + " contains " + castLib(n).member.count + " cast members.")
}
```

Voir aussi

Bibliothèque de distribution

number (caractères)

Syntaxe

```
the number of chars in chunkExpression
```

Description

Expression de sous-chaîne ; renvoie le nombre total de caractères d'une expression de sous-chaîne.

Une expression de sous-chaîne peut être un caractère (y compris des espaces et des caractères de contrôle tels que Tabulation et Retour), un mot, un élément ou la ligne d'un conteneur de caractères. Les conteneurs peuvent être des acteurs champ et des variables contenant des chaînes, ainsi que des caractères, des mots, des éléments, des lignes et des plages spécifiés dans des conteneurs.

Remarque : *la fonction* `count()` *se révèle plus efficace pour déterminer le nombre de caractères d'une sous-chaîne.*

Exemple

L'instruction suivante affiche le nombre de caractères de la chaîne « Adobe, la société multimédia » dans la fenêtre Messages :

```
put the number of chars in "Adobe, the Multimedia Company"
```

Le résultat est 29.

L'instruction suivante attribue à la variable `charCounter` le nombre de caractères du mot i situé dans la chaîne Noms :

```
charCounter = the number of chars in member("Names").word[i]
```

Vous pouvez obtenir les mêmes résultats avec les acteurs texte en utilisant la syntaxe suivante :

```
charCounter = member("Names").word[i].char.count
```

Voir aussi

length(), char...of, count(), number (éléments), number (lignes), number (mots)

number (éléments)

Syntaxe

```
the number of items in chunkExpression
```

Description

Sous-chaîne ; renvoie le nombre total d'éléments d'une expression de sous-chaîne. Un élément de sous-chaîne est une série de caractères délimitée par des virgules.

Une expression de sous-chaîne peut être un caractère, un mot, un élément ou la ligne d'un conteneur de caractères. Les conteneurs peuvent être des acteurs champ et des variables contenant des chaînes, ainsi que des caractères, des mots, des éléments, des lignes et des plages spécifiés dans des conteneurs.

Remarque : *la fonction* `count()` *se révèle plus efficace pour déterminer le nombre d'éléments d'une expression de sous-chaîne.*

Exemple

L'instruction suivante affiche le nombre d'éléments de la chaîne « Adobe, la société multimédia » dans la fenêtre Messages :

```
put the number of items in "Adobe, the Multimedia Company"
```

Le résultat est 2.

L'instruction suivante attribue à la variable `itemCounter` le nombre d'éléments du champ Noms :

```
itemCounter = the number of items in member("Names").text
```

Vous pouvez obtenir les mêmes résultats avec les acteurs texte en utilisant la syntaxe suivante :

```
itemCounter = member("Names").item.count
```

Voir aussi

`item...of`, `count()`, `number (caractères)`, `number (lignes)`, `number (mots)`

number (lignes)

Syntaxe

```
the number of lines in chunkExpression
```

Description

Sous-chaîne ; renvoie le nombre total de lignes d'une expression de sous-chaîne. Les lignes sont délimitées par des retours de chariot et non par des retours à la ligne automatiques.

Une expression de sous-chaîne peut être un caractère, un mot, un élément ou la ligne d'un conteneur de caractères. Les conteneurs peuvent être des acteurs champ et des variables contenant des chaînes, ainsi que des caractères, des mots, des éléments, des lignes et des plages spécifiés dans des conteneurs.

Remarque : *la fonction* `count()` *se révèle plus efficace pour déterminer le nombre de lignes d'une sous-chaîne.*

Exemple

L'instruction suivante affiche le nombre de lignes de la chaîne « Adobe, la société multimédia » dans la fenêtre Messages :

```
put the number of lines in "Adobe, the Multimedia Company"
```

Le résultat est 1.

L'instruction suivante attribue à la variable `lineCounter` le nombre de lignes contenues dans le champ Noms :

```
lineCounter = the number of lines in member("Names").text
```

Vous pouvez obtenir les mêmes résultats avec les acteurs texte en utilisant la syntaxe suivante :

```
lineCounter = member("Names").line.count
```

Voir aussi

line...of, count(), number (caractères), number (éléments), number (mots)

number (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.number

// JavaScript syntax
memberObjRef.number;
```

Description

Propriété d'acteur ; indique le numéro de bibliothèque de distribution d'un acteur spécifié. Lecture seule.

La valeur de cette propriété est un identifiant unique de l'acteur. Il s'agit d'un nombre entier décrivant l'emplacement et la position de l'acteur dans la bibliothèque de distribution.

Exemple

L'instruction suivante attribue le numéro de l'acteur Interrupteur à la variable *whichCastMember* :

```
-- Lingo syntax
whichCastMember = member("Power Switch").number

// JavaScript syntax
var whichCastMember = member("Power Switch").number;
```

L'instruction suivante affecte l'acteur Ballon rouge à l'image-objet 1 :

```
-- Lingo syntax
sprite(1).member = member("Red Balloon").number

// JavaScript syntax
sprite(1).member = member("Red Balloon").number;
```

L'instruction suivante vérifie l'existence d'un acteur avant de changer l'acteur associé à l'image-objet :

```
-- Lingo syntax
property spriteNum

on mouseUp me
    if (member("Mike's face").number > 0) then
        sprite(spriteNum).member = "Mike's face"
    end if
end
```

```
// JavaScript syntax
function mouseUp() {
    if (member("Mike's face").number > 0) {
        sprite(this.spriteNum).member = "Mike's face"
    }
}
```

Voir aussi

`castLib()`, `Acteur`

number (menus)

Syntaxe

```
the number of menus
```

Description

Propriété de menu ; indique le nombre de menus de l'animation actuelle.

Cette propriété peut être testée, mais non définie. Pour créer une barre de menus personnalisée, utilisez la commande `installMenu`.

Remarque : *les menus ne sont pas disponibles dans Shockwave Player.*

Exemple

L'instruction suivante détermine si l'animation contient des menus personnalisés et, si ce n'est pas le cas, installe le menu barreDeMenus :

```
if the number of menus = 0 then installMenu "Menubar"
```

L'instruction suivante affiche dans la fenêtre Messages le nombre de menus de l'animation en cours :

```
put the number of menus
```

Voir aussi

`installMenu`, `number (éléments de menu)`

number (éléments de menu)

Syntaxe

```
the number of menuItems of menu whichMenu
```

Description

Propriété de menu ; indique le nombre d'éléments du menu personnalisé spécifié par *`quelMenu`*. Le paramètre *`quelMenu`* peut être un nom ou un numéro de menu.

Cette propriété peut être testée, mais non définie. Pour créer une barre de menus personnalisée, utilisez la commande `installMenu`.

Remarque : *les menus ne sont pas disponibles dans Shockwave Player.*

Exemple

L'instruction suivante attribue à la variable `fileItems` le nombre d'éléments du menu personnalisé Fichier :

```
fileItems = the number of menuItems of menu "File"
```

L'instruction suivante attribue à la variable `itemCount` le nombre d'éléments du menu personnalisé dont le numéro est égal à la variable i :

```
itemCount = the number of menuItems of menu i
```

Voir aussi

`installMenu`, `number (menus)`

number (piste d'image-objet)

Syntaxe

```
-- Lingo syntax
spriteChannelObjRef.number

// JavaScript syntax
spriteChannelObjRef.number;
```

Description

Propriété de piste d'image-objet ; renvoie le numéro d'une piste d'image-objet. Lecture seule.

Exemple

L'instruction suivante affiche dans la fenêtre Messages le numéro d'une piste d'image-objet spécifiée :

```
-- Lingo syntax
put(channel("Kite String").number)

// JavaScript syntax
put(channel("Kite String").number);
```

Voir aussi

`Image-objet`

number (système)

Syntaxe

```
the number of castLibs
```

Description

Propriété système ; renvoie le nombre de distributions de l'animation actuelle.

Cette propriété peut être testée, mais pas définie.

Exemple

La boucle de répétition suivante utilise la fenêtre Messages pour afficher le nombre d'acteurs contenus dans chaque distribution de l'animation :

```
repeat with n = 1 to the number of castLibs
```

```
    put castLib(n).name && "contains" && the number of members of castLib(n) && "cast
members."
end repeat
```

number (mots)

Syntaxe

```
the number of words in chunkExpression
```

Description

Expression de sous-chaîne ; renvoie le nombre de mots dans l'expression de sous-chaîne spécifiée par *expressionSousChaîne*.

Une expression de sous-chaîne peut être un caractère, un mot, un élément ou une ligne d'un conteneur de caractères. Les conteneurs peuvent être des acteurs champ et des variables contenant des chaînes, ainsi que des caractères, des mots, des éléments, des lignes et des plages spécifiés dans des conteneurs.

Pour calculer le nombre de mots contenus dans les acteurs texte, reportez-vous à l'entrée `count`.

Remarque : *la fonction `count()` se révèle plus efficace pour déterminer le nombre de mots d'une sous-chaîne.*

Exemple

L'instruction suivante affiche dans la fenêtre Messages le nombre de mots de la chaîne « Adobe, la société multimédia » :

```
put the number of words in "Adobe, the multimedia company"
```

Le résultat est 4.

Le gestionnaire suivant inverse l'ordre des mots de la chaîne spécifiée par l'argument `wordList` :

```
on reverse wordList
    theList = EMPTY
    repeat with i = 1 to the number of words in wordList
        put word i of wordList & " " before theList
    end repeat
    delete theList.char[thelist.char.count]
    return theList
end
```

Voir aussi

`count()`, `number (caractères)`, `number (éléments)`, `number (lignes)`, `word...of`

number of members

Syntaxe

```
the number of members of castLib whichCast
```

Description

Propriété d'acteur ; indique le numéro du dernier acteur de la distribution spécifiée.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affiche le type de chaque acteur de la distribution Distribution Centrale dans la fenêtre Messages. La propriété `members of castLib` est utilisée pour déterminer le nombre de répétitions de la boucle.

```
repeat with i = 1 to the number of members of castLib("Central Casting")
    put "Cast member" && i && "is a" && member(i, "Central Casting").type
end repeat
```

number of xtras

Syntaxe

```
the number of xtras
```

Description

Propriété système ; renvoie le nombre d'Xtras de programmation disponibles pour l'animation. Il peut s'agir du nombre d'Xtras ouverts avec la commande `openxlib` ou figurant dans le dossier Configuration\Xtras.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affiche le nombre d'Xtras de programmation disponibles pour l'animation dans la fenêtre Messages :

```
put the number of xtras
```

numChannels

Syntaxe

```
-- Lingo syntax
memberObjRef.numChannels

// JavaScript syntax
memberObjRef.numChannels;
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; renvoie le nombre de pistes de l'acteur SWA spécifié lu en flux continu. Les valeurs sont 1 pour mono ou 2 pour stéréo.

Cette propriété n'est disponible qu'après le début de la lecture en flux continu de l'acteur SWA ou après le préchargement du fichier avec la commande `preLoadBuffer`.

Cette propriété peut être testée, mais pas définie.

Exemple

Dans l'exemple suivant, le nombre de pistes audio de l'acteur SWA transféré Duke Ellington est affecté à l'acteur champ Affichage des pistes :

```
-- Lingo syntax
myVariable = member("Duke Ellington").numChannels
if myVariable = 1 then
    member("Channel Display").text = "Mono"
else
```

```
    member("Channel Display").text = "Stereo"
end if

// JavaScript syntax
var myVariable = member("Duke Ellington").numChannels;
if (myVariable == 1) {
    member("Channel Display").text = "Mono";
} else {
    member("Channel Display").text = "Stereo";
}
```

numParticles

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).emitter.numParticles
modelResourceObjectReference.emitter.numParticles
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir ou de définir la propriété `numParticles` de l'émetteur de particules de la ressource. La valeur doit être supérieure à 0 et inférieure ou égale à 100 000, la valeur par défaut étant 1 000.

Exemple

L'instruction suivante définit le nombre de particules sur 50 000 dans l'acteur objets3D.

```
-- Lingo syntax
member("3Dobjects").modelResource("Particle01").emitter.numParticles = 50000

// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",10).emitter.numParticles = 50000;
```

Voir aussi

emitter

numSegments

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).numSegments
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#cylinder`, cette propriété permet d'obtenir ou de définir la propriété `numSegments` de la ressource de modèle.

La propriété `numSegments` détermine le nombre de segments entre l'extrémité supérieure et l'extrémité inférieure du cylindre. Cette propriété doit être supérieure ou égale à la valeur par défaut de 2.

Le lissé de la surface du cylindre dépend de la valeur spécifiée pour cette propriété. Plus la valeur de la propriété est élevée, plus la surface du cylindre apparaît lisse.

Exemple

L'instruction suivante attribue à la propriété `numSegments` de la ressource de modèle Cylindre01 la valeur 10.

```
-- Lingo syntax
member("3Dobjects").modelResource("Cylinder01").numSegments = 10

// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",11).numSegments = 10;
```

obeyScoreRotation

Syntaxe

```
member(flashMember).obeyScoreRotation
```

Description

Propriété d'acteur Flash ; définie sur `TRUE` ou sur `FALSE` pour déterminer si une image-objet d'animation Flash utilise les informations de rotation du scénario ou l'ancienne propriété de rotation des éléments Flash.

Cette propriété est automatiquement définie sur `FALSE` pour toutes les animations créées par des versions de Director antérieures à la version 7 afin de conserver l'ancienne fonctionnalité qui consistait à utiliser la propriété de rotation d'acteurs pour toutes les images-objets contenant cet acteur Flash.

La propriété des nouveaux éléments créés par la version 7 ou ultérieure est automatiquement définie sur `TRUE`.

Si cette propriété est définie sur `TRUE`, la propriété de rotation de l'acteur est ignorée et les paramètres de rotation du scénario sont utilisés à la place.

Exemple

Le script suivant définit la propriété `obeyScoreRotation` de l'acteur objFlash sur 1 (`TRUE`), puis fait pivoter de 180 degrés l'image-objet qui contient l'acteur.

```
-- Lingo syntax
member("FlashObj").obeyScoreRotation = 1
sprite(1).rotation = sprite(1).rotation + 180

// JavaScript syntax
member("FlashObj").obeyScoreRotation = 1;
sprite(1).rotation = sprite(1).rotation + 180;
```

Voir aussi

`rotation`

optionDown

Syntaxe

```
-- Lingo syntax
_key.optionDown

// JavaScript syntax
_key.optionDown;
```

Description

Propriété de touche ; détermine si l'utilisateur appuie sur la touche Alt (Windows) ou sur la touche Option (Mac). Lecture seule.

Cette propriété renvoie la valeur TRUE si l'utilisateur appuie sur la touche Alt ou Option ; dans le cas contraire, elle renvoie la valeur FALSE.

Sous Windows, la propriété optionDown ne fonctionne pas dans les projections si la touche Alt est utilisée sans être combinée à une touche autre qu'une touche de modification. Evitez d'utiliser optionDown si vous prévoyez de diffuser une animation sous forme de projection Windows et que vous avez uniquement besoin de détecter la sélection de la touche de modification ; dans ce cas, utilisez plutôt controlDown ou shiftDown.

Sur le Mac, la sélection de la touche Option modifie la valeur de la propriété key ; utilisez donc keyCode à la place.

Exemple

Le gestionnaire suivant vérifie si l'utilisateur appuie sur la touche Alt ou Option et, le cas échéant, appelle le gestionnaire doOptionKey :

```
-- Lingo syntax
on keyDown
    if (_key.optionDown) then
        doOptionKey(_key.key)
    end if
end

// JavaScript syntax
function keyDown() {
    if (_key.optionDown) {
        doOptionKey(_key.key);
    }
}
```

Voir aussi

controlDown, Touche, key, keyCode, shiftDown

organizationName

Syntaxe

```
-- Lingo syntax
_player.organizationName

// JavaScript syntax
_player.organizationName;
```

Description

Propriété de lecteur ; contient le nom de société entré lors de l'installation de Director. Lecture seule.

Cette propriété est uniquement disponible dans l'environnement de programmation. Elle est utilisable dans une animation dans une fenêtre personnalisée pour afficher des informations utilisateur.

Exemple

Le gestionnaire suivant serait normalement placé dans un script d'animation d'une animation MIAW. Il place le nom de l'utilisateur et le numéro de série dans une zone d'affichage dès l'ouverture de la fenêtre :

```
-- Lingo syntax
on prepareMovie
    displayString = _player.userName & RETURN & _player.organizationName & RETURN &
_player.serialNumber
```

```
    member("User Info").text = displayString
end

// JavaScript syntax
function prepareMovie() {
    var displayString = _player.userName + "\n" + _player.organizationName+ "\n" +
_player.serialNumber;
    member("User Info").text = displayString;
}
```

Voir aussi

Lecteur

originalFont

Syntaxe

```
-- Lingo syntax
memberObjRef.originalFont

// JavaScript syntax
memberObjRef.originalFont;
```

Description

Propriété d'acteur police ; renvoie le nom exact de la police d'origine importée lors de la création de l'acteur concerné.

Exemple

L'instruction suivante affiche le nom de la police importée lors de la création de l'acteur 11 :

```
-- Lingo syntax
put(member(11).originalFont)

// JavaScript syntax
put(member(11).originalFont);
```

Voir aussi

recordFont, bitmapSizes, characterSet

originH

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.originH

// JavaScript syntax
memberOrSpriteObjRef.originH;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la coordonnée horizontale du point d'origine d'une animation Flash ou d'une forme vectorielle, en pixels. La valeur peut être exprimée sous forme de nombre à virgule flottante.

Le point d'origine est la coordonnée d'une animation Flash ou d'une forme vectorielle autour de laquelle la mise à l'échelle et la rotation se produisent. Le point d'origine peut être défini avec une précision à virgule flottante à l'aide des propriétés distinctes `originH` et `originV` ou avec une précision en nombre entier au moyen de la propriété unique `originPoint`.

Vous ne pouvez définir la propriété `originH` que si la propriété `originMode` présente la valeur `#point`.

Cette propriété peut être testée et définie. La valeur par défaut est 0.

Remarque : *cette propriété doit avoir la valeur par défaut si la propriété scaleMode a pour valeur #autoSize. Sinon, l'image-objet n'est pas correctement affichée.*

Exemple

Le script d'image-objet suivant utilise la propriété `originMode` pour définir une image-objet d'animation Flash de sorte que son point d'origine soit un point spécifique. Il définit ensuite les points d'origine horizontal et vertical.

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).originMode = #point
    sprite(spriteNum).originH = 100
    sprite(spriteNum).originV = 80
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).originMode = symbol("point");
    sprite(this.spriteNum).originH = 100;
    sprite(this.spriteNum).originV = 80;
}
```

Voir aussi

originV, originMode, originPoint, scaleMode

originMode

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.originMode

// JavaScript syntax
memberOrSpriteObjRef.originMode;
```

Description

Propriété d'acteur et d'image-objet ; définit le point d'origine autour duquel la mise à l'échelle et la rotation se produisent, comme suit :

- `#center` (valeur par défaut) : le point d'origine est au centre de l'animation Flash.
- `#topleft` : le point d'origine est dans l'angle supérieur gauche de l'animation Flash.
- `#point` : le point d'origine est un point spécifié par les propriétés `originPoint`, `originH` et `originV`.

Cette propriété peut être testée et définie.

Remarque : *cette propriété doit avoir la valeur par défaut si la propriété scaleMode a pour valeur #autoSize. Sinon, l'image-objet n'est pas correctement affichée.*

Exemple

Le script d'image-objet suivant utilise la propriété `originMode` pour définir une image-objet d'animation Flash de sorte que son point d'origine soit un point spécifique. Il définit ensuite les points d'origine horizontal et vertical.

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).originMode = #point
    sprite(spriteNum).originH = 100
    sprite(spriteNum).originV = 80
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).originMode = symbol("point");
    sprite(this.spriteNum).originH = 100;
    sprite(this.spriteNum).originV = 80;
}
```

Voir aussi

originH, originV, originPoint, scaleMode

originPoint

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.originPoint

// JavaScript syntax
memberOrSpriteObjRef.originPoint;
```

Description

Propriété d'acteur et d'image-objet ; contrôle le point d'origine autour duquel la mise à l'échelle et la rotation se produisent dans une animation Flash ou une forme vectorielle.

La propriété `originPoint` est spécifiée en tant que valeur d'un point de Director : par exemple, point (100, 200). La définition du point d'origine d'une animation Flash ou d'une forme vectorielle à l'aide de la propriété `originPoint` équivaut à définir séparément les propriétés `originH` et `originV`. Ainsi, la définition de la propriété `originPoint` sur la valeur point(50, 75) revient à définir la propriété `originH` sur 50 et la propriété `originV` sur 75.

Les valeurs de point Director spécifiées pour la propriété `originPoint` doivent correspondre à des nombres entiers, alors que `originH` et `originV` peuvent être spécifiées sous forme de nombres à virgule flottante. Lorsque vous testez la propriété `originPoint`, les valeurs de point sont tronquées de façon à apparaître sous forme de nombres entiers. En règle générale, utilisez les propriétés `originH` et `originV` si vous mettez l'accent sur la précision, et utilisez la propriété `originPoint` si vous recherchez la rapidité et la facilité.

Vous ne pouvez définir la propriété `originPoint` que si la propriété `originMode` présente la valeur `#point`.

Cette propriété peut être testée et définie. La valeur par défaut est 0.

Remarque : *cette propriété doit avoir la valeur par défaut si la propriété scaleMode a pour valeur #autoSize. Sinon, l'image-objet n'est pas correctement affichée.*

Exemple

Le script d'image-objet suivant utilise la propriété `originMode` pour définir une image-objet d'animation Flash de sorte que son point d'origine soit un point spécifique. Il définit ensuite les points d'origine.

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).scaleMode = #showAll
    sprite(spriteNum).originMode = #point
    sprite(spriteNum).originPoint = point(100, 80)
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).scaleMode = symbol("showAll");
    sprite(this.spriteNum).originMode = symbol("point");
    sprite(this.spriteNum).originPoint = point(100, 80);
}
```

Voir aussi

originH, originV, scaleMode

originV

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.originv

// JavaScript syntax
memberOrSpriteObjRef.originV;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la coordonnée verticale en pixels du point d'origine d'une animation Flash ou d'une forme vectorielle autour duquel la mise à l'échelle ou la rotation s'effectue. La valeur peut être exprimée sous forme de nombre à virgule flottante.

Le point d'origine peut être défini avec une précision à virgule flottante à l'aide des propriétés distinctes `originH` et `originV` ou avec une précision en nombre entier au moyen de la propriété unique `originPoint`.

Vous ne pouvez définir la propriété `originV` que si la propriété `originMode` présente la valeur `#point`.

Cette propriété peut être testée et définie. La valeur par défaut est 0.

Remarque : *cette propriété doit avoir la valeur par défaut si la propriété scaleMode a pour valeur #autoSize. Sinon, l'image-objet n'est pas correctement affichée.*

Exemple

Le script d'image-objet suivant utilise la propriété `originMode` pour définir une image-objet d'animation Flash de sorte que son point d'origine soit un point spécifique. Il définit ensuite les points d'origine horizontal et vertical.

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).scaleMode = #showAll
    sprite(spriteNum).originMode = #point
    sprite(spriteNum).originH = 100
    sprite(spriteNum).originV = 80
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).scaleMode = symbol("showAll");
    sprite(this.spriteNum).originMode = symbol("point");
    sprite(this.spriteNum).originH = 100;
    sprite(this.spriteNum).originV = 80;
}
```

Voir aussi

originH, originPoint, scaleMode

orthoHeight

Syntaxe

```
member(whichCastmember).camera(whichCamera).orthoHeight
member(whichCastmember).camera[cameraindex].orthoHeight
sprite(whichSprite).camera.orthoHeight
```

Description

Propriété 3D ; lorsque `camera.projection` présente la valeur `#orthographic`, la valeur `camera.orthoHeight` renvoie le nombre d'unités perpendiculaires de l'univers tenant verticalement dans l'image-objet. Les unités de l'univers sont les unités de mesure de l'univers 3D concerné. Elles sont cohérentes en interne mais sont choisies arbitrairement et, de ce fait, sont susceptibles de varier d'un univers 3D à l'autre.

Vous n'avez pas besoin de spécifier l'index de caméra (*`quelleCaméra`*) pour accéder à la première caméra de l'image-objet.

La valeur par défaut de cette propriété est 200,0.

Exemple

L'instruction suivante attribue à la propriété `orthoHeight` de la caméra de l'image-objet 1 la valeur 200. Cette instruction signifie que 200 unités d'univers logent verticalement dans l'image-objet.

```
-- Lingo syntax
sprite(1).camera.orthoheight = 200.0

// JavaScript syntax
sprite(1).camera.orthoheight = 200.0;
```

Voir aussi

projection

overlay

Syntaxe

```
member(whichCastmember).camera(whichCamera).overlay[overlayIndex].propertyName
member(whichCastmember).camera(whichCamera).overlay.count
```

Description

Propriété 3D de caméra ; permet d'obtenir et de définir l'accès aux propriétés des recouvrements contenus dans la liste des recouvrements de la caméra à afficher. Utilisée comme `overlay.count`, cette propriété renvoie le nombre total de recouvrements (contenus dans la liste des recouvrements de la caméra) à afficher.

Les recouvrements sont des textures affichées devant tous les modèles qui apparaissent dans le frustrum de vue d'une caméra donnée. Les recouvrements sont dessinés dans l'ordre dans lequel ils apparaissent dans la liste de recouvrements de la caméra ; le premier élément de la liste apparaît derrière tous les autres recouvrements, tandis que le dernier se trouve devant.

Chaque recouvrement de la liste de recouvrements de la caméra comporte les propriétés suivantes :

- `loc` permet d'obtenir ou de définir la position spécifique de la propriété `regPoint` du recouvrement par rapport au coin supérieur gauche du cadre de la caméra.
- `source` permet d'obtenir ou de définir la texture utilisée comme image source du recouvrement.
- `scale` permet d'obtenir ou de définir la valeur de l'échelle utilisée par le recouvrement. L'échelle détermine l'agrandissement du recouvrement, la valeur par défaut de cette propriété étant 1,0.
- `rotation` permet d'obtenir ou de définir la rotation du recouvrement, en degrés.
- `regPoint` permet d'obtenir ou de définir le point d'alignement du recouvrement par rapport au coin supérieur gauche de la texture.
- `blend` permet d'obtenir ou de définir la fusion du recouvrement à l'aide d'un nombre entier compris entre 0 et 100 qui définit la transparence (0) ou l'opacité (100) du recouvrement.

Exemple

L'instruction suivante affiche la propriété `scale` du premier recouvrement des recouvrements de la caméra de l'image-objet.

```
-- Lingo syntax
put sprite(2).camera.overlay[1].scale
```

Voir aussi

addOverlay, removeOverlay, bevelDepth

pageHeight

Syntaxe

```
-- Lingo syntax
memberObjRef.pageHeight

// JavaScript syntax
memberObjRef.pageHeight;
```

Description

Propriété d'acteur champ ; renvoie la hauteur, en pixels, de la zone de l'acteur champ visible sur la scène.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante renvoie la hauteur de la portion visible de l'acteur champ Nouvelles du jour :

```
--Lingo syntax
trace(member("Today's News").pageHeight)

// JavaScript syntax
trace(member("Today's News").pageHeight);
```

palette

Syntaxe

```
-- Lingo syntax
memberObjRef.palette

// JavaScript syntax
memberObjRef.palette;
```

Description

Propriété d'acteur ; détermine, pour les acteurs bitmap uniquement, la palette associée à l'acteur spécifié par *quelActeur*.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche la palette affectée à l'acteur Feuilles dans la fenêtre Messages :

```
-- Lingo syntax
put(member("Leaves").palette)

// JavaScript syntax
put(member("Leaves").palette);
```

paletteMapping

Syntaxe

```
-- Lingo syntax
_movie.paletteMapping

// JavaScript syntax
_movie.paletteMapping;
```

Description

Propriété d'animation ; détermine si l'animation convertit (`TRUE`) ou non (`FALSE`, valeur par défaut) la palette des acteurs si celle-ci diffère de la palette en cours de l'animation. Lecture/écriture.

L'effet de cette propriété est similaire à celui obtenu avec la case à cocher Conversion automatique des palettes dans la boîte de dialogue Propriétés de l'animation.

Pour afficher simultanément différents bitmaps avec des palettes différentes, attribuez la valeur `TRUE` à `paletteMapping`. Director analyse la palette de référence de chaque acteur affiché (la palette affectée dans sa boîte de dialogue Propriétés de l'acteur) et, si elle diffère de la palette actuelle, trouve le plus proche équivalent pour chaque pixel dans la nouvelle palette.

Les couleurs du bitmap ne correspondant pas sont proches des couleurs d'origine.

La conversion est assez coûteuse en termes d'utilisation du processeur et il est généralement plus judicieux de régler la palette du bitmap à l'avance.

La conversion peut aussi produire des résultats indésirables. Si la palette change au milieu d'une séquence d'image-objet, le bitmap passe immédiatement à la nouvelle palette et apparaît dans des couleurs incorrectes. Toutefois, si quelque chose rafraîchit l'écran, par exemple, une transition ou une image-objet passant sur la scène, le rectangle affecté de l'écran apparaît dans les nouvelles couleurs.

Exemple

L'instruction suivante demande à l'animation de convertir la palette chaque fois que nécessaire :

```
-- Lingo syntax
_movie.paletteMapping = TRUE

// JavaScript syntax
_movie.paletteMapping = true;
```

Voir aussi

Animation

paletteRef

Syntaxe

```
member(whichCastMember). paletteRef
the paletteRef
```

Description

Propriété d'acteur bitmap ; détermine la palette associée à un acteur bitmap. Les palettes intégrées de Director sont indiquées par des symboles (`#systemMac`, `#rainbow`, etc.). Les acteurs palettes sont considérés comme des références d'acteurs. Le comportement de cette propriété diffère de celui de la propriété d'acteur `palette`, qui renvoie un nombre positif pour les palettes de distribution et un nombre négatif pour les palettes intégrées de Director.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affecte la palette système du Mac à l'acteur bitmap Coquille :

```
member("Shell").paletteRef = #systemMac
```

pan

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.pan

// JavaScript syntax
soundChannelObjRef.pan;
```

Description

Propriété de piste audio ; indique la balance gauche/droite du son en cours de lecture dans une piste audio. Lecture/écriture.

La plage de valeurs est comprise entre -100 et 100. La valeur -100 indique que seule la piste gauche est lue. La valeur 100 indique la lecture de la piste droite uniquement. La valeur 0 indique une balance gauche/droite, entraînant le centrage du son. Pour les sons mono, `pan` affecte le haut-parleur (gauche ou droite) par lequel passe le son.

Vous pouvez modifier la balance d'un son à tout moment, mais si la piste audio exécute un fondu, la définition de la balance ne prendra effet qu'après l'exécution du fondu.

Vous pouvez voir un exemple d'utilisation de `pan` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

Les instructions suivantes équilibrent le son de la piste audio 2 de la piste gauche vers la piste droite :

```
-- Lingo syntax
repeat with x = -100 to 100
 sound(2).pan = x
end repeat

// JavaScript syntax
for (var x = -100; x <= 100; x++) {
 sound(2).pan = x;
}
```

Voir aussi

Piste audio

pan (propriété QTVR)

Syntaxe

```
-- Lingo syntax
spriteObjRef.pan

// JavaScript syntax
spriteObjRef.pan;
```

Description

Propriété d'image-objet QuickTime VR ; le panoramique actuel de l'animation QuickTime VR. La valeur est exprimée en degrés.

Cette propriété peut être testée et définie.

paragraph

Syntaxe

```
chunkExpression.paragraph[whichParagraph]
chunkExpression.paragraph[firstParagraph..lastParagraph]
```

Description

Propriété d'acteur texte ; cette expression de sous-chaîne permet d'accéder à différents paragraphes dans un acteur texte.

Le paragraphe est délimité par un retour de chariot.

```
put member("AnimText").paragraph[3]
```

Exemple

L'instruction suivante renvoie le second paragraphe de l'acteur texte monTexte.

```
-- Lingo syntax
put member("myText").paragraph[2]
```

Voir aussi

`line...of`

parent

Syntaxe

```
member(whichCastmember).model(whichModel).parent
member(whichCastmember).camera(whichCamera).parent
member(whichCastmember).light(whichLight).parent
member(whichCastmember).group(whichGroup).parent
```

Description

Propriété 3D ; utilisée avec une référence de modèle, caméra, lumière ou groupe, cette propriété permet d'obtenir ou de définir le nœud parent de l'objet référencé. Le nœud parent peut être tout autre objet de modèle, de caméra, de lumière ou de groupe.

La propriété `transform` d'un objet définit son échelle, sa position et son orientation par rapport à son objet parent.

La définition de la propriété parent d'un objet sur la valeur `VOID` équivaut à supprimer l'objet de l'univers à l'aide de la commande `removeFromWorld()`.

La définition de la propriété parent d'un objet sur la valeur de l'objet de groupe Univers (`group("World")`) équivaut à ajouter un objet à l'univers à l'aide de la commande `addToWorld()`.

Vous pouvez également modifier la valeur de cette propriété à l'aide de la commande `addChild`.

Exemple

L'instruction suivante définit la propriété `parent` du modèle Pneu. Son parent est le modèle Véhicule.

```
-- Lingo syntax
member("3Dobjects").model("Tire").parent = member("3Dobjects").model("Car")
```

```
// JavaScript syntax
member("3Dobjects").getPropRef("model",2).parent =
member("3Dobjects").getPropRef("model",3);
```

Voir aussi

child (3D), addChild

password

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.password

// JavaScript syntax
memberOrSpriteObjRef.password;
```

Description

Propriété d'acteur et image-objet RealMedia ; permet de définir le mot de passe nécessaire à l'accès à un flux RealMedia protégé. Vous ne pouvez pas utiliser cette propriété pour récupérer un mot de passe spécifié auparavant. Si aucun mot de passe n'a encore été défini, la valeur de cette propriété est la chaîne `"********"`. Si aucun mot de passe n'a encore été défini, la valeur de cette propriété est une chaîne vide.

Exemple

Les exemples suivants indiquent que le mot de passe a été défini pour le flux RealMedia de l'acteur Real ou de l'image-objet 2.

```
-- Lingo syntax
put(sprite(2).password) -- "********"
put(member("Real").password) -- "********"

// JavaScript syntax
put(sprite(2).password); // "********"
put(member("Real").password); // "********"
```

Les exemples suivants indiquent que le mot de passe n'a jamais été défini pour le flux RealMedia de l'acteur Real ou de l'image-objet 2.

```
-- Lingo syntax
put(sprite(2).password) -- ""
put(member("Real").password) -- ""

// JavaScript syntax
put(sprite(2).password); // ""
put(member("Real").password); // ""
```

Les exemples suivants définissent le mot de passe pour le flux RealMedia de l'image-objet 2 et l'acteur Real sur `abracadabra`.

```
-- Lingo syntax
sprite(2).password = "abracadabra"
member("Real").password = "abracadabra"

// JavaScript syntax
sprite(2).password = "abracadabra";
member("Real").password = "abracadabra";
```

Voir aussi

userName (RealMedia)

path (animation)

Syntaxe

```
-- Lingo syntax
_movie.path

// JavaScript syntax
_movie.path;
```

Description

Propriété d'animation ; indique le chemin d'accès du dossier de l'animation actuelle. Lecture seule.

Pour les noms de fichier fonctionnant sous Windows et sur Mac, utilisez l'opérateur de chemin d'accès @.

Vous pouvez voir un exemple d'utilisation de `path` dans une animation en consultant l'animation Read and Write Text du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante affiche le chemin d'accès du dossier de l'animation actuelle :

```
-- Lingo syntax
trace(_movie.path)

// JavaScript syntax
trace(_movie.path);
```

L'instruction suivante lit le fichier son Violon.aif stocké dans le sous-dossier Sons du dossier de l'animation en cours :

```
-- Lingo syntax
sound(1).playFile(_movie.path & "Sounds\Crash.aif")

// JavaScript syntax
sound(1).playFile(_movie.path + "Sounds\\Crash.aif");
```

Voir aussi

Animation

path (3D)

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).emitter.path
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir ou de définir la propriété `path` de l'émetteur de particules de la ressource.

Cette propriété est une liste de vecteurs qui définit le chemin suivi par les particules tout au long de leur existence. La valeur par défaut de cette propriété est une liste vide `[]`.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. L'instruction suivante spécifie que les particules de systèmeThermique suivront le chemin décrit par la liste de vecteurs.

```
member("Fires").modelResource("ThermoSystem").emitter.path = [vector(0,0,0),
vector(15,0,0), vector(30,30,-10)]
```

Voir aussi

`pathStrength`, `emitter`

pathName (acteur Flash)

Syntaxe

```
member(whichFlashMember).pathName
the pathName of member whichFlashMember
```

Description

Propriété d'acteur ; contrôle l'emplacement d'un fichier externe dans lequel sont stockés les éléments d'un acteur animation Flash. Vous pouvez lier une animation Flash à un chemin d'accès sur un disque local ou réseau ou vers une URL.

La définition du chemin d'accès d'un acteur non lié le convertit en un acteur lié.

Cette propriété peut être testée et définie. La propriété `pathName` d'un acteur non lié est une chaîne vide.

Cette propriété est identique à la propriété `fileName` pour d'autres types d'acteurs, `fileName` étant utilisable à la place de `pathName`.

Exemple

L'instruction suivante définit la propriété `pathName` de l'acteur sur l'emplacement d'une animation Flash sur le Web.

```
-- Lingo syntax
member("FlashObj").pathName = "http://www.someURL.com/myFlash.swf"

// JavaScript syntax
member("FlashObj").pathName = "http://www.someURL.com/myFlash.swf";
```

Voir aussi

`fileName (acteur)`, `linked`

pathStrength

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).emitter.pathStrength
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#particle`,, cette propriété détermine la façon dont les particules suivent le chemin spécifié par la propriété `path` de l'émetteur. Sa valeur peut être comprise entre 0,0 (aucune force – les particules ne seront pas attirées par le chemin) et l'infini. Sa valeur par défaut est 0,1. La définition de la propriété `pathStrength` sur la valeur 0,0 se révèle utile pour désactiver temporairement le chemin.

Plus la valeur de `pathStrength` est élevée, plus le système de particules est rigide. Des valeurs `pathStrength` élevées entraînent le rebondissement très rapide des particules, à moins qu'une force modératrice soit également utilisée, telle que la propriété de particule `drag`.

Cette propriété peut être testée et définie.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. L'instruction suivante attribue à la propriété `pathStrength` de systèmeThermique la valeur 0,97. Si un chemin est indiqué par la propriété `emitter.path` de systèmeThermique, les particules suivront ce chemin de très près.

```
member("Fires").modelResource("ThermoSystem").emitter.pathStrength = 0.97
```

Voir aussi

path (3D), emitter

pattern

Syntaxe

```
member(whichCastMember).pattern
the pattern of member whichCastMember
```

Description

Propriété d'acteur ; détermine le motif associé à la forme spécifiée. Les valeurs possibles sont les nombres correspondant aux échantillons de la palette des motifs de la fenêtre Outils. Si l'acteur forme est vide, le motif s'applique à son bord externe.

Cette propriété peut être utile dans les animations comportant du contenu Shockwave pour changer des images en modifiant la mosaïque appliquée à une forme, ce qui permet d'économiser la mémoire requise par des bitmaps plus volumineux.

Cette propriété peut être testée et définie.

Exemple

Les instructions suivantes font de l'acteur forme maForme une forme remplie et lui affectent le motif 1 qui est une couleur unie.

```
-- Lingo syntax
member("myShape").filled = TRUE
member("myShape").pattern = 1

// JavaScript syntax
member("myShape").filled = 1;
member("myShape").pattern = 1;
```

pausedAtStart (Flash, vidéo numérique)

Syntaxe

```
member(whichFlashOrDigitalVideoMember).pausedAtStart
the pausedAtStart of member whichFlashOrDigitalVideoMember
```

Description

Propriété d'acteur ; contrôle si la vidéo numérique ou l'animation Flash est lue lorsqu'elle apparaît sur la scène. Si cette propriété présente la valeur TRUE, la vidéo numérique ou l'animation Flash n'est pas lue lorsqu'elle apparaît. Si elle présente la valeur FALSE, la vidéo numérique ou l'animation Flash est lue dès qu'elle apparaît.

Pour un acteur vidéo numérique, la propriété spécifie si la case En pause de la boîte de dialogue Propriétés de l'acteur vidéo numérique est cochée ou non.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante coche la case En pause dans la boîte de dialogue Propriétés de l'acteur vidéo numérique pour l'animation QuickTime Chaise pivotante :

```
member("Rotating Chair").pausedAtStart = TRUE
```

pausedAtStart (RealMedia, Windows Media)

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.pausedAtStart

// JavaScript syntax
memberOrSpriteObjRef.pausedAtStart;
```

Description

Propriété d'acteur ou d'image-objet RealMedia et Windows Media ; permet d'obtenir ou de définir si la lecture d'un flux RealMedia ou Windows Media présent sur la scène démarre automatiquement à la fin de la mise en tampon (FALSE ou 0) ou non (TRUE ou 1). Lecture/écriture.

Cette propriété peut être définie sur une expression renvoyant la valeur TRUE ou FALSE. Les valeurs entières autres que 1 ou 0 sont considérées comme TRUE. La valeur par défaut de cette propriété est FALSE. Vous pouvez définir cette propriété sur TRUE en sélectionnant En pause dans l'affichage graphique de l'Inspecteur des propriétés.

Si cette propriété est définie sur FALSE, vous devez cliquer sur le bouton Lire dans la fenêtre RealMedia ou Windows Media (ou sur un bouton que vous avez créé dans ce but dans votre animation)ou appeler la méthode `play()` pour lire l'image-objet une fois la mise en tampon effectuée.

Cette propriété n'affecte que la lecture basée sur le scénario, et non la lecture dans la fenêtre RealMedia ou Windows Media.

Exemple

Les exemples suivants indiquent que la propriété `pausedAtStart` de l'image-objet 2 et de l'acteur Real présente la valeur FALSE, ce qui signifie que le flux RealMedia est automatiquement lu à la fin de la mise en tampon.

```
-- Lingo syntax
put(sprite(2).pausedAtStart) -- 0
put(member("Real").pausedAtStart) -- 0

// JavaScript syntax
put(sprite(2).pausedAtStart); // 0
put(member("Real").pausedAtStart); // 0
```

Les exemples suivants définissent la propriété `pausedAtStart` de l'image-objet 2 et de l'acteur Real sur `TRUE`, ce qui signifie que le flux RealMedia n'est pas lu tant que la commande play n'a pas été appelée.

```
-- Lingo syntax
sprite(2).pausedAtStart = TRUE
member("Real").pausedAtStart = TRUE

// JavaScript syntax
sprite(2).pausedAtStart = 1;
member("Real").pausedAtStart = 1;
```

L'exemple suivant utilise la propriété `pausedAtStart` pour mettre en tampon une image-objet RealMedia hors de la scène, puis la lire sur la scène une fois la mise en tampon effectuée. Dans cet exemple, la propriété `pausedAtStart` de l'acteur RealMedia est définie sur `TRUE`. Une occurrence de cet acteur est située en dehors de la scène, sur la piste d'image-objet 1. Le script d'image suivant doit être placé dans la plage de l'image-objet :

```
-- Lingo syntax
on exitFrame me
    if sprite(1).state > 3 then -- check to see if buffering is complete
        sprite(1).locH = 162
        sprite(1).locV = 118
        sprite(1).play() -- position and play the sprite
    end if
end

// JavaScript syntax
function exitFrame() {
    var st = sprite(1).state;
    if (st > 3) { // check to see if buffering is complete
        sprite(1).locH = 162;
        sprite(1).locV = 118;
        sprite(1).play(); // position and play the sprite
    }
}
```

L'image-objet RealMedia est mise en tampon en dehors de la scène, puis apparaîtra sur la scène et est lue dès que la mise en tampon a été effectuée.

percentBuffered

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.percentBuffered

// JavaScript syntax
memberOrSpriteObjRef.percentBuffered;
```

Description

Propriété d'acteur ou d'image-objet RealMedia ; renvoie le pourcentage du tampon rempli par le flux RealMedia chargé à partir d'un fichier local ou du serveur. Lorsque cette propriété atteint 100, la mémoire tampon est pleine et la lecture du flux RealMedia démarre si la propriété `pausedAtStart` n'est pas définie sur `TRUE`. Cette propriété est dynamique en cours de lecture et ne peut pas être définie.

La mémoire tampon est un type de mémoire cache qui contient la partie de l'animation qui va être lue, généralement les premières secondes. Le flux entre en mémoire tampon lors de sa lecture en flux continu dans RealPlayer et quitte la mémoire tampon dès le démarrage de la lecture du clip. La mémoire tampon permet de visualiser le contenu sans

télécharger l'ensemble du fichier, permet d'éviter les congestions sur le réseau et empêche que des défaillances de disponibilité de la bande passante interrompent la lecture.

Le processus de mise en mémoire tampon est initialisé par la commande `play` et la lecture de la portion du flux comprise dans la mémoire tampon démarre une fois la mémoire tampon remplie à 100 %. Le processus de mise en mémoire tampon demandant quelques secondes, il existe un certain délai entre l'appel de la commande `play` et le début réel de la lecture du flux. L'utilisation de la commande `pausedAtStart` permet de lire le flux en dehors de la scène pendant le processus de mise en mémoire tampon, puis d'afficher le flux sur la scène lorsque la lecture démarre. Pour plus d'informations, reportez-vous à l'exemple spécifié pour l'entrée « pausedAtStart (RealMedia, Windows Media) », page 957.

Exemple

Les exemples suivants indiquent que 56 % du flux RealMedia de l'image-objet 2 et de l'acteur Real a été mis en tampon.

```
-- Lingo syntax
put(sprite(2).percentBuffered) -- 56
put(member("Real").percentBuffered) -- 56

// JavaScript syntax
put(sprite(2).percentBuffered); // 56
put(member("Real").percentBuffered); // 56
```

Voir aussi

mediaStatus (RealMedia, Windows Media), pausedAtStart (RealMedia, Windows Media), state (RealMedia)

percentPlayed

Syntaxe

```
member(whichCastMember).percentPlayed
the percentPlayed of member whichCastMember
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; renvoie le pourcentage du fichier SWA spécifié qui a été lu.

Cette propriété ne peut être testée qu'une fois la lecture du son SWA déclenchée ou qu'une fois le son préchargé à l'aide de la commande `preLoadBuffer`. Cette propriété ne peut pas être définie.

Exemple

Le gestionnaire suivant affiche le pourcentage de l'acteur SWA Frank Sinatra lu en flux continu et place cette valeur dans l'acteur champ Pourcentage lu :

```
on exitFrame
    whatState = member("Frank Sinatra").state
    if whatState > 1 AND whatState < 9 then
        member("Percent Played").text = string(member("Frank Sinatra").percentPlayed)
    end if
end
```

Voir aussi

percentStreamed (acteur)

percentStreamed (3D)

Syntaxe

```
member(whichCastMember).percentStreamed
```

Description

Propriété 3D ; permet d'obtenir le pourcentage d'un acteur 3D qui a été chargé en mémoire. Cette propriété indique la quantité qui a été chargée du fichier initial ou du dernier fichier demandé. La valeur renvoyée est un nombre entier compris entre 0 et 100. Il n'existe aucune valeur par défaut pour cette propriété.

Exemple

L'instruction suivante indique que le chargement de l'acteur séquenceDeFête est terminé.

```
put member("PartyScene").percentStreamed
-- 100
```

percentStreamed (acteur)

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.percentStreamed

// JavaScript syntax
memberOrSpriteObjRef.percentStreamed;
```

Description

Propriété d'acteur Shockwave Audio (SWA) et Flash, et propriété d'image-objet QuickTime.

Pour les sons SWA en flux continu, prend la valeur du pourcentage d'un fichier SWA déjà lu en flux continu d'un serveur HTTP ou FTP. Pour les sons SWA, cette propriété diffère de la propriété `percentPlayed` par le fait qu'elle indique la quantité du fichier mise en tampon mais non encore lue. Cette propriété ne peut être testée qu'une fois la lecture du son SWA déclenchée ou qu'une fois le son préchargé à l'aide de la commande `preLoadBuffer`.

Pour les acteurs animation Flash, cette propriété a pour valeur le pourcentage d'une animation Flash lue en flux continu dans la mémoire.

Pour les images-objets QuickTime, cette propriété prend la valeur du pourcentage du fichier QuickTime lu.

Cette propriété est une valeur comprise entre 0 et 100 %. Pour un fichier situé sur un disque local, la valeur est de 100. Pour les fichiers lus en continu depuis Internet, la valeur `percentStreamed` augmente à mesure que les octets sont reçus. Cette propriété ne peut pas être définie.

Exemple

L'exemple suivant indique le pourcentage de l'acteur SWA Ray Charles lu en flux continu déjà récupéré et l'affiche dans un champ :

```
-- Lingo syntax
on exitFrame
    whatState = member("Ray Charles").state
    if whatState > 1 AND whatState < 9 then
        member("Percent Streamed Displayer").text = string(member("Ray
Charles").percentStreamed)
```

```
    end if
end

// JavaScript syntax
function exitFrame() {
    var whatState = member("Ray Charles").state;
    var pcStm = new String(member("Ray Charles").percentStreamed);
    if (whatState > 1 && whatState < 9) {
        member("Percent Streamed Displayer").text = pcStm;
    }
}
```

Le script d'image suivant impose une boucle à la tête de lecture dans l'image en cours tant que moins de 60 % de l'animation Flash appelée Ecran d'accueil ont été lus en flux continu dans la mémoire :

```
-- Lingo syntax
on exitFrame
    if member("Splash Screen").percentStreamed < 60 then
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var ssStrm = member("Splash Screen").percentStreamed;
    if (ssStrm < 60) {
        _movie.go(_movie.frame);
    }
}
```

Voir aussi

percentPlayed

period

Syntaxe

```
timeoutObject.period
```

Description

Propriété d'objet ; nombre de millisecondes compris entre les événements de temporisation transmis par `objetDeTemporisation` au gestionnaire de temporisation.

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire `timeout` suivant diminue la valeur `period` de temporisation d'une seconde à chaque appel, jusqu'à ce qu'une période minimale de 2 secondes (2 000 millisecondes) soit atteinte :

```
on handleTimeout timeoutObject
    if timeoutObject.period > 2000 then
        timeoutObject.period = timeoutObject.period - 1000
    end if
end handleTimeout
```

Voir aussi

name (temporisation), persistent, target, time (objet de temporisation), timeout(), timeoutHandler, timeoutList

persistent

Syntaxe

```
timeoutObject.persistent
```

Description

Propriété d'objet ; détermine si *objetDeTemporisation* est supprimé de la liste `timeoutList` lors de l'arrêt de la lecture de l'animation en cours. Si la valeur est `TRUE`, *objetDeTemporisation* reste actif. Si la valeur est `FALSE`, l'objet de temporisation est supprimé lors de l'arrêt de l'animation. La valeur par défaut est `FALSE`.

La définition de cette propriété sur `TRUE` permet à un objet de temporisation de continuer à générer des événements de temporisation dans d'autres animations. Cette opération peut se révéler utile lorsqu'une animation est orientée vers une autre animation par l'intermédiaire de la commande `go to movie`.

Exemple

L'instruction suivante crée un objet de temporisation et le définit comme persistant.

```
-- Lingo syntax
gTO = timeout().new("test",50000,"sampleTimeout",0)
gTO.persistent = TRUE

// JavaScript syntax
_global.gTO = new timeout("test",50000,"sampleTimeout",0)
_global.gTO.persistent = 1;
```

Voir aussi

name (temporisation), period, target, time (objet de temporisation), timeout(), timeoutHandler, timeoutList

picture (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.picture

// JavaScript syntax
memberObjRef.picture;
```

Description

Propriété d'acteur ; détermine l'image associée à un acteur bitmap, texte ou PICT. Pour mettre à jour les modifications du point d'alignement d'un acteur ou les modifications d'une image après l'avoir lié(e) de nouveau à l'aide de la propriété `fileName`, utilisez l'instruction suivante :

```
member(whichCastMember).picture = member(whichCastMember).picture
```

où vous remplacez *quelActeur* par le nom ou le numéro de l'acteur concerné.

Les modifications effectuées sur les acteurs étant conservées en RAM, cette propriété trouve une meilleure utilisation au cours de la phase de programmation. Evitez de l'utiliser dans les projections.

Cette propriété peut être testée et modifiée.

Exemple

L'instruction suivante associe la variable `pictHolder` à l'image de l'acteur Crépuscule :

```
-- Lingo syntax
pictHolder = member("Sunset").picture

// JavaScript syntax
var pictHolder = member("Sunset").picture;
```

Voir aussi

type (image-objet)

picture (fenêtre)

Syntaxe

```
-- Lingo syntax
windowObjRef.picture

// JavaScript syntax
windowObjRef.picture;
```

Description

Propriété de fenêtre ; permet d'obtenir un aperçu du contenu en cours d'une fenêtre (la fenêtre Scène ou une animation dans une fenêtre). Lecture seule.

Vous pouvez appliquer les données du bitmap résultant à un bitmap existant ou les utiliser pour en créer un nouveau.

Si aucun aperçu n'existe, cette propriété renvoie la valeur `VOID` (Lingo) ou `null` (syntaxe JavaScript).

Exemple

L'instruction suivante saisit le contenu actuel de la scène et le place dans un acteur bitmap :

```
-- Lingo syntax
member("Stage image").picture = _movie.stage.picture

// JavaScript syntax
member("Stage image").picture = _movie.stage.picture;
```

Voir aussi

Fenêtre

platform

Syntaxe

```
the platform
```

Description

Propriété système ; indique le type de plate-forme pour lequel la projection a été créée.

Cette propriété peut être testée, mais pas définie.

Les valeurs possibles sont :

Valeur possible	Plate-forme correspondante
`Mac,PowerPC`	PowerPC Mac
`Windows,32`	Windows 95 ou Windows NT

Pour assurer une compatibilité future et permettre l'addition de valeurs, il est préférable de tester la plate-forme au moyen de `contains`.

Exemple

L'instruction suivante renvoie la plate-forme pour laquelle l'animation a été créée.

```
-- Lingo syntax
if the platform contains "Windows,32" then
    alert ("This movie has been created using Windows")
end if
```

Voir aussi

runMode

playBackMode

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.playBackMode

// JavaScript syntax
memberOrSpriteObjRef.playBackMode;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la cadence d'un acteur animation Flash ou d'un acteur GIF animé selon les valeurs suivantes :

- `#normal` (valeur par défaut) – Lit l'animation Flash ou le fichier GIF à une cadence aussi proche que possible de la cadence originale.
- `#lockStep` : lit l'animation Flash ou le fichier GIF à la cadence de l'animation Director.
- `#fixed` : lit l'animation Flash ou le fichier GIF à la cadence spécifiée par la propriété `fixedRate`.

Cette propriété peut être testée et définie.

Exemple

Le script d'image-objet suivant définit la cadence d'une image-objet d'animation Flash comme étant celle de l'animation Director :

```
-- Lingo syntax
property spriteNum

on beginSprite(me)
    sprite(spriteNum).playBackMode = #lockStep
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).playBackMode = symbol("lockStep");
}
```

Voir aussi

fixedRate

playing

Syntaxe

```
-- Lingo syntax
spriteObjRef.playing

// JavaScript syntax
spriteObjRef.playing;
```

Description

Propriété d'image-objet Flash ; indique si une animation Flash est en cours de lecture (TRUE) ou arrêtée (FALSE).

Cette propriété peut être testée, mais pas définie.

Exemple

Le script d'image suivant détermine si l'image-objet animation Flash de la piste 5 est en cours de lecture. Dans la négative, il démarre l'animation :

```
-- Lingo syntax
on enterFrame
    if not sprite(5).playing then
        sprite(5).play()
    end if
end

// JavaScript syntax
function enterFrame() {
    var plg = sprite(5).playing;
    if (plg == 0) {
        sprite(5).play();
    }
}
```

playing (3D)

Syntaxe

```
member(whichCastmember).model(whichModel).keyframePlayer.playing
member(whichCastmember).model(whichModel).bonesPlayer.playing
```

Description

Propriété 3D de modificateur `#keyframePlayer` et `#bonesPlayer` ; indique si le moteur de lecture d'animation du modificateur est en marche (`TRUE`) ou en pause (`FALSE`).

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante indique que le moteur de lecture d'animation `#keyframePlayer` du modèle Martien3 est actuellement en marche.

```
put member("newaliens").model("Alien3").keyframePlayer.playing
-- 1
```

Voir aussi

`play() (3D)`, `pause() (3D)`, `playlist`, `queue() (3D)`

playlist

Syntaxe

```
member(whichCastmember).model(whichModel).keyframePlayer.playlist
member(whichCastmember).model(whichModel).bonesPlayer.playlist
```

Description

Propriété 3D de modificateur `#keyframePlayer` et `#bonesPlayer` ; renvoie une liste linéaire de listes de propriétés, chacune d'elles représentant un mouvement dont la lecture est mise en attente par le modificateur.

Chaque liste de propriétés doit contenir les propriétés suivantes :

- `#name` est le nom du mouvement à lire.
- `#loop` indique si le mouvement doit être lu en boucle.
- `#startTime` est le moment, en millisecondes, où la lecture de l'animation doit commencer.
- `#endTime` est le moment, en millisecondes, où la lecture de l'animation se termine ou le moment où le mouvement doit être mis en boucle. Une valeur négative indique que le mouvement doit être lu jusqu'à la fin.
- `#scale` indique le taux de lecture du mouvement qui doit être multiplié par la propriété playRate du modificateur pour déterminer la cadence réelle de la lecture du mouvement.

La propriété `playlist` peut être testée mais non définie. Utilisez les commandes `queue()`, `play()`, `playNext()` et `removeLast()` pour la manipuler.

Exemple

L'instruction suivante affiche les mouvements mis en attente pour le modèle Promeneur dans la fenêtre Messages.

```
-- Lingo syntax
put member("3Dobjects").model("Stroller").bonesPlayer.playList
```

```
// JavaScript syntax
put(member("3Dobjects").getPropRef("model",2).bonesPlayer.playList);
```

Voir aussi

play() (3D), playNext() (3D), removeLast(), queue() (3D)

playRate (3D)

Syntaxe

```
member(whichCastmember).model(whichModel).bonesPlayer.playRate
member(whichCastmember).model(whichModel).keyframePlayer.playRate
```

Description

Propriété 3D de modificateur `#keyframePlayer` et `#bonesPlayer` ; multiplicateur d'échelle pour la position locale des mouvements lorsqu'ils sont lus. Cette propriété ne détermine que partiellement la cadence à laquelle les mouvements sont exécutés par le modèle.

La lecture d'un mouvement par un modèle est le résultat d'une commande `play()` ou `queue()`. Le paramètre `scale` de la commande `play()` ou `queue()` est multiplié par la propriété `playRate` du modificateur et la valeur résultante est la cadence à laquelle le mouvement en question est lu.

Exemple

L'instruction suivante attribue à la propriété `playRate` du modificateur `keyframePlayer` du modèle martienVert la valeur 3 :

```
member("newAliens").model("GreenAlien").keyframePlayer.playRate = 3
```

Voir aussi

play() (3D), queue() (3D), playlist, currentTime (3D)

playRate (DVD)

Syntaxe

```
-- Lingo syntax
dvdObjRef.playRate

// JavaScript syntax
dvdObjRef.playRate;
```

Description

Propriété de DVD ; indique la cadence de lecture d'un DVD vers l'avant ou vers l'arrière depuis l'emplacement en cours. Lecture/écriture.

Une valeur négative lit le DVD vers l'arrière, et une valeur positive lit le DVD vers l'avant.

Voir aussi

DVD

playRate

Syntaxe

```
-- Lingo syntax
spriteObjRef.playRate

// JavaScript syntax
spriteObjRef.playRate;
```

Description

Propriété d'image-objet vidéo numérique ; contrôle la vitesse de lecture de la vidéo numérique dans une piste spécifique. movieRate est une valeur qui dirige la lecture d'une vidéo numérique. Une valeur de 1 spécifie une lecture normale vers l'avant, une valeur de -1 spécifie une lecture en arrière, et une valeur de 0 spécifie l'arrêt de la lecture. Vous pouvez utiliser des valeurs plus ou moins élevées. Par exemple, une valeur de 0,5 provoque une lecture plus lente que la normale. Toutefois, il est possible que certaines images soient ignorées lorsque la valeur de la propriété d'image-objet `playRate` dépasse 1. La quantité d'images ignorées dépend d'autres facteurs, tels que les performances de l'ordinateur sur lequel l'animation est lue ou de l'étirement possible de l'image-objet vidéo numérique.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante définit une vitesse de lecture normale de la vidéo numérique contenue dans l'image-objet vidéo numérique de la piste 9 :

```
-- Lingo syntax
sprite(9).playRate = 1

// JavaScript syntax
sprite(9).playRate = 1;
```

L'instruction suivante entraîne la lecture en sens inverse de la vidéo numérique dans la piste d'image-objet 9 :

```
-- Lingo syntax
sprite(9).playRate = -1

// JavaScript syntax
sprite(9).playRate = -1;
```

Voir aussi

duration (acteur), currentTime (QuickTime, AVI)

playRate (Windows Media)

Syntaxe

```
-- Lingo syntax
windowsMediaObjRef.playRate

// JavaScript syntax
windowsMediaObjRef.playRate;
```

Description

Propriété Windows Media. Détermine la cadence de lecture d'un acteur Windows Media. Lecture seule.

Exemple

L'instruction suivante affiche dans la fenêtre Messages la cadence de lecture de l'acteur 10 :

```
-- Lingo syntax
trace(member(10).playRate)

// JavaScript syntax
trace(member(10).playRate);
```

Voir aussi

Windows Media

pointAtOrientation

Syntaxe

```
member(whichCastmember).model(whichModel).pointAtOrientation
member(whichCastmember).group(whichGroup).pointAtOrientation
member(whichCastmember).light(whichLight).pointAtOrientation
member(whichCastmember).camera(whichCamera).pointAtOrientation
```

Description

Propriété 3D de modèle, de lumière, de groupe et de caméra ; permet d'obtenir ou de définir la façon dont l'objet référencé répond à la commande `pointAt`. Cette propriété est une liste linéaire de deux vecteurs relatifs à l'objet, le premier définissant la direction avant de l'objet et le second la direction vers le haut de l'objet.

Les directions avant et vers le haut de l'objet n'ont pas besoin d'être perpendiculaires, mais ne doivent pas être parallèles.

Exemple

L'instruction suivante affiche le vecteur de direction avant et le vecteur vertical relatif à l'objet du modèle bip01 :

```
put member("scene").model("bip01").pointAtOrientation
-- [vector(0.0000, 0.0000, 1.0000), vector(0.0000, 1.0000, 0.0000)]
```

Voir aussi

pointAt

pointOfContact

Syntaxe

```
collisionData.pointOfContact
```

Description

Propriété 3D `collisionData` ; renvoie un vecteur décrivant le point de contact d'une collision entre deux modèles.

L'objet `collisionData` est envoyé comme argument avec les événements `#collideWith` et `#collideAny` au gestionnaire spécifié dans les commandes `registerForEvent`, `registerScript` et `setCollisionCallback`.

Les événements `#collideWith` et `#collideAny` sont envoyés en cas de collision entre deux modèles associés à des modificateur de collision. La propriété `resolve` des modificateurs des modèles doit présenter la valeur `TRUE`.

Cette propriété peut être testée, mais pas définie.

Exemple

L'exemple suivant est constitué de deux parties. La première partie constitue la première ligne de code, qui enregistre le gestionnaire #explose pour l'événement `#collideAny`. La seconde partie correspond au gestionnaire `#explose`. Lorsque deux modèles de l'acteur maSéquence entrent en collision, le gestionnaire `#explose` est appelé et l'argument `collisionData` lui est envoyé. Les neuf premières lignes du gestionnaire `#explose` créent la ressource de modèle sourceDétincelles et en définissent les propriétés. Cette ressource de modèle est une simple explosion de particules. La dixième ligne du gestionnaire crée un modèle modèleDétincelles à l'aide de la ressource de modèle sourceDétincelles. La dernière ligne du gestionnaire définit la position de modèleDétincelles à l'emplacement de la collision. L'effet général est une explosion d'étincelles causée par une collision.

```
member("MyScene").registerForEvent(#collideAny, #explode, 0)

on explode me, collisionData
    nmr = member("MyScene").newModelResource("SparkSource",#particle)
    nmr.emitter.mode = #burst
    nmr.emitter.loop = 0
    nmr.emitter.minSpeed = 30
    nmr.emitter.maxSpeed = 50
    nmr.emitter.direction = vector(0, 0, 1)
    nmr.colorRange.start = rgb(0, 0, 255)
    nmr.colorRange.end = rgb(255, 0, 0)
    nmr.lifetime = 5000
    nm = member("MyScene").newModel("SparksModel", nmr)
    nm.transform.position = collisionData.pointOfContact
end
```

Voir aussi

`modelA`, `modelB`

position (transformation)

Syntaxe

```
member(whichCastmember).node(whichNode).transform.position
member(whichCastmember).node(whichNode).getWorldTransform().position
transform.position
```

Description

Propriété 3D ; permet d'obtenir ou de définir le composant de position d'une transformation. Une transformation définit une échelle, une position et une rotation au sein d'un cadre de référence donné. La valeur par défaut de cette propriété est `vector(0,0,0)`.

Un nœud peut être un objet de caméra, groupe, lumière ou modèle. La définition de la `position` d'une transformation de nœud définit la position de cet objet dans le cadre de référence de la transformation. La définition de la propriété position de la transformation d'un objet par rapport à l'univers à l'aide de `getWorldTransform().position` définit la position de l'objet par rapport à l'origine de l'univers. La définition de la propriété position de la transformation d'un objet par rapport à son parent à l'aide de `transform.position` définit la position de l'objet par rapport à son nœud parent.

La propriété `worldPosition` d'un objet de modèle, de lumière, de caméra ou de groupe, est un raccourci de la version `getWorldTransform().position` de cette propriété pour cet objet.

Exemple

L'instruction suivante affiche la position relative au parent du modèle Sphère01.

```
-- Lingo syntax
    put(member("3Dobjects").model("Sphere01").transform.position)

// JavaScript syntax
    put(member("3Dobjects").getPropRef("model",2).transform.position);
```

Voir aussi

transform (propriété), getWorldTransform(), rotation (transformation), scale (transformation)

positionReset

Syntaxe

```
member(whichCastmember).model(whichModel).bonesPlayer.positionReset
member(whichCastmember).model(whichModel).keyframePlayer.positionReset
```

Description

Propriété 3D de modificateur `keyframePlayer` et `bonesPlayer` ; indique si le modèle retourne à sa position de départ à la fin d'un mouvement (`TRUE`) ou non (`FALSE`).

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante empêche le modèle Monstre de retourner à sa position originale lorsqu'il finit l'exécution d'un mouvement.

```
member("NewAlien").model("Monster").keyframePlayer.positionReset = FALSE
```

Voir aussi

currentLoopState

posterFrame

Syntaxe

```
-- Lingo syntax
memberObjRef.posterFrame

// JavaScript syntax
memberObjRef.posterFrame;
```

Description

Propriété d'acteur Flash ; contrôle l'image d'un acteur animation Flash utilisée pour son image miniature. Cette propriété spécifie un nombre entier correspondant au numéro d'une image de l'animation Flash.

Cette propriété peut être testée et définie. La valeur par défaut est 1.

Exemple

Le gestionnaire suivant accepte comme paramètres une référence à un acteur animation Flash et un numéro d'image, puis place la miniature de l'animation spécifiée sur le numéro d'image spécifié :

```
-- Lingo syntax
on resetThumbnail(whichFlashMovie, whichFrame)
    member(whichFlashMovie).posterFrame = whichFrame
end

// JavaScript syntax
function resetThumbnail(whichFlashMovie, whichFrame) {
    member(whichFlashMovie).posterFrame = whichFrame;
}
```

preferred3dRenderer

Syntaxe

```
-- Lingo syntax
_movie.preferred3dRenderer

// JavaScript syntax
_movie.preferred3dRenderer;
```

Description

Propriété d'animation ; permet d'obtenir ou de définir le moteur de rendu par défaut utilisé pour tracer des images-objets 3D d'une animation spécifique si ce moteur de rendu est disponible sur la machine cliente. Lecture/écriture.

Si le moteur de rendu n'est pas disponible sur la machine cliente, l'animation sélectionne le moteur de rendu disponible le plus approprié.

Les valeurs possibles de cette propriété sont les suivantes :

- `#openGL` spécifie les pilotes openGL d'accélération matérielle fonctionnant sur les plates-formes Mac et Windows.
- `#directX9` spécifie les pilotes DirectX® 9 d'accélération matérielle fonctionnant uniquement sur les plates-formes Windows. #auto définit DirectX 9 comme moteur de rendu sous Windows. Dans Mac® -Intel®, seul le moteur de rendu #OpenGL est disponible. DirectX 9 a été ajouté au menu Moteur de rendu 3D préféré dans l'onglet Animation de l'Inspecteur des propriétés. L'utilisation de DirectX 9 n'entraîne aucune modification des fonctionnalités existantes. Toutefois, cette option peut contribuer à améliorer les performances.
- `#directX7_0` spécifie les pilotes DirectX 7 d'accélération matérielle fonctionnant uniquement sur les plates-formes Windows.
- `#directX5_2` spécifie les pilotes DirectX 5.2 d'accélération matérielle fonctionnant uniquement sur les plates-formes Windows.
- `#software` spécifie le moteur de rendu logiciel intégré à Director fonctionnant avec les plates-formes Mac et Windows.
- `#auto` indique que le moteur de rendu le plus approprié doit être choisi. Il s'agit de la valeur par défaut de cette propriété.

La valeur définie pour cette propriété est utilisée par défaut pour la propriété `renderer` de l'objet de services de rendu.

Cette propriété diffère de la propriété `renderer` de l'objet `getRendererServices()` car `preferred3dRenderer` spécifie le moteur de rendu préféré à utiliser, alors que la propriété `renderer` de l'objet `getRendererServices()` indique le moteur de rendu en cours d'utilisation par l'animation.

Shockwave Player permet de spécifier le moteur de rendu souhaité à l'aide du menu contextuel correspondant. Si l'utilisateur sélectionne l'option indiquant de respecter les paramètres de contenu, le moteur de rendu spécifié par les propriétés `renderer` ou `preferred3DRenderer` est utilisé pour dessiner l'animation (s'il est disponible sur le système de l'utilisateur) ; dans les autres cas, c'est le moteur de rendu sélectionné par l'utilisateur qui est utilisé.

Exemple

L'instruction suivante permet à l'animation de sélectionner le meilleur moteur de rendu 3D disponible sur le système de l'utilisateur :

```
-- Lingo syntax
_movie.preferred3dRenderer = #auto

// JavaScript syntax
_movie.preferred3dRenderer = "auto";
```

Voir aussi

`getRendererServices()`, `Animation`, `renderer`

preLoad (3D)

Syntaxe

```
member(whichCastmember).preload
memberReference.preload
```

Description

Propriété 3D ; permet de savoir ou de définir si les données sont chargées avant la lecture (`TRUE`) ou si leur chargement s'effectue en flux continu pendant la lecture (`FALSE`). Cette propriété ne peut être utilisée qu'avec des fichiers liés. La valeur par défaut est `FALSE`.

Dans Director, la définition de la propriété `preLoad` sur `TRUE` entraîne le chargement complet de l'acteur avant le début de la lecture. Dans Shockwave Player, la définition de la propriété `preLoad` sur `TRUE` entraîne la lecture en flux continu de l'acteur au début de la lecture de l'animation. Avant d'exécuter des opérations Lingo sur un acteur 3D en cours de lecture en flux continu, prenez soin de vérifier que la propriété `state` de l'acteur présente une valeur supérieure ou égale à 2.

Exemple

L'instruction suivante attribue à la propriété `preload` de l'acteur séquenceDeFête la valeur `FALSE`, ce qui permet la lecture en flux continu des médias externes liés à l'animation dans l'acteur séquenceDeFête pendant la lecture :

```
member("PartyScene").preload = FALSE
member("3D world").preload
```

Voir aussi

`state (3D)`

preLoad (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.preLoad

// JavaScript syntax
memberObjRef.preLoad;
```

Description

Propriété d'acteur ; détermine si l'acteur vidéo numérique spécifié par *quelActeur* peut être préchargé en mémoire (TRUE) ou non (FALSE, valeur par défaut). L'état TRUE produit le même effet que la sélection de l'option Activer le préchargement dans la boîte de dialogue Propriétés de l'acteur vidéo numérique.

Pour les acteurs animation Flash, cette propriété détermine si une animation Flash doit être entièrement chargée en RAM avant l'affichage de la première image d'une image-objet (TRUE) ou si l'animation peut être transférée en mémoire pendant sa lecture (FALSE, valeur par défaut). Cette propriété ne fonctionne qu'avec les animations Flash liées dont les éléments sont stockés dans un fichier externe ; elle n'a aucun effet sur les acteurs dont les éléments sont stockés dans la distribution. Les propriétés streamMode et bufferSize déterminent la façon dont l'acteur est lu en flux continu dans la mémoire.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante indique dans la fenêtre Messages si l'animation QuickTime Chaise pivotante peut être préchargée en mémoire :

```
-- Lingo syntax
put(member("Rotating Chair").preload)

// JavaScript syntax
put(member("Rotating Chair").preload);
```

Voir aussi

bufferSize, streamMode

preLoadEventAbort

Syntaxe

```
-- Lingo syntax
_movie.preLoadEventAbort

// JavaScript syntax
_movie.preLoadEventAbort;
```

Description

Propriété d'animation; spécifie si le fait d'appuyer sur une touche ou de cliquer avec la souris peut arrêter le préchargement des acteurs (TRUE) ou non (FALSE, valeur par défaut). Lecture/écriture.

La modification de cette propriété affecte l'animation actuelle.

Exemple

L'instruction suivante permet à l'utilisateur d'arrêter le préchargement des acteurs en appuyant sur des touches ou en cliquant avec la souris :

```
-- Lingo syntax
_movie.preLoadEventAbort = TRUE

// JavaScript syntax
_movie.preLoadEventAbort = true;
```

Voir aussi

Animation

preLoadMode

Syntaxe

```
-- Lingo syntax
castObjRef.preLoadMode

// JavaScript syntax
castObjRef.preLoadMode;
```

Description

Propriété de bibliothèque de distribution ; détermine le mode de préchargement d'une bibliothèque de distribution spécifiée. Lecture/écriture.

Les valeurs possibles de `preLoadMode` sont les suivantes :

- 0. La bibliothèque de distribution est chargée lorsqu'elle est requise. C'est la valeur par défaut.
- 1. La bibliothèque de distribution est chargée avant l'image 1.
- 2. La bibliothèque de distribution est chargée après l'image 1.

La définition de cette propriété produit le même effet que la sélection de l'option de chargement de la distribution dans la boîte de dialogue Propriétés de la distribution.

Exemple

L'instruction suivante demande à Director de charger les acteurs de la distribution Boutons avant que l'animation n'entre dans l'image 1 :

```
-- Lingo syntax
castLib("Buttons").preLoadMode = 1

// JavaScript syntax
castLib("Buttons").preLoadMode = 1;
```

Voir aussi

Bibliothèque de distribution

preLoadRAM

Syntaxe

```
the preLoadRAM
```

Description

Propriété système ; spécifie la quantité de mémoire RAM pouvant être utilisée pour le préchargement d'une vidéo numérique. Cette propriété peut être définie et testée.

Cette propriété est utile pour la gestion de la mémoire, puisqu'elle limite les acteurs vidéo numérique à une certaine quantité de mémoire, pour permettre le préchargement d'autres types d'acteurs. Lorsque la propriété `preLoadRAM` reçoit la valeur `FALSE`, toute la mémoire disponible peut être utilisée pour le préchargement des acteurs vidéo numérique.

Il n'est cependant pas possible de prédire de manière fiable la quantité de RAM qu'une vidéo numérique exige une fois préchargée, la mémoire requise étant affectée par le contenu de l'animation, le taux de compression utilisé, le nombre d'images-clés, la modification des images, etc.

Il est généralement possible de précharger sans crainte les deux ou trois premières secondes d'une vidéo puis de continuer la lecture en flux continu à partir de cet endroit.

Si vous connaissez le taux de transfert de votre animation, vous pouvez estimer le paramétrage de `preLoadRAM`. Par exemple, si votre animation utilise un taux de transfert de 300 Ko par seconde, définissez `preLoadRAM` sur 600 Ko si vous souhaitez précharger les 2 premières secondes du fichier vidéo. Même s'il ne s'agit que d'une estimation, elle convient dans la plupart des cas.

Exemple

L'instruction suivante définit la propriété `preLoadRAM` sur 600 Ko pour précharger les 2 premières secondes d'une animation dont le taux de transfert est de 300 Ko par seconde :

```
--Lingo
set the preLoadRAM = 600

// Javascript
_system.preLoadRAM = 600;
```

Voir aussi

`loop (mot-clé)`, `next`

preLoadTime

Syntaxe

```
-- Lingo syntax
memberObjRef.preLoadTime

// JavaScript syntax
memberObjRef.preLoadTime;
```

Description

Propriété d'acteur et de piste audio ; pour les acteurs, spécifie la quantité (en secondes) de l'acteur Shockwave Audio (SWA) en flux continu à télécharger avant que sa lecture ne commence ou lors de l'utilisation d'une commande `preLoadBuffer`. La valeur par défaut est 5 secondes.

Cette propriété ne peut être définie que si l'acteur SWA lu en flux continu est arrêté.

Pour les pistes audio, la valeur concerne le son défini dans la file d'attente ou le son actuel, si aucun son n'est spécifié.

Exemple

Le gestionnaire suivant définit sur 6 secondes le temps de préchargement de l'acteur SWA Louis Armstrong lu en flux continu. Le préchargement réel se produit lorsqu'une commande `preLoadBuffer` ou `play` est émise.

```
-- Lingo syntax
on mouseDown
    member("Louis Armstrong").stop()
    member("Louis Armstrong").preLoadTime = 6
end

// JavaScript syntax
function mouseDown() {
    member("Louis Armstrong").stop();
    member("Louis Armstrong").preLoadTime = 6;
}
```

L'instruction suivante renvoie la valeur preLoadTime du son en cours de lecture dans la piste audio 1 :

```
-- Lingo syntax
put sound(1).preLoadTime

// JavaScript syntax
trace(sound(1).preLoadTime);
```

Voir aussi

preLoadBuffer()

primitives

Syntaxe

```
getRendererServices().primitives
```

Description

Fonction 3D ; renvoie une liste des types de primitives qui peuvent être utilisés pour créer de nouvelles ressources de modèle.

Exemple

L'instruction suivante affiche les types de primitives disponibles.

```
--Lingo
put getRendererServices().primitives

// Javascript
put( getRendererServices().primitives);
```

Voir aussi

getRendererServices(), newModelResource

productName

Syntaxe

```
-- Lingo syntax
_player.productName

// JavaScript syntax
_player.productName;
```

Description

Propriété de lecteur ; renvoie le nom de l'application Director. Lecture seule.

Exemple

L'instruction suivante affiche dans la fenêtre Messages le nom de l'application Director.

```
-- Lingo syntax
trace(_player.productName)

// JavaScript syntax
trace(_player.productName);
```

Voir aussi

Lecteur

productVersion

Syntaxe

```
-- Lingo syntax
_player.productVersion

// JavaScript syntax
_player.productVersion;
```

Description

Propriété de lecteur ; renvoie le numéro de version de l'application Director. Lecture seule.

Exemple

L'instruction suivante affiche dans la fenêtre Messages la version de l'application Director.

```
-- Lingo syntax
trace(_player.productVersion)

// JavaScript syntax
trace(_player.productVersion);
```

Voir aussi

Lecteur

projection

Syntaxe

```
sprite(whichSprite).camera.projection
camera(whichCamera).projection
member(whichCastmember).camera(whichCamera).projection
```

Description

Propriété 3D ; permet d'obtenir ou de définir le style de projection de la caméra. Les valeurs possibles sont `#perspective` (valeur par défaut) et `#orthographic`.

Lorsque la propriété projection présente la valeur `#perspective`, les objets les plus proches de la caméra apparaissent plus grands que les objets les plus éloignés, et les propriétés `projectionAngle` ou `fieldOfView` spécifient l'angle de projection verticale (qui détermine l'espace visible de l'univers). L'angle de projection horizontale est déterminé par le rapport hauteur/largeur de la propriété `rect` de la caméra.

Lorsque la propriété projection présente la valeur `#orthographic`, la taille apparente des objets ne dépend pas de la distance de la caméra et la propriété `orthoHeight` spécifie le nombre d'unités d'univers qui logent verticalement dans l'image-objet (ce qui détermine l'espace visible de l'univers). La largeur de la projection orthographique est déterminée par le rapport hauteur/largeur de la propriété `rect` de la caméra.

Exemple

L'instruction suivante attribue à la propriété `projection` de la caméra de l'image-objet 5 la valeur `#orthographic` :

```
sprite(5).camera.projection = #orthographic
```

Voir aussi

`fieldOfView (3D)`, `orthoHeight`, `fieldOfView (3D)`

purgePriority

Syntaxe

```
-- Lingo syntax
memberObjRef.purgePriority

// JavaScript syntax
memberObjRef.purgePriority;
```

Description

Propriété d'acteur ; spécifie la priorité de purge d'un acteur. Lecture/écriture.

Les priorités de purge des acteurs déterminent l'ordre de priorité suivi par Director pour sélectionner les acteurs à supprimer de la mémoire lorsque celle-ci est saturée. Plus la priorité de purge est élevée, plus il est probable que l'acteur est supprimé. Les valeurs disponibles pour la propriété `purgePriority` sont les suivantes :

- 0 – Jamais
- 1 – Dernier
- 2 – Suivant
- 3 – Normale (valeur par défaut)

Le paramètre Normale permet à Director de purger les acteurs de la mémoire de manière aléatoire. Les paramètres Suivant, Dernier et Jamais permettent de contrôler plus ou moins la purge. Toutefois, si vous sélectionnez les valeurs Dernier ou Jamais pour de nombreux acteurs, votre animation risque de se trouver à court de mémoire.

La définition de la propriété `purgePriority` des acteurs vous permet de gérer la mémoire lorsque la taille de la bibliothèque de distribution de l'animation dépasse la mémoire disponible. En règle générale, vous pouvez minimiser les pauses pendant que l'animation charge les acteurs et réduire le nombre de nouveaux chargements d'acteurs que Director doit exécuter en affectant une faible priorité de purge aux acteurs fréquemment utilisés au cours de l'animation.

Exemple

L'instruction suivante affecte la valeur 3 à la priorité de purge de l'acteur Arrière-plan, ce qui signifie qu'il est l'un des premiers acteurs à être purgés lorsque Director a besoin de mémoire :

```
-- Lingo syntax
member("Background").purgePriority = 3

// JavaScript syntax
member("Background").purgePriority = 3;
```

Voir aussi

Acteur

quad

Syntaxe

```
-- Lingo syntax
spriteObjRef.quad

// JavaScript syntax
spriteObjRef.quad;
```

Description

Propriété d'image-objet ; contient une liste de quatre points, qui sont les valeurs flottantes servant à décrire les coins d'une image-objet sur la scène. Lecture/écriture.

Les points de ce quadrilatère sont organisés dans l'ordre suivant : supérieur gauche, supérieur droit, inférieur droit et inférieur gauche.

Les points eux-mêmes peuvent être manipulés pour obtenir des effets de perspective et autres distorsions des images.

Si vous manipulez le quadrilatère d'une image-objet, vous pouvez lui redonner les valeurs du scénario en désactivant l'image-objet contrôlée par un script au moyen de `puppetSprite(`*`numDimageObjet`*`, FALSE)`. La désactivation du quadrilatère d'une image-objet interdit d'autre part de la faire pivoter ou de l'incliner.

Exemple

L'instruction suivante affiche une liste type décrivant une image-objet :

```
-- Lingo syntax
put(sprite(1).quad)

// JavaScript syntax
put(sprite(1).quad);
```

Lorsque vous modifiez la propriété d'image-objet `quad`, n'oubliez pas que vous devez rétablir la liste de points si vous modifiez l'une de ces valeurs. En effet, lorsque vous affectez la valeur d'une propriété à une variable, vous placez une copie de la liste, et non la liste elle-même, dans la variable. Pour appliquer un changement, utilisez une syntaxe de ce type (uniquement applicable à Lingo) :

```
-- Lingo syntax
currQuadList = sprite(5).quad
currQuadList[1] = currQuadList[1] + point(50, 50)
sprite(5).quad = currQuadList
```

Voir aussi

`point()`, `puppetSprite()`, Image-objet

quality

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.quality

// JavaScript syntax
memberOrSpriteObjRef.quality;
```

Description

Propriété d'acteur et d'image-objet Flash ; contrôle si Director utilise l'anti-aliasing pour le rendu d'une image-objet animation Flash, produisant une haute qualité de rendu, mais aussi une lecture plus lente. La propriété `quality` peut prendre les valeurs suivantes :

- `#autoHigh` : Director commence par effectuer le rendu de l'image-objet avec l'anti-aliasing. Si la cadence d'images tombe en dessous de celle spécifiée pour l'animation, Director désactive l'anti-aliasing. Ce réglage donne priorité à la vitesse de lecture sur la qualité visuelle.
- `#autoLow` : Director commence par effectuer le rendu de l'animation sans anti-aliasing. Le lecteur Flash active l'anti-aliasing s'il détermine que le processeur de l'ordinateur est capable de le gérer. Ce réglage donne priorité à la qualité visuelle si cela est possible.
- `#high` (valeur par défaut) : l'animation est toujours lue avec anti-aliasing.
- `#low` : l'animation est toujours lue sans anti-aliasing.

La propriété `quality` peut être testée et définie.

Exemple

Le script d'image-objet suivant détermine le nombre de couleurs de l'ordinateur sur lequel l'animation est lue. Si le codage des couleurs est défini sur 8 bits ou moins (256 couleurs), le scénario règle la qualité de l'image-objet dans la piste 5 sur `#low`.

```
-- Lingo syntax
on beginSprite me
    if _system.colorDepth <= 8 then
        sprite(1).quality = #low
    end if
end

// JavaScript syntax
function beginSprite() {
```

```
    var clrDp = _system.colorDepth;
    if (clrDp <= 8) {
        sprite(1).quality = symbol("low");
    }
}
```

quality (3D)

Syntaxe

```
member(whichCastmember).texture(whichTexture).quality
member(whichCastmember).shader(whichShader).texture(whichTexture).quality
member(whichCastmember).model(whichModel).shader.texture(whichTexture).quality
member( whichCastmember ).model( whichModel ).shader.texturelist[TextureListIndex].quality
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].
texture(whichTexture).quality
member( whichCastmember ).model( whichModel ).shaderList[ shaderListIndex ]. texturelist[
TextureListIndex ].quality
```

Description

Propriété de texture 3D ; permet d'obtenir ou de définir la qualité d'image d'une texture en contrôlant le niveau de mipmapping qui lui est appliqué. Le mipmapping est un processus qui crée des versions supplémentaires de l'image de texture, créées de tailles variées et plus petites que l'image d'origine. L'Xtra 3D affiche ensuite la version la plus appropriée de l'image en fonction de la taille en cours du modèle et change la version de l'image utilisée selon les besoins. Le mipmapping trilinéaire offre une qualité supérieure au mipmapping bilinéaire, mais requiert davantage de mémoire.

Le mipmapping est différent du filtrage, bien que les deux processus améliorent l'apparence de la texture. Le filtrage répartit les erreurs sur l'ensemble de la texture pour qu'elles soient moins concentrées. Le mipmapping rééchantillonne l'image pour lui donner la taille appropriée.

Cette propriété peut avoir les valeurs suivantes :

- `#low` correspond à la désactivation, le mipmapping n'étant pas utilisé pour la texture.
- `#medium` définit un mipmapping de faible qualité (bilinéaire) pour la texture.
- `#high` définit un mipmapping de qualité élevée (trilinéaire) pour la texture.

La valeur par défaut est `#low`.

Exemple

L'instruction suivante attribue à la propriété `quality` de la texture de placageMars la valeur `#medium` :

```
member("scene").texture("Marsmap").quality = #medium
```

Voir aussi

nearFiltering

radius

Syntaxe

```
modelResourceObjectReference.radius
member(whichCastmember).modelResource(whichModelResource).radius
```

Description

Propriété de modèle 3D ; utilisée avec une ressource de modèle de type `#sphere` ou `#cylinder`, cette propriété permet d'obtenir ou de définir le rayon du modèle.

La propriété `radius` détermine le rayon de balayage utilisé pour générer la ressource de modèle. La valeur de cette propriété doit toujours être supérieure à 0,0 et est définie par défaut sur 25,0.

Exemple

L'instruction suivante affiche le rayon de la ressource de modèle Cylinder01.

```
--Lingo
put member("3Dobjects").modelResource("Cylinder01").radius

// Javascript
put(member("3Dobjects").getPropRef("modelResource",11).radius);
```

randomSeed

Syntaxe

```
the randomSeed
```

Description

Propriété système ; spécifie la valeur de départ utilisée pour générer des nombres aléatoires obtenus au moyen de la fonction `random()`.

L'utilisation de la même valeur de départ produit la même séquence de nombres aléatoires. Cette propriété peut être utile pour le débogage au cours du développement. L'utilisation de la propriété `ticks` facilite la production d'une valeur de départ aléatoire unique car il est hautement improbable que la valeur de `ticks` soit répétée dans les utilisations suivantes.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche le nombre de départ aléatoire dans la fenêtre Messages.

```
--Lingo
put the randomSeed

// Javascript
put(_system.randomSeed);
```

Voir aussi

`random()`, `milliseconds`

recordFont

Syntaxe

```
recordFont(whichCastMember, font {[,face]} {,[bitmapSizes]} {,characterSubset} {,
userFontName})
```

Description

Commande ; inclut une police TrueType ou Type 1 comme acteur. Une fois incluses, ces polices sont disponibles à l'auteur tout comme les autres polices installées sur le système.

Vous devez créer un acteur police vide à l'aide de la commande `new()` avant d'utiliser `recordFont`.

- *`police`* : nom de la police d'origine à enregistrer.
- *`style`* : Liste de symboles indiquant le style de la police d'origine, les valeurs possibles étant `#plain`, `#bold` et `#italic`. Si vous ne définissez aucune valeur pour cet argument, la valeur `#plain` est utilisée par défaut.
- *`tailleDesBitmaps`* : Liste d'entiers spécifiant les tailles pour lesquelles les bitmaps doivent être enregistrés. Cet argument peut être vide. Si vous omettez cet argument, aucun bitmap n'est généré. Ces bitmaps donnent généralement de meilleurs résultats pour les petites tailles (inférieures à 14 points), mais occupent plus de mémoire.
- *`sousEnsembleDeCaractères`* : Chaîne de caractères à coder. Seuls les caractères spécifiés sont disponibles dans la police. Si cet argument est fourni, tous les caractères qu'il contient sont encodés. Si seuls certains caractères sont codés mais qu'un caractère non codé est utilisé, ce caractère apparaît comme une case vide.
- *`nouveauNom`* : Chaîne utilisée comme nom de l'acteur police nouvellement enregistré.

La commande crée une police shockée dans *`quelActeur`* en utilisant la police nommée dans l'argument *`police`*. La valeur renvoyée par la commande indique si l'opération a réussi. La valeur zéro indique que l'opération a réussi.

Exemple

L'instruction suivante crée une police shockée simple n'utilisant que les deux arguments pour l'acteur et la police à enregistrer :

```
myNewFontMember = new(#font)
recordFont(myNewFontMember, "Lunar Lander")
```

L'instruction suivante spécifie les tailles de bitmaps à générer et les caractères pour lesquels les données de police doivent être créées :

```
myNewFontMember = new(#font)
recordfont(mynewmember,"lunar lander",[],[14, 18, 45], "Lunar Lander Game High Score First
Last Name")
```

Remarque : *la propriété recordFont resynthétisant les données de la police au lieu de les utiliser directement, la distribution des polices shockées n'est soumise à aucune restriction légale.*

Voir aussi

`new()`

rect (caméra)

Syntaxe

```
sprite(whichSprite).camera(whichCamera).rect
```

Description

Propriété 3D de caméra ; permet d'obtenir ou de définir le rectangle qui contrôle la taille et la position de la caméra. Ce rectangle est analogue à celui que vous observez dans le viseur d'une caméra réelle.

La valeur par défaut de la propriété rect de toutes les caméras est rect(0, 0, 1, 1) qui les rend invisibles jusqu'à la modification du paramètre. Toutefois, lorsque `sprite.camera(1)` est rendu, sa propriété rect est redéfinie sur `rect(0, 0, sprite(`*`quelleImageObjet`*`).width, sprite(`*`quelleImageObjet`*`).height)` de façon à ce que la caméra remplisse l'écran. Toutes les coordonnées de cadre de la caméra sont calculées par rapport au coin supérieur gauche de l'image-objet.

Si la valeur de *`quelleCaméra`* est supérieure à 1, la propriété `rect` n'est pas redimensionnée en même temps que l'image-objet ; en cas de besoin, il est donc nécessaire de gérer ce redimensionnement à l'aide d'un script.

Si la valeur de *`quelleCaméra`* est supérieure à 1, les propriétés `rect.top` et `rect.left` doivent être supérieures ou égales aux valeurs `rect.top` et `rect.left` de `sprite.camera(1)`.

Exemple

L'instruction suivante affiche les coordonnées de l'acteur bitmap 1.

```
-- Lingo syntax
put(member(1).rect)

// JavaScript syntax
put(member(1).rect);
```

Voir aussi

cameraPosition, cameraRotation

rect (image)

Syntaxe

```
-- Lingo syntax
imageObjRef.rect

// JavaScript syntax
imageObjRef.rect;
```

Description

Propriété d'image. Renvoie un rectangle décrivant la taille d'une image donnée. Lecture seule.

Les coordonnées du rectangle renvoyé sont calculées par rapport au coin supérieur gauche de l'image. Les valeurs gauche et supérieure du rectangle sont donc de 0 et les valeurs inférieure et droite constituent la largeur et la hauteur de l'acteur.

Exemple

L'instruction suivante affiche le rectangle de l'acteur Lever de soleil de 300 x 400 pixels dans la fenêtre Messages :

```
-- Lingo syntax
member("Sunrise").image.rect -- rect(0, 0, 300, 400)

// JavaScript syntax
member("Sunrise").image.rect; // rect(0, 0, 300, 400)
```

L'instruction Lingo suivante examine les 50 premiers acteurs et affiche le rectangle et le nom de chaque acteur bitmap :

```
-- Lingo syntax
on showAllRects
    repeat with x = 1 to 50
```

```
    if member(x).type = #bitmap then
        put member(x).image.rect && "-" && member(x).name
    end if
    end repeat
end

// JavaScript syntax
function showAllRects() {
    var x = 1;
    while(x < 51) {
        var tp = member(x).type;
        if (tp == "bitmap") {
            trace(member(x).image.rect + " - " + member(x).name);
            i++;
        }
    }
}
```

Voir aussi

`height`, `image()`, `width`

rect (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.rect

// JavaScript syntax
memberObjRef.rect;
```

Description

Propriété d'acteur ; spécifie les coordonnées gauche, supérieure, droite et inférieure, sous la forme d'un rectangle, du rectangle de n'importe quel acteur graphique, tel qu'un bitmap, une forme, une animation ou une vidéo numérique. Lecture seule pour tous les acteurs, lecture/écriture uniquement pour les acteurs champ.

Dans le cas d'un bitmap, la propriété `rect` est mesurée à partir du coin supérieur gauche du bitmap, et non à partir du coin supérieur gauche du bord de la fenêtre Dessin.

Dans le cas d'un acteur Xtra, la propriété `rect` est un rectangle dont le coin supérieur gauche est situé à (0, 0).

Exemple

L'instruction suivante affiche les coordonnées de l'acteur bitmap 20 :

```
-- Lingo syntax
put(member(20).rect)

// JavaScript syntax
put(member(20).rect);
```

L'instruction suivante définit les coordonnées de l'acteur bitmap Bandeau :

```
-- Lingo syntax
member("Banner").rect = rect(100, 150, 300, 400)

// JavaScript syntax
member("Banner").rect = rect(100, 150, 300, 400);
```

Voir aussi

Acteur

rect (image-objet)

Syntaxe

```
-- Lingo syntax
spriteObjRef.rect

// JavaScript syntax
spriteObjRef.rect;
```

Description

Propriété d'image-objet ; spécifie les coordonnées gauche, supérieure, droite et inférieure, sous la forme d'un rectangle, du rectangle de n'importe quel image-objet graphique, tel qu'un bitmap, une forme, une animation ou une vidéo numérique. Lecture/écriture.

Exemple

L'instruction suivante affiche les coordonnées de l'image-objet bitmap 20 :

```
-- Lingo syntax
put(sprite(20).rect)

// JavaScript syntax
put(sprite(20).rect);
```

Voir aussi

rect(), Image-objet

rect (fenêtre)

Syntaxe

```
-- Lingo syntax
windowObjRef.rect

// JavaScript syntax
windowObjRef.rect;
```

Description

Propriété de fenêtre ; spécifie les coordonnées gauche, supérieure, droite et inférieure d'une fenêtre, renvoyées sous la forme d'un rectangle. Lecture/écriture.

Si la taille du rectangle spécifié est inférieure à celle de la scène sur laquelle l'animation a été créée, l'animation est recadrée dans la fenêtre, mais n'est pas mise à l'échelle.

Pour effectuer un panoramique ou une mise à l'échelle de l'animation lue dans la fenêtre, définissez la propriété `drawRect` ou `sourceRect` de la fenêtre.

Exemple

L'instruction suivante affiche les coordonnées de la fenêtre Tableau_de_commande :

```
-- Lingo syntax
put(window("Control_panel").rect)

// JavaScript syntax
put(window("Control_panel").rect);
```

Voir aussi

drawRect, sourceRect, Fenêtre

ref

Syntaxe

```
chunkExpression.ref
```

Description

Propriété d'expression de sous-chaîne de texte ; offre un raccourci pratique permettant de faire référence à une expression de sous-chaîne dans un acteur texte.

Exemple

En l'absence de références, vous devriez utiliser une instruction telle que :

```
member(whichTextMember).line[whichLine].word[firstWord..lastWord].font = "Palatino"
member(whichTextMember).line[whichLine].word[firstWord..lastWord].fontSize = 36
member(whichTextMember).line[whichLine].word[firstWord..lastWord].fontStyle = [#bold]
```

Avec une propriété `ref`, en revanche, vous pouvez faire référence à la même sous-chaîne comme dans l'exemple ci-après :

```
myRef = member(whichTextMember).line[whichLine].word[firstWord..lastWord].ref
```

La variable `maRéf` constitue à présent un raccourci pour la totalité de l'expression de sous-chaîne. Cela permet quelque chose comme :

```
put myRef.font
-- "Palatino"
```

Vous pouvez également définir une propriété de la sous-chaîne de la façon suivante :

```
myRef.fontSize = 18
myRef.fontStyle = [#italic]
```

Vous pouvez accéder à la chaîne indiquée par la référence en utilisant la propriété `text` de cette dernière :

```
put myRef.text
```

Cette opération produirait les données réelles de la chaîne et non les informations la concernant.

reflectionMap

Syntaxe

```
member(whichCastmember).shader(whichShader).reflectionMap
```

Description

Propriété 3D de matériau ; permet d'obtenir et de définir la texture utilisée pour créer des reflets à la surface d'un modèle. Cette texture est appliquée à la troisième couche de texture du matériau. Cette propriété est ignorée si le modificateur `toon` est appliqué à la ressource de modèle.

Cette propriété fournit une interface plus simple pour la définition d'une utilisation commune du placage de réflexion. Les propriétés suivantes produisent le même effet :

```
shader.textureModeList[3] = #reflection
shader.blendFunctionList[3] = #blend
shader.blendSourceList[3] = #constant
shader.blendConstantList[3] = 50.0
```

Lorsqu'elle est testée, cette propriété renvoie la texture associée à la troisième couche de texture du modèle. La valeur par défaut est `void`.

Exemple

L'instruction suivante entraîne le reflet de la texture par défaut sur la surface du modèle Sphère01.

```
-- Lingo syntax
member("3Dobjects").model("Sphere01").shader.reflectionMap=member("3Dobjects").texture[1]

// JavaScript syntax
member("3Dobjects").getPropRef("model",2).shader.reflectionMap=member("3Dobjects").getProp
Ref("texture",1);
```

Voir aussi

`textureModeList`, `blendFunctionList`, `blendConstantList`

reflectivity

Syntaxe

```
member(whichCastmember).reflectivity
```

Description

Propriété 3D de matériau ; permet d'obtenir ou de définir le niveau de brillant du matériau par défaut de l'acteur référencé. Cette valeur à virgule flottante représente le pourcentage, de 0,0 à 100,00, de lumière à refléter sur la surface d'un modèle utilisant le matériau par défaut. La valeur par défaut est 0,0.

Exemple

L'instruction suivante définit le niveau de brillant du matériau par défaut de l'acteur Séquence sur 50 %.

```
-- Lingo syntax
member("Scene").reflectivity = 50

// JavaScript syntax
member("Scene").reflectivity = 50;
```

region

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).emitter.region
modelResourceObjectReference.emitter.region
```

Description

Propriété 3D d'émetteur ; utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir et de définir la propriété `region` de l'émetteur de particules de la ressource.

Cette propriété de région définit la position à partir de laquelle les particules sont émises. Si sa valeur est un seul vecteur, ce vecteur en question est utilisé pour définir un point à partir duquel les particules vont être émises dans l'univers 3D.

Si sa valeur est une liste de deux vecteurs, ces vecteurs sont utilisés pour définir les points d'extrémité d'un segment de ligne à partir duquel les particules vont être émises.

Si sa valeur est une liste de quatre vecteurs, ces vecteurs sont utilisés pour définir les sommets du quadrilatère à partir duquel les particules vont être émises.

La valeur par défaut de cette propriété est [vector(0, 0, 0)].

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type #particle. L'instruction suivante spécifie les quatre coins du rectangle d'où proviennent les particules de systèmeThermique.

```
member("Fires").modelResource("ThermoSystem").emitter.region = [vector(20,90,100),
vector(30,90,100), vector(30,100,100), vector(20,100,100)]
```

Voir aussi

emitter

regPoint

Syntaxe

```
-- Lingo syntax
memberObjRef.regPoint

// JavaScript syntax
memberObjRef.regPoint;
```

Description

Propriété d'acteur ; spécifie le point d'alignement d'un acteur. Lecture/écriture.

Le point d'alignement est indiqué en tant que coordonnées horizontale et verticale d'un point sous la forme `point(`*`horizontal, vertical`*`)`. Les acteurs non visuels, tels que les sons, ne possèdent pas de propriété `regPoint` utile.

Vous pouvez utiliser la propriété `regPoint` pour animer des graphiques individuels dans une boucle, ce qui change la position de la boucle par rapport à d'autres objets de la scène.

Vous pouvez également utiliser `regPoint` pour ajuster la position d'un masque sur une image-objet.

Lorsqu'un acteur animation Flash est initialement inséré dans la bibliothèque de distribution, son point d'alignement correspond à son centre et sa propriété `centerRegPoint` présente la valeur `TRUE`. Si vous utilisez par la suite la propriété `regPoint` pour repositionner le point d'alignement, la propriété `centerRegPoint` est automatiquement définie sur `FALSE`.

Exemple

L'instruction suivante affiche le point d'alignement de l'acteur bitmap Bureau dans la fenêtre Messages :

```
-- Lingo syntax
put(member("Desk").regPoint)

// JavaScript syntax
put(member("Desk").regPoint);
```

L'instruction suivante change le point d'alignement de l'acteur bitmap Bureau en fonction des valeurs de la liste :

```
-- Lingo syntax
member("Desk").regPoint = point(300, 400)

// JavaScript syntax
member("Desk").regPoint = point(300, 400);
```

Voir aussi

Acteur, Image-objet

regPoint (3D)

Syntaxe

```
sprite(whichSprite).camera.backdrop[backdropIndex].regPoint
member(whichCastmember).camera(whichCamera).backdrop
[backdropIndex].regPoint
```

Description

Propriété 3D de fond et de recouvrement ; permet d'obtenir ou de définir le point d'alignement du fond ou du recouvrement. Le point d'alignement représente les coordonnées *x*, *y* et *z* du centre du fond ou du recouvrement dans l'espace 3D. La valeur par défaut de cette propriété est point(0, 0).

Exemple

L'instruction suivante modifie le point d'alignement du premier fond de la caméra de l'image-objet 13. Le point d'alignement du fond est point(50, 0), mesuré à partir du coin supérieur gauche du fond.

```
sprite(13).camera.backdrop[1].regPoint = point(50, 0)
```

Voir aussi

loc (fond et recouvrement)

regPointVertex

Syntaxe

```
-- Lingo syntax
memberObjRef.regPointVertex
```

```
// JavaScript syntax
memberObjRef.regPointVertex;
```

Description

Propriété d'acteur ; indique si un sommet de *acteurVecteur* est utilisé comme point d'alignement pour cet acteur. Si la valeur est égale à zéro, le point d'alignement est déterminé normalement, à l'aide des propriétés `centerRegPoint` et `regPoint`. Si la valeur est différente de zéro, elle indique la position dans la liste `vertextList` du sommet utilisé comme point d'alignement. La propriété `centerRegPoint` est définie sur `FALSE` et la propriété `regPoint` est définie sur l'emplacement de ce sommet.

Exemple

L'instruction suivante fait correspondre le point d'alignement de l'acteur forme vectorielle Quelconque à l'emplacement du troisième sommet :

```
-- Lingo syntax
member("squiggle").regPointVertex=3

// JavaScript syntax
member("squiggle").regPointVertex=3;
```

Voir aussi

`centerRegPoint`, `regPoint`

renderer

Syntaxe

```
getRendererServices().renderer
```

Description

Propriété 3D ; permet d'obtenir ou de définir le moteur de rendu actuellement utilisé par une animation. La plage des valeurs de cette propriété est déterminée par la liste des moteurs de rendu disponibles, renvoyée par la propriété `rendererDeviceList` de l'objet de services de rendu.

Shockwave Player permet de spécifier le moteur de rendu souhaité à l'aide du menu contextuel correspondant. Si l'utilisateur sélectionne l'option indiquant de respecter les paramètres de contenu, le moteur de rendu spécifié par les propriétés `renderer` ou `preferred3DRenderer` est utilisé pour dessiner l'animation (s'il est disponible sur le système de l'utilisateur) ; dans les autres cas, c'est le moteur de rendu sélectionné par l'utilisateur qui est utilisé.

La valeur par défaut de cette propriété est déterminée par la propriété `preferred3DRenderer`.

Cette propriété renvoie la même valeur que celle renvoyée par la propriété d'animation `active3dRenderer`.

Exemple

L'instruction suivante indique le moteur de rendu actuellement utilisé par le système de l'utilisateur.

```
-- Lingo syntax
put getRendererServices().renderer

// JavaScript syntax
put( getRendererServices().renderer);
```

Voir aussi

getRendererServices(), preferred3dRenderer, rendererDeviceList, active3dRenderer

rendererDeviceList

Syntaxe

```
getRendererServices().rendererDeviceList
```

Description

Propriété 3D de moteur de rendu ; renvoie une liste de symboles identifiant les moteurs de rendu disponibles sur la machine cliente. Le contenu de cette liste détermine la plage des valeurs qui peuvent être spécifiées pour les propriétés `renderer` et `preferred3DRenderer`. Cette propriété peut être testée, mais pas définie.

Cette propriété est une liste contenant les valeurs possibles suivantes :

- `#openGL` spécifie les pilotes openGL d'accélération matérielle fonctionnant sur les plates-formes Mac et Windows.
- `#directX9` spécifie les pilotes DirectX 9 d'accélération matérielle fonctionnant uniquement sur les plates-formes Windows. #auto définit automatiquement DirectX 9 comme moteur de rendu. Dans Mac-Intel, seul le moteur de rendu #OpenGL est disponible.
- `#directX7_0` spécifie les pilotes DirectX 7 d'accélération matérielle fonctionnant uniquement sur les plates-formes Windows.
- `#directX5_2` spécifie les pilotes DirectX 5.2 d'accélération matérielle fonctionnant uniquement sur les plates-formes Windows.
- `#software` spécifie le moteur de rendu logiciel intégré à Director fonctionnant avec les plates-formes Mac et Windows.

Exemple

L'instruction suivante indique les moteurs de rendu disponibles sur le système actuel.

```
-- Lingo syntax
put getRendererServices().rendererDeviceList

// JavaScript syntax
put( getRendererServices().rendererDeviceList);
```

Voir aussi

getRendererServices(), renderer, preferred3dRenderer, active3dRenderer

renderFormat

Syntaxe

```
member(whichCastmember).texture(whichTexture).renderFormat
member(whichCastmember).texture[index].renderFormat
member(whichCastmember).shader(whichShader).texture.renderFormat
member(whichCastmember).model(whichModel).shader.texture.renderFormat
member(whichCastmember).model(whichModel).shader.textureList[index].renderFormat
```

```
member(whichCastmember).model(whichModel).shaderList[index].texture(whichTexture).renderFo
rmat
member(whichCastmember).model(whichModel).shaderList[index].textureList[index].renderFormat
```

Description

Propriété 3D ; permet d'obtenir ou de définir la propriété `textureRenderFormat` pour une texture spécifique en spécifiant l'une des valeurs suivantes :

`#default` utilise la valeur renvoyée par `getRendererServices().textureRenderFormat`.

```
#rgba8888
#rgba8880
#rgba5650
#rgba5550
#rgba5551
#rgba4444
```

Pour plus d'informations sur ces valeurs, reportez-vous à l'entrée `textureRenderFormat`.

La définition de cette propriété pour une texture individuelle remplace le paramétrage global défini à l'aide de `textureRenderFormat`.

La propriété `renderFormat` détermine le format de pixel utilisé par le moteur de rendu lorsqu'il effectue le rendu de la texture spécifiée. Chaque format de pixel est composé de chiffres, indiquant chacun le codage chromatique utilisé pour le rouge, le vert, le bleu et l'alpha. La valeur choisie détermine le niveau de fidélité des couleurs (y compris la précision du canal alpha facultatif) et donc la quantité de mémoire utilisée sur la carte vidéo. Vous pouvez choisir une valeur qui améliore la fidélité des couleurs ou une valeur qui vous permet de stocker davantage de textures en mémoire sur la carte vidéo. Vous pouvez stocker à peu près deux fois plus de textures 16 bits que de textures 32 bits dans un même espace.

Exemple

L'instruction suivante attribue à la propriété `renderFormat` de la texture ImageTexte la valeur #rgba4444. Les composants rouge, bleu, vert et alpha de la texture sont chacun dessinés en utilisant 4 bits d'information.

```
-- Lingo syntax
member("3Dobjects").texture[1].renderFormat = #rgba4444
put(member("3Dobjects").texture[1].renderFormat )
```

Voir aussi

`textureRenderFormat`, `getHardwareInfo()`

renderStyle

Syntaxe

```
member(whichCastmember).shader(whichShader).renderStyle
```

Description

Propriété 3D de matériau standard ; permet d'obtenir ou de définir la propriété `renderStyle` d'un matériau, tel que déterminé par la géométrie de la ressource de modèle sous-jacente. Cette propriété peut présenter les valeurs suivantes :

#fill spécifie que le matériau remplit totalement la surface de la ressource de modèle.

#wire spécifie que le matériau n'apparaît qu'au bord des faces de la ressource de modèle.

#point spécifie que le matériau n'apparaît qu'aux sommets de la ressource de modèle.

Tous les matériaux ont accès aux propriétés de matériau `#standard` ; outre ces propriétés standard, les matériaux de type `#engraver`, `#newsprint` et `#painter` possèdent des propriétés propres à leur type. Pour plus d'informations, reportez-vous à l'entrée `newShader`.

Exemple

L'instruction suivante entraîne le rendu du premier matériau uniquement au niveau d'un sommet de la ressource de modèle sous-jacente.

```
-- Lingo syntax
member("3Dobjects").shader[1].renderStyle = #point

// JavaScript syntax
 member("3Dobjects").getPropRef("shader",1).renderStyle = symbol("point");
```

resizable

Syntaxe

```
-- Lingo syntax
windowObjRef.resizable

// JavaScript syntax
windowObjRef.resizable;
```

Description

Propriété de fenêtre ; indique si la fenêtre est redimensionnable (`TRUE`, valeur par défaut) ou non (`FALSE`). Lecture/écriture.

Exemple

Les instructions suivantes agrandissent la fenêtre Empire si cette dernière est redimensionnable.

```
-- Lingo syntax
if (window("Empire").resizable = TRUE) then
    window("Empire").maximize()
end if

// JavaScript syntax
if (window("Empire").resizable == true) {
    window("Empire").maximize();
}
```

Voir aussi

`Fenêtre`

resolution (3D)

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).resolution
```

Description

Propriété 3D ; permet d'obtenir ou de définir la propriété de résolution d'une ressource de modèle de type `#sphere` ou `#cylinder`.

La résolution contrôle le nombre de polygones utilisés pour générer la géométrie de la ressource de modèle. Une valeur plus élevée génère des polygones plus nombreux produisant une surface plus régulière. La valeur par défaut de cette propriété est 20.

Exemple

L'instruction suivante donne à la résolution de la ressource de modèle sphère01 la valeur 10,0.

```
member("3D World").modelResource("sphere01").resolution = 10.0
```

resolution (DVD)

Syntaxe

```
-- Lingo syntax
dvdObjRef.resolution

// JavaScript syntax
dvdObjRef.resolution;
```

Description

Propriété de DVD. Renvoie une liste de propriétés contenant la résolution de source d'axe des `x` (largeur) et d'axe des `y` (hauteur). Lecture seule.

Exemple

L'instruction suivante renvoie un exemple de liste de propriétés de résolutions :

```
-- Lingo syntax
trace(member(1).resolution) -- [#width: 720, #height: 480]

// JavaScript syntax
trace(member(1).resolution); // ["width": 720, "height": 480]
```

Voir aussi

DVD

resolve

Syntaxe

```
member(whichCastmember).model(whichModel).collision.resolve
```

Description

Propriété 3D de collision ; permet de savoir ou de définir si les collisions sont résolues à la collision de deux modèles. Si cette propriété présente la valeur `TRUE` pour deux modèles impliqués dans une collision, tous deux s'arrêtent au point de collision. Si la propriété `resolve` d'un seul de ces modèles présente la valeur `TRUE`, ce modèle s'arrête et l'autre modèle, dont la propriété est soit non définie, soit définie sur `FALSE`, continue son mouvement. La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante attribue à la propriété `resolve` du modificateur `collision` appliqué au modèle Boîte la valeur `TRUE`. Lorsque le modèle Boîte entre en collision avec un autre modèle auquel le modificateur #collision est associé, il s'arrête.

```
member("3d world").model("Box").collision.resolve = TRUE
```

Voir aussi

collisionData, collisionNormal, modelA, modelB, pointOfContact

resource

Syntaxe

```
member(whichCastmember).model(whichModel).resource
```

Description

Propriété 3D ; permet d'obtenir ou de définir la propriété de ressource qui définit la géométrie de la ressource de modèle référencée. Cette propriété permet également d'accéder à l'objet de ressource de modèle référencé ainsi qu'à ses propriétés associées.

Exemple

L'instruction suivante affiche la propriété `radius` de la ressource de modèle utilisée par le second modèle.

```
-- Lingo syntax
put member("3Dobjects").model[2].resource.radius

// JavaScript syntax
put(member("3Dobjects").getPropRef("model",2).resource.radius);
```

right

Syntaxe

```
-- Lingo syntax
spriteObjRef.right

// JavaScript syntax
spriteObjRef.right;
```

Description

Propriété d'image-objet ; indique la distance, en pixels, séparant le bord droit d'une image-objet du bord gauche de la scène. Lecture/écriture.

Les coordonnées des images-objets sont exprimées par rapport au coin supérieur gauche de la scène.

Exemple

L'instruction suivante renvoie la distance du bord droit d'une image-objet :

```
-- Lingo syntax
put(sprite(6).right)
```

```
// JavaScript syntax
put(sprite(6).right);
```

Voir aussi

bottom, height, left, locH, locV, Image-objet, top, width

right (3D)

Syntaxe

```
member(whichCastmember).modelResource
(whichModelResource).right
modelResourceObjectReference.right
```

Description

Propriété 3D ; permet d'obtenir ou de définir la propriété `right` d'une ressource de modèle de type `#box`.

La propriété `right` détermine si le côté droit de la boîte est fermé (`TRUE`) ou ouvert (`FALSE`). La valeur par défaut est `TRUE`.

Exemple

L'instruction suivante attribue à la propriété right de la ressource de modèle Caisse la valeur `TRUE`, ce qui signifie que le côté droit de la caisse est fermé :

```
member("3D World").modelResource("crate").right = TRUE
```

Voir aussi

bottom (3D), left (3D), top (3D)

rightIndent

Syntaxe

```
chunkExpression.rightIndent
```

Description

L'instruction suivante définit un retrait à droite de 10 pixels pour l'acteur texte monTexte.

```
-- Lingo syntax
member("myText").rightIndent=10

// JavaScript syntax
member("myText").rightIndent=10;
```

Voir aussi

firstIndent, leftIndent

rightMouseDown

Syntaxe

```
-- Lingo syntax
_mouse.rightMouseDown

// JavaScript syntax
_mouse.rightMouseDown;
```

Description

Propriété de souris ; indique si l'utilisateur est en train d'appuyer sur le bouton droit de la souris (Windows) ou sur Contrôle+bouton de la souris (Mac) (`TRUE`) ou non (`FALSE`). Lecture seule.

Sur le Mac, la propriété `rightMouseDown` présente la valeur `TRUE` uniquement si la propriété `emulateMultiButtonMouse` renvoie elle-même la valeur `TRUE`.

Exemple

L'instruction suivante vérifie si le bouton droit de la souris (sous Windows) est enfoncé et, le cas échéant, lit le son Désolé dans la piste audio 2 :

```
-- Lingo syntax
if (_mouse.rightMouseDown) then
    sound(2).play(member("Oops"))
end if

// JavaScript syntax
if (_mouse.rightMouseDown) {
    sound(2).play(member("Oops"));
}
```

Voir aussi

`emulateMultibuttonMouse`, `Souris`

rightMouseUp

Syntaxe

```
-- Lingo syntax
_mouse.rightMouseUp

// JavaScript syntax
_mouse.rightMouseUp;
```

Description

Propriété de souris ; indique si le bouton droit de la souris (Windows) ou le bouton de la souris et la touche Ctrl (Mac) sont relâchés (`TRUE`) ou enfoncés (`FALSE`). Lecture seule.

Sur le Mac, la propriété `rightMouseUp` présente la valeur `TRUE` uniquement si la propriété `emulateMultiButtonMouse` renvoie elle-même la valeur `TRUE`.

Exemple

L'instruction suivante vérifie si le bouton droit de la souris (sous Windows) est relâché et, le cas échéant, lit le son Cliquez :

```
-- Lingo syntax
if (_mouse.rightMouseUp) then
    sound(2).play(member("Click Me"))
end if

// JavaScript syntax
if (_mouse.rightMouseUp) {
    sound(2).play(member("Click Me"));
}
```

Voir aussi

emulateMultibuttonMouse, Souris

romanLingo

Syntaxe

```
the romanLingo
```

Description

Propriété système ; spécifie si Lingo utilise un interprète simple octet (TRUE) ou double octet (FALSE).

L'interprète Lingo est plus rapide avec les jeux de caractères codés sur un octet. Certaines versions du logiciel système Mac (la version japonaise par exemple) utilisent un jeu de caractères sur deux octets. En revanche, le système français utilise un jeu de caractères sur un octet. Normalement, la propriété romanLingo est définie au démarrage de Director en fonction de la version locale du logiciel système.

Si vous utilisez un système de script double octet, mais que vous n'utilisez aucun caractère codé sur deux octets dans votre script, attribuez la valeur TRUE à cette propriété afin d'accélérer l'exécution des scripts Lingo.

Exemple

L'instruction suivante définit la propriété romanLingo sur la valeur TRUE, ce qui entraîne l'utilisation par Lingo d'un jeu de caractères sur un octet :

```
-- Lingo
set the romanLingo to TRUE
```

Voir aussi

inlineImeEnabled

rootLock

Syntaxe

```
member(whichCastmember).model(whichModel).keyframePlayer.rootLock
member(whichCastmember).model(whichModel).bonesPlayer.rootLock
```

Description

Propriété 3D de modificateur #keyframePlayer et #bonesPlayer ; indique si les composants de translation d'un mouvement sont utilisés (FALSE) ou ignorés (TRUE).

La valeur par défaut de cette propriété est FALSE.

Exemple

L'instruction suivante oblige le modèle Martien3, qui constitue le premier modèle, à garder sa position de départ tout en exécutant ses mouvements, produisant ainsi un personnage marchant sur place :

```
-- Lingo
member("newalien").model("Alien3").keyframePlayer.rootLock = 1

// Javascript
member("newalien").getPropRef("model",1).keyframePlayer.rootLock=1
```

rootNode

Syntaxe

```
member(whichCastmember).camera(whichCamera).rootNode
sprite(whichSprite).camera.rootNode
```

Description

Propriété 3D ; permet de connaître ou de définir les objets visibles dans une image-objet. Lors de la première création d'une caméra, elle présente tous les nœuds de l'univers. La propriété `rootNode` permet de modifier cet affichage en créant une autre vue par défaut qui limite l'affichage à un nœud spécifique et à ses enfants.

Supposons par exemple que la lumière C soit un enfant du modèle A. Si vous attribuez à la propriété `rootNode` la valeur `camera("defaultView").rootNode=model(A)`, l'image-objet affiche uniquement le modèle A éclairé par la lumière C. La valeur par défaut `group("univers")`, indique que tous les nœuds sont utilisés.

Exemple

L'instruction suivante affecte le modèle Pluton à la propriété `rootNode` de la caméra de l'image-objet 5. Seuls Pluton et ses enfants sont visibles dans l'image-objet 5.

```
-- Lingo
sprite(5).camera.rootNode = member("Scene").model("Pluto")

// Javascript
sprite(5).getPropRef("camera",1).rootNode=member("Scene").getPropRef("model",1);
```

rotation

Syntaxe

```
-- Lingo syntax
spriteObjRef.rotation

// JavaScript syntax
spriteObjRef.rotation;
```

Description

Propriété d'image-objet ; contrôle la rotation d'une image-objet animation QuickTime, GIF animé, animation Flash ou bitmap dans son rectangle de délimitation, sans faire pivoter ce rectangle ou le contrôleur de l'image-objet (dans le cas de QuickTime). Lecture/écriture.

En fait, le rectangle de délimitation de l'image-objet agit comme une fenêtre à travers laquelle vous pouvez voir l'animation Flash ou QuickTime. Le rectangle de délimitation d'un bitmap et d'un GIF animé change en fonction de la rotation de l'image.

La rotation du scénario ne fonctionne dans une animation Flash que si la propriété `obeyScoreRotation` est définie sur `TRUE`.

Une animation Flash pivote autour de son point d'origine spécifié par sa propriété `originMode`. Une animation QuickTime pivote autour du centre du rectangle de délimitation de l'image-objet. Un bitmap pivote autour du point d'alignement de l'image.

Pour un média QuickTime, si la propriété `crop` de l'image-objet présente la valeur `TRUE`, une rotation fréquente de l'image-objet déplace une partie de l'image hors de la zone visible ; lorsque la propriété `crop` de l'image-objet présente la valeur `FALSE`, l'image est mise à l'échelle pour tenir dans le rectangle de délimitation (ce qui peut provoquer une distorsion de l'image).

Vous spécifiez la rotation en degrés sous la forme d'un nombre à virgule flottante.

Le scénario peut conserver des informations sur la rotation d'une image de +21 474 836,47° à -21 474 836,48°, ce qui permet 59 652 rotations complètes dans chaque direction.

Lorsque la limite de rotation est atteinte (juste après la 59 652ème rotation), la commande rétablit le réglage sur +116,47° ou -116,48° – et non 0,00°. Cela s'explique par le fait que +21 474 836,47° est égal à +116,47° et -21 474 836,48° est égal à -116,48° (ou +243,12°). Pour éviter ce réglage, vous devez contraindre les angles à ±360° lorsque vous effectuez une rotation continue à l'aide d'un script.

La valeur par défaut de cette propriété est 0.

Exemple

Le comportement suivant fait pivoter continuellement l'image-objet de 2 degrés chaque fois que la tête de lecture avance, tout en limitant l'angle à 360 degrés :

```
-- Lingo syntax
property spriteNum

on prepareFrame me
    sprite(spriteNum).rotation = integer(sprite(spriteNum).rotation + 2) mod 360
end

// JavaScript syntax
function prepareFrame() {
    sprite(this.spriteNum).rotation = parseInt(sprite(this.spriteNum).rotation + 2) % 360;
}
```

Le script d'image suivant définit une boucle pour la tête de lecture sur l'image en cours pendant qu'il fait pivoter une image-objet QuickTime dans la piste 5 de 360 degrés par incréments de 16 degrés. Lorsque l'image-objet a pivoté de 360 degrés, la tête de lecture continue avec l'image suivante.

```
-- Lingo syntax
on rotateMovie(whichSprite)
    repeat with i = 1 to 36
        sprite(whichSprite).rotation = i * 10
        _movie.updateStage()
    end repeat
end

// JavaScript syntax
function rotateMovie(whichSprite) {
```

```
    for (var i = 1; i <= 36; i++) {
        sprite(whichSprite).rotation = i * 10;
        _movie.updateStage();
    }
}
```

Voir aussi

obeyScoreRotation, originMode, Image-objet

rotation (fond et recouvrement)

Syntaxe

```
sprite(whichSprite).camera.backdrop[backdropIndex].rotation
member(whichCastmember).camera(whichCamera).backdrop
[backdropIndex].rotation
sprite(whichSprite).camera.overlay[overlayIndex].rotation
member(whichCastmember).camera[cameraIndex].overlay
[overlayIndex].rotation
```

Description

Propriété 3D ; permet de connaître ou de définir la rotation du fond ou du recouvrement vers la caméra par défaut. La valeur par défaut de cette propriété est 0,0.

Exemple

L'instruction suivante fait tourner un fond de 60 degrés autour de son point d'alignement.

```
sprite(4).camera.backdrop[1].rotation = 60.0
```

Voir aussi

bevelDepth, transform (propriété)

rotation (matériau de gravure)

Syntaxe

```
member(whichCastmember).shader(whichShader).rotation
member(whichCastmember).model(whichModel).shader.rotation
member(whichCastmember).model(whichModel).shaderList[index].rotation
```

Description

Propriété 3D de matériau de gravure ; permet d'obtenir ou de définir un angle en degrés (exprimé sous forme de nombre à virgule flottante), qui décrit un décalage de rotation 2D pour les lignes gravées. La valeur par défaut de cette propriété est 0,0.

Exemple

L'instruction suivante fait pivoter de 1 degré les lignes utilisées pour dessiner le matériau de gravure du modèle gbCyl3 :

```
member("scene").model("gbCyl3").shader.rotation = \
member("scene").model("gbCyl3").shader.rotation + 1
```

Voir aussi

transform (propriété)

rotation (transformation)

Syntaxe

```
member(whichCastmember).node(whichNode).transform.rotation
member(whichCastmember).node(whichNode).getWorldTransform().rotation
transform.rotation
```

Description

Propriété 3D ; permet d'obtenir ou de définir le composant de rotation d'une transformation. Une transformation définit une échelle, une position et une rotation au sein d'un cadre de référence donné. La valeur par défaut de cette propriété est `vector(0,0,0)`.

Un nœud peut être un objet de caméra, groupe, lumière ou modèle. La définition de la propriété `rotation` d'une transformation de nœud définit la rotation de cet objet dans le cadre de référence de la transformation. La définition de la propriété `rotation` de la transformation d'un objet par rapport à l'univers à l'aide de `getWorldTransform().rotation` définit la rotation de l'objet par rapport à l'origine de l'univers. La définition de la propriété `rotation` de la transformation d'un objet par rapport à son parent à l'aide de `transform.rotation` définit la rotation de l'objet par rapport à son nœud parent.

Si vous souhaitez modifier l'orientation d'une transformation, il est recommandé d'utiliser les méthodes `rotate` et `prerotate` au lieu de définir cette propriété.

Exemple

L'instruction suivante définit la rotation de la première caméra de l'acteur par rapport au parent sur la valeur `vector(0,0,0)` :

```
member("Space").camera[1].transform.rotation = vector(0, 0, 0)
```

L'exemple suivant affiche la rotation par rapport au parent du modèle Lune, ajuste ensuite l'orientation du modèle à l'aide de la commande rotate, puis affiche la rotation résultante du modèle par rapport à l'univers :

```
put member("SolarSys").model("Moon").transform.rotation
-- vector( 0.0000, 0.0000, 45.0000)
member("SolarSys").model("Moon").rotate(15,15,15)
put member("SolarSys").model("Moon").getWorldTransform().rotation
--vector( 51.3810, 16.5191, 65.8771 )
```

Voir aussi

getWorldTransform(), preRotate, rotate, transform (propriété), position (transformation), scale (transformation)

rotationReset

Syntaxe

```
member(whichCastmember).model(whichModel).bonesPlayer.rotationReset
member(whichCastmember).model(whichModel).keyframePlayer.rotationReset
```

Description

Propriété 3D de modificateur `keyframePlayer` et `bonesPlayer` ; indique les axes autour desquels les modifications de rotation sont conservées de la fin d'un mouvement au début du suivant ou de la fin de l'itération d'un mouvement en boucle au début de l'itération suivante.

Les valeurs possibles de cette propriété sont `#none`, `#x`, `#y`, `#z`, `#xy`, `#yz`, `#xz` et `#all`. La valeur par défaut est `#all`.

Exemple

L'instruction suivante attribue à la propriété `rotationReset` du modèle Monstre la valeur de l'axe des *z*. Le modèle conserve sa rotation autour de son axe des *z* lorsque le mouvement ou la boucle en cours d'exécution prend fin.

```
member("NewAlien").model("Monster").bonesPlayer.rotationReset = #z
```

Voir aussi

`positionReset`, `bonesPlayer (modificateur)`

RTF

Syntaxe

```
-- Lingo syntax
memberObjRef.RTF

// JavaScript syntax
memberObjRef.RTF;
```

Description

Propriété d'acteur ; permet l'accès au texte et aux balises contrôlant la mise en page du texte d'un acteur texte contenant du texte RTF.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche dans la fenêtre Messages les informations de formatage RTF incluses dans l'acteur texte CV :

```
--Lingo syntax
put(member("Resume").RTF)

// JavaScript syntax
trace(member("Resume").RTF);
```

Voir aussi

`HTML`, `importFileInto()`

safePlayer

Syntaxe

```
-- Lingo syntax
_player.safePlayer
```

```
// JavaScript syntax
_player.safePlayer;
```

Description

Propriété de lecteur ; contrôle si les fonctions de sécurité de Director sont activées ou non. Lecture/écriture.

Dans une animation comportant du contenu Shockwave, cette propriété peut être testée, mais non définie. Elle présente toujours la valeur `TRUE` dans Shockwave Player.

Dans l'environnement auteur et dans les projections, la valeur par défaut est `FALSE`. Cette propriété peut être renvoyée, mais ne peut être définie que sur `TRUE`. Une fois qu'elle est définie sur `TRUE`, il est impossible de la redéfinir sur `FALSE` sans redémarrer Director ou la projection.

Lorsque la valeur de `safePlayer` est `TRUE`, les fonctions de sécurité suivantes sont activées :

- Seuls les Xtras sûrs peuvent être utilisés.
- La propriété `safePlayer` ne peut pas être remise à zéro.
- Le collage du contenu du Presse-papiers avec la méthode `pasteClipBoardInto()` génère une boîte de dialogue d'avertissement et permet à l'utilisateur d'annuler l'opération.
- L'enregistrement d'une animation ou d'une distribution à l'aide d'un script est désactivé.
- L'impression à l'aide de la méthode `printFrom()` est désactivée.
- L'ouverture d'une application avec la méthode `open()` est désactivée.
- La possibilité d'arrêter une application ou d'éteindre l'ordinateur de l'utilisateur à l'aide des méthodes `restart()` or `shutDown()` est désactivée.
- L'ouverture d'un fichier qui n'est pas dans le dossier DSWMedia est désactivée.
- La détermination d'un nom de fichier local est désactivée.
- L'utilisation de `getNetText()` ou de `postNetText()` ou l'accès à une adresse URL ne possédant pas le même domaine que l'animation entraînent l'affichage d'une boîte de dialogue de sécurité.

Exemple

L'instruction suivante recherche la valeur définie pour la propriété `safePlayer` dans l'environnement auteur.

```
-- Lingo
put( _player.safePlayer)
--0

// Javascript
trace(_player.safePlayer)
// 0
```

Voir aussi

Lecteur

sampleCount

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.sampleCount
```

```
// JavaScript syntax
soundChannelObjRef.sampleCount;
```

Description

Propriété de piste audio ; indique le nombre de sons échantillonnés dans le son en cours de lecture dans une piste audio. Lecture seule.

Il s'agit du nombre total d'échantillons qui dépend des propriétés `sampleRate` et `duration` du son. Ce nombre ne dépend pas de la propriété `channelCount` du son.

Un son de 1 seconde et de 44,1 KHz contient 44 100 échantillons.

Exemple

L'instruction suivante affiche dans la fenêtre Messages le nom et la valeur `sampleCount` de l'acteur lu dans la piste audio 1 :

```
-- Lingo syntax
put("Sound cast member" && sound(1).member.name && "contains" && sound(1).sampleCount &&
"samples.")

// JavaScript syntax
put("Sound cast member " + sound(1).member.name + " contains " +sound(1).sampleCount + "
samples.");
```

Voir aussi

sampleRate, Piste audio

sampleRate

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.sampleRate

// JavaScript syntax
soundChannelObjRef.sampleRate;
```

Description

Propriété de piste audio ; renvoie, en échantillons par seconde, la fréquence d'échantillonnage de l'acteur son ou, dans le cas d'un son SWA, du fichier d'origine codé en Shockwave Audio. Lecture seule.

Cette propriété n'est disponible qu'après le début de la lecture du son SWA ou après le préchargement du fichier à l'aide de la méthode `preLoadBuffer()`. Lorsqu'une une piste audio est donnée, le résultat est le taux d'échantillonnage de l'acteur son en cours de lecture dans cette piste audio.

Les valeurs types sont 8 000, 11 025, 16 000, 22 050 et 44 100.

Lorsque plusieurs sons sont placés en file d'attente dans une piste audio, Director les lit tous avec les valeurs `channelCount`, `sampleRate` et `sampleSize` du premier son en attente et effectue un nouvel échantillonnage des autres pour une lecture plus fluide. Director ne réinitialise ces propriétés qu'après le traitement de tous les sons de la file d'attente ou lors de l'exécution d'une méthode `stop()`. Le prochain son placé en file d'attente ou lu détermine ensuite les nouveaux paramètres.

Exemple

L'instruction suivante affecte la fréquence d'échantillonnage d'origine du fichier utilisé dans l'acteur SWA en flux continu Paul Robin à l'acteur champ Qualité audio :

```
-- Lingo syntax
member("Sound Quality").text = string(member("Paul Robeson").sampleRate)

// JavaScript syntax
member("Sound Quality").text = member("Paul Robeson").sampleRate.toString();
```

L'instruction suivante affiche dans la fenêtre Messages le taux d'échantillonnage du son lu dans la piste audio 1 :

```
-- Lingo syntax
trace(sound(1).sampleRate)

// JavaScript syntax
trace(sound(1).sampleRate);
```

Voir aussi

channelCount, sampleSize, preLoadBuffer(), Piste audio, stop() (piste audio)

sampleSize

Syntaxe

```
-- Lingo syntax
memberObjRef.sampleSize

// JavaScript syntax
memberObjRef.sampleSize;
```

Description

Propriété d'acteur ; détermine la taille d'échantillonnage de l'acteur spécifié. Le résultat est généralement une taille de 8 ou 16 bits. Si une piste audio est donnée, la valeur est celle de l'acteur son en cours de lecture dans la piste audio concernée.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la taille d'échantillonnage de l'acteur son Voix-off et affecte cette valeur à la variable `soundSize` :

```
-- Lingo syntax
soundSize = member("Voice Over").sampleSize

// JavaScript syntax
var soundSize = member("Voice Over").sampleSize;
```

L'instruction suivante affiche dans la fenêtre Messages la taille d'échantillonnage du son lu dans la piste audio 1 :

```
-- Lingo syntax
put(sound(1).sampleSize)

// JavaScript syntax
put(sound(1).sampleSize);
```

savew3d

Syntaxe

```
--Lingo Syntax
member(whichcastmember).savew3d(Absolute path (optional argument))

// JavaScript syntax
member(whichcastmember).savew3d(Absolute path (optional argument));
```

Description

Enregistre l'univers 3D à partir d'une projection.

Le comportement de cette méthode est le suivant :

- Si l'emplacement absolu est indiqué, cette méthode enregistre la scène 3D à cet instant précis dans le fichier dont le chemin est spécifié (sauf si le chemin est erroné). L'enregistrement s'effectue en mode auteur ou dans la projection.
- Si l'emplacement absolu n'est pas indiqué, c'est-à-dire que l'appel est membername.savew3d(), le comportement est le suivant :
  - Si le membre w3d est lié de façon externe et que savew3d est enregistré en mode de projection, la scène 3D est alors enregistrée à cet instant précis dans le fichier w3d lié de façon externe, en écrasant le fichier.
  - Si le membre w3d est interne et que savew3d est enregistré en mode de projection, cet appel de méthode est alors ignoré.

En mode auteur, la méthode savew3d se comporte de la même manière que saveWorld. Le membre est sélectionné pour l'écriture et l'enregistrement 3D est effectué après l'enregistrement du fichier Director.

Exemple

Cette instruction enregistre les modifications apportées au membre (« 3dworld ») dans un fichier séparé intitulé savedworld.w3d.

```
-- Lingo syntax
member("3dworld").savew3d(the moviepath & "savedworld.w3d")

// JavaScript syntax
member("3dworld").savew3d(the moviepath & "savedworld.w3d");
```

saveWorld

Syntaxe

```
--Lingo Syntax
member(whichcastmember).saveWorld()

// JavaScript syntax
member(whichcastmember).saveWorld();
```

Description

Enregistre l'univers 3D existant de l'acteur 3D après avoir enregistré l'animation.

Exemple

Cette instruction enregistre les modifications apportées au membre (« 3dworld ») dans un fichier séparé intitulé savedworld.w3d.

```
-- Lingo syntax
member("3dworld").saveWorld()

// JavaScript syntax
member("3dworld").saveWorld();
```

scale (3D)

Syntaxe

```
member(whichCastmember).camera(whichCamera).backdrop[backdropIndex].scale
member(whichCastmember).camera(whichCamera).overlay[overlayIndex].scale
```

Description

Propriété 3D ; permet d'obtenir ou de définir la valeur d'échelle utilisée par un recouvrement ou un fond spécifique dans la liste des recouvrements ou des fonds de la caméra référencée à afficher. La largeur et la hauteur du fond ou du recouvrement sont multipliées par la valeur de l'échelle. La valeur par défaut de cette propriété est 1,0.

Exemple

L'instruction suivante double la taille d'un fond.

```
-- Lingo
sprite(25).camera.backdrop[1].scale = 2.0

// Javascript
sprite(25).getPropRef("camera",1).getProp("backDrop",1).scale=2.0;
```

Voir aussi

bevelDepth, overlay

scale (fond et recouvrement)

Syntaxe

```
member(whichCastmember).camera(whichCamera).backdrop[backdropIndex].scale
member(whichCastmember).camera(whichCamera).overlay[overlayIndex].scale
```

Description

Propriété 3D ; permet d'obtenir ou de définir la valeur d'échelle utilisée par un recouvrement ou un fond spécifique dans la liste des recouvrements ou des fonds de la caméra référencée à afficher. La largeur et la hauteur du fond ou du recouvrement sont multipliées par la valeur de l'échelle. La valeur par défaut de cette propriété est 1,0.

Exemple

L'instruction suivante double la taille d'un fond.

```
sprite(25).camera.backdrop[1].scale = 2.0
```

Voir aussi

bevelDepth, overlay

scale (acteur)

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.scale

// JavaScript syntax
memberOrSpriteObjRef.scale;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la mise à l'échelle d'une image-objet animation QuickTime, forme vectorielle ou Flash.

Pour QuickTime, cette propriété ne met pas à l'échelle le rectangle de délimitation ou le contrôleur de l'image-objet. En revanche, elle met à l'échelle l'image autour de son centre dans le rectangle de délimitation. La mise à l'échelle est spécifiée sous forme d'une liste Director contenant deux pourcentages enregistrés en tant que nombres à virgule flottante :

```
[xPercent, yPercent]
```

Le paramètre *pourcentageX* spécifie la mise à l'échelle horizontale et le paramètre *pourcentageY* spécifie la mise à l'échelle verticale.

Lorsque la propriété `crop` de l'image-objet est définie sur `TRUE`, la propriété `scale` permet de simuler un zoom dans le rectangle de délimitation de l'image-objet. Lorsque la propriété `crop` de l'image-objet est définie sur `FALSE`, la propriété `scale` est ignorée.

Cette propriété peut être testée et définie. La valeur par défaut est `[1.0000,1.0000]` .

Pour les acteurs animation Flash ou forme vectorielle, la mise à l'échelle est exprimée sous la forme d'un nombre à virgule flottante. L'animation est mise à l'échelle depuis son point d'origine, comme spécifié par sa propriété `originMode`.

Remarque : *cette propriété doit avoir la valeur par défaut si la propriété scaleMode a pour valeur #autoSize. Sinon, l'image-objet n'est pas correctement affichée.*

Exemple

Le gestionnaire suivant accepte une référence à une image-objet d'animation Flash sous la forme d'un paramètre, réduit l'échelle de l'animation à 0 % (de sorte qu'elle disparaisse), puis la remet à l'échelle par incréments de 5 % jusqu'à ce qu'elle retrouve sa taille normale (100 %) :

```
-- Lingo syntax
on scaleMovie whichSprite
    sprite(whichSprite).scale = 0
    _movie.updatestage()
    repeat with i = 1 to 20
        sprite(whichSprite).scale = i * 5
        _movie.updatestage()
    end repeat
end

// JavaScript syntax
function scaleMovie(whichSprite) {
    sprite(whichSprite).scale = 0;
    _movie.updateStage();
    var i = 1;
    while (i < 21) {
```

```
        sprite(whichSprite).scale = i * 5;
        _movie.updateStage();
        i++;
    }
}
```

Voir aussi

scaleMode, originMode

scale (transformation)

Syntaxe

```
member(whichCastmember).node(whichNode).transform.scale
member(whichCastmember).node(whichNode).getWorldTransform().scale
transform.scale
```

Description

Propriété 3D ; permet d'obtenir ou de définir le composant de redimensionnement d'une transformation. Une transformation définit une échelle, une position et une rotation au sein d'un cadre de référence donné. La propriété `scale` permet d'obtenir ou de définir le degré de redimensionnement de la transformation pour chacun des trois axes. La valeur par défaut de cette propriété est `vector(1.0,1.0,1.0)`.

Un nœud peut être un objet de caméra, groupe, lumière ou modèle. Cette commande n'a aucun effet visuel sur les lumières ou les caméras car elles ne contiennent pas de géométrie. La définition de la propriété `scale` d'une transformation de nœud définit le redimensionnement de cet objet le long des axes x, y et z, dans le cadre de référence de la transformation. L'obtention de la propriété `scale` de la transformation d'un objet par rapport à l'univers à l'aide de `getWorldTransform().scale` renvoie le redimensionnement de l'objet par rapport à l'origine de l'univers. La définition de la propriété `scale` de la transformation d'un objet par rapport à son parent à l'aide de `transform.scale` définit le redimensionnement de l'objet par rapport à son nœud parent.

Exemple

L'instruction suivante attribue à la propriété `scale` de la transformation du modèle Lune la valeur vector(2, 5, 3) :

```
member("Scene").model("Moon").transform.scale = vector(2,5,3)
```

Voir aussi

transform (propriété), getWorldTransform(), position (transformation), rotation (transformation), scale (commande)

scaleMode

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.scaleMode

// JavaScript syntax
memberOrSpriteObjRef.scaleMode;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la manière dont une animation Flash ou une forme vectorielle est mise à l'échelle dans le rectangle de délimitation d'une image-objet. Lorsque vous mettez à l'échelle une image-objet d'animation Flash en définissant ses propriétés `scale` et `viewScale`, cette image n'est pas mise à l'échelle ; seule la vue de l'animation dans l'image-objet l'est. La propriété `scaleMode` peut prendre les valeurs suivantes :

- `#showAll` (valeur par défaut pour les animations Director antérieures à la version 7) : Conserve les proportions de l'acteur animation Flash d'origine. Si nécessaire, remplissez tout intervalle vide dans la dimension horizontale ou verticale avec la couleur d'arrière-plan.
- `#noBorder` : Conserve les proportions de l'acteur animation Flash d'origine. Si nécessaire, recadrez la dimension horizontale ou verticale.
- `#exactFit` : Ne conserve pas les proportions de l'acteur animation Flash d'origine. Etirez l'animation Flash pour lui donner les dimensions exactes de l'image-objet.
- `#noScale` : Conserve la taille d'origine du média Flash, quel que soit le mode de mise à l'échelle de l'image-objet sur la scène. Si l'image-objet devient plus petite que l'animation Flash d'origine, l'animation affichée dans l'image-objet est recadrée pour tenir dans les limites de l'image-objet.
- `#autoSize` (valeur par défaut) : spécifie que le rectangle de l'image-objet est automatiquement mis à l'échelle et positionné en fonction des propriétés `rotation`, `skew`, `flipH` et `flipV`. Autrement dit, lorsqu'une image-objet Flash pivote, elle n'est pas recadrée comme dans les versions précédentes de Director. Le paramètre `#autoSize` ne fonctionne correctement que si les propriétés `scale`, `viewScale`, `originPoint` et `viewPoint` sont définies sur leur valeur par défaut.

Cette propriété peut être testée et définie.

Exemple

Le script d'image-objet suivant vérifie la couleur de la scène de l'animation Director et, si cette couleur est indexée à la position 0 de la palette en cours, il définit la propriété `scaleMode` d'une image-objet d'animation Flash sur la valeur `#showAll`. Dans le cas contraire, il définit la propriété `scaleMode` sur `#noBorder`.

```
-- Lingo syntax
property spriteNum

on beginsprite me
    if _movie.stage.bgColor = 0 then
        sprite(spriteNum).scaleMode = #showAll
    else
        sprite(spriteNum).scaleMode = #noBorder
    end if
end

// JavaScript syntax
function beginsprite() {
    var stgClr = _movie.stage.bgColor;
    if (stgClr == 0) {
        sprite(this.spriteNum).scaleMode = symbol("showAll");
    } else {
        sprite(this.spriteNum).scaleMode = symbol("noBorder");
    }
}
```

Voir aussi

scale (acteur)

score

Syntaxe

```
-- Lingo syntax
_movie.score

// JavaScript syntax
_movie.score;
```

Description

Propriété d'animation ; détermine le scénario associé à l'animation actuelle. Lecture/écriture.

Cette propriété peut servir à conserver le contenu actuel du scénario avant de l'effacer, ainsi qu'à générer un nouveau contenu de scénario ou à affecter le contenu actuel à une boucle.

Exemple

L'instruction suivante affecte l'acteur boucle Cascade au scénario de l'animation en cours :

```
-- Lingo syntax
_movie.score = member("Waterfall").media

// JavaScript syntax
_movie.score = member("Waterfall").media;
```

Voir aussi

Animation

scoreColor

Syntaxe

```
sprite(whichSprite).scoreColor
the scoreColor of sprite whichSprite
```

Description

Propriété d'image-objet ; indique la couleur de scénario affectée à l'image-objet spécifiée par *quelleImageObjet*. Les valeurs possibles correspondent aux puces de couleur 0 à 5 de la palette actuelle.

Cette propriété peut être testée et définie. La définition de cette propriété n'est utile que pendant la programmation et l'enregistrement du scénario.

Exemple

L'instruction suivante affiche dans la fenêtre Messages la valeur de la couleur de scénario affectée à l'image-objet 7 :

```
put sprite(7).scorecolor
```

scoreSelection

Syntaxe

```
-- Lingo syntax
_movie.scoreSelection
```

```
// JavaScript syntax
_movie.scoreSelection;
```

Description

Propriété d'animation ; détermine les pistes sélectionnées dans la fenêtre Scénario. Lecture/écriture.

Les informations sont formatées sous forme d'une liste linéaire de listes linéaires. Chaque sélection contiguë figure dans un format de liste comprenant le numéro de la piste initiale, le numéro de la piste finale, le numéro de l'image initiale et celui de l'image finale. Spécifiez les pistes d'images-objets en indiquant leur numéro de piste. Utilisez les numéros suivants pour indiquer d'autres pistes.

Pour spécifier :	**Utilisez :**
Piste des scripts de l'image	0
Piste audio 1	-1
Piste audio 2	-2
Piste des transitions	-3
Piste des palettes	-4
Piste des cadences	-5

Vous pouvez sélectionner des pistes ou des images non adjacentes.

Exemple

L'instruction suivante sélectionne les pistes d'images-objets 15 à 25 dans les images 100 à 200 :

```
-- Lingo syntax
_movie.scoreSelection = [[15, 25, 100, 200]]

// JavaScript syntax
_movie.scoreSelection = list(list(15, 25, 100, 200));
```

L'instruction suivante sélectionne les pistes d'images-objets 15 à 25 et 40 à 50 dans les images 100 à 200 :

```
-- Lingo syntax
_movie.scoreSelection = [[15, 25, 100, 200], [40, 50, 100, 200]]

// JavaScript syntax
_movie.scoreSelection = list(list(15, 25, 100, 200), list(40, 50, 100, 200));
```

L'instruction suivante sélectionne le script d'image des images 100 à 200 :

```
-- Lingo syntax
_movie.scoreSelection = [[0, 0, 100, 200]]

// JavaScript syntax
_movie.scoreSelection = list(list(0, 0, 100, 200));
```

Voir aussi

Animation

script

Syntaxe

```
-- Lingo syntax
_movie.script[scriptNameOrNum]

// JavaScript syntax
_movie.script[scriptNameOrNum];
```

Description

Propriété d'animation ; permet d'accéder par index ou par nom aux acteurs script d'une animation. Lecture seule.

L'argument *nomOuNumScript* peut être une chaîne spécifiant le nom de l'acteur script ou un nombre entier indiquant le numéro de cet acteur.

- Si *nomOuNumScript* est une chaîne, la propriété `script` permet d'accéder à l'acteur script, quelle que soit la bibliothèque de distribution contenant cet acteur.
- Si *nomOuNumScript* est un nombre entier, la propriété `script` permet uniquement d'accéder à l'acteur script trouvé dans la première bibliothèque de distribution de l'animation référencée ; vous ne pouvez pas utiliser un accès par index pour spécifier une bibliothèque de distribution différente de la première.

Exemple

L'instruction suivante accède au script dont le nom est indiqué.

```
-- Lingo syntax
bugScript = _movie.script["Warrior Ant"]

// JavaScript syntax
var bugScript = _movie.script["Warrior Ant"];
```

Voir aussi

Animation

scripted

Syntaxe

```
-- Lingo syntax
spriteChannelObjRef.scripted

// JavaScript syntax
spriteChannelObjRef.scripted;
```

Description

Propriété de piste d'image-objet ; spécifie si une piste d'image-objet est contrôlée par un script (`TRUE`) ou par le scénario (`FALSE`). Lecture seule.

Exemple

Les instructions suivantes créent une image-objet contrôlée par un script à partir de l'acteur cerf-volant dans la piste d'image-objet 5 si cette piste n'est pas déjà sous le contrôle d'un script.

```
-- Lingo syntax
if (channel(5).scripted = FALSE) then
```

```
    channel(5).makeScriptedSprite(member("kite"))
end if

// JavaScript syntax
if (channel(5).scripted == false) {
    channel(5).makeScriptedSprite(member("kite"));
}
```

Voir aussi

Piste d'image-objet

scriptingXtraList

Syntaxe

```
-- Lingo syntax
_player.scriptingXtraList

// JavaScript syntax
_player.scriptingXtraList;
```

Description

Propriété de lecteur ; renvoie une liste linéaire de tous les Xtras de programmation disponibles dans le lecteur Director. Lecture seule.

Il s'agit des Xtras figurant dans le dossier Configuration\Xtras.

Exemple

L'instruction suivante affiche dans la fenêtre Messages tous les Xtras de programmation disponibles :

```
-- Lingo syntax
trace(_player.scriptingXtraList)

// JavaScript syntax
trace(_player.scriptingXtraList);
```

Voir aussi

mediaXtraList, Lecteur, Objets de programmation, toolXtraList, transitionXtraList, xtraList (lecteur)

scriptInstanceList

Syntaxe

```
sprite(whichSprite).scriptInstanceList
the scriptInstanceList of sprite whichSprite
```

Description

Propriété d'image-objet ; crée une liste des références de script liées à une image-objet. Cette propriété n'est disponible que pendant l'exécution. La liste est vide lorsque l'animation n'est pas en cours de lecture. Les modifications apportées à la liste ne sont pas enregistrées dans le scénario. Cette propriété est utile aux tâches suivantes :

- associer un comportement à une image-objet pour une utilisation pendant l'exécution ;

- déterminer si des comportements sont associés à une image-objet et déterminer quels sont ces comportements ;
- rechercher une référence de script de comportement à utiliser avec la commande `sendSprite`.

Cette propriété peut être testée et définie. Elle ne peut être modifiée que si l'image-objet existe déjà et qu'au moins une instance d'un comportement y est liée.

Exemple

Le gestionnaire suivant affiche la liste des références de scripts liées à une image-objet :

```
-- Lingo
on showScriptRefs spriteNum
    put sprite(spriteNum).scriptInstanceList
end

// Javascript
function showScriptRefs(spriteNum)
{
    trace(sprite(spriteNum).scriptInstanceList);
}
```

Les instructions suivantes lient le script Grand bruit à l'image-objet 5 :

```
x = script("Big Noise").new()
sprite(5).scriptInstanceList.add(x)
```

Voir aussi

`scriptNum`, `sendSprite()`

scriptList

Syntaxe

```
sprite(whichSprite).scriptList
the scriptList of sprite whichSprite
```

Description

Propriété d'image-objet ; renvoie la liste des comportements associés à l'image-objet concernée, ainsi que leurs propriétés. Cette propriété ne peut être définie qu'avec `setScriptList()`. Elle ne peut pas être définie au cours d'une session d'enregistrement du scénario.

Exemple

L'instruction suivante affiche la liste des scripts associés à l'image-objet 1 dans la fenêtre Messages :

```
put sprite(1).scriptList
-- [[(member 2 of castLib 1), "[#myRotateAngle: 10.0000, #myClockwise: 1, #myInitialAngle:
0.0000]"], [(member 3 of castLib 1), "[#myAnglePerFrame: 10.0000, #myTurnFrames: 10,
#myHShiftPerFrame: 10, #myShiftFrames: 10, #myTotalFrames: 60, #mySurfaceHeight: 0]"]]
```

Voir aussi

`setScriptList()`, `value()`

scriptNum

Syntaxe

```
sprite(whichSprite). scriptNum
scriptNum of sprite whichSprite
```

Description

Propriété d'image-objet ; indique le numéro du script lié à l'image-objet spécifiée par *quelleImageObjet*. Si plusieurs scripts sont liés à l'image-objet, la propriété d'image-objet `scriptNum` renvoie le numéro du premier script. Pour une liste complète des scripts liés à une image-objet, consultez la liste de ses comportements dans l'Inspecteur de comportement.

Cette propriété peut être testée et définie pendant l'enregistrement du scénario.

Exemple

L'instruction suivante affiche le numéro du script lié à l'image-objet 4 :

```
-- Lingo
put sprite(4).scriptNum

// Javascript
trace(sprite(4).scriptNum);
```

Voir aussi

`scriptInstanceList`

scriptsEnabled

Syntaxe

```
-- Lingo syntax
memberObjRef.scriptsEnabled

// JavaScript syntax
memberObjRef.scriptsEnabled;
```

Description

Propriété d'acteur animation Director ; détermine si les scripts d'une animation liée sont activés (`TRUE` ou 1) ou désactivés (`FALSE` ou 0).

Cette propriété n'est disponible que pour les acteurs animation Director liés.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante désactive les scripts dans l'animation liée Chronique Jazz :

```
-- Lingo syntax
member("Jazz Chronicle").scriptsEnabled = FALSE

// JavaScript syntax
member("Jazz Chronicle").scriptsEnabled = 0;
```

scriptText

Syntaxe

```
-- Lingo syntax
memberObjRef.scriptText

// JavaScript syntax
memberObjRef.scriptText;
```

Description

Propriété d'acteur ; indique le contenu du script éventuellement affecté à un acteur.
Lecture/écriture.

Le texte d'un script est supprimé lorsqu'une animation est convertie en projection, protégée ou compressée au format Shockwave Player. De telles animations perdent alors les valeurs de leur propriété `scriptText`. Par conséquent, les valeurs de la propriété `scriptText` de l'animation ne peuvent pas être récupérées lorsque cette animation est lue par une projection. Toutefois, Director peut définir de nouvelles valeurs pour la propriété `scriptText` dans la projection. Ces scripts nouvellement affectés sont automatiquement compilés pour en garantir l'exécution rapide.

Exemple

L'instruction suivante fait du contenu de l'acteur champ 20 le script de l'acteur 30 :

```
-- Lingo syntax
member(20).text = member(30).scriptText

// JavaScript syntax
member(20).text = member(30).scriptText;
```

Voir aussi

Acteur

scriptType

Syntaxe

```
member whichScript.scriptType
the scriptType of member whichScript
```

Description

Propriété d'acteur ; indique le type du script spécifié. Les valeurs possibles sont `#movie`, `#score` et `#parent`.

Exemple

L'instruction suivante fait de l'acteur script Script principal un script d'animation :

```
-- Lingo
member("Main Script").scriptType = #movie

// Javascript
member("Main Script").scriptType = Symbol("movie");
```

scrollTop

Syntaxe

```
-- Lingo syntax
memberObjRef.scrollTop

// JavaScript syntax
memberObjRef.scrollTop;
```

Description

Propriété d'acteur ; détermine la distance, en pixels, séparant le haut d'un acteur champ et le haut du champ actuellement visible dans la zone de défilement. En changeant la valeur de la propriété d'acteur `scrollTop` pendant la lecture de l'animation, vous pouvez modifier la section du champ qui apparaît dans la zone de défilement.

Cette procédure permet de produire des comportements de défilement personnalisés pour des acteurs texte et champ.

Par exemple, le script Lingo suivant déplace l'acteur champ Crédits vers le haut ou le bas dans la zone d'un champ, suivant la valeur de la variable `valeurDeGlissière` :

```
global sliderVal

on prepareFrame
    newVal = sliderVal * 100
    member("Credits").scrolltop = newVal
end
```

La variable globale `valeurDeGlissière` pourrait mesurer l'ampleur du déplacement d'une glissière par l'utilisateur. L'instruction `set nouvelleValeur = valeurDeGlissière * 100` multiplie `valeurDeGlissière` pour produire une valeur supérieure à celle du déplacement de la glissière par l'utilisateur. Si `valeurDeGlissière` a une valeur positive, le texte se déplace vers le haut ; si `valeurDeGlissière` a une valeur négative, le texte se déplace vers le bas.

Exemple

La boucle de répétition suivante fait défiler le champ Crédits en augmentant continuellement la valeur de la propriété `scrollTop` :

```
--Lingo syntax
on wa
    member("Credits").scrollTop = 1
    repeat with count = 1 to 150
        member("Credits").scrollTop = member("Credits").scrollTop + 1
        _movie.updateStage()
    end repeat
end

// JavaScript syntax
function wa() {
    member("Credits").scrollTop = 1;
    for (var count = 1; count <= 150; count++) {
        member("Credits").scrollTop = member("Credits").scrollTop + 1;
        _movie.updateStage();
    }
}
```

sds (modificateur)

Syntaxe

```
member(whichCastmember).model(whichModel).sds.whichProperty
```

Description

Modificateur 3D ; ajoute des détails géométriques aux modèles et synthétise ces détails supplémentaires pour adoucir les courbes au fur et à mesure que le modèle s'approche de la caméra. Vous pouvez définir les propriétés du modificateur `sds` après avoir ajouté le modificateur `sds` à un modèle avec `addModifier()`.

Le modificateur `sds` affecte directement la ressource de modèle. Faites attention lorsque vous utilisez les modificateurs `sds` et `lod` conjointement, car ils exécutent des opérations opposées (le modificateur `sds` ajoute des détails géométriques alors que le modificateur `lod` les supprime). Avant d'ajouter le modificateur `sds`, il est recommandé d'attribuer à la propriété de modificateur `lod.auto` la valeur `FALSE` et de choisir la résolution souhaitée pour la propriété `lod.level` en procédant comme suit :

```
member("myMember").model("myModel").lod.auto = 0
member("myMember").model("myModel").lod.level = 100
member("myMember").model("myModel").addmodifier(#sds)
```

Le modificateur `sds` n'est pas utilisable avec les modèles utilisant déjà les modificateurs `inker` ou `toon`.

Après avoir ajouté le modificateur `sds` à une ressource de modèle, vous pouvez obtenir ou définir les propriétés suivantes :

`enabled` indique si la fonctionnalité de fractionnement de surface est activée (`TRUE`) ou désactivée (`FALSE`). La valeur par défaut de cette propriété est `TRUE`.

`depth` spécifie le nombre maximal de niveaux de résolution que le modèle peut afficher lors de l'utilisation du modificateur `sds`.

`error` indique le niveau de tolérance des erreurs pour la fonctionnalité de fractionnement de surface. Cette propriété s'applique uniquement lorsque la propriété `sds.subdivision` présente la valeur `#adaptive`.

`subdivision` indique le mode de fonctionnement du modificateur de fractionnement de surface. Les valeurs possibles sont :

- `#uniform` spécifie que la maille est mise à l'échelle en détail de façon uniforme, chaque face étant fractionnée le même nombre de fois.
- `#adaptive` spécifie que des détails supplémentaires ne sont ajoutés qu'en présence de changements d'orientation majeurs et uniquement aux zones actuellement visibles de la maille.

Remarque : *pour plus d'informations, consultez les entrées des différentes propriétés.*

Exemple

L'instruction suivante affiche la valeur de la propriété `sds.depth` du modèle Terrain :

```
-- Lingo
put member("3D").model("Terrain").sds.depth
-- 2

// Javascript
trace(member("3D").getPropRef("model",1).sds.depth
// 2
```

Voir aussi

lod (modificateur), toon (modificateur), inker (modificateur), depth (3D), enabled (sds), error, subdivision, addModifier

searchCurrentFolder

Syntaxe

```
-- Lingo syntax
_player.searchCurrentFolder

// JavaScript syntax
_player.searchCurrentFolder;
```

Description

Propriété de lecteur ; détermine si Director recherche des noms de fichier dans le dossier en cours. Lecture/écriture.

- Lorsque la propriété `searchCurrentFolder` présente la valeur `TRUE` (1), Director recherche les noms de fichier dans le dossier en cours.
- Lorsque la propriété `searchCurrentFolder` présente la valeur `FALSE` (0), Director ne recherche pas les noms de fichier dans le dossier en cours.

La valeur par défaut de cette propriété est `TRUE`.

Remarque : *_player.searchCurrentFolder est uniquement disponible en mode d'exécution Author (environnement auteur) et Projector (projection) et n'est pas accessible en mode Plugin (module de navigateur Web).*

Exemple

L'instruction suivante affiche l'état de la propriété `searchCurrentFolder` dans la fenêtre Messages. Le résultat est 1, qui constitue l'équivalent numérique de `TRUE` :

```
-- Lingo syntax
put(_player.searchCurrentFolder)

// JavaScript syntax
put(_player.searchCurrentFolder);
```

Voir aussi

Lecteur

searchPathList

Syntaxe

```
-- Lingo syntax
_player.searchPathList

// JavaScript syntax
_player.searchPathList;
```

Description

Propriété de lecteur ; liste des chemins d'accès dans lesquels Director effectue une recherche pour trouver des médias liés tels que les fichiers vidéo numérique, GIF, bitmap ou audio. Lecture/écriture.

Chaque élément de la liste des chemins d'accès est un nom de chemin d'accès complet tel qu'il apparaît sur la plate-forme en cours pendant l'exécution.

La valeur de `searchPathList` est une liste linéaire que vous pouvez manipuler comme n'importe quelle autre liste à l'aide de commandes telles que `add()`, `addAt()`, `append()`, `deleteAt()` et `setAt()`. La valeur par défaut est une liste vide.

Les adresses URL ne doivent pas être utilisées comme référence de fichier pour la recherche.

L'ajout d'un grand nombre de chemins d'accès à `searchPaths` ralentit la recherche. Essayez de minimiser le nombre de chemins figurant dans la liste.

Remarque : *cette propriété fonctionne sur toutes les animations ultérieures après avoir été définie. Les éléments de l'animation actuelle ayant déjà été chargés, la modification du paramètre n'affecte aucun de ces éléments.*

Exemple

L'instruction suivante affiche les chemins d'accès que Director prend en compte lors de la recherche de fichiers :

```
-- Lingo syntax
trace(_player.searchPathList)

// JavaScript syntax
trace(_player.searchPathList);
```

L'instruction suivante affecte deux dossiers à `searchPaths` sous Windows :

```
-- Lingo syntax
_player.searchPathList = ["C:\Director\Projects\", "D:\CDROM\Sources\"]

// JavaScript syntax
_player.searchPathList = list("C:\\Director\\Projects\\","D:\\CDROM\\Sources\\");
```

L'instruction suivante affecte deux dossiers à searchPaths sur un Mac :

```
-- Lingo syntax
_player.searchPathList = ["Hard Drive:Director:Projects:", "CDROM:Sources:"]

// JavaScript syntax
_player.searchPathList = list("Hard Drive:Director:Projects:","CDROM:Sources:");
```

Voir aussi

Lecteur, searchCurrentFolder

selectedButton

Syntaxe

```
-- Lingo syntax
dvdObjRef.selectedButton

// JavaScript syntax
dvdObjRef.selectedButton;
```

Description

Propriété de DVD ; renvoie l'index du bouton actuellement activé. Lecture seule.

Exemple

L'instruction suivante affiche le bouton sélectionné dans l'objet DVD.

```
-- Lingo
put sprite(1).selectedButton
--1

// Javascript
trace(sprite(1).selectedButton)
//1
```

Voir aussi

DVD

selectedText

Syntaxe

```
-- Lingo syntax
memberObjRef.selectedText

// JavaScript syntax
memberObjRef.selectedText;
```

Description

Propriété d'acteur texte ; renvoie le bloc de texte sélectionné sous forme de référence à seul objet. Cela permet l'accès aux caractéristiques de police, de même qu'aux informations sur les caractères de la chaîne.

Exemple

Le gestionnaire suivant affiche le texte sélectionné et placé dans une variable locale `objet`. Cet objet est ensuite utilisé pour faire référence à différentes caractéristiques du texte, détaillées dans la fenêtre Messages.

```
--Lingo syntax
property spriteNum

on mouseUp(me)
    mySelectionObject = sprite(spriteNum).member.selectedText
    put(mySelectionObject.text)
    put(mySelectionObject.font)
    put(mySelectionObject.fontSize)
    put(mySelectionObject.fontStyle)
end

// JavaScript syntax
function mouseUp() {
    var mySelectionObject = sprite(this.spriteNum).member.selectedText;
    trace(mySelectionObject.text);
    trace(mySelectionObject.font);
    trace(mySelectionObject.fontSize);
    trace(mySelectionObject.fontStyle);
}
```

selection

Syntaxe

```
-- Lingo syntax
castObjRef.selection

// JavaScript syntax
castObjRef.selection;
```

Description

Propriété de bibliothèque de distribution ; renvoie les acteurs sélectionnés dans une fenêtre Distribution donnée. Lecture/écriture.

Exemple

L'instruction suivante sélectionne les acteurs 1 à 10 de la distribution numéro 1 :

```
-- Lingo syntax
castLib(1).selection = [[1, 10]]

// JavaScript syntax
castLib(1).selection = list( list(1, 10) );
```

L'instruction suivante sélectionne les acteurs 1 à 10 et 30 à 40 de la bibliothèque de distribution 1 :

```
-- Lingo syntax
castLib(1).selection = [[1, 10], [30, 40]]

// JavaScript syntax
castLib(1).selection = list( list(1, 10), list(30, 40) );
```

Voir aussi

Bibliothèque de distribution

selection (propriété d'acteur texte)

Syntaxe

```
member(whichTextMember).selection
```

Description

Propriété d'acteur texte ; renvoie une liste du premier et du dernier caractère sélectionnés dans l'acteur texte.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante définit la sélection affichée par l'image-objet de l'acteur texte maRéponse de manière à mettre en surbrillance les caractères 6 à 10 :

```
member("myAnswer").selection = [6, 10]
```

Voir aussi

color(), selStart, selEnd

selEnd

Syntaxe

```
-- Lingo syntax
_movie.selEnd

// JavaScript syntax
_movie.selEnd;
```

Description

Propriété d'animation spécifiant le dernier caractère d'une sélection. Cette propriété s'utilise avec selStart pour identifier une sélection dans le champ modifiable en cours en partant du premier caractère. La valeur maximale de selEnd correspond au nombre de caractères du champ modifiable en cours. Cette propriété s'applique aux images-objets d'acteur champ, mais non aux images-objets d'acteur texte. Si vous définissez une valeur selStart supérieure à la valeur de selEnd, la valeur de selEnd est redéfinie sur celle de selStart.

Cette propriété peut être testée et définie. La valeur par défaut est 0.

Exemple

Les instructions suivantes sélectionnent « cde » à partir du champ « abcdefg » :

```
-- Lingo syntax
_movie.selStart = 2
_movie.selEnd = 5

// JavaScript syntax
_movie.selStart = 2;
_movie.selEnd = 5;
```

L'instruction suivante effectue une sélection d'une longueur de 20 caractères :

```
-- Lingo syntax
_movie.selEnd = _movie.selStart + 20

// JavaScript syntax
_movie.selEnd = _movie.selStart + 20;
```

Voir aussi

editable, hilite (commande), selection() (fonction), selStart, text

selStart

Syntaxe

```
-- Lingo syntax
_movie.selStart

// JavaScript syntax
_movie.selStart;
```

Description

Propriété d'animation spécifiant la position qui précède le caractère initial d'une sélection. Cette propriété s'utilise avec selEnd pour identifier une sélection dans le champ modifiable en cours. La valeur 0 indique une position avant le premier caractère. La valeur maximale de selStart correspond au nombre de caractères du champ modifiable en cours. Cette propriété s'applique aux images-objets d'acteur champ, mais non aux images-objets d'acteur texte. Si vous définissez une valeur selStart supérieure à la valeur de selEnd, la valeur de selEnd est redéfinie sur celle de selStart.

Cette propriété peut être testée et définie. La valeur par défaut est 0.

Exemple

Les instructions suivantes sélectionnent « cde » à partir du champ « abcdefg » :

```
-- Lingo syntax
_movie.selStart = 2
_movie.selEnd = 5

// JavaScript syntax
_movie.selStart = 2;
_movie.selEnd = 5;
```

L'instruction suivante effectue une sélection d'une longueur de 20 caractères :

```
-- Lingo syntax
_movie.selEnd = _movie.selStart + 20

// JavaScript syntax
_movie.selEnd = _movie.selStart + 20;
```

Voir aussi

selection() (fonction), selEnd, text

serialNumber

Syntaxe

```
-- Lingo syntax
_player.serialNumber

// JavaScript syntax
_player.serialNumber;
```

Description

Propriété d'animation ; renvoie une chaîne comprenant le numéro de série saisi pendant l'installation de Director. Lecture seule.

Cette propriété est uniquement disponible dans l'environnement de programmation. Elle peut s'utiliser dans une animation dans une fenêtre personnalisée pour afficher des informations utilisateur.

Exemple

Le gestionnaire suivant serait normalement placé dans un script d'animation dans une fenêtre. Il place le nom de l'utilisateur et le numéro de série dans une zone d'affichage dès l'ouverture de la fenêtre :

```
-- Lingo syntax
on prepareMovie
```

```
displayString = _player.userName & RETURN & _player.organizationName & RETURN &
_player.serialNumber
member("User Info").text = displayString
end

// JavaScript syntax
function prepareMovie() {
var displayString = _player.userName + "\n" + _player.organizationName + "\n" +
_player.serialNumber;
member("User Info").text = displayString;
}
```

Voir aussi

Lecteur

shader

Syntaxe

```
member(whichCastmember).shader(whichShader)
member(whichCastmember).shader[index]
member(whichCastmember).model(whichModel).shader
member(whichCastmember).modelResource(whichModelResource).face[index].shader
```

Description

Propriété 3D d'élément, de modèle et de face ; objet utilisé pour définir l'apparence de la surface du modèle. Le matériau est la « peau » qui entoure la ressource de modèle utilisée par le modèle.

Le matériau même n'est pas une image. Le composant visible d'un matériau est créé avec un maximum de huit couches de `texture`. Ces huit couches de texture ont été créées à partir d'acteurs bitmap ou d'objets images dans Director ou importées avec des modèles de programmes de modélisation 3D. Pour plus d'informations, reportez-vous à l'entrée `texture`.

Tout modèle dispose d'une liste linéaire de matériaux appelée `shaderList`. Le nombre d'entrées de la liste est égal au nombre de mailles de la ressource utilisée par le modèle. Un seul matériau peut être appliqué à chaque maille.

L'acteur 3D possède un matériau par défaut nommé `DefaultShader`, qui ne peut pas être supprimé. Ce matériau est utilisé lorsque aucun matériau n'est affecté à un modèle et lorsqu'un matériau utilisé par un modèle est supprimé.

La syntaxe `member(`*`quelActeur`*`).model(`*`quelModèle`*`).shader` donne accès au premier matériau de la liste de matériaux `shaderlist` du modèle et équivaut à `member(`*`quelActeur`*`).model(`*`quelModèle`*`).shaderList[1]`.

Les matériaux sont créés et supprimés à l'aide des commandes `newShader()` et `deleteShader()`.

Les matériaux sont enregistrés dans la palette des matériaux de l'acteur 3D. Ils peuvent être référencés par nom (*`quelMatériau`*) ou par index de palette (*`indexDeMatériau`*). Un matériau peut être utilisé par n'importe quel nombre de modèles. Les modifications apportées à un matériau apparaissent dans tous les modèles qui l'utilisent.

Il existe quatre types de matériaux :

Les matériaux `#standard` présentent leurs textures de façon réaliste.

Les matériaux `#painter`, `#engraver` et `#newsprint` stylisent leurs textures pour qu'elles aient l'apparence d'une peinture, d'une gravure ou d'un journal. Ils comportent des propriétés spéciales en plus des propriétés de matériau `#standard`.

Pour plus d'informations sur les propriétés de matériau, reportez-vous aux rubriques Utilisation de Director dans l'Aide de Director.

Les matériaux utilisés par les différentes faces des primitives `#mesh` peuvent être définis à l'aide de la syntaxe `member(quelActeur).modelResource(quelleRessDeMod).face[index].shader`. Pour modifier cette propriété, il est nécessaire d'appeler la commande `build()`.

Exemple

L'instruction suivante affecte le matériau surfaceDuMur à la propriété `shader` du modèle Mur :

```
member("Room").model("Wall").shader = member("Room").shader("WallSurface")
```

Voir aussi

`shaderList`, `newShader`, `deleteShader`, `face`, `texture`

shaderList

Syntaxe

```
member(whichCastmember).model(whichModel).shaderList
member(whichCastmember).model(whichModel).shaderList[index]
```

Description

Propriété 3D de modèle ; liste linéaire de `shadowPercentage` appliqué au modèle. Le nombre d'entrées de la liste est égal au nombre de mailles de la ressource utilisée par le modèle. Un seul matériau peut être appliqué à chaque maille.

Définissez le matériau sur la position d'*index* spécifiée dans `shaderList` à l'aide de la syntaxe suivante :
`member(quelActeur).model(quelModèle).shaderList[index] = référenceDeMatériau`

Dans le texte 3D, chaque caractère est constitué d'une maille distincte. Définissez la valeur d'`index` sur le nombre de caractères du matériau que vous souhaitez définir.

Définissez le même matériau sur toutes les positions d'*index* de `shaderList` à l'aide de la syntaxe suivante (notez l'absence d'index pour `shaderList`) : `member(quelActeur).model(quelModèle).shaderList = référenceDeMatériau`

Définissez la propriété d'un matériau de `shaderList` à l'aide de la syntaxe suivante :
`member(quelActeur).model(quelModèle).shaderList[index].\`
`quellePropriété = valeurDePropriété`

Définissez la même valeur pour tous les matériaux d'un modèle à l'aide de la syntaxe suivante (remarquez l'absence d'index pour shaderList) :

```
member(whichCastmember).model(whichModel).shaderList.whichProperty = propValue
```

Exemple

L'instruction suivante définit le second matériau de la liste `shaderList` du modèle pareChocs sur le matériau Chrome :

```
member("Car").model("Bumper").shaderList[2] = member("Car").shader("Chrome")
```

L'instruction suivante définit tous les matériaux de la liste `shaderList` du modèle pareChocs sur le matériau Chrome :

```
member("Car").model("Bumper").shaderList = member("Car").shader("Chrome")
```

Voir aussi

shadowPercentage

shadowPercentage

Syntaxe

```
member(whichCastmember).model(whichModel).toon.shadowPercentage
member(whichCastmember).model(whichModel).shader.shadowPercentage
member(whichCastmember).shader(whichShader).shadowPercentage
```

Description

Propriété 3D de modificateur `toon` et de matériau `#painter` ; indique le pourcentage de couleurs disponibles utilisé dans la zone éclairée de la surface du modèle sur laquelle la lumière n'apparaît pas.

La plage de cette propriété s'étend de 0 à 100, la valeur par défaut étant 50.

Le nombre de couleurs utilisées par le modificateur `toon` et le matériau `painter` pour un modèle est déterminé par la propriété `colorSteps` du modificateur `toon` ou du matériau `painter` du modèle.

Exemple

L'instruction suivante attribue la valeur 50 à la propriété `shadowPercentage` du modificateur `toon` du modèle Théière. La moitié des couleurs disponibles pour le modificateur `toon` est utilisée pour la zone ombragée de la surface du modèle.

```
-- Lingo
member("shapes").model("Teapot").toon.shadowPercentage = 50

// Javascript
member("shapes").getPropRef("model",1).getProp("toon").shadowPercentage=50;
```

Voir aussi

colorSteps, shadowStrength

shadowStrength

Syntaxe

```
member(whichCastmember).model(whichModel).toon.shadowStrength
member(whichCastmember).model(whichModel).shader.shadowStrength
member(whichCastmember).shader(whichShader).shadowStrength
```

Description

Propriété 3D de modificateur `toon` et de matériau `#painter` ; indique la luminosité de la zone de la surface du modèle sur laquelle la lumière n'apparaît pas.

La valeur par défaut de cette propriété est 1,0.

Exemple

L'instruction suivante définit la propriété `shadowStrength` du modificateur toon du modèle Sphère sur la valeur 0,1. La zone de la surface du modèle non éclairée est très foncée.

```
member("Shapes").model("Sphere").toon.shadowStrength = 0.1
```

shapeType

Syntaxe

```
member(whichCastMember).shapeType
the shapeType of member whichCastMember
```

Description

Propriété d'acteur forme ; indique le type de forme spécifiée. Les types possibles sont `#rect`, `#roundRect`, `#oval` et `#line`. Vous pouvez utiliser cette propriété pour spécifier le type d'un acteur forme après avoir créé ce dernier avec Lingo.

Exemple

Les instructions suivantes créent un nouvel acteur forme avec le numéro 100, puis le définissent comme un ovale :

```
-- Lingo
new(#shape, member 100)
member(100).shapeType = #oval

// Javascript
var t = _movie.newMember(symbol("shape"));
t.shapeType=symbol("oval");
```

shiftDown

Syntaxe

```
-- Lingo syntax
_key.shiftDown

// JavaScript syntax
_key.shiftDown;
```

Description

Propriété de touche ; indique si l'utilisateur appuie sur la touche Maj. Lecture seule.

Cette propriété renvoie la valeur `TRUE` si l'utilisateur appuie sur la touche Maj ; dans le cas contraire, elle renvoie la valeur `FALSE`.

Cette propriété doit être testée conjointement à une autre touche.

Exemple

L'instruction suivante vérifie si la touche Maj est enfoncée et, le cas échéant, appelle le gestionnaire `doCapitalA` :

```
-- Lingo syntax
if (_key.shiftDown) then
    doCapitalA(_key.key)
end if

// JavaScript syntax
if (_key.shiftDown) {
    doCapitalA(_key.key);
}
```

Voir aussi

controlDown, Touche, key, keyCode, optionDown

shininess

Syntaxe

```
member(whichCastmember).shader(whichShader).shininess
member(whichCastmember).model(whichModel).shader.shininess
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].shininess
```

Description

Propriété 3D de matériau standard ; permet d'obtenir ou de définir le brillant d'une surface. Le brillant est défini en tant que pourcentage de la surface de matériau réservée aux taches spéculaires. La valeur est un nombre entier compris entre 0 et 100, avec une valeur par défaut de 30.

Tous les matériaux ont accès aux propriétés de matériau `#standard` ; outre ces propriétés standard, les matériaux de type `#engraver`, `#newsprint` et `#painter` possèdent des propriétés propres à leur type. Pour plus d'informations, reportez-vous à l'entrée `newShader`.

Exemple

L'instruction suivante définit la propriété `shininess` du premier matériau de la liste des matériaux du modèle gbCyl3 sur la valeur 60. Soixante pour-cent de la surface du matériau sont dédiés aux reflets.

```
-- Lingo
member("Scene").model("gbCyl3").shader.shininess = 60

// Javascript
member("Scene").getPropRef("model",1).getProp("shader").shininess=60;
```

silhouettes

Syntaxe

```
member(whichCastmember).model(whichModel).inker.silhouettes
member(whichCastmember).model(whichModel).toon.silhouettes
```

Description

Propriété 3D de modificateur `toon` et `inker` ; indique la présence (`TRUE`) ou l'absence (`FALSE`) de lignes dessinées par le modificateur sur les bords visibles du modèle.

Des lignes de silhouette sont dessinées autour de l'image 2D du modèle sur le plan de projection de la caméra. Leur relation avec la maille du modèle n'est pas fixée, à la différence des lignes de plis ou de délimitation qui sont dessinées sur la maille.

Les lignes de silhouette sont similaires aux lignes de contour des images dans un livre de coloriage pour enfant.

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante définit la propriété `silhouettes` du modificateur `inker` du modèle Sphère sur la valeur `FALSE`. Les lignes sont dessinées autour du profil du modèle.

```
-- Lingo
member("Shapes").model("Sphere").inker.silhouettes = FALSE

// Javascript
member("Shapes").getPropRef("model",1).inker.silhouetess = false;
```

size

Syntaxe

```
-- Lingo syntax
memberObjRef.size

// JavaScript syntax
memberObjRef.size;
```

Description

Propriété d'acteur ; renvoie la taille, en octets, qu'un acteur spécifique occupe en mémoire. Lecture seule.

Multipliez les octets par 1 024 pour obtenir des kilo-octets.

Exemple

La ligne suivante renvoie la taille de l'acteur Temple dans un champ intitulé Taille :

```
-- Lingo syntax
member("How Big").text = string(member("shrine").size)

// JavaScript syntax
member("How Big").text = member("shrine").size.toString();
```

Voir aussi

Acteur

sizeRange

Syntaxe

```
member(whichCastmember).modelResource
(whichModelResource).sizeRange.start
modelResourceObjectReference.sizeRange.start
member(whichCastmember).modelResource
(whichModelResource).sizeRange.end
modelResourceObjectReference.sizeRange.end
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir ou de définir les propriétés `start` et `end` de la propriété `sizeRange` de la ressource de modèle. Les particules sont mesurées en unités d'univers.

La taille des particules du système est interpolée de façon linéaire entre `sizeRange.start` et `sizeRange.end` pendant la durée de vie de chaque particule.

Cette propriété doit être un entier supérieur à 0 et a une valeur par défaut de 1.

Exemple

Dans l'exemple suivant, mrFont est une ressource de modèle de type `#particle`. L'instruction suivante définit les propriétés `sizeRange` de mrFont. La première ligne donne la valeur de départ 4 et la seconde donne la valeur de fin 1. Cette instruction donne aux particules de mrFont une taille de 4 lorsqu'elles apparaissent pour la première fois, puis les réduit petit à petit à une taille de 1.

```
-- Lingo
member("fountain").modelResource("mrFount").sizeRange.start = 4
member("fountain").modelResource("mrFount").sizeRange.end = 1

// Javascript
member("fountain").getPropRef("modelResource",1).getProp("sizeRange").start=4;
member("fountain").getPropRef("modelResource",1).getProp("sizeRange").end=4;
```

sizeState

Syntaxe

```
-- Lingo syntax
windowObjRef.sizeState

// JavaScript syntax
windowObjRef.sizeState;
```

Description

Propriété de fenêtre ; renvoie l'état de dimensionnement d'une fenêtre. Lecture seule.

L'état de dimensionnement renvoyé peut correspondre à l'une des valeurs suivantes :

Etat de dimensionnement	Description
#minimized	Indique que la fenêtre est réduite.
#maximized	Indique que la fenêtre est agrandie.
#normal	Indique que la fenêtre n'est ni réduite, ni agrandie.

Exemple

Les instructions suivantes agrandissent la fenêtre Artistes si elle ne l'est pas déjà.

```
-- Lingo syntax
if (window("Artists").sizeState <> #maximized) then
    window("Artists").maximize()
end if

// JavaScript syntax
if (window("Artists").sizeState != symbol("maximized")) {
    window("Artists").maximize();
}
```

Voir aussi

Fenêtre

skew

Syntaxe

```
-- Lingo syntax
spriteObjRef.skew
```

```
// JavaScript syntax
spriteObjRef.skew;
```

Description

Propriété d'image-objet ; renvoie sous forme de valeur flottante exprimée en centièmes de degré, l'angle d'inclinaison des bords verticaux de l'image-objet d'un point de vue vertical. Lecture/écriture.

Des valeurs négatives signalent une inclinaison vers la gauche, tandis que des valeurs positives indiquent une inclinaison vers la droite. Des valeurs supérieures à 90° renversent l'image verticalement.

Le scénario peut conserver des informations sur l'inclinaison d'une image de +21 474 836,47° à -21 474 836,48°, ce qui permet 59 652 rotations complètes dans chaque direction.

Lorsque la limite d'inclinaison est atteinte (juste après la 59 652ème rotation), la commande rétablit le réglage sur +116,47° ou -116,48°, et non sur 0,00°. Cela s'explique par le fait que +21 474 836,47° est égal à +116,47° et -21 474 836,48° est égal à -116,48° (ou +243,12°). Pour éviter ce réglage, vous devez contraindre les angles à ±360° dans chaque direction lorsque vous effectuez une inclinaison continue à l'aide d'un script.

Exemple

Le comportement suivant incline continuellement une image-objet de 2 degrés chaque fois que la tête de lecture avance, tout en limitant l'angle à 360 degrés :

```
-- Lingo syntax
property spriteNum

on prepareFrame me
    sprite(spriteNum).skew = integer(sprite(spriteNum).skew + 2) mod 360
end

// JavaScript syntax
function prepareFrame() {
    sprite(this.spriteNum).skew = parseInt(sprite(this.spriteNum).skew + 2) % 360;
}
```

Voir aussi

flipH, flipV, rotation, Image-objet

smoothness

Syntaxe

```
member(whichTextmember).smoothness
member(whichCastMember).modelResource(whichExtruderModelResource).smoothness
```

Description

Propriété 3D d'extrudeur et de texte ; permet d'obtenir ou de définir un nombre entier contrôlant le nombre de segments utilisés pour la création d'un acteur texte 3D. Plus ce nombre est élevé, plus le texte est lisse. La plage de cette propriété s'étend de 1 à 10, la valeur par défaut étant 5.

Pour plus d'informations sur l'utilisation des ressources de modèle d'extrudeur et des acteurs texte, reportez-vous à l'entrée extrude3D.

Exemple

Dans l'exemple suivant, l'acteur Logo est un acteur texte. L'instruction suivante définit le lissé de Logo à 8 ; le bord de ses lettres est très lisse en mode 3D.

```
-- Lingo
member("Logo").smoothness = 8

// Javascript
member("Logo").smoothness = 8;
```

Dans l'exemple suivant, la ressource du modèle Slogan correspond à du texte extrudé. L'instruction suivante définit la propriété de lissé de la ressource de modèle Slogan sur la valeur 1, ce qui donne un aspect très anguleux aux lettres de Slogan.

```
-- Lingo
member("Scene").model("Slogan").resource.smoothness = 1

// Javascript
member("Scene").getPropRef("model",1).getProp("resource").smoothness=1;
```

Voir aussi

extrude3D

sound (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.sound

// JavaScript syntax
memberObjRef.sound;
```

Description

Propriété d'acteur ; permet de définir si le son d'une animation, d'une vidéo numérique ou d'une animation Flash est activé (TRUE, valeur par défaut) ou désactivé (FALSE). Lecture/écriture.

Dans les acteurs Flash, le nouveau paramètre prend effet à l'issue de la lecture du son en cours.

Vous pouvez voir un exemple d'utilisation de `sound` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

Le gestionnaire suivant accepte une référence à un acteur et bascule la propriété `sound` de cet acteur entre les états activé et désactivé :

```
-- Lingo syntax
on ToggleSound(whichMember)
    member(whichMember).sound = not(member(whichMember).sound)
end

// JavaScript syntax
function ToggleSound(whichMember) {
    member(whichMember).sound = !(member(whichMember).sound);
}
```

Voir aussi

Animation Flash

sound (lecteur)

Syntaxe

```
-- Lingo syntax
_player.sound[intSoundChannelNum]

// JavaScript syntax
_player.sound[intSoundChannelNum];
```

Description

Propriété de lecteur ; permet d'accéder par index à un objet piste audio à l'aide d'une propriété de lecteur. Lecture seule.

L'argument *numPisteAudio* est un nombre entier spécifiant le numéro de la piste audio à laquelle accéder.

Cette propriété a la même fonction que la méthode de haut niveau `sound()`.

Exemple

L'instruction suivante définit la variable `mySound` sur le son de la piste audio 3 :

```
-- Lingo syntax
mySound = _player.sound[3]

// JavaScript syntax
var mySound = _player.sound[3];
```

Voir aussi

Lecteur, sound(), Piste audio

soundChannel (SWA)

Syntaxe

```
-- Lingo syntax
memberObjRef.soundChannel

// JavaScript syntax
memberObjRef.soundChannel;
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; spécifie la piste audio dans laquelle le son SWA doit être lu.

Si aucun numéro de piste n'est spécifié ou si la piste 0 est indiquée, l'acteur SWA en flux continu affecte le son à la piste audio inutilisée dotée du plus haut numéro.

Les sons Shockwave Audio en flux continu peuvent apparaître sous la forme d'images-objets dans les pistes d'images-objets, mais exécutent des sons dans une piste audio. Vous devez faire référence aux images-objets son SWA par leur numéro de piste d'image-objet et non par le numéro de piste audio.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante indique à l'acteur SWA lu en flux continu Frank Zappa qu'il doit être lu dans la piste audio 3 :

```
-- Lingo syntax
member("Frank Zappa").soundChannel = 3

// JavaScript syntax
member("Frank Zappa").soundChannel = 3;
```

soundChannel (RealMedia)

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.soundChannel

// JavaScript syntax
memberOrSpriteObjRef.soundChannel;
```

Description

Propriété d'acteur ou image-objet RealMedia ; permet d'obtenir ou de définir la piste audio utilisée pour la lecture de l'audio du flux RealMedia. La définition de cette propriété permet de contrôler l'audio d'un flux RealMedia à l'aide des méthodes et propriétés audio Lingo. La définition de cette propriété à une valeur non comprise entre 0 et 8 entraîne une erreur Lingo. Cette propriété n'a aucun effet si `realPlayerNativeAudio()` présente la valeur `TRUE`.

Le paramètre par défaut de cette propriété est 0, ce qui signifie que le composant audio RealMedia est lu sur la piste audio portant le numéro le plus élevé parmi les pistes disponibles et que la valeur de la propriété change en cours de lecture en fonction de la piste utilisée. Lorsque l'acteur RealMedia est en cours de lecture, cette propriété correspond à la piste audio active. Lorsque l'acteur RealMedia s'arrête, la valeur de cette propriété repasse à 0.

Si vous spécifiez une piste (de 1 à 8) pour cette propriété et que des sons sont actuellement en cours de lecture sur cette piste (provenant d'autres parties de l'animation), ils sont interrompus et l'audio RealMedia est lu dans la piste.

La lecture simultanée de plusieurs acteurs RealMedia n'est pas prise en charge. Si votre animation contient des acteurs RealMedia qui sont lus simultanément, leurs sons sont lus en même temps dans la même piste audio.

Exemple

Les exemples suivants indiquent que le son du flux RealMedia de l'image-objet 2 et de l'acteur Real est lu dans la piste audio 2.

```
-- Lingo syntax
put(sprite(2).soundChannel) -- 2
put(member("Real").soundChannel) -- 2

// JavaScript syntax
put(sprite(2).soundChannel); // 2
put(member("Real").soundChannel); // 2
```

Les exemples suivants affectent la piste audio 1 au flux RealMedia de l'image-objet 2 et de l'acteur Real.

```
-- Lingo syntax
sprite(2).soundChannel = 1
member("Real").soundChannel = 1

// JavaScript syntax
sprite(2).soundChannel = 1;
member("Real").soundChannel = 1;
```

Voir aussi

realPlayerNativeAudio()

soundDevice

Syntaxe

```
-- Lingo syntax
_sound.soundDevice

// JavaScript syntax
_sound.soundDevice;
```

Description

Propriété audio ; permet le réglage du dispositif de mixage audio pendant la lecture de l'animation. Lecture/écriture.

Les valeurs possibles pour `soundDevice` correspondent aux dispositifs répertoriés par `soundDeviceList`.

Plusieurs dispositifs audio peuvent être répertoriés. Les différents dispositifs audio pour Windows présentent différents avantages.

- MacroMix (Windows®) : Plus petit dénominateur commun pour une lecture audio Windows. Ce dispositif fonctionne sur n'importe quel ordinateur Windows, mais avec une latence moins bonne que celle des autres dispositifs.
- QT3Mix (Windows) : Mixe le son avec l'audio QuickTime et parfois d'autres applications si celles-ci utilisent DirectSound. Ce dispositif ne fonctionne que si QuickTime est installé et offre un meilleur temps d'exécution que MacroMix.
- DirectSound® (Windows) : semblable à QT3Mix, mais offre une latence supérieure.
- MacSoundManager (Mac) : seul dispositif audio disponible sur Mac.

Exemple

L'instruction suivante règle le dispositif audio sur MacroMix pour un ordinateur tournant sous Windows. En cas de défaillance du dispositif nouvellement affecté, la propriété `soundDevice` n'est pas modifiée.

```
-- Lingo syntax
_sound.soundDevice = "MacroMix"

// JavaScript syntax
_sound.soundDevice = "MacroMix";
```

Voir aussi

Son, soundDeviceList

soundDeviceList

Syntaxe

```
-- Lingo syntax
_sound.soundDeviceList
```

```
// JavaScript syntax
_sound.soundDeviceList;
```

Description

Propriété audio ; crée une liste linéaire des dispositifs audio disponibles sur l'ordinateur utilisé. Lecture seule.

Pour le Mac, cette propriété ne recense qu'un dispositif, à savoir MacSoundManager.

Exemple

L'instruction suivante affiche une liste de types de dispositifs audio typique sur un ordinateur Windows :

```
-- Lingo syntax
trace(_sound.soundDeviceList)

// JavaScript syntax
trace(_sound.soundDeviceList);
```

Voir aussi

Son, soundDevice

soundEnabled

Syntaxe

```
-- Lingo syntax
_sound.soundEnabled

// JavaScript syntax
_sound.soundEnabled;
```

Description

Propriété audio ; détermine si le son est activé (TRUE, valeur par défaut) ou désactivé (FALSE). Lecture/écriture.

Lorsque vous attribuez à cette propriété la valeur FALSE, le son est désactivé, mais le réglage du volume ne change pas.

Exemple

L'instruction suivante inverse la valeur en cours de la propriété soundEnabled (active le son s'il est désactivé et vice-versa) :

```
-- Lingo syntax
_sound.soundEnabled = not(_sound.soundEnabled)

// JavaScript syntax
_sound.soundEnabled = !(_sound.soundEnabled);
```

Voir aussi

Son

soundKeepDevice

Syntaxe

```
-- Lingo syntax
_sound.soundKeepDevice

// JavaScript syntax
_sound.soundKeepDevice;
```

Description

Propriété audio ; pour Windows uniquement, détermine si le pilote audio se décharge et se recharge chaque fois qu'un son doit être lu. Lecture/écriture.

La valeur par défaut de cette propriété est `TRUE`, ce qui empêche le déchargement et le rechargement du pilote audio chaque fois qu'un son doit être lu.

Il peut s'avérer nécessaire de définir cette propriété sur `FALSE` avant de lire un son pour s'assurer que le dispositif audio est déchargé et disponible pour les autres applications ou processus de l'ordinateur une fois que la lecture du son est terminée.

La définition de cette propriété sur la valeur `FALSE` risque d'affecter les performances si le son est lu fréquemment par l'application Director.

Exemple

L'instruction suivante attribue à la propriété `soundKeepDevice` la valeur `FALSE` :

```
-- Lingo syntax
_sound.soundKeepDevice = FALSE

// JavaScript syntax
_sound.soundKeepDevice = false;
```

Voir aussi

Son

soundLevel

Syntaxe

```
-- Lingo syntax
_sound.soundLevel

// JavaScript syntax
_sound.soundLevel;
```

Description

Propriété audio ; renvoie ou définit le niveau de volume du son lu dans le haut-parleur de l'ordinateur. Lecture/écriture.

La plage des valeurs possibles s'étend de 0 (silence) à 7 (valeur maximale, par défaut).

Sous Windows, le réglage audio du système est combiné au contrôle du volume des haut-parleurs externes. Par conséquent, le volume réel obtenu après le réglage du son peut varier. Evitez de régler la propriété `soundLevel` à moins d'être certain que le résultat est acceptable pour l'utilisateur. Il est préférable de régler le volume des pistes et des images-objets individuellement à l'aide de la propriété `volume` de l'objet Piste audio.

Ces valeurs correspondent aux paramètres présents dans le tableau de commande Son du Mac. L'utilisation de cette propriété permet à un script de modifier directement le volume du son ou d'effectuer une autre opération lorsque le son est situé à un niveau spécifié.

Vous pouvez voir un exemple d'utilisation de `soundLevel` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante définit la variable `oldSound` comme étant égale au niveau sonore en cours :

```
-- Lingo syntax
oldSound = _sound.soundLevel

// JavaScript syntax
var oldSound = _sound.soundLevel;
```

L'instruction suivante définit le niveau sonore sur 5 :

```
-- Lingo syntax
_sound.soundLevel = 5

// JavaScript syntax
_sound.soundLevel = 5;
```

Voir aussi

Son, volume (Windows Media)

soundMixMedia

Syntaxe

```
-- Lingo syntax
_sound.soundMixMedia

// JavaScript syntax
_sound.soundMixMedia;
```

Description

Propriété audio ; détermine si les sons des acteurs Flash sont mixés avec les sons des pistes audio du scénario. Lecture/écriture.

Cette propriété est définie par défaut sur `TRUE` pour les animations créées dans Director 7 et les versions ultérieures et sur `FALSE` dans les versions précédentes. Elle ne fonctionne que sous Windows.

Si cette propriété présente la valeur `TRUE`, les sons des acteurs Flash sont mixés avec les sons des pistes audio du scénario. Director prend en charge le mixage et la lecture des sons des acteurs Flash.

Il est possible de constater de légères différences dans la façon dont les sons Flash sont lus. Pour entendre le son exact des sons Flash, définissez cette propriété sur `FALSE`.

Lorsque cette propriété est définie sur `FALSE`, les sons Flash ne sont pas mixés et doivent être lus séparément.

Exemple

Les instructions suivantes affichent la valeur par défaut de la propriété `soundMixMedia` dans l'environnement auteur.

```
-- Lingo
put _sound.soundMixMedia
-- 1

// Javascript
trace( _sound.soundMixMedia);
// 1
```

Voir aussi

Son

source

Syntaxe

```
sprite(whichSprite).camera.backdrop[backdropIndex].source
member(whichCastmember).camera(whichCamera).backdrop
[backdropIndex].source
sprite(whichSprite).camera.overlay[overlayIndex].source
member(whichCastmember).camera(whichCamera).overlay
[overlayIndex].source
```

Description

Propriété 3D de fond et de recouvrement ; permet d'obtenir ou de définir la texture utilisée comme image source du recouvrement ou du fond.

Exemple

L'instruction suivante donne à la source du fond 1 la texture Cèdre.

```
sprite(3).camera.backdrop[1].source =
sprite(3).member.texture("Cedar")
```

Voir aussi

bevelDepth, overlay

sourceFileName

Syntaxe

```
flashCastMember.sourceFileName
```

Description

Propriété d'acteur Flash ; spécifie le chemin d'accès du fichier FLA source à utiliser lors des opérations de lancement et d'édition. Cette propriété peut être testée et définie. La valeur par défaut est une chaîne vide.

Exemple

L'instruction suivante définit la propriété sourceFileName de l'acteur Flash SWF sur `C:\FlashFiles\myFile.fla` :

```
-- Lingo
member("SWF").sourceFileName = "C:\FlashFiles\myFile.fla"
```

```
// Javascript
member("SWF").sourceFileName = "C:\FlashFiles\myFile.fla";
```

sourceRect

Syntaxe

```
-- Lingo syntax
windowObjRect.sourceRect

// JavaScript syntax
windowObjRect.sourceRect;
```

Description

Propriété de fenêtre ; détermine les coordonnées initiales de la scène de l'animation en cours de lecture dans une fenêtre. Lecture seule.

Cette propriété est utile pour rétablir la taille et la position d'origine d'une fenêtre lorsque celle-ci a été modifiée par le curseur ou la définition de son rectangle.

Exemple

L'instruction suivante affiche les coordonnées d'origine de la scène intitulée Tableau_de_commande dans la fenêtre Messages :

```
-- Lingo syntax
put(window("Control_panel").sourceRect)

// JavaScript syntax
put(window("Control_panel").sourceRect);
```

Voir aussi

Fenêtre

specular (lumière)

Syntaxe

```
member(whichCastmember).light(whichLight).specular
```

Description

Propriété 3D de lumière ; permet de savoir ou de définir si la spécularité est activée (TRUE) ou non (FALSE). La spécularité est la capacité de reflet sur un modèle lorsque la lumière dirigée vers le modèle est reflétée vers la caméra. La brillance du matériau d'un modèle détermine sa proportion spéculaire. La valeur de cette propriété est ignorée pour les lumières d'ambiance. La valeur par défaut de cette propriété est TRUE.

Remarque : *la désactivation de cette propriété peut améliorer la performance.*

Exemple

L'instruction suivante attribue à la propriété `specular` de la lumière omni2 la valeur FALSE. Cette lumière n'entraîne pas de reflets. S'il s'agit de la seule lumière éclairant la scène, il n'y a aucun reflet spéculaire sur les matériaux de la scène.

```
-- Lingo
member("3d world").light("omni2").specular = FALSE

// Javascript
member("3d world").getPropRef("light",1).specular = false;
```

Voir aussi

silhouettes, specularLightMap

specular (matériau)

Syntaxe

```
member(whichCastmember).shader(whichShader).specular
```

Description

Propriété 3D de matériau standard ; permet d'obtenir ou de définir la couleur spéculaire d'un matériau donné. La couleur spéculaire est la couleur de la tache spéculaire générée lorsque la spécularité est activée. Pour que cette propriété ait un effet visible, la propriété spéculaire de certaines lumières de la scène doit présenter la valeur `TRUE`. La couleur spéculaire est influencée par la couleur des lumières qui éclairent l'objet. Si la couleur spéculaire est blanche mais que la couleur d'une lumière est rouge, un reflet rouge spéculaire apparaîtra sur l'objet. La valeur par défaut de cette propriété est `rgb(255, 255, 255)`, qui est le blanc.

Tous les matériaux ont accès aux propriétés de matériau `#standard` ; outre ces propriétés standard, les matériaux de type `#engraver`, `#newsprint` et `#painter` possèdent des propriétés propres à leur type. Pour plus d'informations, reportez-vous à l'entrée newShader.

Exemple

```
-- Lingo
put member("scene").shader("plutomat").specular
--rgb(11, 11, 11)

// Javascript
trace(member("scene").getPropRef("shader",1).specular
// <rgb(11,11,11)>
```

Voir aussi

silhouettes, specular (lumière), specularColor, emissive

specularColor

Syntaxe

```
member(whichCastmember).specularColor
```

Description

Propriété 3D d'acteur ; permet d'obtenir ou de définir la valeur rvb de la couleur spéculaire du premier matériau de l'acteur. Le premier matériau de la palette d'un acteur constitue toujours le matériau par défaut. Comme toutes les autres propriétés d'acteur 3D, cette propriété est enregistrée avec l'acteur et est restaurée la prochaine fois que vous ouvrirez l'animation. La valeur par défaut de cette propriété est `rgb(255, 255, 255)`, qui est le blanc.

Exemple

L'instruction suivante donne à la couleur spéculaire du premier matériau de l'acteur Séquence la valeur `rgb(255, 0, 0)`. Cette instruction est l'équivalent de `member("Scene").shader[1].specular = rgb(255, 0, 0)`. Toutefois, cette dernière syntaxe n'enregistre pas la nouvelle valeur avec l'acteur lorsque l'animation est enregistrée. Seul `member.specularColor` enregistre la nouvelle valeur de couleur.

```
-- Lingo
member("Scene").specularColor = rgb(255, 0, 0)

// Javascript
member("Scene").specularColor = rgb(255, 0, 0);
```

Voir aussi

silhouettes, specular (lumière), specular (matériau)

specularLightMap

Syntaxe

```
member(whichCastmember).shader(whichShader).specularLightMap
member(whichCastmember).model(whichModel).shader.specularLightMap
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].specularLightMap
```

Description

Propriété 3D de matériau standard ; permet d'obtenir ou de définir la cinquième couche de texture d'un matériau standard donné. Cette propriété est ignorée si le modificateur `toon` est appliqué à la ressource de modèle.

Les valeurs peuvent être définies comme suit :

- `textureModeList[5] = #specular`
- `blendFunctionList[5] = #add`
- `blendFunctionList[1] = #replace`
- `default = void`

Cette propriété fournit une interface simple pour la définition d'une utilisation commune du placage de lumière spéculaire.

Tous les matériaux ont accès aux propriétés de matériau `#standard` ; outre ces propriétés standard, les matériaux de type `#engraver`, `#newsprint` et `#painter` possèdent des propriétés propres à leur type. Pour plus d'informations, reportez-vous à l'entrée `newShader`.

Exemple

L'instruction suivante donne la texture Ovale comme propriété specularLightMap au matériau utilisé par le modèle BoîteEnVerre.

```
-- Lingo
member("3DPlanet").model("GlassBox").shader.specularLightMap = \
member("3DPlanet").texture("Oval")

// Javascript
member("3DPlanet").getPropRef("model",1).getProp("shader").specularLightMap
=member("3DPlanet").getPropRef("texture",1);
```

Voir aussi

diffuseLightMap

spotAngle

Syntaxe

```
member(whichCastmember).light(whichLight).spotAngle
```

Description

Propriété 3D ; permet d'obtenir ou de définir l'angle du cône de projection de lumière. La lumière extérieure à l'angle spécifié pour cette propriété n'apporte aucune intensité. Cette propriété accepte les valeurs à virgule flottante comprises entre 0,0 et 180,00, sa valeur par défaut étant 90,0. La valeur flottante que vous spécifiez correspond à la moitié de l'angle. Par exemple, si vous souhaitez spécifier un angle de 90 degrés, vous devez indiquer une valeur de 45,0.

Exemple

L'instruction suivante attribue à la propriété `spotAngle` de la lumière unidirectionnelle la valeur 8. L'angle du cône de projection de la lumière est de 16 degrés :

```
-- Lingo
member("3d world").light("unidirectional").spotAngle = 8

// Javascript
member("3d world").getPropRef("light",1).spotAngle = 8;
```

spotDecay

Syntaxe

```
member(whichCastmember).light(whichLight).spotDecay
```

Description

Propriété 3D de lumière ; permet de savoir ou de définir si l'intensité d'un projecteur diminue avec l'éloignement de la caméra. La valeur par défaut de cette propriété est `FALSE`.

Exemple

L'instruction suivante attribue à la propriété `spotDecay` de la lumière 1 la valeur `TRUE`. Les modèles les plus éloignés de la lumière 1 sont moins éclairés que les modèles les plus proches.

```
-- Lingo
member("Scene").light[1].spotDecay = TRUE

// Javascript
member("Scene").getPropRef("light",1).spotDecay = true;
```

sprite (animation)

Syntaxe

```
-- Lingo syntax
_movie.sprite[spriteNameOrNum]
```

```
// JavaScript syntax
_movie.sprite[spriteNameOrNum];
```

Description

Propriété d'animation ; offre un accès par index ou par nom à une image-objet d'animation. Lecture seule.

L'argument *nomOuNumImageObjet* peut être une chaîne spécifiant le nom de l'image-objet ou un nombre entier indiquant le numéro de cette image-objet.

Exemple

L'instruction suivante définit la variable `sportSprite` sur l'image-objet d'animation 5 :

```
-- Lingo syntax
sportSprite = _movie.sprite[5]

// JavaScript syntax
var sportSprite = _movie.sprite[5];
```

Voir aussi

Animation

sprite (piste d'image-objet)

Syntaxe

```
-- Lingo syntax
spriteChannelObjRef.sprite

// JavaScript syntax
spriteChannelObjRef.sprite;
```

Description

Propriété de piste d'image-objet ; renvoie une référence à l'image-objet dans l'image en cours d'une piste d'image-objet. Lecture seule.

Exemple

L'instruction suivante définit la variable `mySprite` sur l'image-objet de la piste d'image-objet Ruban.

```
-- Lingo syntax
mySprite = channel("Ribbon").sprite

// JavaScript syntax
var mySprite = channel("Ribbon").sprite;
```

Voir aussi

Piste d'image-objet

spriteNum

Syntaxe

```
-- Lingo syntax
spriteObjRef.spriteNum

// JavaScript syntax
spriteObjRef.spriteNum;
```

Description

Propriété d'image-objet ; détermine le numéro de piste dans laquelle se trouve l'image-objet d'un comportement et la rend accessible à n'importe quel comportement. Lecture seule.

Il suffit simplement de déclarer la propriété en haut du comportement, avec toutes les autres propriétés que celui-ci pourra utiliser.

Si vous utilisez un gestionnaire `new()` pour créer une instance du comportement, le gestionnaire `new()` du script doit explicitement affecter la propriété `spriteNum` au numéro de l'image-objet. Cela permet d'identifier l'image-objet à laquelle le script est associé. Le numéro de l'image-objet doit être transmis au gestionnaire `new()` sous la forme d'un argument lors de l'appel de ce gestionnaire.

Exemple

Dans le gestionnaire suivant, la propriété `spriteNum` est automatiquement définie pour les instances de script créées par le système :

```
-- Lingo syntax
property spriteNum, pMySpriteRef

on mouseDown me
    sprite(spriteNum).member = member("DownPict")
end

// JavaScript syntax
function mouseDown() {
    sprite(this.spriteNum).member = member("DownPict");
}
```

Le gestionnaire suivant utilise la valeur automatique insérée dans la propriété `spriteNum` pour affecter la référence de l'image-objet à une nouvelle variable de propriété `pMySpriteRef` à des fins de commodité :

```
-- Lingo syntax
property spriteNum, pMySpriteRef

on beginSprite me
    pMySpriteRef = sprite(me.spriteNum)
end

// JavaScript syntax
function beginSprite() {
    this.pMySpriteRef = sprite(this.spriteNum);
}
```

Cette approche permet d'utiliser la référence `pMySpriteRef` dans la suite du script, le gestionnaire utilisant la syntaxe :

```
-- Lingo syntax
currMember = pMySpriteRef.member
```

```
// JavaScript syntax
var currMember = pMySpriteRef.member
```

à la place de la syntaxe suivante, légèrement plus longue :

```
Lingo syntax
currMember = sprite(spriteNum).member

// JavaScript syntax
var currMember = sprite(this.spriteNum).member
```

Cette seconde approche existe uniquement par souci de commodité et n'assure aucune autre fonction.

Voir aussi

new(), Image-objet

stage

Syntaxe

```
-- Lingo syntax
_movie.stage

// JavaScript syntax
_movie.stage;
```

Description

Propriété d'animation ; fait référence à l'animation principale. Lecture seule.

Cette propriété se révèle utile lors de l'envoi d'un message à l'animation principale à partir d'une animation enfant.

Exemple

L'instruction suivante affiche le paramétrage en cours de la scène :

```
-- Lingo syntax
put(_movie.stage.rect)

// JavaScript syntax
put(_movie.stage.rect);
```

Voir aussi

Animation

startAngle

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).
startAngle
modelResourceObjectReference.startAngle
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#cylinder` ou `#sphere`, cette commande permet d'obtenir et de définir la propriété `startAngle` de la ressource de modèle référencée sous la forme d'une valeur à virgule flottante comprise entre 0,0 et 360,0. La valeur par défaut de cette propriété est 0,0.

La propriété `startAngle` détermine l'angle de démarrage du balayage de la ressource de modèle et fonctionne conjointement avec la propriété `endAngle` pour dessiner des sphères et des cylindres. Par exemple, pour obtenir une demi-sphère, attribuez à `startAngle` la valeur 0,0 et à `endAngle` la valeur 180,0.

Exemple

L'instruction suivante donne à la propriété startAngle de la ressource de modèle Sphère01 la valeur 0,0. Si la propriété `endAngle` de cette ressource de modèle présente la valeur 90, seul un quart des modèles utilisant cette ressource apparaîtra.

```
-- Lingo
put member("3D World").modelResource(1).startAngle
-- 0.0

// Javascript
trace(member("3D World").getPropRef("modelResource",1).startAngle);
// 0
```

Voir aussi

endAngle

startFrame

Syntaxe

```
-- Lingo syntax
spriteObjRef.startFrame

// JavaScript syntax
spriteObjRef.startFrame;
```

Description

Propriété d'image-objet ; renvoie le numéro de la première image de l'étendue d'une image-objet. Lecture seule.

Cette propriété permet de déterminer la plage couverte par une image-objet spécifique dans le scénario. Elle n'est disponible que dans une image contenant l'image-objet et n'est pas applicable à des images-objets situées dans d'autres images de l'animation.

Exemple

L'instruction suivante affiche l'image de départ de l'image-objet dans la piste 5 dans la fenêtre Messages :

```
-- Lingo syntax
put(sprite(5).startFrame)

// JavaScript syntax
put(sprite(5).startFrame);
```

Voir aussi

endFrame, Image-objet

startTime

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.startTime

// JavaScript syntax
soundChannelObjRef.startTime;
```

Description

Propriété de piste audio ; indique la position de départ du son en cours de lecture ou en pause défini lorsque ce son était placé en file d'attente. Lecture seule.

Cette propriété ne peut pas être définie après le placement du son en file d'attente. Si aucune valeur n'a été définie au moment du placement du son en file d'attente, cette propriété renvoie la valeur 0.

Exemple

L'instruction suivante fait démarrer l'image-objet vidéo numérique de la piste 5 à 100 battements :

```
-- Lingo syntax
sprite(5).startTime = 100

// JavaScript syntax
sprite(5).startTime = 100;
```

Voir aussi

Piste audio

startTimeList

Syntaxe

```
-- Lingo syntax
dvdObjRef.startTimeList

// JavaScript syntax
dvdObjRef.startTimeList;
```

Description

Propriété de DVD ; liste de propriétés spécifiant le moment ou le chapitre au niveau duquel la lecture démarre. Lecture/écriture.

La propriété `startTimeList` correspond à une liste de propriétés pouvant reposer sur un chapitre ou sur une heure.

Une propriété `startTimeList` reposant sur un chapitre contient les propriétés suivantes :

- `title`. Spécifie le titre auquel appartient le chapitre à lire.
- `chapter`. Spécifie le chapitre à lire.

La propriété `startTimeList` suivante indique que la lecture doit commencer au chapitre 2 du titre 1 :

```
[#title:1, #chapter:2]
```

Une propriété `startTimeList` reposant sur une heure contient les propriétés suivantes :

- `title`. Spécifie le titre.

- `hours`. Spécifie l'heure du début de la lecture.
- `min`. Spécifie la minute à laquelle la lecture démarre.
- `sec`. Spécifie la seconde à laquelle la lecture démarre.
- `frames`. Spécifie les images au niveau desquelles la lecture démarre.

La propriété `startTimeList` suivante indique que la lecture doit commencer à une heure spécifique du titre 1 :

```
[#title:1, #hours:0, #minutes:45, #seconds:15, #frames:15]
```

La propriété `startTimeList` suivante ne répertorie qu'un seul paramètre temporel :

```
[#title:1, #seconds:15]
```

Vous pouvez annuler l'effet de la propriété startTimeList en lui attribuant la valeur 0.

Exemple

L'instruction suivante définit la propriété startTimeList de l'objet DVD sur 0.

```
-- Lingo
sprite(1).startTimeList = 0

// Javascript
sprite(1).startTimeList = 0;
```

Voir aussi

`DVD`, `play() (DVD)`, `stopTimeList`

state (3D)

Syntaxe

```
member(whichCastmember).state
```

Description

Propriété 3D ; renvoie l'état actuel de l'acteur référencé au cours du chargement et de la lecture en flux continu. Cette propriété indique la quantité qui a été chargée du fichier initial ou du dernier fichier demandé.

La propriété `state` de l'acteur détermine le code Lingo 3D pouvant éventuellement être exécuté sur l'acteur.

Cette propriété peut avoir une des valeurs suivantes :

- `0` indique que l'acteur n'est pas chargé et que, par conséquent, aucun média 3D n'est disponible. Aucune opération Lingo 3D ne devrait être réalisée sur l'acteur.
- `1` indique que le chargement du média a commencé.
- `2` indique que le segment de chargement initial de l'acteur est chargé. Tous les objets dont la propriété de chargement en flux continu est zéro (tel que cela est défini à la création du fichier du modèle) sont alors chargés, car ils font partie du segment de chargement initial. Vous pouvez exécuter la plupart des instructions Lingo 3D associées aux objets dont la priorité de chargement est zéro. N'utilisez pas les commandes `loadFile` et `resetWorld` dans cet état.
- `3` indique que tous les médias supplémentaires de l'acteur sont chargés et décompressés. La plupart des instructions Lingo 3D peuvent alors être exécutées. N'utilisez pas les commandes `loadFile` et `resetWorld` dans cet état.

- `4` indique que tous les médias de l'acteur ont été entièrement chargés et décompressés. Toutes les instructions Lingo 3D peuvent maintenant être exécutées sur l'acteur.
- `-1` indique qu'une erreur non définie a eu lieu pendant le processus de lecture en flux continu des médias. Etant donné que l'erreur peut avoir eu lieu à n'importe quel moment du chargement, l'état de l'acteur n'est pas fiable.

En règle générale, évitez d'utiliser Lingo avec des acteurs 3D dont l'état actuel est inférieur à 3.

Exemple

L'instruction suivante indique que le chargement de l'acteur séquenceDeFête est terminé, qu'il a eu lieu sans erreur et que l'acteur est prêt pour la lecture.

```
-- Lingo
put member("PartyScene").state
-- 4

// Javascript
trace(member("PartyScene").state);
// 4
```

state (Flash, SWA)

Syntaxe

```
-- Lingo syntax
memberObjRef.state

// JavaScript syntax
memberObjRef.state;
```

Description

Propriété d'acteur ; pour les acteurs Shockwave Audio (SWA) lus en flux continu et les acteurs animation Flash, détermine l'état actuel du fichier en flux continu. Les propriétés `streamName`, `URL` et `preLoadTime` ne peuvent être modifiées que lorsque le son SWA est arrêté.

Les propriétés de fichier SWA suivantes ne renvoient des informations utiles qu'une fois la lecture en flux continu commencée : `cuePointNames`, `cuePointTimes`, `currentTime`, `duration`, `percentPlayed`, `percentStreamed`, `bitRate`, `sampleRate` et `numChannels`.

Pour les acteurs SWA lus en flux continu, les valeurs suivantes sont possibles :

- 0 – Lecture en flux continu de l'acteur interrompue.
- 1 – L'acteur est en cours de rechargement.
- 2 – Préchargement terminé avec succès.
- 3 – Lecture de l'acteur.
- 4 – Interruption de l'acteur (pause).
- 5 – Lecture en flux continu de l'acteur terminée.
- 9 – Une erreur est survenue.
- 10 – Espace processeur insuffisant.

Pour les acteurs animation Flash, cette propriété renvoie une valeur valide uniquement lorsque l'animation Director est exécutée. Les valeurs suivantes sont possibles :

- 0 – L'acteur n'est pas en mémoire.
- 1 – L'en-tête est en cours de chargement.
- 2 – Le chargement de l'en-tête est terminé.
- 3 – Le média de l'acteur est en cours de chargement.
- 4 – Le chargement du média de l'acteur est terminé.
- -1 – Une erreur est survenue.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante émet une alerte si une erreur est détectée pour l'acteur SWA lu en flux continu :

```
-- Lingo syntax
on mouseDown
    if member("Ella Fitzgerald").state = 9 then
        _player.alert("Sorry, can't find an audio file to stream.")
    end if
end

// JavaScript syntax
function mouseDown() {
    var ellaSt = member("Ella Fitzgerald").state;
    if (ellaSt == 9) {
        _player.alert("Sorry, can't find an audio file to stream.");
    }
}
```

Voir aussi

`clearError()`, `getError()` (Flash, SWA)

state (RealMedia)

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.state

// JavaScript syntax
memberOrSpriteObjRef.state;
```

Description

Propriété d'acteur ou image-objet RealMedia ; renvoie l'état actuel du flux RealMedia, exprimé par un nombre entier compris entre 1 et 4. Chaque valeur d'état correspond à un point spécifique dans le processus de lecture en flux continu. Cette propriété est dynamique en cours de lecture et peut être testée, mais pas définie.

Le processus de lecture en flux continu est initialisé lors de l'entrée de la tête de lecture sur l'image-objet RealMedia dans le scénario, lors de l'appel de la méthode `play` dans une image-objet ou un acteur RealMedia ou lorsqu'un utilisateur clique sur le bouton Lire dans la fenêtre RealMedia. L'appel de cette propriété renvoie une valeur numérique indiquant l'état du processus de lecture en flux continu de l'acteur RealMedia. A chaque état correspond une ou plusieurs valeurs de propriété `mediaStatus (RealMedia, Windows Media)` et chaque valeur `mediaStatus` n'est observée que dans un état. Par exemple, les valeurs `#seeking` et `#buffering` de la propriété `mediaStatus` ne sont présentes que lorsque `state` présente la valeur 3.

La valeur de la propriété `state` fournit d'importantes informations utiles pour l'exécution de Lingo sur un acteur. Si la valeur de `member.state` est inférieure à 2, il est possible que certaines propriétés Lingo soient incorrectes et, par conséquent, que les commandes Lingo dépendant de ces propriétés le soient également. Si la valeur `member.state` est supérieure ou égale à 2 et inférieure à 4, l'acteur RealMedia ne s'affiche pas, mais toutes les propriétés et méthodes Lingo comportent des valeurs bien définies qui sont utilisables pour effectuer des opérations Lingo sur l'acteur.

Au démarrage du processus de lecture en flux continu, la propriété `state` passe par les états suivants, à moins qu'une erreur (-1) ne survienne et empêche le démarrage de la lecture en flux continu :

-1 (erreur) indique qu'un problème est survenu, par exemple une erreur due à des éléments restants du flux RealMedia précédent. Vous obtiendrez des informations supplémentaires en examinant la propriété `lastError`. Cet état est l'équivalent de `#error` pour la propriété `mediaStatus`.

0 (fermé) indique que la lecture en flux continu n'a pas démarré ou bien que les propriétés d'acteur sont toujours dans leur état initial ou qu'elles sont constituées de copies provenant d'une lecture antérieure de l'acteur. Cet état est l'équivalent de `#closed` pour la propriété `mediaStatus`.

1 (connexion) indique que la lecture en flux continu a démarré mais se trouve dans les premières étapes de connexion au serveur et que les informations disponibles localement sont encore insuffisantes pour un traitement quelconque de l'acteur. Cet état est l'équivalent de `#connecting` pour la propriété `mediaStatus`.

2 (ouvert) indique que les propriétés Lingo ont été actualisées à partir du flux réel. Lorsque la valeur de `state` est supérieure ou égale à 2, les propriétés `height`, `width` et `duration` de la lecture RealMedia sont connues. Cet état est transitoire et passe rapidement à l'état 3. Cet état est l'équivalent de `#opened` pour la propriété `mediaStatus`.

3 (recherche ou mise en tampon) indique que toutes les propriétés Lingo d'acteur RealMedia sont actives, mais que la lecture de l'acteur n'est pas prête à démarrer. Les fenêtres Scène ou RealMedia affichent un rectangle noir ou le logo RealNetworks. Si cet état est le résultat d'une mise en mémoire tampon due à une congestion réseau, `state` reprend rapidement la valeur `4` (lecture). Cet état est l'équivalent de `#buffering` ou de `#seeking` pour la propriété `mediaStatus`.

4 (lecture) indique que le flux RealMedia est en cours de lecture (ou en pause) et qu'aucune erreur ni problème n'est détecté. Il s'agit de l'état correspondant à une lecture normale. Cet état est l'équivalent de `#playing` ou de `#paused` pour la propriété `mediaStatus`.

Exemple

Les exemples suivants indiquent que l'état des flux de l'image-objet 2 et de l'acteur Real est 0, qui indique la fermeture.

```
-- Lingo syntax
put(sprite(2).state) -- 0
put(member("Real").state) -- 0

// JavaScript syntax
put(sprite(2).state); // 0
put(member("Real").state); // 0
```

Voir aussi

`mediaStatus (RealMedia, Windows Media)`, `percentBuffered`, `lastError`

static

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.static

// JavaScript syntax
memberOrSpriteObjRef.static;
```

Description

Propriété d'acteur et d'image-objet ; associe la performance de lecture d'une image-objet animation Flash à la présence ou non d'éléments d'animation. Si l'animation contient des éléments d'animation (FALSE, valeur par défaut) la propriété redessine l'image-objet pour chaque image ; dans le cas contraire (TRUE), la propriété ne redessine l'image-objet que lorsque cette dernière se déplace ou change de taille.

Cette propriété peut être testée et définie.

Remarque : *ne définissez la propriété* `static` *sur* `TRUE` *que lorsque l'image-objet d'animation Flash ne croise pas d'autres images-objets Director en mouvement. Si l'animation Flash croise des images-objets Director en mouvement, elle risque de ne pas être rafraîchie correctement.*

Exemple

Le script d'image-objet suivant affiche dans la fenêtre Messages le numéro de piste d'une image-objet animation Flash et indique si celle-ci contient des éléments d'animation :

```
-- Lingo syntax
property spriteNum

on beginSprite me
    if sprite(spriteNum).static then
        animationType = "does not have animation."
    else
        animationType = "has animation."
    end if
    put("The Flash movie in channel" && spriteNum && animationType)
end

// JavaScript syntax
function beginSprite() {
    var st = sprite(this.spriteNum).static;
    if (st == 1) {
        animationType = "does not have animation.";
    } else {
        animationType = "has animation.";
    }
    trace("The Flash movie in channel " + this.spriteNum + animationType);
}
```

staticQuality

Syntaxe

```
-- Lingo syntax
spriteObjRef.staticQuality
```

```
// JavaScript syntax
spriteObjRef.staticQuality;
```

Description

Propriété d'image-objet QuickTime VR ; indique la qualité de codec utilisée lorsque l'image panoramique est statique. Les valeurs possibles sont `#minQuality`, `#maxQuality` et `#normalQuality`.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche la propriété staticQuality de l'image-objet QuickTime dans la fenêtre Messages.

```
-- Lingo
put sprite(1).staticQuality
-- #normalQuality

// Javascript
trace(sprite(1).staticQuality)
// symbol(normalQuality)
```

status

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.status

// JavaScript syntax
soundChannelObjRef.status;
```

Description

Propriété de piste audio ; indique l'état d'une piste audio. Lecture seule.

Les valeurs possibles sont :

Etat	Nom	Signification
0	Idle	Aucun son n'est lu ni placé en file d'attente.
1	Loading	Un son en attente est préchargé mais n'est pas encore lu.
2	Queued	La piste audio a terminé le préchargement d'un son en attente mais ne lit pas encore le son.
3	Playing	Un son est en cours de lecture.
4	Paused	Un son est en pause.

Exemple

L'instruction suivante affiche l'état actuel de la piste d'image-objet 2 dans la fenêtre Messages :

```
-- Lingo syntax
put(sound(2).status)

// JavaScript syntax
put(sound(2).status);
```

Voir aussi

Piste audio

stillDown

Syntaxe

```
-- Lingo syntax
_mouse.stillDown

// JavaScript syntax
_mouse.stillDown;
```

Description

Propriété de souris ; indique si l'utilisateur appuie sur le bouton de la souris (TRUE) ou non (FALSE). Lecture seule.

Cette fonction est utile dans les scripts mouseDown pour ne déclencher certains événements qu'après l'exécution de la fonction mouseUp.

Un script ne peut pas tester stillDown si cette propriété est utilisée dans une boucle. Dans une boucle, utilisez plutôt la fonction mouseDown.

Exemple

L'instruction suivante vérifie si le bouton de la souris est enfoncé et, le cas échéant, appelle le gestionnaire dragProcedure :

```
-- Lingo syntax
if (_mouse.stillDown) then
    dragProcedure
end if

// JavaScript syntax
if (_mouse.stillDown) {
    dragProcedure();
}
```

Voir aussi

Souris, mouseDown, mouseUp

stopTime

Syntaxe

```
sprite(whichSprite).stopTime
the stopTime of sprite whichSprite
```

Description

Propriété d'image-objet ; détermine le moment auquel l'image-objet vidéo numérique spécifiée s'arrête. La valeur de stopTime est exprimée en battements.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante arrête l'image-objet vidéo numérique de la piste 5 à 100 battements :

```
-- Lingo
sprite(5).stopTime = 100

// Javascript
sprite(5).stopTime = 100;
```

stopTimeList

Syntaxe

```
-- Lingo syntax
dvdObjRef.stopTimeList

// JavaScript syntax
dvdObjRef.stopTimeList;
```

Description

Propriété de DVD ; liste de propriétés spécifiant le moment ou le chapitre au niveau duquel la lecture s'arrête Lecture/écriture.

La propriété `stopTimeList` correspond à une liste de propriétés pouvant reposer sur un chapitre ou sur une heure.

Une propriété `stopTimeList` reposant sur un chapitre contient les propriétés suivantes :

- `title`. Spécifie le titre.
- `chapter`. Spécifie le chapitre. La lecture s'arrête dès que la lecture de ce chapitre est achevée.

La propriété stopTimeList suivante indique que la lecture doit s'arrêter au chapitre 4 du titre 1 :

```
[#title:1, #chapter:4]
```

Une propriété `stopTimeList` reposant sur une heure contient les propriétés suivantes :

- `title`. Spécifie le titre.
- `hours`. Spécifie l'heure de fin de la lecture.
- `min`. Spécifie la minute à laquelle la lecture prend fin.
- `sec`. Spécifie la seconde à laquelle la lecture prend fin.
- `frames`. Spécifie les images au niveau desquelles la lecture prend fin.

La propriété `stopTimeList` suivante indique que la lecture doit s'arrêter à une heure spécifique du titre 1 :

```
[#title:1, #hours:0, #minutes:55, #seconds:45, #frames:15]
```

La propriété `stopTimeList` suivante ne répertorie qu'un seul paramètre temporel :

```
[#title:1, #seconds:45]
```

Vous pouvez annuler l'effet de la propriété stopTimeList en lui attribuant la valeur 0.

Voir aussi

DVD, play() (DVD), startTimeList

streamMode

Syntaxe

```
-- Lingo syntax
memberObjRef.streamMode

// JavaScript syntax
memberObjRef.streamMode;
```

Description

Propriété d'acteur Flash ; contrôle le mode de transfert en mémoire d'un acteur animation Flash liée, de la manière suivante :

- `#frame` (valeur par défaut) : transfère une partie de l'acteur chaque fois que l'image Director avance pendant que l'image-objet est sur la scène.
- `#idle` : transfère une partie de l'acteur chaque fois qu'un événement d'inactivité est généré ou au moins une fois par image Director pendant que l'image-objet est sur la scène.
- `#manual` : transfère une partie de l'acteur en mémoire uniquement lorsque la commande `stream` est émise pour cet acteur.

Cette propriété peut être testée et définie.

Exemple

Le script `startMovie` suivant effectue une recherche d'acteurs animation Flash dans la distribution interne et attribue à leur propriété `streamMode` la valeur `#manual` :

```
-- Lingo syntax
on startMovie
    repeat with i = 1 to castLib(1).member.count
    if member(i, 1).type = #flash then
        member(i, 1).streamMode = #manual
    end if
    end repeat
end

// JavaScript syntax
function startMovie() {
    i = 1;
    while( i < (castLib(whichCast).member.count) + 1)
        var tp = member(i, whichCast).type;
        if (tp == "flash") {
            member(i, 1).streamMode = symbol("manual");
            i++;
        }
    }
}
```

streamName

Syntaxe

```
-- Lingo syntax
memberObjRef.streamName

// JavaScript syntax
memberObjRef.streamName;
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; spécifie une adresse URL ou un nom de fichier pour un acteur SWA lu en flux continu. Cette propriété fonctionne comme la propriété d'acteur `URL`.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante lie le fichier Orchestre.swa à un acteur SWA lu en flux continu. Le fichier lié se trouve sur le disque nommé monDisque dans le dossier Sons.

```
-- Lingo syntax
member("SWAstream").streamName = "MyDisk/sounds/BigBand.swa"

// JavaScript syntax
member("SWAstream").streamName = "MyDisk/sounds/BigBand.swa";
```

streamSize

Syntaxe

```
-- Lingo syntax
memberObjRef.streamSize

// JavaScript syntax
memberObjRef.streamSize;
```

Description

Propriété d'acteur ; indique un nombre entier correspondant au nombre total d'octets à transférer pour l'acteur spécifié. La propriété `streamSize` ne renvoie une valeur que lorsque l'animation Director est en cours de lecture.

Cette propriété peut être testée, mais pas définie.

Exemple

Le script d'image suivant vérifie si une animation Flash intitulée Intro a été complètement transférée en mémoire. Dans le cas contraire, le script met à jour un acteur champ pour indiquer le nombre d'octets déjà chargés (à l'aide de la propriété d'acteur `bytesStreamed`) et le nombre total d'octets de l'acteur (grâce à la propriété d'acteur `streamSize`). Le script contraint la tête de lecture à passer en boucle sur l'image actuelle jusqu'à ce que l'animation soit complètement chargée en mémoire.

```
-- Lingo syntax
on exitFrame
    if member("Intro Movie").percentStreamed < 100 then
        member("Message Line").text = string(member("Intro Movie").bytesStreamed) && "of" \
&& string(member("Intro Movie").streamSize) && "bytes have downloaded so far."
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var pctStm = member("Intro Movie").percentStreamed;
    var strSs = new String(member("Intro Movie").streamSize);
    var strIm = new String(member("Intro Movie").bytesStreamed);
    if (pctStm < 100) {
        member("Message Line").text = strStm + " of " + strSS+ " bytes have downloaded
so far.";
```

```
        _ movie.go(_movie.frame);
    }
}
```

streamSize (3D)

Syntaxe

```
member(whichCastmember).streamSize
```

Description

Propriété 3D ; permet d'obtenir la taille du flux de données à télécharger, de 0 au maximum. Cette commande fait référence à l'importation du fichier initial ou à la dernière commande `loadFile()` émise.

Exemple

L'instruction suivante indique que le dernier chargement de fichier associé à l'acteur Séquence a une taille totale de 325 300 octets.

```
put member("Scene").streamSize
-- 325300
```

Voir aussi

`bytesStreamed (3D)`, `percentStreamed (3D)`, `state (3D)`, `preLoad (3D)`

strokeColor

Syntaxe

```
-- Lingo syntax
memberObjRef.strokeColor

// JavaScript syntax
memberObjRef.strokeColor;
```

Description

Propriété d'acteur forme vectorielle ; indique la couleur RVB du trait formant les contours de la forme.

Vous pouvez voir un exemple d'utilisation de `strokeColor` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante donne à la propriété strokeColor de l'acteur Trait la couleur rouge :

```
-- Lingo syntax
member("line").strokeColor = color(255, 0, 0)

// JavaScript syntax
member("line").strokeColor = color(255, 0, 0);
```

Voir aussi

`color()`, `fillColor`, `endColor`, `backgroundColor`

strokeWidth

Syntaxe

```
-- Lingo syntax
memberObjRef.strokeWidth

// JavaScript syntax
memberObjRef.strokeWidth;
```

Description

Propriété d'acteur forme vectorielle ; indique la largeur, en pixels, du trait formant les contours de la forme.

Cette valeur est un nombre à virgule flottante compris entre 0 et 100 et peut être testée et définie.

Vous pouvez voir un exemple d'utilisation de `strokeWidth` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante attribue à la propriété `strokeWidth` de l'acteur Trait la valeur de 10 pixels :

```
-- Lingo syntax
member("line").strokeWidth = 10

// JavaScript syntax
member("line").strokeWidth = 10;
```

style

Syntaxe

```
member(whichCastmember).model(whichModel).toon.style
member(whichCastmember).model(whichModel).shader.style
member(whichCastmember).shader(whichShader).style
```

Description

Propriété 3D de modificateur `toon` et de matériau `painter` ; indique la façon dont le modificateur `toon` et le matériau `painter` appliquent la couleur à un modèle. Les valeurs possibles sont :

- `#toon` utilise des transitions aiguës entre les couleurs.
- `#gradient` utilise des transitions fluides entre les couleurs. C'est la valeur par défaut.
- `#blackAndWhite` utilise les couleurs noir et blanc.

Le nombre de couleurs utilisées par le modificateur `toon` et le matériau `painter` est défini à l'aide de la propriété `colorSteps` du modificateur ou du matériau.

Exemple

L'instruction suivante attribue à la propriété `style` du modificateur toon du modèle Théière la valeur `#blackAndWhite`. Le modèle est rendu en noir et blanc.

```
member("Shapes").model("Teapot").toon.style = #blackAndWhite
```

subdivision

Syntaxe

```
member(whichCastmember).model(whichModel).sds.subdivision
```

Description

Propriété 3D de modificateur `sds` ; permet d'obtenir ou de définir le mode opérationnel du modificateur de fractionnement de surface. Les valeurs possibles sont :

- `#uniform` spécifie que la maille est mise à l'échelle en détail de façon uniforme, chaque face étant fractionnée le même nombre de fois.
- `#adaptive` spécifie que des détails supplémentaires ne sont ajoutés qu'en présence d'un changement d'orientation majeur et uniquement aux zones actuellement visibles de la maille.

Le modificateur `sds` ne peut pas être utilisé avec les modificateurs `inker` ou `toon` ; faites également preuve de vigilance lors de l'utilisation du modificateur `sds` avec le modificateur `lod`. Pour plus d'informations, reportez-vous à l'entrée du modificateur `sds`.

Exemple

L'instruction suivante attribue à la propriété `subdivision` du modificateur `sds` du modèle Bébé la valeur `#adaptive`. La géométrie de Bébé n'est pas modifiée de façon uniforme.

```
-- Lingo
member("Scene").model(1).sds.subdivision = #adaptive

// Javascript
Member("Scene").getPropRef("model",1).sds.subdivision = symbol("adaptive");
```

Voir aussi

`sds (modificateur)`, `error`, `enabled (sds)`, `depth (3D)`, `tension`

subPicture

Syntaxe

```
-- Lingo syntax
dvdObjRef.subPicture

// JavaScript syntax
dvdObjRef.subPicture;
```

Description

Propriété de DVD. Détermine la sous-image en cours si elle existe. Lecture/écriture.

La valeur de `subPicture` est un nombre entier. La valeur 0 désactive la propriété `subPicture`.

Voir aussi

`DVD`

subPictureCount

Syntaxe

```
-- Lingo syntax
dvdObjRef.subPictureCount

// JavaScript syntax
dvdObjRef.subPictureCount;
```

Description

Propriété de DVD. Renvoie le nombre de sous-images disponibles. Lecture seule.

Voir aussi

DVD

suspendUpdates

Syntaxe

```
sprite(which3dSprite).suspendUpdates
```

Description

Propriété 3D d'image-objet ; lorsqu'elle est définie sur `TRUE`, elle empêche la mise à jour de l'image-objet dans le cadre des opérations de rafraîchissement normales de l'écran. Ceci peut améliorer les performances de lecture de l'animation. Certains types d'actualisation d'écran peuvent encore affecter l'image-objet, par exemple ceux qui sont engendrés par le glissement d'une autre fenêtre sur l'image-objet. Lorsque la propriété `suspendUpdates` est définie sur `FALSE`, l'image-objet est redessinée normalement.

Il est fondamental que la propriété `suspendUpdates` conserve la valeur `FALSE` lors de l'animation d'un élément quelconque dans l'image-objet 3D.

Exemple

L'instruction suivante suspend les mises à jour sur l'image-objet(3).

```
-- Lingo
sprite(3).suspendUpdates = FALSE

// Javascript
sprite(3).suspendUpdates = false;
```

switchColorDepth

Syntaxe

```
-- Lingo syntax
_player.switchColorDepth

// JavaScript syntax
_player.switchColorDepth;
```

Description

Propriété de lecteur ; détermine si Director change le codage des couleurs du moniteur que la scène occupe conformément au codage des couleurs de l'animation en cours de chargement (TRUE) ou laisse le codage des couleurs du moniteur inchangé lors du chargement de l'animation (FALSE, valeur par défaut). Lecture/écriture.

Si la propriété switchColorDepth présente la valeur TRUE, rien ne se passe tant qu'une nouvelle animation n'est pas chargée.

Il est recommandé d'adapter le codage des couleurs du moniteur à celui de l'animation.

- Si le réglage du codage des couleurs du moniteur est inférieur à celui de l'animation, le fait de l'adapter aux paramètres définis pour l'animation (en supposant que le moniteur puisse offrir ce type de codage) permet de conserver l'aspect initial de l'animation.
- Si le codage des couleurs du moniteur est supérieur à celui de l'animation, la réduction du codage permet d'utiliser moins de mémoire pour exécuter les animations, de charger les acteurs de manière plus efficace et d'accélérer l'animation.

La valeur de cette propriété est également définissable à l'aide de l'option Adapter le moniteur aux couleurs de l'animation dans la boîte de dialogue Préférences générales.

Exemple

L'instruction suivante attribue à la variable intitulée switcher la valeur en cours de la propriété switchColorDepth :

```
-- Lingo syntax
switcher = _player.switchColorDepth

// JavaScript syntax
var switcher = _player.switchColorDepth;
```

L'instruction suivante vérifie si le codage des couleurs en cours est de 8 bits et, le cas échéant, active la propriété switchColorDepth:

```
-- Lingo syntax
if (_system.colorDepth = 8) then
    _player.switchColorDepth = TRUE
end if

// JavaScript syntax
if (_system.colorDepth == 8) {
    _player.switchColorDepth = true;
}
```

Voir aussi

colorDepth, Lecteur

systemTrayIcon

Syntaxe

```
-- Lingo syntax
_movie.displayTemplate.systemTrayIcon
windowObjRef.systemTrayIcon

// JavaScript syntax
```

```
_movie.displayTemplate.systemTrayIcon;
windowObjRef.systemTrayIcon;
```

Description

Propriété d'animation et de fenêtre (Microsoft Windows uniquement). Détermine si une fenêtre est associée à une icône dans la barre d'état système du Bureau d'un utilisateur. Lecture/écriture.

Si la propriété `systemTrayIcon` présente la valeur `TRUE`, une icône de fenêtre est placée dans la barre d'état système.

Si `systemTrayIcon` présente la valeur `FALSE`, aucune icône n'apparaît dans la barre d'état système.

Exemple

L'instruction suivante affiche l'icône de la barre d'état système pendant la lecture de l'animation.

```
-- Lingo
_movie.displayTemplate.systemTrayIcon = TRUE

// Javascript
_movie.displayTemplate.systemTrayIcon = true;
```

Voir aussi

displayTemplate, Animation, systemTrayTooltip, Fenêtre

systemTrayTooltip

Syntaxe

```
-- Lingo syntax
_movie.displayTemplate.systemTrayTooltip
windowObjRef.systemTrayTooltip

// JavaScript syntax
_movie.displayTemplate.systemTrayTooltip;
windowObjRef.systemTrayTooltip;
```

Description

Propriété d'animation et de fenêtre (Microsoft Windows uniquement). Détermine la chaîne qui apparaît dans l'info-bulle associée à l'icône de la barre d'état système. Lecture/écriture.

Cette propriété n'est applicable que si la propriété `systemTrayIcon` est définie sur `TRUE`. Si la propriété `systemTrayIcon` présente la valeur `TRUE`, l'info-bulle apparaît lorsqu'un utilisateur positionne la souris sur l'icône de la barre d'état système.

La valeur par défaut de `systemTrayTooltip` correspond au titre de la fenêtre.

Voir aussi

displayTemplate, Animation, systemTrayIcon, Fenêtre

tabCount

Syntaxe

```
chunkExpression.tabCount
```

Description

Propriété d'acteur texte ; indique le nombre d'arrêts de tabulation uniques présents dans l'expression de sous-chaîne de l'acteur texte.

La valeur est un nombre entier égal ou supérieur à 0 et peut être testée, mais pas définie.

Exemple

L'instruction suivante affiche le nombre de tabulations dans un champ de texte.

```
-- Lingo
put member(1).tabCount
-- 3

// Javascript
trace(member(1).tabCount);
// 3
```

tabs

Syntaxe

```
member(whichTextMember).tabs
```

Description

Propriété d'acteur texte ; contient une liste de tous les arrêts de tabulations définis dans l'acteur texte.

Chaque élément de la liste comporte des informations concernant ces tabulations pour l'acteur. Les propriétés possibles de la liste sont les suivantes :

#type	Peut correspondre à `#left` (gauche), `#center` (centre), `#right` (droite) ou `#decimal` (décimale).
#position	Valeur entière indiquant la position de la tabulation en points.

Cette propriété peut être testée et définie. Lorsque la propriété `tabs` est définie, la propriété `type` est facultative. Si la propriété `type` n'est pas définie, le type de tabulation prend par défaut la valeur `#left`.

Exemple

L'instruction suivante récupère toutes les tabulations de l'acteur texte Crédits et les affiche dans la fenêtre Messages :

```
-- Lingo
put member("Intro credits").tabs
-- [[#type: #left, #position: 36], [#type: #Decimal, #position: 141], [#type: #right,
#position: 216]]

// Javascript
trace(member("Intro credits").tabs)
// < [[#type: #left, #position: 36], [#type: #Decimal, #position: 141], [#type: #right,
#position: 216]]>
```

target

Syntaxe

```
timeoutObject.target
```

Description

Propriété d'objet de temporisation ; indique l'objet enfant auquel l'*objetDeTemporisation* donné enverra ses événements de temporisation. Les objets de temporisation dont la propriété target présente la valeur VOID envoient leurs événements à un gestionnaire dans un script d'animation.

Cette propriété s'avère pratique pour le débogage des comportements et des scripts parents utilisant les objets de temporisation.

Exemple

L'instruction suivante affiche dans la fenêtre Messages le nom de l'objet enfant qui recevra les événements de temporisation de l'objet de temporisation timerOne :

```
-- Lingo
put timeout("timerOne").target

// Javascript
trace(timeout("timerOne").target)
```

Voir aussi

name (temporisation), timeout(), timeoutHandler, timeoutList

targetFrameRate

Syntaxe

```
sprite(which3dSprite).targetFrameRate
```

Description

Propriété 3D d'image-objet ; détermine le nombre d'images par seconde à utiliser lors du rendu d'une image-objet 3D. La valeur par défaut est de 30 images par seconde. La propriété targetFrameRate n'est utilisée que si la propriété useTargetFrameRate est définie sur TRUE. Si la propriété useTargetFrameRate présente la valeur TRUE, Director réduira le nombre de polygones des modèles de l'image-objet pour atteindre la cadence d'image spécifiée ciblée.

Exemple

Les instructions suivantes attribuent à la propriété targetFrameRate de l'image-objet 3 la valeur 45 et appliquent la cadence d'image en définissant la propriété useTargetFrameRate de l'image-objet sur la valeur TRUE :

```
-- Lingo
sprite(3).targetFrameRate = 45
sprite(3).useTargetFrameRate = TRUE

// Javascript
sprite(3).targetFrameRate = 45;
sprite(3).useTargetFrameRate = true;
```

Voir aussi

useTargetFrameRate

tension

Syntaxe

```
member(whichCastmember).model(whichModel).sds.tension
```

Description

Propriété 3D de modificateur `sds` ; permet d'obtenir ou de définir un pourcentage à virgule flottante compris entre 0,0 et 100,0 pour déterminer à quel point la surface nouvellement générée doit refléter la surface d'origine. Plus cette valeur est élevée, plus les surfaces fractionnées correspondent à la surface d'origine. La valeur par défaut est 65,0.

Exemple

L'instruction suivante définit la propriété `tension` du modificateur `sds` du modèle Bébé sur 35. Si la valeur `error` du modificateur `sds` est faible et que sa valeur `depth` est élevée, cette instruction produira un effet très prononcé sur la géométrie de Bébé.

```
-- Lingo
member("scene").model("Baby").sds.tension = 35

// Javascript
member("scene").getPropRef("model",1).sds.tension = 35;
```

Voir aussi

`sds (modificateur)`, `error`, `depth (3D)`

text

Syntaxe

```
-- Lingo syntax
memberObjRef.text

// JavaScript syntax
memberObjRef.text;
```

Description

Propriété d'acteur texte ; détermine la chaîne de caractères de l'acteur champ spécifié par *`quelActeur`*.

La propriété d'acteur `text` est utile pour afficher des messages et enregistrer ce que saisit l'utilisateur.

Cette propriété peut être testée et définie.

Les changements apportés par Lingo au texte d'un acteur suppriment tout formatage que vous auriez pu appliquer aux mots ou lignes. La modification de la propriété d'acteur `text` réapplique le formatage global. Pour changer des portions de texte particulières, vous devez faire référence aux lignes, mots ou éléments qu'il contient.

Lorsque l'animation est lue sous forme d'applet, la valeur de cette propriété est `""` (chaîne vide) pour un acteur champ dont le texte n'a pas encore été lu en flux continu.

Vous pouvez voir un exemple d'utilisation de `text` dans une animation en consultant les animations Forms and Post et Text du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante place Merci dans l'acteur vide Réponse:

```
--Lingo syntax
if (member("Response").text = EMPTY) then
    member("Response").text = "Thank You."
end if

// JavaScript syntax
if (member("Response").text == " ") {
    member("Response").text = "Thank You.";
}
```

L'instruction suivante affecte à l'acteur Remarque la phrase « Vous avez pris la bonne décision ! » :

```
--Lingo syntax
member("Notice").text = "You have made the right decision!"

// JavaScript syntax
member("Notice").text = "You have made the right decision!";
```

Voir aussi

`selEnd`, `selStart`

texture

Syntaxe

```
member(whichCastmember).texture(whichTexture)
member(whichCastmember).texture[index]
member(whichCastmember).shader(whichShader).texture
member(whichCastmember).model(whichModel).shader.texture
member(whichCastmember).model(whichModel).shaderList.texture
member(whichCastmember).model(whichModel).shaderList[index].texture
member(whichCastmember).modelResource(whichParticleSystemModelResource).texture
```

Description

Propriété 3D d'élément et de matériau ; objet image utilisé par un matériau pour définir l'apparence de la surface d'un modèle. L'image est enrobée sur la géométrie du modèle par le matériau.

Le composant visible d'un matériau est créé avec un maximum de huit couches de textures. Ces huit couches de texture ont été créées à partir d'acteurs bitmap ou d'objets images dans Director ou importées avec des modèles de programmes de modélisation 3D.

Créez et supprimez des textures avec les commandes `newTexture()` et `deleteTexture()`.

Les textures sont enregistrées dans la palette des textures de l'acteur 3D. Elles peuvent être référencées par nom (*`quelleTexture`*) ou par index de palette (*`indexDeTexture`*). Une texture peut être utilisée par n'importe quel nombre de matériaux. Les modifications apportées à une texture apparaissent dans tous les matériaux qui l'utilisent.

Il existe trois types de textures :

`#fromCastmember` ; la texture est créée à partir d'un acteur bitmap avec la commande `newTexture()`.

`#fromImageObject` ; la texture est créée à partir d'un objet image Lingo avec la commande `newTexture()`.

`#importedFromFile` ; la texture est importée avec un modèle à partir d'un programme de modélisation 3D.

Pour plus d'informations sur les propriétés de texture, reportez-vous aux rubriques Utilisation de Director dans l'Aide de Director.

La texture d'un système de particules est une propriété de la ressource de modèle, dont le type est `#particle`.

Exemple

L'instruction suivante attribue à la propriété `texture` du matériau surfaceDuMur la texture peintureBleue :

```
-- Lingo
member("Room").shader("WallSurface").texture = member("Room").texture("BluePaint")

// Javascript
member("Room").getPropRef("shader",1).texture = member("Room").getPropRef("texture",1);
```

Voir aussi

`newTexture`, `deleteTexture`

textureCoordinateList

Syntaxe

```
member(whichCastmember).modelResource(whichmodelResource).
textureCoordinateList
modelResourceObjectReference.textureCoordinateList
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#mesh` ou avec un modificateur `meshDeform` associé à un modèle, cette propriété permet d'obtenir ou de définir la propriété `textureCoordinateList` de la ressource de modèle.

La propriété `textureCoordinateList` est une liste de sous-listes identifiant les positions à utiliser dans une image pour le placage de texture sur un triangle. Chaque sous-liste est composée de deux valeurs indiquant une position dans une texture. Ces valeurs doivent être comprises entre 0,0 et 1,0 pour pouvoir être redimensionnées en textures de taille arbitraire. La valeur par défaut est une liste vide.

Manipulez `modelResource.face[index].textureCoordinates` ou `model.meshdeform.mesh[index].face[index]` pour changer la correspondance entre `textureCoordinates` et les coins d'une face de la maille.

Exemple

```
-- Lingo
put member(5).modelResource("mesh square").textureCoordinateList
--[ [0.1, 0.1], [0.2, 0.1], [0.3, 0.1], [0.1, 0.2], [0.2, 0.2], [0.3, 0.2], [0.1, 0.3], [0.2,
0.3], [0.3, 0.3] ]

// Javascript
trace(member(5).getPropRef("modelResource",1).textureCoordinateList;
// <[ [0.1, 0.1], [0.2, 0.1], [0.3, 0.1], [0.1, 0.2], [0.2, 0.2], [0.3, 0.2], [0.1, 0.3],
[0.2, 0.3], [0.3, 0.3] ]>
```

Voir aussi

`face`, `texture`, `meshDeform (modificateur)`

textureCoordinates

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).face[faceIndex].textureCoordinat
es
modelResourceObject.face[faceIndex].textureCoordinates
```

Description

Propriété 3D ; identifie les éléments de la propriété `textureCoordinateList` à utiliser pour la face d'*`indexDeFace`*. Cette propriété doit être une liste de trois entiers spécifiant les index de la propriété `textureCoordinateList`, correspondant aux coordonnées de texture à utiliser pour chaque coin de la face de la maille.

Voir aussi

`face`, `textureCoordinateList`

textureLayer

Syntaxe

```
member(whichCastmember).model(whichModel).meshDeform.mesh[index].textureLayer.count
member(whichCastmember).model(whichModel).meshdeform.mesh[index].texturelayer.add()
member(whichCastmember).model(whichModel).meshdeform.mesh[index].texturelayer[index].textu
reCoordinateList.
```

Description

Propriétés 3D de modificateur `meshdeform` ; ces propriétés permettent d'obtenir et de définir des informations concernant les couches de texture d'une maille spécifiée.

Un matériau peut contenir jusqu'à huit couches de texture, chaque couche ne pouvant contenir qu'une seule texture, la même texture pouvant être spécifiée pour plus d'une couche. Les couches de texture sont celles utilisées par les matériaux.

Utilisez les propriétés suivantes pour accéder aux couches de texture et les manipuler :

`meshdeform.mesh[index].texturelayer.count` renvoie le nombre de couches de texture pour la maille spécifiée.

`model.meshdeform.mesh[index].texturelayer.add()` ajoute une couche de texture vide à la maille spécifiée.

`model.meshdeform.mesh[index].texturelayer[index].texturecoordinatelist` permet d'obtenir ou de définir une liste de coordonnées de texture pour une couche spécifique de la maille spécifiée. Vous pouvez également utiliser cette propriété pour copier les coordonnées de texture entre différentes couches, tel qu'indiqué ci-dessous :

```
model.meshdeform.texturelayer[a].texturecoordinatelist =
model.meshdeform.texturelayer[b].texturecoordinatelist
```

Voir aussi

`meshDeform (modificateur)`, `mesh (propriété)`, `textureCoordinateList`, `add (texture 3D)`, `count`, `texture`, `textureModeList`

textureList

Syntaxe

```
member(whichMember).model(whichModel).shader(whichShader).textureList
member(whichMember).model(whichModel).shader(whichShader).textureList[index]
```

Description

Propriété 3D de matériau ; détermine la liste des textures appliquées au matériau. Un matériau peut utiliser jusqu'à huit couches de texture. Lorsque testée, cette propriété renvoie une liste linéaire d'objets de texture. Lorsque définie sans spécifier d'index, cette propriété spécifie un objet de texture à appliquer à toutes les couches. La définition de la propriété `textureList` sur la valeur `VOID` désactive l'application de textures pour toutes les couches. La valeur par défaut est `VOID`.

Pour tester ou définir l'objet de texture d'une couche de texture spécifique, vous devez inclure une valeur d'index.

Exemple

L'instruction suivante définit la troisième couche de texture du matériau SurfaceDuMur sur la texture PeintureBleue dans l'acteur Pièce :

```
-- Lingo
member(3).model("Car").shader("WallSurface").textureList[3] =
member("Room").texture("BluePaint")

// Javascript
member(3).getPropRef("model",1).getPropRef("shader",1).textureList[3] =
member("Room").getPropRef("texture",1);
```

Voir aussi

`textureModeList`

textureMember

Syntaxe

```
member(whichCastmember).textureMember
```

Description

Propriété 3D d'acteur ; indique le nom de l'acteur bitmap utilisé comme source de la texture par défaut pour l'acteur 3D.

La propriété `textureType` de l'acteur 3D doit présenter la valeur `#member` pour que la propriété `textureMember` soit effective.

Exemple

L'instruction suivante attribue à la propriété `textureMember` de l'acteur scèneDeJardin la valeur Haie. Si la propriété `textureType` de scèneDeJardin présente la valeur `#member`, l'acteur Haie devient le bitmap source pour la texture par défaut de scèneDeJardin.

```
-- Lingo
member("YardScene").textureMember = "Fence"

// Javascript
member("YardScene").textureMember = "Fence";
```

Voir aussi

textureType

textureMode

Syntaxe

```
member(whichCastmember).shader(whichShader).textureMode
member(whichCastmember).model(whichModel).shader.textureMode
member(whichCastmember).model(whichModel).shaderList{[index]}.textureMode
```

Description

Propriété 3D de matériau #standard ; spécifie la façon dont la première couche de texture est plaquée sur la surface du modèle. Utilisez la propriété textureModeList pour spécifier les textures des couches autres que la première. Cette propriété est ignorée si le modificateur #toon est appliqué à la ressource de modèle.

Les valeurs possibles de cette propriété sont #none, #wrapPlanar, #wrapCylindrical, #wrapSpherical, #reflection, #diffuseLight et #specularLight. Pour obtenir la description de ces termes, reportez-vous à l'entrée textureModeList.

Exemple

L'instruction suivante définit la propriété textureMode de la première couche de texture du matériau du modèle Balle sur la valeur #wrapSpherical :

```
-- Lingo
member("scene").model("Ball").shader.textureMode = #wrapSpherical

// Javascript
member("scene").getPropRef("model",1).getProp("shader").textureMode =
symbol("wrapSpherical");
```

Voir aussi

textureModeList

textureModeList

Syntaxe

```
member(whichCastmember).shader(whichShader).textureModeList
member(whichCastmember).shader(whichShader).
textureModeList[textureLayerIndex]
member(whichCastmember).model(whichModel).shader.textureModeList
member(whichCastmember).model(whichModel).shader.
textureModeList[textureLayerndex]
```

Description

Propriété 3D de matériau standard ; permet de changer la façon dont une couche de texture est plaquée sur la surface d'un modèle. Cette propriété est ignorée si le modificateur #toon est appliqué à la ressource de modèle. Les valeurs possibles sont :

- #none utilise les valeurs de coordonnées de texture définies à l'origine pour la ressource de modèle. Ce paramètre désactive wrapTransform et wrapTransformList[indexDeCoucheDeTexture].

- `#wrapPlanar` enrobe la surface du modèle avec la texture comme s'il s'agissait d'une rétroprojection. La propriété `wrapTransformList[indexDeCoucheDeTexture]` du matériau est appliquée à l'espace de placage avant la génération des coordonnées de texture dans l'espace du modèle. Avec une propriété `wrapTransformList[indexDeCoucheDeTexture]` d'identité (valeur par défaut), le placage planaire est orienté de façon à extruder la texture le long de l'axe des z, avec le haut de la texture le long de l'axe des y.

- `#wrapCylindrical` enrobe la texture autour de la surface comme si la surface était placée au milieu de la texture, puis que la texture était enroulée autour de la surface pour former un cylindre. La propriété `wrapTransformList[indexDeCoucheDeTexture]` est appliquée à l'espace de placage avant la génération des coordonnées de texture dans l'espace du modèle. Avec une propriété `wrapTransformList[indexDeCoucheDeTexture]` d'identité (valeur par défaut), le placage cylindrique est orienté de façon à ce que la texture soit enrobée depuis l'axe des y négatif, depuis le bord gauche de la texture vers l'axe des x positif autour de l'axe des z. Le haut de la texture suit l'axe des z positif.

- `#wrapSpherical` enrobe la texture autour de la surface comme si la surface était placée au milieu de la texture, puis que les quatre coins de la texture étaient tirés pour se rencontrer en haut. La propriété `wrapTransformList[indexDeCoucheDeTexture]` est appliquée à l'espace de placage avant la génération des coordonnées de texture dans l'espace du modèle. Avec une propriété `wrapTransformList[indexDeCoucheDeTexture]` d'identité, le placage sphérique est placé à l'origine de l'espace du modèle et est orienté de façon à ce que la texture soit enrobée depuis l'axe des y négatif, depuis le bord gauche de la texture vers l'axe des x positif autour de l'axe des z. Le haut de la texture suit l'axe des z positif.

- `#reflection` est similaire à `#wrapSpherical`, à l'exception du fait que les nouvelles coordonnées de texture sont continuellement projetées sur la surface à partir d'un point fixe. Les coordonnées de texture ne pivotent pas en même temps que le modèle. Simule la lumière réfléchie sur un objet par son environnement. Ce paramètre désactive la propriété `wrapTransform`.

- `#diffuseLight` génère des valeurs de coordonnées de texture de lumière diffuse, une par sommet, puis stocke les résultats dans la maille référencée. Ce paramètre désactive la propriété `wrapTransform`.

- `#specularLight` génère des valeurs de coordonnées de texture de lumière spéculaire, une par sommet, puis stocke les résultats dans la maille référencée. Ce paramètre désactive la propriété `wrapTransform`.

Exemple

Dans l'exemple suivant, un matériau est configuré pour simuler une balle de jardin avec des reflets. La première couche de texture du matériau est un placage sphérique et la troisième couche utilise un placage de style `#reflection`. L'entrée textureList[3] du matériau semble se refléter depuis l'environnement sur tous les modèles utilisant ce matériau.

```
-- Lingo
member("scene").shader("GardenBall).textureList[1] =
member("scene").texture("FlatShinyBall")
member("scene").shader("GardenBall").textureModeList[1] = #wrapSpherical
member("scene").shader("GardenBall").textureList[3] =
member("scene").texture("GardenEnvironment")
member("scene").shader("GardenBall").textureModeList[3] = #reflection

// Javascript
member("scene").getPropRef("shader",1).textureList[1] =
member("scene").getPropRef("texture",1);
member("scene").getPropRef("shader",1).textureModeList[1] = symbol("wrapSpherical");
member("scene").getPropRef("shader",1).textureList[3] =
member("scene").getPropRef("texture",1);
member("scene").getPropRef("shader",1).textureModeList[3] = symbol("reflection");
```

Voir aussi

`textureTransformList`, `wrapTransform`

textureRenderFormat

Syntaxe

```
getRenderServices().textureRenderFormat
```

Description

Propriété 3D de `rendererServices` ; permet d'obtenir ou de définir le format binaire utilisé par toutes les textures des acteurs 3D. Utilisez la propriété `textureRenderFormat` pour annuler ce paramètre pour certaines textures uniquement. Les formats binaires plus réduits (c'est-à-dire les variantes 16 bits telles que `#rgba5551`) utilisent une quantité plus réduite de RAM vidéo de l'accélérateur matériel, ce qui permet d'utiliser un plus grand nombre de textures avant d'être forcé de passer au rendu logiciel. Les formats binaires plus élevés (c'est-à-dire les variantes 32 bits telles que `#rgba8888`) offrent généralement un meilleur résultat. Pour utiliser la transparence alpha dans une texture, le dernier bit ne doit pas être nul. Pour obtenir un bon effet de transparence, le canal alpha doit avoir une précision supérieure à un bit.

Les formats de pixels comptent chacun quatre chiffres, chaque chiffre indiquant le degré de précision pour le rouge, le vert, le bleu et l'alpha. La valeur choisie détermine la précision des couleurs (la précision du canal alpha) et la quantité de mémoire utilisée par le tampon des textures du matériel. Vous pouvez choisir une valeur qui améliore la fidélité des couleurs ou une valeur vous permettant de stocker davantage de textures sur la carte. Vous pouvez faire tenir deux fois plus de textures 16 bits que de textures 32 bits dans un même espace. Lorsqu'une animation utilise plus de textures que la carte ne peut en contenir, Director passe au rendu `#software`.

Vous pouvez spécifier l'une des valeurs suivantes pour `textureRenderFormat` :

- `#rgba8888` : mode de couleur 32 bits avec 8 bits chaque pour rouge, vert, bleu et alpha.
- `#rgba8880` : comme ci-dessus, sans valeur alpha.
- `#rgba5650` : mode de couleur 16 bits sans alpha ; 5 bits pour rouge, 6 bits pour vert et 5 bits pour bleu.
- `#rgba5550` : mode de couleur 16 bits sans alpha ; 5 bits chaque pour rouge, vert et bleu, sans mesure alpha.
- `#rgba5551` : mode de couleur 16 bits avec 5 bits chaque pour rouge, vert et bleu ; 1 bit pour alpha.
- `#rgba4444` : mode de couleur 16 bits avec 4 bits chaque pour rouge, vert, bleu et alpha.

La valeur par défaut est `#rgba5551`.

Exemple

L'instruction suivante attribue à la propriété globale `textureRenderFormat` de l'acteur 3D la valeur `#rgba8888`. Chaque texture de cette animation est rendue avec des couleurs 32 bits, sauf si sa propriété `textureRenderFormat` présente une autre valeur que `#default`.

```
-- Lingo
getRendererServices().textureRenderFormat = #rgba8888

// Javascript
getRendererServices().textureRenderFormat = symbol("rgba8888");
```

Voir aussi

`renderer`, `preferred3dRenderer`, `renderFormat`, `getRendererServices()`

textureRepeat

Syntaxe

```
member(whichCastmember).shader(whichShader).textureRepeat
member(whichCastmember).model(whichModel).shader.textureRepeat
member(whichCastmember).model(whichModel).shaderList{[index]}.textureRepeat
```

Description

Propriété 3D de matériau `#standard` ; contrôle le comportement de verrouillage de la première couche de texture du matériau. Utilisez la propriété `textureRepeatList` pour contrôler cette propriété pour les couches de texture autres que la première.

Lorsque `textureRepeat` présente la valeur `TRUE` et que la valeur des composants x et/ou y de `référenceDeMatériau.textureTransform.scale` est inférieure à 1, la texture est juxtaposée (répétée) sur la surface du modèle.

Lorsque `textureRepeat` présente la valeur `FALSE`, la texture n'est pas juxtaposée. Si la valeur des composants x et/ou y de `référenceDeMatériau.textureTransform.scale` est inférieure à 1, les zones du modèle qui ne sont pas recouvertes par la texture sont noires. Si la valeur des composants x et/ou y de `référenceDeMatériau.textureTransform.scale` est supérieure à 1, les portions de la texture dépassant la plage de coordonnées sont rognées.

La valeur par défaut de cette propriété est `TRUE`. Cette propriété présente toujours la valeur `TRUE` avec le moteur de rendu `#software`.

Exemple

L'instruction suivante attribue à la propriété `textureRepeat` du premier matériau utilisé par le modèle gbCyl3 la valeur `TRUE`. La première texture de ce matériau est juxtaposée si la valeur du composant x ou y de sa propriété `textureTransform` ou `textureTransformList` est inférieure à 1.

```
-- Lingo
member("scene").model("gbCyl3").shader.textureRepeat = TRUE

// Javascript
member("scene").getPropRef("model",1).getProp("shader").textureRepeat = true;
```

Voir aussi

textureTransform, textureTransformList

textureRepeatList

Syntaxe

```
shaderReference.textureRepeatList[textureLayerIndex]
member(whichCastmember).shader(whichShader).textureRepeatList[textureLayerIndex]
member(whichCastmember).shader[shaderListIndex].textureRepeatList[textureLayerIndex]
member(whichCastmember).model(whichModel).shader.textureRepeatList[textureLayerIndex]
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].
textureRepeatList[textureLayerIndex]
```

Description

Propriété 3D de matériau standard ; permet d'obtenir ou de définir le comportement de verrouillage de texture de toute couche de texture. Lorsque cette propriété présente la valeur par défaut `TRUE`, la texture de `indexDeCoucheDeTexture` peut être juxtaposée (répétée) plusieurs fois sur les surfaces du modèle. Vous pouvez obtenir ce résultat en définissant `référenceDeMatériau.textureTransform[indexDeCoucheDeTexture].scale` sur une valeur inférieure à 1 dans x ou y. Lorsque cette propriété est définie sur FALSE, la texture s'applique à une portion plus réduite des surfaces du modèle au lieu d'être juxtaposée sur ces surfaces lorsque la valeur de `référenceDeMatériau.textureTransform[indexDeCoucheDeTexture].scale` est inférieure à 1 dans x ou y. Une illustration de cet effet serait la réduction de l'image source dans le cadre de l'image d'origine et le remplissage de l'espace en noir. De même, si la valeur de `référenceDeMatériau.textureTransform[indexDeCoucheDeTexture].scale` est supérieure à 1 dans x ou y, les portions de la texture dépassant la plage de coordonnées sont rognées.

Exemple

Le code suivant donne à une sphère une texture granitée répétée quatre fois sur toute la surface, ainsi qu'un logo qui ne couvrira qu'un quart de la surface.

```
-- Lingo
m = member(2).model("mySphere")
f = member(2).newTexture("granite", #fromCastmember, member("granite"))
g = member(2).newTexture("logo", #fromCastmember, member("logo"))
s = member(2).newShader("s", #standard)
s.textureList[1] = g
s.textureList[2] = f
s.textureRepeatList[2] = false
s.textureRepeatList[1] = true
s.textureTransformList[1].scale(0.5,0.5,1.0)
s.textureTransformList[2].scale(0.5,0.5,1.0)
s.textureModeList[2] = #wrapPlanar
s.blendFunctionList[2] = #add
m.shaderList = s

// Javascript
m = member(2).getPropRef("model","1");
f = member(2).newTexture("granite", symbol("fromCastmember"), member("granite"));
g = member(2).newTexture("logo", symbol("fromCastmember"), member("logo"));
s = member(2).newShader("s", symbol("standard"));
s.textureList[1] = g;
s.textureList[2] = f;
s.textureRepeatList[2] = false;
s.textureRepeatList[1] = true;
s.textureTransformList[1].scale(0.5,0.5,1.0);
s.textureTransformList[2].scale(0.5,0.5,1.0);
s.textureModeList[2] = symbol("wrapPlanar");
s.blendFunctionList[2] = symbol("add");
m.shaderList = s;
```

textureTransform

Syntaxe

```
member(whichCastmember).shader(whichShader).textureTransform
member(whichCastmember).model(whichModel).shader.textureTransform
member(whichCastmember).model(whichModel).shaderList{[index]}.textureTransform
```

Description

Propriété 3D de matériau `#standard` ; donne accès à une transformation qui modifie les coordonnées de la première couche de texture du matériau. Manipulez ces transformations pour juxtaposer, faire pivoter ou déplacer la texture avant de l'appliquer à la surface du modèle. La texture même n'est pas affectée, la transformation ne modifiant que la façon dont le matériau applique la texture. La propriété `textureTransform` est appliquée à toutes les coordonnées de texture, quelle que soit la valeur de la propriété `textureMode`. Il s'agit de la dernière modification des coordonnées de la texture avant leur envoi au moteur de rendu. La propriété `textureTransform` est une matrice qui opère sur la texture de l'espace textureImage. L'espace `TextureImage` est défini pour exister uniquement sur le plan x,y.

Pour juxtaposer l'image deux fois le long de son axe horizontal, utilisez `référenceDeMatériau.textureTransform.scale(0.5, 1.0, 1.0)`. Le redimensionnement sur l'axe des z est ignoré.

Pour décaler l'image de `point(décalageX,décalageY)`, utilisez `référenceDeMatériau.textureTransform.translate(décalageX,décalageY,0.0)`. La translation en fonction d'entiers lorsque la propriété `textureRepeat` du matériau présente la valeur `TRUE` n'a aucun effet étant donné que la largeur et la hauteur de la texture ont une valeur comprise entre 0,0 et 1,0 dans ce cas.

Pour appliquer une rotation à une couche de texture, utilisez `référenceDeMatériau.textureTransform.rotate(0,0,angle)`. Les rotations sur l'axe des z sont effectuées autour du second point (0, 0) qui correspond au coin supérieur gauche de la texture. Les rotations sur les axes des x et des y sont ignorées.

Tout comme pour les transformations de modèle, les modifications `textureTransform` peuvent être multicouches. Pour faire pivoter la texture autour de point(décalageX, décalageY) au lieu de point(0, 0), effectuez d'abord une translation vers point(0 – décalageX, 0 – décalageY), faites pivoter, puis effectuez une translation vers point(décalageX, décalageY). La propriété `textureTransform` est similaire à la propriété `wrapTransform` du matériau, avec les exceptions suivantes. Elle est appliquée dans un espace image 2D plutôt que dans un espace d'univers 3D. Seules les rotations autour de l'axe des z et les translations et redimensionnements sur les axes des x et y ont un résultat. La transformation est appliquée quel que soit le paramètre `référenceDeMatériau.textureMode`. En comparaison, la propriété `wrapTransform` n'est efficace que lorsque `textureMode` présente la valeur `#wrapPlanar`, `#wrapCylindrical` ou `#wrapSpherical`.

Exemple

L'instruction suivante indique la propriété `textureTransform` de la première texture du premier matériau utilisé par le modèle gbCyl3 :

```
-- Lingo
put member("Scene").model("gbCyl3").shader.textureTransform
-- transform(1.0000, 0.0000, 0.0000,0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000,
1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000)

// Javascript
trace(member("Scene").getPropRef("model",1).getProp("shader").textureTransform);
// <transform(1.0000, 0.0000, 0.0000,0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000,
1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000)>
```

L'instruction suivante divise par deux la hauteur et la largeur de la première texture utilisée par le matériau gbCyl3. Si la propriété `textureRepeat` de gbCyl3 présente la valeur `TRUE`, quatre copies de la texture sont juxtaposées sur le matériau.

```
-- Lingo
member("Scene").shader("gbCyl3").textureTransform.scale = vector(0.5, 0.5, 1)
```

```
// Javascript
member("Scene").getPropRef("shader",1).textureTransform.scale = vector(0.5,0.5,1);
```

L'instruction suivante fait pivoter la première texture utilisée par le matériau gbCyl3 de 90 degrés à partir de vector(0, 0, 0) :

```
-- Lingo
member("Scene").shader("gbCyl3").textureTransform.rotation = vector(0, 0, 90)

// Javascript
member("Scene").getPropRef("shader",1).textureTransform.rotation = vector(0,0,90);
```

textureTransformList

Syntaxe

```
shaderReference .textureTransformList[textureLayerIndex]
member(whichCastmember).shader(ShaderName).textureTransformList[textureLayerIndex]
member(whichCastmember).shader[shaderListIndex].textureTransformList[textureLayerIndex]
member(whichCastmember).model(modelName).shader.textureTransformList[textureLayerIndex]
member(whichCastmember).model(modelName).shaderList[shaderListIndex].
textureTransformList[textureLayerIndex]
```

Description

Propriété 3D de matériau standard ; donne accès à une transformation qui modifie les coordonnées de texture d'une couche de texture spécifiée. Manipulez ces transformations pour juxtaposer, faire pivoter ou déplacer une image de texture avant de l'appliquer à la surface d'un modèle. La texture même n'est pas affectée, la transformation ne modifiant que la façon dont le matériau applique la texture.

Pour juxtaposer l'image deux fois le long de son axe horizontal, utilisez `textureTransformList[quelleCoucheDeTexture].scale(0.5, 1.0, 1.0)`. Les redimensionnements sur l'axe des z sont ignorés étant donné que les images sont en 2D. Les échelles 0,0 doivent être évitées (même dans z) afin de ne pas annuler l'effet de la texture tout entière.

Pour décaler l'image de point(décalageX,décalageY), utilisez `textureTransformList[quelleCoucheDeTexture].translate(décalageX,décalageY,0.0)`. La translation en fonction d'entiers lorsque la propriété textureRepeat de cette couche de texture présente la valeur `TRUE` n'a aucun effet étant donné que la largeur et la hauteur de la texture ont une valeur comprise entre 0,0 et 1,0 dans ce cas.

Pour appliquer une rotation à une couche de texture, utilisez `textureTransformList[quelleCoucheDeTexture].rotate(0,0,angle)`. Les rotations sur l'axe des z sont effectuées autour du second point (0,0) qui correspond au coin supérieur gauche de la texture. Les rotations sur les axes des x et des y sont ignorées étant donné que les images sont en 2D par nature.

Tout comme pour les transformations de modèle, les modifications `textureTransform` peuvent être multicouche. Pour faire pivoter l'image autour de point(décalageX,décalageY) au lieu de point(0, 0), effectuez d'abord une translation vers point(0 - décalageX, 0 -décalageY), faites pivoter, puis effectuez une translation vers point(décalageX, décalageY).

La propriété `textureTransformList` est similaire à la propriété `wrapTransformList` du matériau, avec les exceptions suivantes.

Elle est appliquée dans un espace image 2D plutôt que dans un espace d'univers 3D. Seules les rotations autour de l'axe des z et les translations et redimensionnements sur les axes des x et y ont un résultat.

La transformation est appliquée quel que soit le paramètre `référenceDeMatériau.textureModeList[index]`. En comparaison, la propriété `wrapTransform` n'est efficace que lorsque `textureMode` présente la valeur `#wrapPlanar`, `#wrapCylindrical` ou `#wrapSpherical`.

Exemple

L'instruction suivante indique la transformation de texture de la troisième texture du premier matériau utilisé par le modèle gbCyl3.

```
-- Lingo
put member("scene").model("gbCyl3").shader.textureTransformList[3]
-- transform(1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000,
1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000)

// Javascript
trace(member("scene").getPropRef("model",1).getProp("shader").textureTransformList[3])
//< transform(1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000,
0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000)>
```

L'instruction suivante divise par deux la hauteur et la largeur de la cinquième texture utilisée par le matériau gbCyl3. Si la propriété `textureRepeatList[5]` de gbCyl3 présente la valeur `TRUE`, quatre copies de la texture sont juxtaposées sur le matériau.

```
-- Lingo
member("scene").shader("gbCyl3").textureTransformList[5].scale = vector(0.5, 0.5, 1)

// Javascript
member("scene").getPropRef("shader","1").textureTransformList[5].scale = vector(0.5, 0.5,
1);
```

L'instruction suivante fait pivoter la quatrième texture utilisée par le matériau gbCyl3 de 90 degrés à partir de vector(0, 0, 0) :

```
-- Lingo
member("scene").shader("gbCyl3").textureTransformList[4].rotation = vector(0, 0, 90)

// Javascript
member("scene").getPropRef("shader","1 ").textureTransformList[4].rotation = vector(0, 0, 90);
```

Les instructions suivantes font pivoter la troisième texture utilisée par le matériau gbCyl3 de 90 degrés autour de son centre en supposant que textureList[3] est une texture de 128x128 :

```
-- Lingo
s = member("scene").shader("gbCyl3")
s.textureTransformList[3].translate(-64,-64,0)
s.textureTransformList[3].rotate(0,0,90)
s.textureTransformList[3].translate(64,64,0)

// Javascript
s = member("scene").getPropRef("shader","1");
s.textureTransformList[3].translate(-64,-64,0);
s.textureTransformList[3].rotate(0,0,90);
s.textureTransformList[3].translate(64,64,0);
```

textureType

Syntaxe

```
member(whichCastmember).textureType
```

Description

Propriété 3D de texture ; permet d'obtenir ou de définir le type de la texture par défaut. Les valeurs possibles sont :

- `#none` ne spécifie aucun type de texture.
- `#default` utilise la texture du matériau d'origine.
- `#member` utilise comme texture l'image de l'acteur spécifié.

La valeur par défaut de cette propriété est `#default`. Vous devez définir cette propriété sur la valeur `#member` pour pouvoir utiliser la propriété `textureMember`.

Exemple

L'instruction suivante attribue à la propriété `textureType` de l'acteur Séquence la valeur `#member`.

```
member("Scene").textureType = #member
```

Ceci permet d'utiliser un acteur bitmap comme source de texture par défaut en définissant la propriété `textureMember`. L'acteur bitmap est appelé « herbe ».

```
member("Scene").textureMember = "grass"
```

Voir aussi

`textureMember`

thumbNail

Syntaxe

```
-- Lingo syntax
memberObjRef.thumbNail

// JavaScript syntax
memberObjRef.thumbNail;
```

Description

Propriété d'acteur ; contient l'image à utiliser pour afficher l'aperçu d'un acteur dans la fenêtre Distribution. Lecture/écriture uniquement en phase de création.

Cette image peut être personnalisée pour n'importe quel acteur.

Exemple

L'exemple suivant indique comment utiliser un acteur servant de repère d'emplacement pour afficher une miniature sur la scène. Cet acteur est placé sur la scène, puis son image est définie sur la miniature de l'acteur 10. Ceci permet d'afficher une image réduite sans devoir procéder à une mise à l'échelle ou à une manipulation du graphique :

```
-- Lingo syntax
member("Placeholder").picture = member(10).thumbNail

// JavaScript syntax
member("Placeholder").picture = member(10).thumbNail;
```

Voir aussi

Acteur

tilt

Syntaxe

```
-- Lingo syntax
spriteObjRef.tilt

// JavaScript syntax
spriteObjRef.tilt;
```

Description

Propriété d'image-objet QuickTime VR ; inclinaison actuelle, en degrés, de l'animation QuickTime VR.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante indique comment obtenir l'inclinaison de l'animation QuickTime en cours dans l'image-objet(3).

```
-- Lingo
putsprite(3).tilt

// Javascript
trace(sprite(3).tilt);
```

time (objet de temporisation)

Syntaxe

```
timeoutObject.time
```

Description

Propriété d'objet de temporisation ; heure système, exprimée en millisecondes, de l'envoi du prochain événement de temporisation par l'*objetDeTemporisation* donné.

Il ne s'agit pas du temps restant jusqu'au prochain événement, mais de la position temporelle absolue du prochain événement de temporisation.

Exemple

Le gestionnaire suivant détermine le temps restant jusqu'à l'envoi du prochain événement `timeout` par l'objet de temporisation `Update` en calculant la différence entre sa propriété `time` et la valeur en cours des millisecondes, puis en affichant le résultat dans le champ Temps restant :

```
-- Lingo
on prepareFrame
    msBeforeUpdate = timeout("Update").time - the milliseconds
    secondsBeforeUpdate = msBeforeUpdate / 1000
    minutesBeforeUpdate = secondsBeforeUpdate / 60
    member("Time Until").text = string(minutesBeforeUpdate) && "minutes before next \
```

```
    timeout"
end

// Javascript
Function prepareFrame()
{
    var msBeforeUpdate = timeout("Update").time - _system.milliseconds;
    var secondsBeforeUpdate = msBeforeUpdate / 1000;
    var minutesBeforeUpdate = secondsBeforeUpdate / 60;
    member("Time Until").text = string(minutesBeforeUpdate) + " minutes before next
timeout";
}
```

Voir aussi

milliseconds, period, persistent, target, timeout(), timeoutHandler

timeoutHandler

Syntaxe

```
timeoutObject.timeoutHandler
```

Description

Propriété système ; représente le nom du gestionnaire qui recevra les messages de temporisation de l'*objetDeTemporisation* donné. Sa valeur est un symbole, tel que `#délaiDépassé`. La propriété `timeoutHandler` est toujours un gestionnaire compris dans l'objet `target` de l'objet de temporisation ou dans un script d'animation si aucune propriété `target` n'a été définie pour l'objet de temporisation.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche la propriété `timeoutHandler` de l'objet Minuterie du jeu dans la fenêtre Messages :

```
-- Lingo
put timeout("Quiz Timer").timeoutHandler

// Javascript
trace(timeout("Quiz Timer").timeoutHandler);
```

Voir aussi

target, timeout(), timeoutList

timeoutList

Syntaxe

```
-- Lingo syntax
_movie.timeoutList

// JavaScript syntax
_movie.timeoutList;
```

Description

Propriété d'animation ; liste linéaire contenant tous les objets de temporisation actuellement actifs. Lecture seule.

Pour supprimer un objet de temporisation, utilisez la méthode `forget()`.

Pour ajouter des objets de temporisation à la propriété `timeoutList`, utilisez la méthode `new()`.

Exemple

L'instruction suivante supprime le troisième objet de temporisation de la liste `timeoutList` :

```
-- Lingo syntax
_movie.timeoutList[3].forget()

// JavaScript syntax
_movie.timeoutList[3].forget();
```

Voir aussi

`forget() (fenêtre)`, `Animation`, `new()`, `forget() (temporisation)`, `timeout()`

timeScale

Syntaxe

```
member(whichCastMember).timeScale
the timeScale of member whichCastMember
```

Description

Propriété d'acteur ; renvoie l'unité temporelle par seconde sur laquelle reposent les images de la vidéo numérique. Par exemple, une vidéo numérique QuickTime est mesurée en 1/600ème de seconde.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affiche la propriété timescale d'une vidéo numérique QuickTime dans l'image-objet(3).

```
-- Lingo
put(sprite(3).timeScale)

// Javascript
trace(sprite(3).timeScale)
```

Voir aussi

`digitalVideoTimeScale`

title (DVD)

Syntaxe

```
-- Lingo syntax
dvdObjRef.title

// JavaScript syntax
dvdObjRef.title;
```

Description

Propriété de DVD ; spécifie le titre en cours. Lecture/écriture.

Cette propriété renvoie un nombre entier spécifiant le numéro du titre en cours.

Exemple

L'instruction suivante renvoie le titre en cours :

```
-- Lingo syntax
trace (member(1).title)-- 1

// JavaScript syntax
trace (member(1).title);// 1
```

Voir aussi

DVD

title (fenêtre)

Syntaxe

```
-- Lingo syntax
windowObjRef.title

// JavaScript syntax
windowObjRef.title;
```

Description

Propriété de fenêtre ; affecte un titre à une fenêtre. Lecture/écriture.

Exemple

L'instruction suivante affecte le titre Planètes à la cinquième fenêtre :

```
-- Lingo syntax
_player.windowList[5].title = "Planets"

// JavaScript syntax
_player.windowList[5].title = "Planets";
```

Voir aussi

Fenêtre

titlebarOptions

Syntaxe

```
-- Lingo syntax
windowObjRef.titlebarOptions

// JavaScript syntax
windowObjRef.titlebarOptions;
```

Description

Propriété de fenêtre ; spécifie une liste de propriétés stockant les options de barre de titre d'une fenêtre. Lecture/écriture.

Cette liste de propriétés contient les propriétés ci-après :

Propriété	Description
#icon	Spécifie l'icône d'acteur à utiliser dans la barre de titre. Cette propriété n'est disponible que si la barre de titre est visible (propriété `#visible` définie sur `TRUE`).
#visible	Spécifie si la barre de titre est visible. Si cette propriété présente la valeur `FALSE`, la barre de titre est invisible et aucune autre propriété de barre de titre et de fenêtre n'est affectée. Si cette propriété présente la valeur `TRUE`, la barre de titre est visible et la fenêtre conserve l'état de toutes les autres propriétés de barre de titre et de fenêtre. La valeur par défaut est `TRUE`.
#closebox	Spécifie si une case de fermeture apparaît dans le coin supérieur droit de la fenêtre. Si cette propriété présente la valeur `TRUE`, une case de fermeture apparaît. Si elle présente la valeur `FALSE`, aucune case de fermeture n'apparaît. La valeur par défaut est `TRUE`.
#minimizebox	Spécifie si une case de réduction apparaît dans le coin supérieur droit de la fenêtre. Si cette propriété présente la valeur `TRUE`, une case de réduction apparaît. Si elle présente la valeur `FALSE`, aucune case de réduction n'apparaît. La valeur par défaut est `TRUE`.
#maximizebox	Spécifie si une case d'agrandissement apparaît dans le coin supérieur droit de la fenêtre. Si cette propriété présente la valeur `TRUE`, une case d'agrandissement apparaît. Si elle présente la valeur `FALSE`, aucune case d'agrandissement n'apparaît. La valeur par défaut est `TRUE`.
#sideTitlebar	(Mac uniquement) Spécifie si la barre de titre doit apparaître sur le côté de la fenêtre. Si cette propriété présente la valeur `TRUE`, la barre de titre apparaît sur le côté de la fenêtre. Si elle présente la valeur `FALSE`, la barre de titre n'apparaît pas sur le côté de la fenêtre. La valeur par défaut est `FALSE`.

Ces propriétés sont également accessibles par l'intermédiaire de la propriété `displayTemplate` de l'objet animation.

Exemple

L'instruction suivante affiche dans la fenêtre Messages les options de barre de titre disponibles pour la fenêtre Eléments :

```
-- Lingo syntax
trace(window("Elements").titlebarOptions)

// JavaScript syntax
trace(window("Elements").titlebarOptions);
```

Les instructions suivantes définissent la propriété `icon` sur l'acteur bitmap `smallIcon` :

```
-- Lingo syntax
window("Elements").titlebarOptions.icon = member("smallIcon")

// JavaScript syntax
window("Elements").titlebarOptions.icon = member("smallIcon");
```

Voir aussi

`appearanceOptions`, `displayTemplate`, `Fenêtre`

titleCount

Syntaxe

```
-- Lingo syntax
dvdObjRef.titleCount

// JavaScript syntax
dvdObjRef.titleCount;
```

Description

Propriété de DVD ; renvoie le nombre de titres disponibles. Lecture seule.

Le nombre de titres disponibles peut être compris entre 1 et 99.

Exemple

L'instruction suivante renvoie le nombre de titres de l'objet DVD.

```
-- Lingo
put sprite(1).titleCount

// Javascript
trace(sprite(1).titleCount)
```

Voir aussi

DVD

toolXtraList

Syntaxe

```
-- Lingo syntax
_player.toolXtraList

// JavaScript syntax
_player.toolXtraList;
```

Description

Propriété de lecteur ; renvoie une liste linéaire de tous les Xtras de type outil disponibles dans le lecteur Director. Lecture seule.

Exemple

L'instruction suivante affiche dans la fenêtre Messages tous les Xtras de type outil disponibles.

```
-- Lingo syntax
put(_player.toolXtraList)

// JavaScript syntax
put(_player.toolXtraList);
```

Voir aussi

mediaXtraList, Lecteur, scriptingXtraList, transitionXtraList, xtraList (lecteur)

toon (modificateur)

Syntaxe

```
member(whichCastmember).model(whichModel).toon.toonModifierProperty
```

Description

Modificateur 3D ; une fois le modificateur `#toon` ajouté à un modèle, vous pouvez en obtenir et en définir les propriétés.

Le modificateur toon dessine un modèle en n'utilisant qu'un nombre réduit de couleurs, ce qui permet d'obtenir un style de dessin animé pour la surface du modèle. Lorsque le modificateur `#toon` est appliqué, les propriétés `texture`, `reflectionMap`, `diffuseLightMap`, `specularLightMap` et `glossMap` du matériau du modèle sont ignorées.

Lorsque le modificateur `#toon` est utilisé en conjonction avec le modificateur `#inker`, l'effet de rendu est cumulatif et varie en fonction du premier modificateur appliqué. La liste des modificateurs renvoyée par la propriété `modifier` indique #inker ou #toon (selon le premier des deux qui a été ajouté), mais non les deux. Le modificateur toon n'est pas utilisable en conjonction avec le modificateur `#sds`.

Le modificateur `#toon` comporte les propriétés suivantes :

Remarque : *pour plus d'informations, consultez les entrées des différentes propriétés.*

- `style` permet d'obtenir ou de définir le style appliqué aux transitions des couleurs. Les valeurs possibles sont les suivantes :

  `#toon` donne des transitions aiguës entre les couleurs disponibles.

  `#gradient` donne des transitions fluides entre les couleurs disponibles.

  `#blackAndWhite` donne des transitions aiguës entre le noir et le blanc.

- `colorSteps` permet d'obtenir ou de définir le nombre de couleurs utilisées pour le calcul de l'éclairage. Cette valeur est arrondie à la puissance de 2 inférieure la plus proche. Les valeurs permises sont 2, 4, 8 et 16 ; la valeur par défaut est 2.

- `shadowPercentage` permet d'obtenir ou de définir le pourcentage des couleurs (`colorSteps`) définies pour l'éclairage utilisé pour le rendu de la portion ombragée de la surface du modèle. Les valeurs possibles s'étalent de 0 à 100. La valeur par défaut est 50.

- `shadowStrength` permet d'obtenir ou de définir le niveau d'obscurité appliqué à la portion ombragée de la surface du modèle. Les valeurs possibles sont n'importe quel nombre à virgule flottante non négatif. La valeur par défaut est 1,0.

- `highlightPercentage` permet d'obtenir ou de définir le pourcentage des couleurs (`colorSteps`) définies pour l'éclairage utilisé pour le rendu de la portion éclairée de la surface du modèle. Les valeurs possibles s'étalent de 0 à 100. La valeur par défaut est 50.

- `highlightStrength` permet d'obtenir ou de définir le niveau de luminosité appliqué à la portion éclairée de la surface du modèle. Les valeurs possibles sont n'importe quel nombre à virgule flottante non négatif. La valeur par défaut est 1,0.

- `lineColor` permet d'obtenir ou de définir la couleur des lignes dessinées par le modificateur. Les valeurs possibles sont n'importe quel objet de couleur Lingo. La valeur par défaut est `rgb (0, 0, 0)`, qui correspond au noir.

- `creases` permet de savoir ou de définir si les lignes sont dessinées avec des plis. Il s'agit d'une valeur booléenne ; la valeur par défaut est `TRUE`.

• Si `creases` présente la valeur `TRUE`, `creaseAngle` permet de savoir ou de définir la façon dont la fonction de dessin des lignes du modificateur toon répond à la présence des plis.

• `boundary` permet de savoir ou de définir si les lignes sont dessinées autour de la limite de la surface. Il s'agit d'une valeur booléenne ; la valeur par défaut est `TRUE`.

• `lineOffset` permet de savoir ou de définir si les lignes sont dessinées en fonction de la surface et de la caméra. Les valeurs négatives déplacent les lignes vers la caméra. Les valeurs positives éloignent les lignes de la caméra. Les valeurs possibles sont des nombres à virgule flottante compris entre -100,0 et 100,0. La valeur par défaut est -2,0.

• `useLineOffset` permet de savoir ou de définir si `lineOffset` est activé ou désactivé. Il s'agit d'une valeur booléenne ; la valeur par défaut est `FALSE`.

• `silhouettes` permet de savoir ou de définir si les lignes sont dessinées de façon à définir les bords le long de la bordure d'un modèle, en soulignant la forme. Il s'agit d'une valeur booléenne ; la valeur par défaut est `TRUE`.

Voir aussi

`addModifier`, `modifiers`, `sds (modificateur)`, `inker (modificateur)`

top

Syntaxe

```
-- Lingo syntax
spriteObjRef.top

// JavaScript syntax
spriteObjRef.top;
```

Description

Propriété d'image-objet ; renvoie ou définit la coordonnée verticale du bord supérieur du rectangle de délimitation d'une image-objet en nombre de pixels à partir du coin supérieur gauche de la scène. Lecture/écriture.

Exemple

L'instruction suivante vérifie si le haut de l'image-objet 3 dépasse le haut de la scène et, le cas échéant, appelle le gestionnaire `offTopEdge` :

```
-- Lingo syntax
if (sprite(3).top < 0) then
    offTopEdge()
end if

// JavaScript syntax
if (sprite(3).top < 0) {
    offTopEdge();
}
```

Voir aussi

`bottom`, `height`, `left`, `locH`, `locV`, `right`, `Image-objet`, `width`

topSpacing

Syntaxe

```
chunkExpression.topSpacing
```

Description

Propriété d'acteur texte ; permet de spécifier l'espacement supplémentaire appliqué en haut de chaque paragraphe dans la partie *expressionSousChaîne* de l'acteur texte.

La valeur est un entier, qui indique un espacement moindre entre les paragraphes s'il est inférieur à 0 et un espacement plus important s'il est supérieur à 0.

La valeur par défaut est 0 ; elle correspond à l'espacement par défaut entre les paragraphes.

Exemple

L'instruction suivante définit la propriété `topSpacing` du second paragraphe de l'acteur texte monTexte sur 20 :

```
-- Lingo
member(1).paragraph[2].topSpacing = 20

// Javascript
member(1).getPropRef("paragraph",2).topSpacing = 20;
```

Voir aussi

bottomSpacing

traceLoad

Syntaxe

```
-- Lingo syntax
_movie.traceLoad

// JavaScript syntax
_movie.traceLoad;
```

Description

Propriété d'animation ; indique la quantité d'informations sur les acteurs affichées pendant le chargement de ces derniers. Lecture/écriture.

Les valeurs possibles de la propriété traceLoad sont les suivantes :

- 0 : n'affiche aucune information (valeur par défaut).
- 1 : affiche le nom des acteurs.
- 2 : affiche le nom des acteurs, le numéro de l'image en cours, le nom de l'animation et le décalage de recherche du fichier (la quantité relative de déplacement que le lecteur doit effectuer pour charger les médias).

Exemple

L'instruction suivante demande à l'animation d'afficher le nom des acteurs lorsque ces derniers sont chargés :

```
-- Lingo syntax
_movie.traceLoad = 1
```

```
// JavaScript syntax
_movie.traceLoad = 1;
```

Voir aussi

Animation

traceLogFile

Syntaxe

```
-- Lingo syntax
_movie.traceLogFile

// JavaScript syntax
_movie.traceLogFile;
```

Description

Propriété d'animation ; spécifie le nom du fichier dans lequel le contenu de la fenêtre Messages est enregistré. Lecture/écriture.

Vous pouvez fermer ce fichier en définissant la propriété `traceLogFile` sur la valeur `EMPTY` (Lingo) ou sur une chaîne vide " " (syntaxe JavaScript). Tout élément généré devant apparaître dans la fenêtre Messages est écrit dans ce fichier. Cette propriété est utile pour le processus de débogage lors de l'exécution d'une animation dans une projection et dans le cadre de la programmation.

Exemple

L'instruction suivante demande à Lingo d'écrire le contenu de la fenêtre Messages dans le fichier Messages.txt situé dans le dossier de l'animation en cours :

```
-- Lingo syntax
_movie.traceLogFile = _movie.path & "Messages.txt"

// JavaScript syntax
_movie.traceLogFile = _movie.path + "Messages.txt";
```

L'instruction suivante ferme le fichier dans lequel est écrit le contenu de la fenêtre Messages :

```
-- Lingo syntax
_movie.traceLogFile = ""

// JavaScript syntax
_movie.traceLogFile = "";
```

Voir aussi

Animation

traceScript

Syntaxe

```
-- Lingo syntax
_movie.traceScript
```

```
// JavaScript syntax
_movie.traceScript;
```

Description

Propriété d'animation ; indique si la fonction de suivi de l'animation est activée (TRUE) ou non (FALSE). Lecture/écriture.

Lorsque la propriété traceScript est activée, la fenêtre Messages affiche chaque ligne de script exécutée.

Exemple

L'instruction suivante active la propriété traceScript.

```
-- Lingo syntax
_movie.traceScript = TRUE

// JavaScript syntax
_movie.traceScript = true;
```

Voir aussi

Animation

trackCount (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.trackCount()

// JavaScript syntax
memberObjRef.trackCount();
```

Description

Propriété d'acteur vidéo numérique ; renvoie le nombre de pistes dans l'acteur vidéo numérique spécifié.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine le nombre de pistes de l'acteur vidéo numérique Chronique jazz et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(member("Jazz Chronicle").trackCount())

// JavaScript syntax
trace(member("Jazz Chronicle").trackCount());
```

trackCount (image-objet)

Syntaxe

```
-- Lingo syntax
spriteObjRef.trackCount()
```

```
// JavaScript syntax
spriteObjRef.trackCount();
```

Description

Propriété d'image-objet vidéo numérique ; renvoie le nombre de pistes dans l'image-objet vidéo numérique spécifiée.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine le nombre de pistes de l'image-objet vidéo numérique affectée à la piste 10 et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(sprite(10).trackCount())

// JavaScript syntax
trace(sprite(10).trackCount());
```

trackEnabled

Syntaxe

```
-- Lingo syntax
spriteObjRef.trackEnabled(whichTrack)

// JavaScript syntax
spriteObjRef.trackEnabled(whichTrack);
```

Description

Propriété d'image-objet vidéo numérique ; indique l'état de la piste spécifiée d'une vidéo numérique. Cette propriété présente la valeur `TRUE` si la piste est activée et en cours de lecture. Cette propriété présente la valeur `FALSE` si la piste est désactivée et n'est plus en cours de lecture ou de mise à jour.

Cette propriété ne peut pas être définie. Vous devez utiliser la propriété `setTrackEnabled` à la place.

Exemple

L'instruction suivante vérifie si la piste 2 de l'image-objet vidéo numérique 1 est activée :

```
-- Lingo syntax
put(sprite(1).trackEnabled(2))

// JavaScript syntax
put(sprite(1).trackEnabled(2));
```

Voir aussi

`setTrackEnabled()`

trackNextKeyTime

Syntaxe

```
-- Lingo syntax
spriteObjRef.trackNextKeyTime(whichTrack)
```

```
// JavaScript syntax
spriteObjRef.trackNextKeyTime(whichTrack);
```

Description

Propriété d'image-objet vidéo numérique ; indique la position temporelle de l'image-clé suivant la position temporelle actuelle de la piste vidéo numérique spécifiée.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la position temporelle de l'image-clé suivant la position temporelle de la piste 5 de la vidéo numérique affectée à la piste d'image-objet 15 et l'affiche dans la fenêtre Messages :

```
-- Lingo syntax
put(sprite(15).trackNextKeyTime(5))

// JavaScript syntax
put(sprite(15).trackNextKeyTime(5));
```

trackNextSampleTime

Syntaxe

```
-- Lingo syntax
spriteObjRef.trackNextSampleTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackNextSampleTime(whichTrack);
```

Description

Propriété d'image-objet vidéo numérique ; indique la position temporelle de l'échantillon qui suit la position temporelle en cours de la vidéo numérique. Cette propriété est pratique pour localiser les pistes texte dans une vidéo numérique.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la position temporelle de l'échantillon qui suit la position temporelle en cours dans la piste 5 de la vidéo numérique affectée à l'image-objet 15 :

```
-- Lingo syntax
put(sprite(15).trackNextSampleTime(5))

// JavaScript syntax
put(sprite(15).trackNextSampleTime(5));
```

trackPreviousKeyTime

Syntaxe

```
-- Lingo syntax
spriteObjRef.trackPreviousKeyTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackPreviousKeyTime(whichTrack);
```

Description

Propriété d'image-objet vidéo numérique ; renvoie la position temporelle de l'image-clé qui précède la position temporelle actuelle.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la position temporelle de l'image-clé de la piste 5 qui précède la position temporelle actuelle de l'image-objet vidéo numérique affectée à la piste 15 et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(sprite(15).trackPreviousKeyTime(5))

// JavaScript syntax
put(sprite(15).trackPreviousKeyTime(5));
```

trackPreviousSampleTime

Syntaxe

```
-- Lingo syntax
spriteObjRef.trackPreviousSampleTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackPreviousSampleTime(whichTrack);
```

Description

Propriété d'image-objet vidéo numérique ; indique la position temporelle de l'échantillon qui précède la position temporelle en cours de la vidéo numérique. Cette propriété est pratique pour localiser les pistes texte dans une vidéo numérique.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la position temporelle de l'échantillon de la piste 5 qui précède la position temporelle actuelle de l'image-objet vidéo numérique affectée à la piste 15 et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(sprite(15).trackPreviousSampleTime(5))

// JavaScript syntax
put(sprite(15).trackPreviousSampleTime(5));
```

trackStartTime (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.trackStartTime(whichTrack)

// JavaScript syntax
memberObjRef.trackStartTime(whichTrack);
```

Description

Propriété d'acteur vidéo numérique ; renvoie la position temporelle du début de la piste de l'acteur vidéo numérique spécifié.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la position temporelle de début de la lecture de la piste 5 de l'acteur vidéo numérique Chronique Jazz et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(member("Jazz Chronicle").trackStartTime(5))

// JavaScript syntax
put(member("Jazz Chronicle").trackStartTime(5));
```

trackStartTime (image-objet)

Syntaxe

```
-- Lingo syntax
spriteObjRef.trackStartTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackStartTime(whichTrack);
```

Description

Propriété d'image-objet vidéo numérique ; définit la position temporelle de début d'une animation vidéo numérique dans la piste d'image-objet spécifiée. La valeur de `trackStartTime` est mesurée en battements.

Cette propriété peut être testée, mais pas définie.

Exemple

Dans la fenêtre Messages, l'instruction suivante signale le début de la lecture de la piste 5 de la piste d'image-objet 10. La position temporelle de début est à 120 battements (2 secondes).

```
-- Lingo syntax
put(sprite(10).trackStartTime(5))

// JavaScript syntax
put(sprite(10).trackStartTime(5))
```

Voir aussi

duration (acteur), playRate, currentTime (QuickTime, AVI)

trackStopTime (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.trackStopTime(whichTrack)

// JavaScript syntax
memberObjRef.trackStopTime(whichTrack);
```

Description

Propriété d'acteur vidéo numérique ; renvoie la position temporelle d'arrêt de la piste de l'acteur vidéo numérique spécifié. Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la position temporelle d'arrêt de la piste 5 de l'acteur vidéo numérique Chronique Jazz et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(member("Jazz Chronicle").trackStopTime(5))

// JavaScript syntax
put(member("Jazz Chronicle").trackStopTime(5));
```

trackStopTime (image-objet)

Syntaxe

```
-- Lingo syntax
spriteObjRef.trackStopTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackStopTime(whichTrack);
```

Description

Propriété d'image-objet vidéo numérique ; renvoie la position temporelle d'arrêt de la piste de l'image-objet vidéo numérique spécifiée.

Lors de la lecture d'une animation vidéo numérique, la propriété `trackStopTime` correspond à l'endroit où la lecture s'arrête ou effectue une boucle si la propriété `loop` est activée.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la position temporelle d'arrêt de la piste 5 dans la vidéo numérique affectée à l'image-objet 6 et affiche le résultat dans la fenêtre Messages :

```
-- Lingo syntax
put(sprite(6).trackStopTime(5))

// JavaScript syntax
put(sprite(6).trackStopTime(5));
```

Voir aussi

playRate, currentTime (QuickTime, AVI), trackStartTime (acteur)

trackText

Syntaxe

```
-- Lingo syntax
spriteObjRef.trackText(whichTrack)
```

```
// JavaScript syntax
spriteObjRef.trackText(whichTrack);
```

Description

Propriété d'image-objet vidéo numérique ; renvoie le texte situé dans la piste spécifiée de la vidéo numérique à la position temporelle actuelle. Le résultat est une chaîne de caractères, qui peut avoir une longueur de 32 Ko. Cette propriété ne s'applique qu'aux pistes de texte.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affecte le texte de la piste 5 de la vidéo numérique affectée à la position temporelle actuelle à l'image-objet 20 à l'acteur champ Archives :

```
-- Lingo syntax
member("Archives").text = string(sprite(20).trackText(5))

// JavaScript syntax
member("Archives").text = sprite(20).trackText(5).toString();
```

trackType (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.trackType(whichTrack)

// JavaScript syntax
memberObjRef.trackType(whichTrack);
```

Description

Propriété d'acteur vidéo numérique ; indique le type de média qui se trouve dans la piste spécifiée de l'acteur indiqué. Les valeurs possibles sont `#video`, `#sound`, `#text` et `#music`.

Cette propriété peut être testée, mais pas définie.

Exemple

Le gestionnaire suivant vérifie si la piste 5 de l'acteur vidéo numérique Nouvelles du jour est une piste texte et, le cas échéant, exécute le gestionnaire `textFormat` :

```
-- Lingo syntax
on checkForText
    if member("Today's News").trackType(5) = #text then
        textFormat
    end if
end

// JavaScript syntax
function checkForText() {
    var tt = member("Today's News").trackType(5);
    if (tt == "text") {
        textFormat();
    }
}
```

trackType (image-objet)

Syntaxe

```
-- Lingo syntax
spriteObjRef.trackType(whichTrack)

// JavaScript syntax
spriteObjRef.trackType(whichTrack);
```

Description

Propriété d'image-objet vidéo numérique ; renvoie le type de média qui se trouve dans la piste spécifiée de l'image-objet indiquée. Les valeurs possibles sont `#video`, `#sound`, `#text` et `#music`.

Cette propriété peut être testée, mais pas définie.

Exemple

Le gestionnaire suivant vérifie si la piste 5 de l'image-objet vidéo numérique affectée à la piste 10 est une piste texte et, le cas échéant, exécute le gestionnaire `textFormat` :

```
-- Lingo syntax
on checkForText
    if sprite(10).trackType(5) = #text then
        textFormat
    end if
end

// JavaScript syntax
function checkForText() {
    var tt = sprite(10).trackType(5);
    if (tt == "text") {
        textFormat();
    }
}
```

trails

Syntaxe

```
sprite(whichSprite).trails
the trails of sprite whichSprite
```

Description

Propriété d'image-objet ; active (1 ou `TRUE`) ou désactive (0 ou `FALSE`) l'effet de traces d'encre pour l'image-objet spécifiée par *`quelleImageObjet`*. Pour que la valeur définie par Lingo dure au-delà de l'image-objet en cours, l'image-objet doit être contrôlée par un script.

Pour supprimer les traces, animez une autre image-objet sur ces pixels ou utilisez une transition.

Exemple

L'instruction suivante active les traces pour l'image-objet 7 :

```
-- Lingo
sprite(7).trails = 1
```

```
// Javascript
sprite(7).trails = 1;
```

Voir aussi

directToStage

transform (propriété)

Syntaxe

```
member(whichCastmember).node(whichNode).transform
member(whichCastmember).node(whichNode).transform.transformProperty
member(whichCastmember).model(whichModel).bonesPlayer.bone[boneID].transform
member(whichCastmember).model(whichModel).bonesPlayer.bone[boneID].transform.transformProp
erty
```

Description

Propriété et commande 3D ; permet d'obtenir ou de définir la transformation associée à un nœud ou segment spécifique au sein d'un modèle utilisant le modificateur `bonesPlayer`. En tant que commande, `transform` donne accès aux différentes commandes et propriétés de l'objet de transformation. Un nœud peut être un objet de caméra, groupe, lumière ou modèle.

Pour les objets de nœud, cette propriété est, par défaut, la transformation d'identité. Une transformation de nœud définit la position, la rotation, et l'échelle du nœud en fonction de son objet parent. Lorsque le parent d'un nœud est l'objet groupe Univers, la propriété `transform` du nœud a la même valeur que celle renvoyée par la commande `getWorldTransform()`.

Pour les segments des modèles utilisant le modificateur `bonesPlayer`, cette propriété prend comme valeur par défaut la valeur de la transformation affectée au segment à la création du fichier du modèle. La transformation d'un segment représente la rotation du segment en fonction de son segment parent et sa position en fonction de la position d'origine de l'articulation. La position d'articulation d'origine est déterminée au moment de la création du fichier du modèle.

Vous pouvez utiliser les commandes et propriétés de transformation suivantes avec la propriété `transform` des objets nœuds :

Remarque : *cette section ne contenant qu'un récapitulatif, vous trouverez de plus amples informations en consultant les entrées correspondantes.*

- `preScale` applique une mise à l'échelle avant les décalages de position, de rotation et d'échelle de la transformation.
- `preTranslate` applique une translation avant les décalages de position, de rotation et d'échelle de la transformation.
- `preRotate` applique une rotation avant les décalages de position, de rotation et d'échelle de la transformation.
- `scale` (commande) applique une mise à l'échelle après les décalages de position, de rotation et d'échelle de la transformation.
- `scale` (transformation) permet d'obtenir ou de définir le degré de redimensionnement de la transformation.
- `translate` applique une translation après les décalages de position, de rotation et d'échelle de la transformation.
- `rotate` applique une rotation après les décalages de position, de rotation et d'échelle de la transformation.
- `position` (transformation) permet d'obtenir ou de définir le décalage de position de la transformation.

- `rotation` (transformation) permet d'obtenir ou de définir le décalage de rotation de la transformation.

Pour modifier la propriété `transform` d'un segment au sein d'un modèle, vous devrez enregistrer une copie de la transformation d'origine du segment, modifier la copie enregistrée à l'aide des commandes et propriétés indiquées ci-dessus, puis réinitialiser la propriété `transform` du segment de façon à la rendre égale à la transformation modifiée. Par exemple :

```
t = member("character").model("biped").bonesPlayer.bone[38].transform.duplicate()
t.translate(25,0,-3)
member("character").model("biped").bonesPlayer.bone[38].transform = t
```

Paramètres

Aucune.

Exemple

L'instruction suivante indique la transformation du modèle Boîte, suivie des propriétés de position et de rotation de la transformation.

```
put member("3d world").model("box").transform
-- transform(1.000000,0.000000,0.000000,0.000000, 0.000000,1.000000,0.000000,0.000000,
0.000000,0.000000,1.000000,0.000000, -94.144844,119.012825,0.000000,1.000000)
put member("3d world").model("box").transform.position
-- vector(-94.1448, 119.0128, 0.0000)
put member("3d world").model("box").transform.rotation
--vector(0.0000, 0.0000, 0.0000)
```

Voir aussi

interpolateTo(), scale (transformation), rotation (transformation), position (transformation), bone, worldTransform, preRotate, preScale(), preTranslate()

transitionType

Syntaxe

```
member(whichCastMember).transitionType
the transitionType of member whichCastMember
```

Description

Propriété d'acteur transition ; détermine le type d'une transition, spécifié sous la forme d'un nombre. Les valeurs possibles sont identiques aux codes affectés aux transitions pour la commande `puppetTransition`.

Exemple

L'instruction suivante affecte le type d'acteur de transition 51 à l'acteur 3, qui est un acteur de type fondu pixels :

```
member(3).transitionType = 51
```

transitionXtraList

Syntaxe

```
-- Lingo syntax
_player.transitionXtraList
```

```
// JavaScript syntax
_player.transitionXtraList;
```

Description

Propriété de lecteur ; renvoie une liste linéaire de tous les Xtras de transition disponibles dans le lecteur Director. Lecture seule.

Exemple

L'instruction suivante affiche dans la fenêtre Messages tous les Xtras de transition disponibles.

```
-- Lingo syntax
put(_player.transitionXtraList)

// JavaScript syntax
put(_player.transitionXtraList);
```

Voir aussi

mediaXtraList, Lecteur, scriptingXtraList, toolXtraList, xtraList (lecteur)

translation

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.translation

// JavaScript syntax
memberOrSpriteObjRef.translation;
```

Description

Propriété d'acteur et d'image-objet QuickTime ; contrôle le décalage d'une image d'image-objet QuickTime à l'intérieur du cadre de délimitation de l'image-objet.

Ce décalage est exprimé par rapport à l'emplacement par défaut de l'image-objet tel qu'il est défini par sa propriété `center`. Lorsque la propriété `center` présente la valeur `TRUE`, l'image-objet est décalée par rapport au centre du rectangle de délimitation ; lorsque `center` présente la valeur `FALSE`, l'image-objet est décalée par rapport au coin supérieur gauche du rectangle de délimitation.

Le décalage, indiqué en pixels sous forme de nombres entiers positifs ou négatifs, est défini comme une liste Director : [*transX*, *transY*]. Le paramètre *transX* indique le décalage horizontal à partir de l'emplacement par défaut de l'image-objet, tandis que le paramètre *transY* indique le décalage vertical. La valeur par défaut est [0, 0].

Lorsque la propriété `crop` de l'image-objet est définie sur `TRUE`, la propriété `translation` peut être utilisée pour masquer des parties de l'animation QuickTime en les déplaçant à l'extérieur du rectangle de délimitation. Lorsque la propriété `crop` est définie sur `FALSE`, la propriété `translation` n'est pas prise en compte et l'image-objet est toujours placée dans le coin supérieur gauche du rectangle de délimitation de l'image-objet.

Cette propriété peut être testée et définie.

Exemple

Le script d'image suivant suppose que la propriété `center` de l'acteur d'une image-objet QuickTime d'une largeur de 320 pixels placée dans la piste 5 est définie sur `FALSE` et que sa propriété `crop` présente la valeur `TRUE`. Elle garde la tête de lecture dans l'image courante jusqu'à ce que le point de translation horizontal de l'animation se soit déplacé vers le bord droit de l'image-objet, par incréments de 10 pixels. Cela crée un effet de balayage vers la droite, qui met l'image-objet hors de vue en la déplaçant vers la droite. Lorsque l'image-objet ne figure plus à l'écran, la tête de lecture passe à l'image suivante.

```
-- Lingo syntax
on exitFrame
    horizontalPosition = sprite(5).translation[1]
    if horizontalPosition < 320 then
        sprite(5).translation = sprite(5).translation + [10, 0]
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var horizontalPosition = sprite(5).translation[1];
    if (horizontalPosition < 320 ) {
        sprite(5).translation = sprite(5).translation + list(10, 0);
        _movie.go(_movie.frame);
    }
}
```

transparent

Syntaxe

```
member(whichCastmember).shader(whichShader).transparent
member(whichCastmember).model(whichModel).shader.transparent
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].transparent
```

Description

Propriété 3D de matériau standard ; permet de savoir ou de définir si l'opacité d'un modèle est réalisée à l'aide de valeurs alpha (`TRUE`) ou s'il s'agit d'un rendu opaque (`FALSE`). La valeur par défaut de cette propriété est `TRUE` (avec valeurs alpha).

La fonctionnalité `shader.blend` dépend de cette propriété.

Tous les matériaux ont accès aux propriétés de matériau `#standard` ; outre ces propriétés standard, les matériaux de type `#engraver`, `#newsprint` et `#painter` possèdent des propriétés propres à leur type. Pour plus d'informations, reportez-vous à l'entrée `newShader`.

Exemple

L'instruction suivante entraîne un rendu opaque du modèle Pluton. Le paramétrage de la propriété `blend` du matériau de ce modèle n'a aucun effet.

```
-- Lingo
member("scene").model("Pluto").shader.transparent = FALSE

// Javascript
member("scene").getPropRef("model",1).getProp("shader").transparent = false;
```

Voir aussi

blendFactor, blend (3D)

triggerCallback

Syntaxe

```
-- Lingo syntax
spriteObjRef.triggerCallback

// JavaScript syntax
spriteObjRef.triggerCallback;
```

Description

Propriété d'image-objet QuickTime VR ; contient le nom du gestionnaire exécuté lorsque l'utilisateur clique sur une zone référencée dans une animation QuickTime VR. Le gestionnaire reçoit deux arguments : le paramètre `me` et l'identifiant de la zone référencée sur laquelle l'utilisateur a cliqué.

La valeur que le gestionnaire renvoie détermine la façon dont la zone référencée est gérée par l'animation. Si le gestionnaire renvoie la valeur `#continue`, l'image-objet QuickTime VR continue à traiter normalement la zone référencée. S'il renvoie la valeur `#cancel`, le comportement par défaut de la zone référencée est annulé.

Cette propriété doit être réglée sur 0 pour effacer l'instruction d'appel.

L'image-objet QuickTime VR reçoit le message en premier.

Pour obtenir des performances optimales, ne définissez la propriété `triggerCallback` qu'en cas d'absolue nécessité.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affecte le gestionnaire `MyHotSpotCallback` au gestionnaire d'appel d'une image-objet QuickTime VR dès que la tête de lecture entre dans l'étendue de l'image-objet. Le gestionnaire `MyHotSpotCallback` est exécuté chaque fois que cette zone référencée est déclenchée. L'appel est annulé lorsque la tête de lecture quitte l'étendue de l'image-objet.

```
-- Lingo syntax
property pMySpriteNum, spriteNum

on beginSprite(me)
    pMySpriteNum = spriteNum
    sprite(pMySpriteNum).triggerCallback = #MyHotSpotCallback
end

on MyHotSpotCallback(me, hotSpotID)
    put "Hotspot" && hotSpotID && "was just triggered"
end

on endSprite me
    sprite(pMySpriteNum).triggerCallback = 0
end

// JavaScript syntax
function beginSprite() {
    pMySpriteNum = this.spriteNum;
    sprite(this.pMySpriteNum).triggerCallback = symbol("MyHotSpotCallback");
```

```
}

function MyHotSpotCallback(hotSpotID) {
    trace("Hotspot " + hotSpotID + " was just triggered");
}

function endSprite() {
    sprite(pMySpriteNum).triggerCallback = 0;
}
```

trimWhiteSpace

Syntaxe

```
-- Lingo syntax
memberObjRef.trimWhiteSpace

// JavaScript syntax
memberObjRef.trimWhiteSpace;
```

Description

Propriété d'acteur ; détermine si les pixels blancs situés autour du bord d'un acteur bitmap sont supprimés ou laissés en place. Cette propriété est définie à l'importation de l'acteur. Elle peut être modifiée dans Lingo ou dans le volet Bitmap de l'Inspecteur des propriétés.

Exemple

L'instruction suivante entraîne la suppression des espaces blancs dans l'acteur texte.

```
-- Lingo
member(1).trimWhiteSpace;

// Javascript
member(1).trimWhiteSpace;
```

tunnelDepth

Syntaxe

```
member(whichTextmember).tunnelDepth
member(whichCastMember).modelResource(whichExtruderModelResource).tunnelDepth
```

Description

Propriété 3D de ressource de modèle d'extrudeur et d'acteur texte. Cette propriété permet d'obtenir ou de définir la profondeur d'extrusion (la distance séparant les faces avant et arrière) d'une ressource de modèle 3D. Les valeurs possibles sont des nombres à virgule flottante compris entre 1,0 et 100,0. La valeur par défaut est 50,0.

Pour plus d'informations sur l'utilisation des ressources de modèle d'extrudeur et des acteurs texte, reportez-vous à l'entrée `extrudeToMember`.

Exemple

Dans l'exemple suivant, l'acteur Logo est un acteur texte. L'instruction suivante définit la profondeur du tunnel du logo à 5 ; la profondeur de ses lettres est très réduite lorsqu'elles sont affichées en mode 3D.

```
-- Lingo
member("logo").tunnelDepth = 5

// Javascript
member("logo").tunnelDepth = 5;
```

Dans l'exemple suivant, la ressource du modèle Slogan correspond à du texte extrudé. L'instruction suivante donne à la propriété `tunnelDepth` de la ressource de modèle Slogan la valeur 1 000 ; ses lettres seront très profondes.

```
-- Lingo
member("scene").model("Slogan").resource.tunnelDepth = 1000

// Javascript
member("scene").getPropRef("model",1).getProp("resource").tunnelDepth = 1000;
```

Voir aussi

`extrude3D`

tweened

Syntaxe

```
sprite(whichSprite).tweened
the tweened of sprite whichSprite
```

Description

Propriété d'image-objet ; détermine si seule la première image d'une nouvelle image-objet est une image-clé (`TRUE`) ou si toutes les images de la nouvelle image-objet sont des images-clés (`FALSE`).

Cette propriété n'affecte pas la lecture et n'est utile que lors de l'enregistrement du scénario.

Cette propriété peut être testée et définie.

Exemple

Lorsque l'instruction suivante est émise, les nouvelles images-objets créées dans la piste 25 n'ont une image-clé que dans la première image de l'étendue de l'image-objet :

```
-- Lingo
sprite(25).tweened = 1

// Javascript
sprite(25).tweened = 1;
```

tweenMode

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).tweenMode
modelResourceObjectReference.tweenMode
```

Description

Propriété 3D de particule ; permet de savoir ou de définir si la couleur d'une particule varie en fonction de sa vitesse ou de son âge. La propriété `tweenMode` peut prendre les valeurs suivantes :

- `#velocity` modifie la couleur de la particule entre `colorRange.start` et `colorRange.end` en fonction de la vitesse de la particule.
- `#age` modifie la couleur de la particule en effectuant une interpolation linéaire de la couleur entre `colorRange.start` et `colorRange.end` sur la durée de vie de la particule. Il s'agit de la valeur par défaut de cette propriété.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type #particle. L'instruction suivante attribue à la propriété tweenMode de systèmeThermique la valeur #velocity, de façon à ce que les particules les plus lentes n'atteignent pas la couleur spécifiée par colorRange.end, contrairement aux particules les plus rapides.

```
-- Lingo
member(8).modelResource("thermoSystem").tweenMode = #velocitytype

// Javascript
member(8).getPropRef("modelResource",1).tweenMode =symbol("velocitytype");
```

type (lumière)

Syntaxe

```
member(whichCastmember).light(whichLight).type
```

Description

Propriété 3D de lumière ; type de la lumière référencée. Les valeurs possibles de cette propriété sont les suivantes :

- `#ambient` entraîne une lumière uniforme sur toutes les surfaces. L'intensité des lumières ambiantes n'est pas affectée par la distance les séparant de la source lumineuse.
- `#directional` entraîne une lumière qui semble dirigée dans une direction spécifique, sans pour autant être aussi précise que les lumières de type `#spot`. L'intensité des lumières directionnelles diminue avec la distance les séparant de la source lumineuse.
- `#point` entraîne une lumière éclairant dans toutes les directions à partir d'un emplacement spécifique de l'univers 3D. L'effet est semblable à celui produit par une ampoule. L'intensité des lumières `#point` diminue avec la distance les séparant de la source lumineuse.
- `#spot` entraîne une lumière partant d'un point spécifique et dans le cône défini par la lumière vers l'avant et par la propriété `spotAngle`. L'intensité de ces lumières décline avec la distance de la source lumineuse en fonction des valeurs définies dans la propriété `attenuation` de la lumière.

Exemple

L'instruction suivante affiche la propriété `type` de la lumière lumièrePrincipale :

```
-- Lingo
put member("3D").motion("MainLight").type
-- #spot

// Javascript
trace(member("3D").getPropRef("motion",1).type);
// symbol("spot")
```

Voir aussi

spotAngle, attenuation

type (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.type

// JavaScript syntax
memberObjRef.type;
```

Description

Propriété d'acteur ; indique le type d'un acteur. Lecture seule.

La propriété `type` peut prendre l'une des valeurs suivantes :

#animgif	#palette
#bitmap	#physics
#button	#picture
#cursor	#QuickTimeMedia
#digitalVideo	#realMedia
#DVD	#script
#empty	#shape
#field	#shockwave3D
#filmLoop	#sound
#flash	#swa
#flashcomponent	#text
#font	#transition
#havok	#vectorShape
#movie	#windowsMedia
#ole	

Cette liste comprend les types d'acteurs disponibles dans Director et les Xtras l'accompagnant. Vous pouvez également définir des types d'acteurs spéciaux correspondant pour des acteurs personnalisés.

Pour les animations créées dans Director 5, 6 et 6.5, la propriété type renvoie la valeur #field pour les acteurs champ et la valeur #richText pour les acteurs texte. Dans toutes les versions à partir de Director 7, la valeur renvoyée est #field pour les acteurs champ et #text pour les acteurs texte.

Exemple

Le gestionnaire suivant vérifie si l'acteur Nouvelles du jour est un acteur champ et, dans le cas contraire, affiche un message d'alerte :

```
-- Lingo syntax
on checkFormat
    if (member("Today's News").type <> #field) then
        _player.alert("Sorry, this cast member must be a field.")
    end if
end
```

```
// JavaScript syntax
function checkFormat() {
    if (member("Today's News").type != "field") {
        _player.alert("Sorry, this cast member must be a field.");
    }
}
```

Voir aussi

Acteur

type (ressource de modèle)

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).type
```

Description

Propriété 3D de la ressource de modèle ; type de la ressource de modèle référencée. Les valeurs possibles de cette propriété sont les suivantes :

- `#box` indique que cette ressource de modèle est une ressource de boîte primitive créée avec la commande `newModelResource`.
- `#cylinder` indique que cette ressource de modèle est une ressource de cylindre primitive créée avec la commande `newModelResource`.
- `#extruder` indique que cette ressource de modèle est une ressource d'extrudeur de texte primitive créée avec la commande `extrude3d`.
- `#mesh` indique que cette ressource de modèle est une ressource de générateur de maille primitive créée avec la commande `newMesh`.
- `#particle` indique que cette ressource de modèle est une ressource de système de particules primitive créée avec la commande `newModelResource`.
- `#plane` indique que cette ressource de modèle est une ressource de plan primitive créée avec la commande `newModelResource`.
- `#sphere` indique que cette ressource de modèle est une ressource de sphère primitive créée avec la commande `newModelResource`.
- `#fromFile` indique que cette ressource de modèle a été créée dans un autre programme que Director et chargée à partir d'un fichier ou acteur externe.

Exemple

L'instruction suivante affiche la propriété `type` de la ressource de modèle Hélice.

```
put member("helix models").modelResource("Helix").type
-- #fromFile
```

Voir aussi

newModelResource, newMesh, extrude3D

type (mouvement)

Syntaxe

```
member(whichCastmember).motion(whichMotion).type
```

Description

Propriété 3D de mouvement ; type du mouvement référencé. Les valeurs possibles de cette propriété sont les suivantes :

- `#bonesPlayer` indique que ce mouvement est une animation reposant sur des segments dont la lecture requiert l'utilisation du modificateur `#bonesPlayer`.
- `#keyFramePlayer` indique que ce mouvement est une animation reposant sur des images-clés dont la lecture requiert l'utilisation du modificateur `#keyFramePlayer`.
- `#none` indique que ce mouvement n'a aucun mouvement correspondant et peut être lu avec le modificateur `#bonesPlayer` ou `#keyFramePlayer`. Les mouvements par défaut des acteurs 3D sont de ce type.

Exemple

L'instruction suivante affiche la propriété `type` du mouvement Course.

```
put member("scene").motion("Run").type
-- #bonesPlayer
```

L'instruction suivante affiche la propriété `type` du mouvement mouvementParDéfaut.

```
put member("scene").motion("DefaultMotion").type
-- #none
```

Voir aussi

bonesPlayer (modificateur), keyframePlayer (modificateur)

type (matériau)

Syntaxe

```
member(whichCastmember).shader(whichShader).type
```

Description

Propriété 3D de matériau ; type de matériau du matériau référencé. Les valeurs possibles de cette propriété sont les suivantes :

- `#standard` indique qu'il s'agit d'un matériau standard.
- `#painter` indique qu'il s'agit d'un matériau peintre.
- `#newsprint` indique qu'il s'agit d'un matériau journal.
- `#engraver` indique qu'il s'agit d'un matériau gravure.

Exemple

L'instruction suivante indique que le matériau utilisé par le modèle boîte2 est un matériau peintre :

```
put member("Scene").model("box2").shader.type
-- #painter
```

Voir aussi

newShader

type (image-objet)

Syntaxe

```
sprite(whichSprite).type
the type of sprite whichSprite
```

Description

Propriété d'image-objet ; libère des pistes d'image-objet pendant l'enregistrement du scénario en attribuant à la propriété d'image-objet `type` la valeur 0 pour ces pistes.

Remarque : *l'acteur d'une image-objet doit être remplacé uniquement par un autre acteur du même type afin d'éviter toute modification des propriétés de l'image-objet lors de l'échange.*

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante libère la piste d'image-objet 1 lors d'une session d'enregistrement de scénario :

```
sprite(1).type = 0
```

type (texture)

Syntaxe

```
member(whichCastmember).shader(whichShader).type
```

Description

Propriété 3D de texture ; type de texture de la texture référencée. Les valeurs possibles de cette propriété sont les suivantes :

- `#fromCastMember` indique qu'il s'agit d'une texture créée à partir d'un acteur Director prenant en charge la propriété `image` à l'aide de la commande `newTexture`.
- `#fromImageObject` indique qu'il s'agit d'une texture créée à partir d'un objet image à l'aide de la commande `newTexture`.
- `#importedFromFile` indique qu'il s'agit d'une texture créée dans un autre programme que Director et créée au moment de l'importation d'un fichier ou du chargement d'un acteur.

Exemple

L'instruction suivante indique que la texture utilisée par le matériau du modèle Pluton a été créée à partir d'un objet image.

```
put member("scene").model("Pluto").shader.texture.type
-- #fromImageObject
```

Voir aussi

newTexture

type (fenêtre)

Syntaxe

```
-- Lingo syntax
windowObjRef.type

// JavaScript syntax
windowObjRef.type;
```

Description

Propriété de fenêtre ; spécifie le type d'une fenêtre. Lecture/écriture.

Si la propriété `type` est définie, toutes les propriétés relatives à la nouvelle fenêtre sont définies en conséquence.

Cette propriété peut prendre l'une des valeurs suivantes :

Propriété	Description
#document	Indique que la fenêtre apparaîtra avec une barre de titre standard, une case de fermeture, une case de réduction et une case d'agrandissement. Les fenêtres de ce type sont déplaçables.
#tool	Indique que la fenêtre apparaîtra avec une barre de titre plus courte et uniquement une petite case de fermeture dans le coin supérieur droit. Les fenêtres de type `#tool` ne reçoivent plus les événements d'activation ou de désactivation car elles sont toujours actives. Les fenêtres de ce type s'affichent toujours en couches et apparaissent toujours au-dessus des fenêtres de type `#document`.
#dialog	Indique que la fenêtre apparaîtra avec une barre de titre standard, une case de fermeture et aucune icône. Les fenêtres de ce type sont modales et s'affichent toujours au-dessus de toutes les autres fenêtres.

Ces propriétés sont également accessibles par l'intermédiaire de la propriété `displayTemplate` de l'objet animation.

Le comportement des fenêtres dépend également des valeurs de la propriété `type` et de la propriété `dockingEnabled` de l'objet animation.

- Si la propriété `dockingEnabled` reçoit la valeur `TRUE` et que la propriété `type` est définie sur `#document`, l'animation dans une fenêtre présente l'aspect et le comportement d'une fenêtre de type Document dans Director. La fenêtre s'affiche dans la zone appelée « espace principal » et pourra s'ancrer avec les fenêtres Scène, Scénario et Distribution, les éditeurs de médias, ainsi que les fenêtres de messages. Toutefois, il vous est impossible de regrouper la fenêtre avec l'une de ces autres fenêtres.

- Si la propriété `dockingEnabled` reçoit la valeur `TRUE` et que la propriété `type` est définie sur `#tool`, l'animation dans une fenêtre présente l'aspect et le comportement d'une fenêtre de type Outil dans Director. Vous pouvez regrouper cette fenêtre avec toutes les fenêtres de type Outil, à l'exception de l'Inspecteur des propriétés et de la Palette des outils.

- Si la propriété `dockingEnabled` reçoit la valeur `TRUE` et que la propriété `type` est définie sur `#fullscreen` ou sur `#dialog`, le type est ignoré et la fenêtre constitue une fenêtre de création.

Exemple

L'instruction suivante définit le type de la fenêtre Planètes sur la valeur `#tool`.

```
-- Lingo syntax
window("Planets").type = #tool

// JavaScript syntax
window("Planets").type = "tool";
```

Voir aussi

appearanceOptions, displayTemplate, dockingEnabled, titlebarOptions, Fenêtre

updateLock

Syntaxe

```
-- Lingo syntax
_movie.updateLock

// JavaScript syntax
_movie.updateLock;
```

Description

Propriété d'animation ; détermine si la scène est mise à jour pendant l'enregistrement du scénario (FALSE) ou non (TRUE). Lecture/écriture.

Vous pouvez éviter la modification de l'affichage de la scène pendant une session d'enregistrement du scénario en attribuant la valeur TRUE à updateLock avant qu'un script ne mette à jour le scénario. Si updateLock présente la valeur FALSE, la scène est mise à jour et affiche une nouvelle image chaque fois que la commande atteint une nouvelle image.

Vous pouvez également utiliser la propriété updateLock pour empêcher les mises à jour involontaires du scénario à la sortie d'une image, par exemple lorsque vous quittez temporairement une image pour examiner les propriétés d'une autre image.

Bien que cette propriété puisse être utilisée pour masquer les changements apportés à une image pendant l'exécution, il faut savoir que les acteurs champ portent la marque de leurs changements dès la modification de leur contenu, contrairement aux modifications apportées aux emplacements ou aux acteurs avec d'autres images-objets, qui ne sont mis à jour que lorsque cette propriété est désactivée.

Exemple

L'instruction suivante affiche la valeur de la propriété updateLock dans la fenêtre Messages.

```
-- Lingo
put _movie.updateLock

// Javascript
trace(_movie.updateLock);
```

Voir aussi

Animation

updateMovieEnabled

Syntaxe

```
the updateMovieEnabled
```

Description

Propriété système ; indique si les changements apportés à l'animation en cours sont automatiquement enregistrés (TRUE) ou non (FALSE, valeur par défaut) lorsque l'animation est orientée vers une autre animation.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante demande à Director d'enregistrer les changements de l'animation actuelle à chaque fois qu'elle passe à une autre animation :

```
-- Lingo
the updateMovieEnabled = TRUE

// Javascript
_movie.updateMovieEnabled = true;
```

URL

Syntaxe

```
-- Lingo syntax
memberObjRef.URL

// JavaScript syntax
memberObjRef.URL;
```

Description

Propriété d'acteur ; spécifie l'URL des acteurs Shockwave Audio (SWA) et acteurs animation Flash.

Pour les acteurs animation Flash, cette propriété est identique à la propriété d'acteur `pathName`.

La propriété `URL` peut être testée et définie. Pour les acteurs SWA, cette propriété ne peut être définie que lorsque l'acteur SWA lu en flux continu est arrêté.

Exemple

L'instruction suivante fait d'un fichier sur un serveur Internet l'adresse URL de l'acteur SWA Benny G. :

```
-- Lingo syntax
on mouseDown
    member("Benny Goodman").URL = "http://audio.adobe.com/samples/classic.swa"
end

// JavaScript syntax
function mouseDown() {
    member("Benny Goodman").URL = "http://audio.adobe.com/samples/classic.swa"
}
```

useAlpha

Syntaxe

```
-- Lingo syntax
memberObjRef.useAlpha
imageObjRef.useAlpha

// JavaScript syntax
memberObjRef.useAlpha;
imageObjRef.useAlpha;
```

Description

Propriété d'acteur bitmap et d'objet image ; pour les objets images et les acteurs 32 bits comportant des données de couche alpha, cette propriété détermine si Director utilise ces données pour dessiner l'image sur la scène (TRUE) ou si Director ne tient pas compte de ces informations (FALSE).

Exemple

L'exemple suivant bascule l'état d'activation de la couche alpha de l'acteur Premier plan :

```
-- Lingo syntax
member("foreground").useAlpha = not(member("foreground").useAlpha)

// JavaScript syntax
switch(member("foreground").useAlpha) {
    case 0:
        member("foreground").useAlpha = 1;
        break;
    case 1:
        member("foreground").useAlpha = 0;
        break;
    }
}
```

useDiffuseWithTexture

Syntaxe

```
member(whichCastmember).shader(whichShader).useDiffuseWithTexture
```

Description

Propriété 3D de matériau standard ; permet de savoir ou de définir si la couleur diffuse est utilisée pour moduler la texture (TRUE) ou non (FALSE).

Lorsque cette propriété présente la valeur TRUE, elle fonctionne en conjonction avec les propriétés blendFunction et blendConstant : lorsque blendFunction présente la valeur #blend, la couleur diffuse est équilibrée avec la couleur de la texture afin de déterminer la couleur finale. Par exemple, si la propriété blendFunction est définie sur #blend et que la propriété blendConstant reçoit la valeur 100,0, la couleur finale correspond à la couleur pure de la texture. Si vous attribuez à blendConstant la valeur 0,0, la couleur finale est la couleur diffuse. Si vous attribuez à blendConstant la valeur 10,0, la couleur finale correspond à 10 % de la couleur de la texture et à 90 % de la couleur diffuse.

La valeur par défaut de cette propriété est FALSE.

Tous les matériaux ont accès aux propriétés de matériau #standard ; outre ces propriétés standard, les matériaux de type #engraver, #newsprint et #painter possèdent des propriétés propres à leur type. Pour plus d'informations, reportez-vous à l'entrée newShader.

Exemple

Dans l'exemple suivant, la propriété `shaderList` du modèle Mystère contient six matériaux. Chaque matériau possède une liste de textures pouvant contenir jusqu'à huit textures. La propriété `diffuseColor` de l'acteur (Niveau2) est rgb(255, 0, 0). La propriété `blendFunction` des six matériaux est définie sur #blend et la propriété `blendConstant` de ces matériaux est définie sur 80. L'instruction suivante définit la propriété `useDiffuseWithTexture` de tous les matériaux utilisés par Mystère sur la valeur TRUE. Un peu de rouge est mélangé à la surface du modèle. Cette propriété est affectée par le paramétrage des propriétés `blendFunction`, `blendFunctionList`, `blendSource`, `blendSourceList`, `blendConstant` et `blendConstantList`.

```
-- Lingo
member("Level2").model("MysteryBox").shaderlist.useDiffuseWithTexture = TRUE

// Javascript
member("Level2").getPropRef("model",1).getProp("shaderlist").useDiffuseWithTexture = true;
```

Voir aussi

`blendFunction`, `blendConstant`

useFastQuads

Syntaxe

```
-- Lingo syntax
_movie.useFastQuads

// JavaScript syntax
_movie.useFastQuads;
```

Description

Propriété d'animation ; détermine s'il convient d'utiliser des opérations de calcul de quadrilatères plus rapides (`TRUE`) ou plus lentes (`FALSE`, valeur par défaut). Lecture/écriture.

Lorsque cette propriété présente la valeur `TRUE`, Director utilise une méthode de calcul plus rapide et moins précise pour les opérations de calcul de quadrilatères. Les calculs rapides de quadrilatères sont suffisants pour les effets simples de rotation et d'inclinaison des images-objets.

Si cette propriété est définie sur `FALSE`, Director utilise la méthode de calcul de quadrilatères par défaut, c'est-à-dire la méthode la plus lente, qui fournit des résultats visuellement plus attractifs en cas d'utilisation de quadrilatères pour des effets de distorsion et d'autres effets aléatoires.

Indépendamment de ce paramètre, les opérations de simple rotation et d'inclinaison d'images-objets utilisent toujours la méthode de calcul rapide de quadrilatères. Si vous définissez `useFastQuads` sur la valeur `TRUE`, la vitesse de calcul de ces opérations simples n'est pas accélérée.

Exemple

L'instruction suivante entraîne Director à utiliser son code de calcul rapide de quadrilatères pour toutes les opérations de quadrilatères de l'animation :

```
- Lingo syntax
_movie.useFastQuads = TRUE

// JavaScript syntax
_movie.useFastQuads = true;
```

Voir aussi

Animation

useHypertextStyles

Syntaxe

```
-- Lingo syntax
memberObjRef.useHypertextStyles

// JavaScript syntax
memberObjRef.useHypertextStyles;
```

Description

Propriété d'acteur texte ; contrôle l'affichage des liens hypertexte dans l'acteur texte spécifié.

Si `useHypertextStyles` présente la valeur `TRUE`, tous les liens sont automatiquement colorés en bleu et soulignés et le curseur prend la forme d'un doigt lorsqu'il est positionné sur un lien.

Si cette propriété présente la valeur `FALSE`, le formatage automatique et le changement de forme du curseur sont désactivés.

Exemple

Le comportement suivant bascule l'état d'activation du formatage des liens hypertexte dans l'acteur texte monTexte :

```
--Lingo syntax
on mouseUp
    member("myText").usehypertextStyles = not(member("myText").usehypertextStyles)
end

// JavaScript syntax
function mouseUp() {
    member("myText").usehypertextStyles = !(member("myText").usehypertextStyles)
}
```

useLineOffset

Syntaxe

```
member(whichCastmember).model(whichModel).toon.useLineOffset
member(whichCastmember).model(whichModel).inker.useLineOffset
```

Description

Propriété 3D de modificateur `toon` et `inker` ; indique si la propriété `lineOffset` du modificateur est utilisée par le modificateur lorsqu'il dessine des lignes sur la surface du modèle.

La valeur par défaut de cette propriété est `FALSE`.

Exemple

L'instruction suivante attribue à la propriété `useLineOffset` du modificateur `toon` du modèle Théière la valeur `FALSE`. La propriété `lineOffset` du modificateur `toon` n'a aucun effet.

```
-- Lingo syntax
member("tp").model("Teapot").toon.useLineOffset = FALSE
```

```
// JavaScript syntax
member("tp").getPropRef("model",1).getProp("toon").useLineOffset = false;
```

Voir aussi

lineOffset

userData

Syntaxe

```
member(whichCastmember).model(whichModel).userData
member(whichCastmember).light(whichLight).userData
member(whichCastmember).camera(whichCamera).userData
member(whichCastmember).group(whichCamera).userData
```

Description

Propriété 3D ; renvoie la liste de propriétés `userData` d'un modèle, d'un groupe, d'une caméra ou d'une lumière. La valeur par défaut de cette propriété pour un objet créé dans un autre programme que Director est une liste de toutes les propriétés affectées à la propriété `userData` du modèle dans le programme de modélisation 3D. La valeur par défaut de cette propriété pour un objet créé dans Director est une liste de propriétés vide [:], à moins que l'objet n'ait été créé à l'aide de commandes de clonage. Lorsqu'une commande de clonage a été utilisée pour créer l'objet, la propriété `userData` du nouvel objet prend une valeur égale à celle de l'objet source d'origine.

Pour modifier les éléments de cette liste, vous devrez utiliser les commandes `addProp` et `deleteProp` documentées dans le dictionnaire Lingo principal.

Exemple

L'instruction suivante affiche la propriété `userData` du modèle nouvelleCarrosserie :

```
put member("Car").model("New Body").userData
-- [#driver: "Bob", #damage: 34]
```

L'instruction suivante ajoute la propriété `#health` avec la valeur 100 à la liste de propriétés `userData` du modèle Lecteur :

```
member("scene").model("Player").userData.addProp(#health,100)
```

userName

Syntaxe

```
-- Lingo syntax
_player.userName

// JavaScript syntax
_player.userName;
```

Description

Propriété de lecteur ; renvoie une chaîne contenant le nom d'utilisateur saisi pendant l'installation de Director. Lecture seule.

Cette propriété est uniquement disponible dans l'environnement de programmation. Elle peut être utilisée dans une animation dans une fenêtre personnalisée pour afficher des informations utilisateur.

Exemple

Le gestionnaire suivant place le nom de l'utilisateur et le numéro de série dans une zone d'affichage dès l'ouverture de la fenêtre. Un script d'animation dans une fenêtre constitue l'endroit idéal pour ce gestionnaire.

```
-- Lingo syntax
on prepareMovie
    displayString = _player.userName & RETURN & _player.organizationName & RETURN &
_player.serialNumber
    member("User Info").text = displayString
end

// JavaScript syntax
function prepareMovie() {
    var displayString = _player.userName + "\n" + _player.organizationName+ "\n" +
_player.serialNumber;
    member("User Info").text = displayString;
}
```

Voir aussi

Lecteur

userName (RealMedia)

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.userName

// JavaScript syntax
memberOrSpriteObjRef.userName;
```

Description

Propriété d'acteur et image-objet RealMedia ; permet de définir le nom d'utilisateur nécessaire à l'accès à un flux RealMedia protégé. Vous ne pouvez pas utiliser cette propriété pour récupérer un nom d'utilisateur spécifié auparavant. Si aucun nom d'utilisateur n'a encore été défini, la valeur de cette propriété est la chaîne `"********"`. La valeur par défaut de cette propriété est une chaîne vide, ce qui signifie qu'aucun nom d'utilisateur n'a été spécifié.

Exemple

Les exemples suivants indiquent que le nom d'utilisateur pour le flux RealMedia de l'acteur Real ou de l'image-objet 2 a été défini.

```
-- Lingo syntax
put(sprite(2).userName) -- "********"
put(member("Real").userName) -- "********"

// JavaScript syntax
put(sprite(2).userName); // "********"
put(member("Real").userName); // "********"
```

Les exemples suivants indiquent que le nom d'utilisateur pour le flux RealMedia de l'acteur Real ou de l'image-objet 2 n'a jamais été défini.

```
-- Lingo syntax
put(sprite(2).userName) -- ""
put(member("Real").userName) -- ""

// JavaScript syntax
put(sprite(2).userName); // ""
put(member("Real").userName); // ""
```

Les exemples suivants indiquent que le nom d'utilisateur pour le flux RealMedia de l'acteur Real et de l'image-objet 2 est Marcel.

```
-- Lingo syntax
member("Real").userName = "Marcelle"
sprite(2).userName = "Marcelle"

// JavaScript syntax
member("Real").userName = "Marcelle";
sprite(2).userName = "Marcelle";
```

Voir aussi

password

useTargetFrameRate

Syntaxe

```
sprite(which3dSprite).useTargetFrameRate
```

Description

Propriété 3D d'image-objet ; détermine si la propriété `targetFrameRate` de l'image-objet est appliquée. Si la propriété `useTargetFrameRate` présente la valeur `TRUE`, le nombre de polygones des modèles de l'image-objet est réduit pour atteindre la cadence d'image spécifiée ciblée.

Exemple

Les instructions suivantes attribuent à la propriété `targetFrameRate` de l'image-objet 3 la valeur 45 et appliquent la cadence d'image en définissant la propriété `useTargetFrameRate` de l'image-objet sur la valeur `TRUE` :

```
-- Lingo syntax
sprite(3).targetFrameRate = 45
sprite(3).useTargetFrameRate = TRUE

// JavaScript syntax
sprite(3).targetFrameRate = 45;
sprite(3).useTargetFrameRate = true;
```

Voir aussi

targetFrameRate

vertex

Syntaxe

```
-- Lingo syntax
memberObjRef.vertex[whichVertexPosition]
```

```
// JavaScript syntax
memberObjRef.vertex[whichVertexPosition];
```

Description

Expression de sous-chaîne ; permet d'accéder directement à certaines parties d'une liste de sommets d'un acteur forme vectorielle.

Utilisez cette sous-chaîne pour éviter de devoir analyser différentes sous-chaînes de la liste de sommets. L'utilisation de ce type d'expression de sous-chaîne permet à la fois de tester et de définir les valeurs de la liste de sommets.

Exemple

Le code suivant indique comment déterminer le nombre de points de sommet dans un acteur :

```
-- Lingo syntax
put(member("Archie").vertex.count) -- 2

// JavaScript syntax
put(member("Archie").vertex.count); // 2
```

Pour obtenir le second sommet de l'acteur, vous pouvez utiliser le code suivant :

```
-- Lingo syntax
put(member("Archie").vertex[2]) -- point(66.0000, -5.0000)

// JavaScript syntax
put(member("Archie").vertex[2]); // point(66.0000, -5.0000)
```

Vous pouvez également définir la valeur dans une poignée de contrôle :

```
-- Lingo syntax
member("Archie").vertex[2].handle1 = point(-63.0000, -16.0000)

// JavaScript syntax
member("Archie").vertex[2].handle1 = point(-63.0000, -16.0000);
```

Voir aussi

vertexList

vertexList

Syntaxe

```
-- Lingo syntax
memberObjRef.vertexList

// JavaScript syntax
memberObjRef.vertexList;
```

Description

Propriété d'acteur ; renvoie une liste linéaire contenant des listes de propriétés, une pour chaque sommet d'une forme vectorielle. La liste de propriétés contient l'emplacement du sommet et la poignée de contrôle. Si l'emplacement a la valeur (0, 0), il n'y a pas de poignée de contrôle.

Chaque sommet peut avoir deux poignées de contrôle déterminant la courbe entre ce sommet et les sommets adjacents. Dans `vertexList`, les coordonnées des poignées de contrôle d'un sommet ont des valeurs relatives au sommet, plutôt que des valeurs absolues dans le système des coordonnées de la forme. Si la première poignée de contrôle d'un sommet est située 10 pixels à gauche de ce sommet, son emplacement est enregistré sous la forme (-10, 0). Par conséquent, lorsque l'emplacement d'un sommet est modifié avec Lingo, les poignées de contrôle se déplacent avec le sommet et n'ont pas besoin d'être mises à jour (sauf si l'utilisateur veut absolument changer l'emplacement ou la taille de la poignée).

En cas de modification de cette propriété, sachez que vous devez réinitialiser le contenu de la liste après avoir changé l'une des valeurs. En effet, lorsque vous affectez une variable à la valeur de la propriété, vous placez une copie de la liste, et non la liste elle-même, dans la variable. Pour appliquer un changement, utilisez une syntaxe comme :

```
- Get the current property contents
currVertList = member(1).vertexList
-- Add 25 pixels to the horizontal and vertical positions of the first vertex in the list
currVertList[1] .vertex = currVertList[1] .vertex + point(25, 25)
-- Reset the actual property to the newly computed position
member(1).vertexList = currVertList
```

Exemple

L'instruction suivante affiche la valeur vertexList pour une ligne arquée comportant deux sommets :

```
-- Lingo syntax
put(member("Archie").vertexList)-- [[#vertex: point(-66.0000, 37.0000), #handle1: point(-
70.0000, -36.0000), #handle2: point(-62.0000, 110.0000)], [#vertex: point(66.0000, -5.0000),
#handle1: point(121.0000, 56.0000), #handle2: point(11.0000, -66.0000)]]

// JavaScript syntax
put(member("Archie").vertexList);//[[#vertex: point(-66.0000, 37.0000),#handle1: point(-
70.0000, -36.0000),#handle2: point(-62.0000, 110.0000)], [#vertex: point(66.0000, -
5.0000),#handle1: point(121.0000, 56.0000), #handle2: point(11.0000, -66.0000)]]
```

Voir aussi

addVertex(), count(), deleteVertex(), moveVertex(), moveVertexHandle(), originMode, vertex

vertexList (générateur de maille)

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).vertexList
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#mesh`, cette propriété permet d'obtenir ou de définir la propriété `vertexList` de la ressource de modèle.

La propriété `vertexList` est une liste linéaire de chaque sommet utilisé dans la maille. Un seul sommet peut être partagé par plusieurs faces de la maille. Vous pouvez spécifier une liste de n'importe quelle taille pour cette propriété ; toutefois, celle-ci n'enregistre que le nombre d'éléments spécifié lors de l'utilisation de la commande `newMesh()` pour créer la ressource de modèle `#mesh`.

Exemple

L'instruction suivante définit la propriété vertexList de la ressource de modèle Triangle :

```
member("Shapes").modelResource("Triangle").vertexList = [vector(0,0,0), vector(20,0,0),
vector(20, 20, 0)]
```

Voir aussi

newMesh, face, vertices

vertexList (déformation de maille)

Syntaxe

```
member(whichCastmember).model(whichModel).meshDeform.mesh[index].vertexList
```

Description

Propriété 3D ; lorsqu'elle est utilisée avec un modèle associé au modificateur `#meshDeform`, cette propriété permet d'obtenir ou de définir la propriété `vertexList` de la maille spécifiée dans le modèle référencé.

La propriété `vertexList` est une liste linéaire de chaque sommet utilisé dans la maille spécifiée. Un seul sommet peut être partagé par plusieurs faces de la maille.

Si un modèle utilise les modificateurs `#sds` ou `#lod` en complément du modificateur `#meshDeform`, il est important de ne pas oublier que la valeur de cette propriété change en fonction des modificateurs `#sds` ou `#lod`.

Exemple

L'instruction suivante affiche la propriété vertexList du modificateur `#meshDeform` pour la première maille du modèle Triangle :

```
put member("Shapes").model("Triangle").meshDeform.mesh[1].vertexList
-- [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)]
```

Voir aussi

face, vertices, mesh (propriété)

vertices

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).face[faceIndex].vertices
```

Description

Propriété 3D de face ; lorsqu'elle est utilisée avec une ressource de modèle de type `#mesh`, cette propriété permet d'obtenir ou de définir les sommets de la propriété `vertexList` de la ressource à utiliser pour la face de la maille spécifiée par *`indexDeFace`*.

Cette propriété est une liste linéaire de trois entiers correspondant aux positions d'index des trois sommets, comme indiqué dans la propriété `vertexList` de la maille, qui comprend la face spécifiée.

Les sommets doivent être spécifiés dans la liste dans un ordre antihoraire de façon à obtenir une normale de surface pointant vers l'extérieur.

Si vous apportez des modifications à cette propriété ou que vous utilisez la commande `generateNormals()`, vous devrez appeler la commande `build()` pour reconstruire la maille.

Exemple

L'exemple suivant affiche la propriété vertexList de la ressource de modèle de maille carréSimple, puis affiche la propriété `vertices` pour la seconde face de cette maille :

```
put member("3D").modelResource("SimpleSquare").vertexList
-- [vector( 0.0000, 0.0000, 0.0000), vector( 0.0000, 5.0000, 0.0000), vector( 5.0000, 0.0000,
0.0000), vector( 5.0000, 5.0000, 0.0000)]
put member("3D").modelResource("SimpleSquare").face[1].vertices
-- [3, 4, 1]
```

Voir aussi

face, vertexList (déformation de maille), generateNormals()

video (QuickTime, AVI)

Syntaxe

```
member(whichCastMember).video
the video of member whichCastMember
```

Description

Propriété d'acteur vidéo numérique ; détermine si l'image graphique de l'acteur vidéo numérique spécifié est lue (TRUE ou 1) ou non (FALSE ou 0).

Seul l'élément visuel de l'acteur vidéo numérique est affecté. Par exemple, lorsque la propriété `video` présente la valeur FALSE, la lecture de la piste audio de la vidéo (s'il en existe une) se poursuit.

Exemple

L'instruction suivante désactive la vidéo associée à l'acteur Entrevue :

```
-- Lingo syntax
member("Interview").video = FALSE

// JavaScript syntax
member("Interview").video = false;
```

Voir aussi

setTrackEnabled(), trackEnabled

video (RealMedia, Windows Media)

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.video

// JavaScript syntax
memberOrSpriteObjRef.video;
```

Description

Propriété RealMedia et Windows Media ; permet de savoir ou de définir si l'image-objet ou l'acteur rend la vidéo (TRUE ou 1) ou uniquement l'audio (FALSE ou 0). Lecture/écriture.

Les valeurs entières autres que 1 ou 0 sont considérées comme TRUE.

Utilisez cette propriété pour supprimer la vidéo lors de la lecture du composant audio d'un acteur RealMedia ou Windows Media ou pour activer/désactiver la vidéo pendant la lecture.

Exemple

Les exemples suivants indiquent que la propriété `video` de l'image-objet 2 et de l'acteur Real présente la valeur `TRUE`.

```
-- Lingo syntax
put(sprite(2).video) -- 1
put(member("Real").video) -- 1

// JavaScript syntax
put(sprite(2).video); // 1
put(member("Real").video); // 1
```

Les exemples suivants définissent la propriété `video` sur la valeur `FALSE` pour l'élément vidéo RealMedia de l'image-objet 2 et de l'acteur Real.

```
-- Lingo syntax
sprite(2).video = FALSE
member("Real").video = FALSE

// JavaScript syntax
sprite(2).video = 0;
member("Real").video = 0;
```

videoFormat

Syntaxe

```
-- Lingo syntax
dvdObjRef.videoFormat

// JavaScript syntax
dvdObjRef.videoFormat;
```

Description

Propriété de DVD. Renvoie un symbole indiquant le format vidéo. Lecture seule.

Les symboles possibles sont les suivants :

Symbole	Description
#MPEG1	Le format vidéo est MPEG-1.
#MPEG2	Le format vidéo est MPEG-2.
#unknown	Le format vidéo est inconnu.

Exemple

L'instruction suivante indique le format vidéo de l'image-objet de l'objet DVD.

```
-- Lingo syntax
put sprite(1).videoFormat

// JavaScript syntax
trace(sprite(1).videoFormat)
```

Voir aussi

DVD

videoForWindowsPresent

Syntaxe

```
the videoForWindowsPresent
```

Description

Propriété système ; indique si un logiciel AVI est installé sur l'ordinateur.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante vérifie si Vidéo pour Windows est présent et, dans le cas contraire, fait passer la tête de lecture au repère Autre séquence :

```
if the videoForWindowsPresent= FALSE then go to "Alternate Scene"
```

Voir aussi

`QuickTimeVersion()`

viewH

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.viewH

// JavaScript syntax
memberOrSpriteObjRef.viewH;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la coordonnée horizontale d'une animation Flash et le point de vue d'une forme vectorielle, exprimés en pixels. Les valeurs peuvent être des nombres à virgule flottante. La valeur par défaut est 0.

Le point de vue d'une animation Flash est défini par rapport à son point d'origine.

La définition d'une valeur positive pour `viewH` décale l'animation vers la gauche à l'intérieur de l'image-objet ; le choix d'une valeur négative décale l'animation vers la droite. Par conséquent, la modification de la propriété `viewH` peut entraîner le recadrage de l'animation, voire son retrait total de l'écran.

Cette propriété peut être testée et définie.

Remarque : *cette propriété doit avoir la valeur par défaut si la propriété scaleMode a pour valeur #autoSize. Sinon, l'image-objet n'est pas correctement affichée.*

Exemple

Le gestionnaire suivant accepte une référence d'image-objet comme paramètre et déplace la vue d'une image-objet d'animation Flash de la gauche vers la droite à l'intérieur du rectangle de délimitation de cette image-objet :

```
-- Lingo syntax
on panRight whichSprite
    repeat with i = 120 down to -120
        sprite(whichSprite).viewH = i
        _movie.updateStage()
```

```
    end repeat
end

// JavaScript syntax
function panRight(whichSprite) {
    var i = 120;
    while(i > -121) {
        sprite(whichSprite).viewH = i;
        _movie.updateStage();
        i--;
    }
}
```

Voir aussi

scaleMode, viewV, viewPoint, viewScale

viewPoint

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.viewPoint

// JavaScript syntax
memberOrSpriteObjRef.viewPoint;
```

Description

Propriété d'acteur et d'image-objet ; contrôle le point présent dans une animation Flash ou une forme vectorielle affiché au centre du rectangle de délimitation de l'image-objet, en pixels. Ces valeurs sont des entiers.

La modification du point de vue d'un acteur ne fait que modifier l'affichage d'une animation dans le rectangle de délimitation de l'image-objet et non l'emplacement de l'image-objet sur la scène. Le point de vue correspond à la coordonnée d'un acteur affiché au centre du rectangle de délimitation de l'image-objet et est toujours exprimé par rapport au point d'origine de l'animation (tel que défini par les propriétés `originPoint`, `originH` et `originV`). Par exemple, si vous définissez le point de vue d'une animation Flash sur point(100, 100), le centre de l'image-objet est le point de l'animation Flash situé à 100 pixels (unités d'animation Flash) vers la droite et 100 pixels (unités d'animation Flash) vers le bas à partir du point d'origine, quel que soit l'endroit où vous déplacez le point d'origine.

La propriété `viewPoint` est spécifiée sous la forme d'une valeur de point Director : par exemple, point (100, 200). La définition du point de vue d'une animation Flash avec la propriété `viewPoint` équivaut à définir séparément les propriétés `viewH` et `viewV`. Par exemple, la définition de la propriété `viewPoint` sur point (50, 75) équivaut à définir la propriété `viewH` sur 50 et la propriété `viewV` sur 75.

Les valeurs de point Director spécifiées pour la propriété `viewPoint` doivent correspondre à des nombres entiers, alors que les valeurs `viewH` et `viewV` peuvent être spécifiées sous forme de nombres à virgule flottante. Lorsque vous testez la propriété `viewPoint`, les valeurs de point sont tronquées de façon à apparaître sous forme de nombres entiers. En règle générale, utilisez les propriétés `viewH` et `viewV` si vous mettez l'accent sur la précision et utilisez la propriété `originPoint` si vous recherchez la rapidité et la facilité.

Cette propriété peut être testée et définie. La valeur par défaut est point (0, 0).

Remarque : *cette propriété doit avoir la valeur par défaut si la propriété scaleMode a pour valeur #autoSize. Sinon, l'image-objet n'est pas correctement affichée.*

Exemple

Le gestionnaire suivant a pour effet de déplacer une image-objet animation Flash vers le bas et vers la droite par incréments de cinq pixels en unités d'animation Flash :

```
-- Lingo syntax
on panAcross(whichSprite)
    repeat with i = 1 to 10
        sprite(whichSprite).viewPoint = sprite(whichSprite).viewPoint + point(i * -5, i * -
5)
        _movie.updateStage()
    end repeat
end

// JavaScript syntax
function panAcross(whichSprite) {
    var i = 1;
    while(i < 11) {
        sprite(whichSprite).viewPoint = sprite(whichSprite).viewPoint + point(i * -5, i * -
5);
        _movie.updateStage();
        i++
    }
}
```

Voir aussi

scaleMode, viewV, viewH, viewScale

viewScale

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.viewScale

// JavaScript syntax
memberOrSpriteObjRef.viewScale;
```

Description

Propriété d'acteur et d'image-objet ; définit la valeur d'ensemble permettant de mettre à l'échelle l'affichage d'une image-objet d'animation Flash ou de forme vectorielle à l'intérieur du rectangle de délimitation de l'image-objet. Vous spécifiez la valeur en degrés sous la forme d'un nombre à virgule flottante. La valeur par défaut est 100.

Le rectangle de l'image-objet n'est pas mis à l'échelle ; seul l'affichage de l'acteur dans le rectangle l'est. La définition de la propriété `viewScale` d'une image-objet est semblable au choix d'un objectif pour un appareil-photo. A mesure que la valeur `viewScale` diminue, la taille apparente de l'animation dans l'image-objet augmente, et vice versa. Par exemple, la définition de la propriété `viewScale` sur 200 % signifie que la zone affichée à l'intérieur de l'image-objet est deux fois plus importante que précédemment et que l'acteur placé dans l'image-objet est réduit de moitié par rapport à sa taille d'origine.

Une différence notable entre les propriétés `viewScale` et `scale` réside dans le fait que la propriété `viewScale` effectue toujours une mise à l'échelle en partant du centre du rectangle de délimitation de l'image-objet, tandis que la propriété `scale` effectue la mise à l'échelle en partant d'un point déterminé par la propriété `originMode` de l'animation Flash.

Cette propriété peut être testée et définie.

Remarque : *cette propriété doit avoir la valeur par défaut si la propriété scaleMode a pour valeur #autoSize. Sinon, l'image-objet n'est pas correctement affichée.*

Exemple

Le script d'image-objet suivant configure l'image-objet animation Flash et double l'échelle de son affichage :

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).viewScale = 200
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).viewScale = 200;
}
```

Voir aussi

scaleMode, viewV, viewPoint, viewH

viewV

Syntaxe

```
-- Lingo syntax
memberOrSpriteObjRef.viewV

// JavaScript syntax
memberOrSpriteObjRef.viewV;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la coordonnée verticale d'une animation Flash et le point de vue d'une forme vectorielle, exprimés en pixels. Les valeurs peuvent être des nombres à virgule flottante. La valeur par défaut est 0.

Le point de vue d'une animation Flash est défini par rapport à son point d'origine.

La définition d'une valeur positive pour `viewV` décale l'animation vers le haut à l'intérieur de l'image-objet ; le choix d'une valeur négative décale l'animation vers le bas. Par conséquent, la modification de la propriété `viewV` peut entraîner le recadrage de l'animation, voire son retrait total de l'écran.

Cette propriété peut être testée et définie.

Remarque : *cette propriété doit avoir la valeur par défaut si la propriété scaleMode a pour valeur #autoSize. Sinon, l'image-objet n'est pas correctement affichée.*

Exemple

Le gestionnaire suivant accepte une référence d'image-objet comme paramètre et déplace la vue d'une image-objet d'animation Flash du haut vers le bas à l'intérieur du rectangle de délimitation de cette image-objet :

```
-- Lingo syntax
on panDown(whichSprite)
    repeat with i = 120 down to -120
        sprite(whichSprite).viewV = i
        _movie.updateStage()
```

```
    end repeat
end

// JavaScript syntax
function panDown(whichSprite) {
    var i = 120;
    while(i > -121) {
        sprite(whichSprite).viewV = i;
        _movie.updateStage();
        i--;
    }
}
```

Voir aussi

scaleMode, viewV, viewPoint, viewH

visible

Syntaxe

```
-- Lingo syntax
windowObjRef.visible

// JavaScript syntax
windowObjRef.visible;
```

Description

Propriété de fenêtre ; détermine si une fenêtre est visible (TRUE) ou non (FALSE). Lecture/écriture.

Exemple

L'instruction suivante rend la fenêtre Tableau_de_commande visible :

```
-- Lingo syntax
window("Control_Panel").visible = TRUE

// JavaScript syntax
window("Control_Panel").visible = true;
```

Voir aussi

Fenêtre

visible (image-objet)

Syntaxe

```
sprite(whichSprite).visible
the visible of sprite whichSprite
```

Description

Propriété d'image-objet ; détermine si l'image-objet spécifiée par *quelleImageObjet* est visible (TRUE) ou non (FALSE). Cette propriété affecte toutes les images-objets de la piste, quelle que soit leur position dans le scénario.

Remarque : *si la propriété* `visible` *d'une piste d'image-objet présente la valeur* `FALSE`*, l'image-objet est invisible et seuls les événements liés à la souris ne sont pas envoyés à cette piste. L'envoi des événements* `beginSprite`*,* `endSprite`*,* `prepareFrame`*,* `enterFrame` *et* `exitFrame` *se poursuit, quel que soit le réglage de visibilité de l'image-objet. Toutefois, la sélection du bouton Désactiver dans cette piste du scénario a pour effet de définir la propriété* `visible` *sur la valeur* `FALSE` *et d'empêcher l'envoi de tous les événements à cette piste. La sélection de ce bouton désactive une piste, tandis que le réglage de la propriété* `visible` *d'une image-objet sur la valeur* `FALSE` *n'affecte qu'une propriété graphique.*

Cette propriété peut être testée et définie. Si cette propriété présente la valeur `FALSE`, elle n'est pas automatiquement réinitialisée sur `TRUE` à la fin de l'image-objet. Vous devez définir la propriété `visible` de l'image-objet sur la valeur `TRUE` afin de voir tous les autres acteurs utilisant cette piste.

Exemple

L'instruction suivante rend l'image-objet 8 visible :

```
sprite(8).visible = TRUE
```

visibility

Syntaxe

```
member(whichCastmember).model(whichModel).visibility
modelObjectReference.visibility
```

Description

Propriété 3D ; permet d'obtenir ou de définir la propriété `visibility` du modèle référencé. Cette propriété détermine la façon dont la géométrie du modèle est dessinée. Elle peut avoir une des valeurs suivantes :

- `#none` indique de ne pas tracer les polygones et que le modèle est invisible.
- `#front` indique que seuls les polygones faisant face à la caméra sont dessinés. Cette méthode optimise la vitesse de rendu. Il s'agit du paramétrage par défaut de cette propriété.
- `#back` indique que seuls les polygones orientés dans la direction opposée à la caméra sont dessinés. Utilisez ce paramètre lorsque vous souhaitez dessiner l'intérieur d'un modèle ou pour les modèles qui ne sont pas correctement dessinés, peut-être parce qu'ils ont été importés à partir d'un format de fichier utilisant une valeur différente pour le calcul des normales.
- `#both` indique que les deux côtés de tous les polygones sont dessinés. Utilisez ce paramètre lorsque vous souhaitez voir le plan quelle que soit la direction et pour les modèles qui ne sont pas dessinés correctement.

Exemple

L'instruction suivante indique que la propriété `visibility` du modèle Monstre02 présente la valeur `#none`. Le modèle est invisible.

```
-- Lingo syntax
put member("3D").model("Monster02").visibility
-- #none

// JavaScript syntax
trace(member("3D").getPropRef("model",1).visibility)
// symbol("none")
```

volume (DVD)

Syntaxe

```
-- Lingo syntax
dvdObjRef.volume

// JavaScript syntax
dvdObjRef.volume;
```

Description

Propriété de DVD. Détermine le volume sonore du DVD en cours. Lecture/écriture.

Le volume doit être un nombre entier compris entre 0 (volume désactivé) et 100 (volume maximal).

Sous Windows, l'échelle du volume est logarithmique. Sur Mac, cette échelle est linéaire.

Exemple

L'instruction suivante définit le volume de l'acteur DVD.

```
-- Lingo syntax
member(1).volume = 20

// JavaScript syntax
member(1).volume = 20;
```

Voir aussi

DVD

volume (acteur)

Syntaxe

```
-- Lingo syntax
memberObjRef.volume

// JavaScript syntax
memberObjRef.volume;
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; détermine le volume de l'acteur SWA lu en flux continu spécifié. Les valeurs utilisables sont comprises entre 0 et 255.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante place le volume de l'acteur SWA lu en flux continu sur la moitié du volume maximum :

```
-- Lingo syntax
member("SWAfile").volume = 128

// JavaScript syntax
member("SWAfile").volume = 128;
```

volume (piste audio)

Syntaxe

```
-- Lingo syntax
soundChannelObjRef.volume

// JavaScript syntax
soundChannelObjRef.volume;
```

Description

Propriété de piste audio ; détermine le volume d'une piste audio. Lecture/écriture.

Les pistes audio sont numérotées 1, 2, 3, etc., jusqu'à 8. Les pistes 1 et 2 sont celles qui apparaissent dans le scénario.

La valeur de la propriété `volume` est comprise entre 0 (son désactivé) et 255 (volume maximal). La valeur 255 correspond au volume maximal du système, contrôlé par la propriété `soundLevel` de l'objet son, et les valeurs inférieures sont établies en fonction du volume total. Cette propriété permet à plusieurs pistes d'avoir des paramètres indépendants dans la plage disponible.

Plus la valeur de la propriété de son `volume` est faible, plus vous entendrez de bruit ou de souffle. L'utilisation de la propriété `soundLevel` peut produire moins de bruit, mais offre moins de contrôle.

Vous pouvez voir un exemple d'utilisation de `volume` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo Examples, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante règle le volume de la piste audio 2 sur 130, ce qui représente un niveau sonore moyen :

```
-- Lingo syntax
sound(2).volume = 130

// JavaScript syntax
sound(2).volume = 130;
```

Voir aussi

`Piste audio`, `soundLevel`

volume (image-objet)

Syntaxe

```
-- Lingo syntax
spriteObjRef.volume

// JavaScript syntax
spriteObjRef.volume;
```

Description

Propriété d'image-objet ; contrôle le volume d'un acteur animation vidéo numérique ou Windows Media spécifié par un nom ou un numéro. Les valeurs du volume varient entre 0 et 256. Les valeurs inférieures ou égales à 0 correspondent au son désactivé. Les valeurs supérieures à 256 correspondent à un niveau sonore très élevé et provoquent une distorsion considérable.

Exemple

L'instruction suivante règle le volume de l'animation QuickTime exécutée dans la piste d'image-objet 7 sur 256, ce qui représente le volume sonore maximum :

```
-- Lingo syntax
sprite(7).volume = 256

// JavaScript syntax
sprite(7).volume = 256;
```

Voir aussi

soundLevel

volume (Windows Media)

Syntaxe

```
-- Lingo syntax
windowsMediaObjRef.volume

// JavaScript syntax
windowsMediaObjRef.volume;
```

Description

Propriété d'image-objet Windows Media ; détermine le volume d'une image-objet Windows Media.

La valeur de cette propriété est un nombre entier compris entre 0 (volume désactivé) et 7 (volume très élevé).

Vous pouvez également définir cette propriété à l'aide du menu Contrôle > Volume dans Director.

Exemple

L'instruction suivante définit le volume de l'image-objet 7 sur la valeur 2 :

```
-- Lingo syntax
sprite(7).volume = 2

// JavaScript syntax
sprite(7).volume = 2;
```

Voir aussi

Windows Media

warpMode

Syntaxe

```
-- Lingo syntax
spriteObjRef.warpMode

// JavaScript syntax
spriteObjRef.warpMode;
```

Description

Propriété d'image-objet QuickTime VR ; indique le type d'effet de torsion appliqué à un panorama.

Les valeurs possibles sont `#full` (complet), `#partial` (partiel) et `#none` (aucun).

Cette propriété peut être testée et définie. Lorsque cette propriété est testée et que les valeurs des modes statiques et de mouvement diffèrent, la valeur de la propriété est la valeur définie pour le mode en cours. Lorsque définie, cette propriété détermine l'effet de torsion à la fois pour les modes statiques et de mouvement.

Exemple

L'instruction suivante donne à la propriété `warpMode` de l'image-objet 1 la valeur `#full` :

```
-- Lingo syntax
sprite(1).warpMode = #full

// JavaScript syntax
sprite(1).warpMode = symbol("full");
```

width

Syntaxe

```
-- Lingo syntax
memberObjRef.width
imageObjRef.width
spriteObjRef.width

// JavaScript syntax
memberObjRef.width;
imageObjRef.width;
spriteObjRef.width;
```

Description

Propriété d'acteur, d'image et d'image-objet ; pour les acteurs forme vectorielle, Flash, GIF animé, RealMedia, Windows Media, bitmap et forme, détermine la largeur d'un acteur en pixels. Lecture seule pour les acteurs et les objets images, lecture/écriture pour les images-objets.

Cette propriété n'affecte pas les acteurs champ et bouton.

Exemple

L'instruction suivante affecte la largeur de l'acteur 50 à la variable `theHeight` :

```
-- Lingo syntax
theHeight = member(50).width

// JavaScript syntax
var theHeight = member(50).width;
```

L'instruction suivante règle la largeur de l'image-objet 10 sur 26 pixels :

```
-- Lingo syntax
sprite(10).width = 26

// JavaScript syntax
sprite(10).width = 26;
```

L'instruction suivante affecte la largeur de l'image-objet numéro i + 1 à la variable `howWide` :

```
-- Lingo syntax
howWide = sprite(i + 1).width

// JavaScript syntax
var howWide = sprite(i + 1).width;
```

Voir aussi

height, image (image), Acteur, Image-objet

width (3D)

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).width
modelResourceObjectReference.width
```

Description

Propriété 3D ; permet d'obtenir ou de définir la largeur du plan d'une ressource de modèle de type `#box` ou `#plane`. Cette propriété doit être supérieure à 0,0 et a un paramètre par défaut de 1,0. Pour les objets de type `#box`, la valeur par défaut de `width` est 50,0. Pour les objets de type `#plane`, la valeur par défaut est de 1,0. La propriété `width` est mesurée le long de l'axe des x.

Exemple

L'instruction suivante donne au plan de la ressource de modèle Gazon une largeur de 250,0.

```
-- Lingo syntax
member("3D World").modelResource("Grass plane").width = 250.0

// JavaScript syntax
Member("3D World").getPropRef("modelResource",1).width = 250.0;
```

widthVertices

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).
widthVertices
modelResourceObjectReference.widthVertices
```

Description

Propriété 3D ; permet d'obtenir ou de définir le nombre de sommets (sous forme d'un nombre entier) sur l'axe des x d'une ressource de modèle de type `#box` ou `#plane`. Cette propriété doit être supérieure ou égale à 2 et a une valeur par défaut de 2.

Exemple

L'instruction suivante attribue à la propriété `widthVertices` de la ressource de modèle Tour la valeur 10. Dix-huit polygones (2 * (10-1) triangles) sont utilisés pour définir la géométrie de la ressource de modèle le long de son axe des x.

```
member("3D World").modelResource("Tower").widthVertices = 10
```

wind

Syntaxe

```
member(whichCastmember).modelResource(whichModelResource).wind
modelResourceObjectReference.wind
```

Description

Propriété 3D ; permet d'obtenir ou de définir la propriété `wind` d'une ressource de modèle de type `#particle` sous la forme d'un vecteur.

Cette propriété `wind` définit la direction et la force du vent appliquées à toutes les particules au cours de chaque étape de la simulation. La valeur par défaut de cette propriété est `vector(0, 0, 0)`, qui spécifie qu'aucun vent n'est appliqué.

Exemple

```
-- Lingo syntax
put member("3D").modelResource("fog bank").wind
-- vector(10.5,0,0)

// JavaScript syntax
trace(member("3D").getPropRef("modelResource","1").wind)
// vector(10.5,0,0)
```

window

Syntaxe

```
-- Lingo syntax
_player.window[windowNameOrNum]

// JavaScript syntax
_player.window[windowNameOrNum];
```

Description

Propriété de lecteur ; offre un accès par index ou par nom aux objets fenêtre créés par le lecteur Director. Lecture seule.

L'argument *`nomOuNumFenêtre`* peut être une chaîne spécifiant le nom de la fenêtre à laquelle accéder ou un nombre entier indiquant la position d'index de cette fenêtre.

Cette propriété a la même fonction que la méthode de haut niveau `window()`.

Exemple

Les instructions suivantes définissent la variable `myWindow` sur le troisième objet fenêtre :

```
-- Lingo syntax
myWindow = _player.window[3]

// JavaScript syntax
var myWindow = _player.window[3];
```

Voir aussi

`Lecteur`, `window()`

windowBehind

Syntaxe

```
-- Lingo syntax
windowObjRef.windowBehind

// JavaScript syntax
windowObjRef.windowBehind;
```

Description

Propriété de fenêtre ; renvoie une référence à la fenêtre située derrière toutes les autres fenêtres. Lecture seule.

Exemple

Les instructions suivantes définissent la variable `backWindow` sur la fenêtre située derrière toutes les autres fenêtres, puis déplacent cette fenêtre au premier plan :

```
-- Lingo syntax
backWindow = _player.windowList[5].windowBehind
backWindow.moveToFront()

// JavaScript syntax
var backWindow = _player.windowList[5].windowBehind;
backWindow.moveToFront();
```

Voir aussi

moveToBack(), moveToFront(), Fenêtre, windowInFront, windowList

windowInFront

Syntaxe

```
-- Lingo syntax
windowObjRef.windowInFront

// JavaScript syntax
windowObjRef.windowInFront;
```

Description

Propriété de fenêtre ; renvoie une référence à la fenêtre située devant toutes les autres fenêtres. Lecture seule.

Exemple

Les instructions suivantes définissent la variable `frontWindow` sur la fenêtre située devant toutes les autres fenêtres, puis déplacent cette fenêtre à l'arrière-plan :

```
-- Lingo syntax
frontWindow = _player.windowList[5].windowInFront
frontWindow.moveToBack()

// JavaScript syntax
var frontWindow = _player.windowList[5].windowInFront
frontWindow.moveToBack();
```

Voir aussi

moveToBack(), moveToFront(), Fenêtre, windowBehind, windowList

windowList

Syntaxe

```
-- Lingo syntax
_player.windowList

// JavaScript syntax
_player.windowList;
```

Description

Propriété de lecteur ; affiche la liste des références à toutes les fenêtres d'animation recensées. Lecture seule.

La scène est également considérée comme une fenêtre.

Exemple

L'instruction suivante affiche dans la fenêtre Messages la liste de toutes les fenêtres d'animation recensées :

```
-- Lingo syntax
trace(_player.windowList)

// JavaScript syntax
trace(_player.windowList);
```

Voir aussi

Lecteur

wordWrap

Syntaxe

```
-- Lingo syntax
memberObjRef.wordWrap

// JavaScript syntax
memberObjRef.wordWrap;
```

Description

Propriété d'acteur ; détermine si le retour à la ligne automatique est autorisé (TRUE) ou non (FALSE).

Exemple

L'instruction suivante désactive la fonction de retour à la ligne automatique pour l'acteur champ Pierre :

```
--Lingo syntax
member("Rokujo").wordWrap = FALSE

// JavaScript syntax
member("Rokujo").wordWrap = false;
```

worldPosition

Syntaxe

```
member(whichCastmember).model(whichModel).worldPosition
```

```
member(whichCastmember).light(whichLight).worldPosition
member(whichCastmember).camera(whichCamera).worldPosition
member(whichCastmember).group(whichGroup).worldPosition
```

Description

Propriété 3D ; permet d'obtenir, mais pas de définir, la position d'un nœud avec les coordonnées de l'univers. Un nœud peut être un modèle, un groupe, une caméra ou une lumière. Cette propriété produit un résultat équivalent à celui de la commande `getWorldTransform().position`. La position d'un nœud est représentée par un objet vecteur.

Exemple

L'instruction suivante indique que la position du modèle Mars, en coordonnées de l'univers, est `vector(-1333.2097, 0.0000, -211.0973)`.

```
--Lingo syntax
put member("scene").model("Mars").worldPosition
-- vector(-1333.2097, 0.0000, -211.0973)

// JavaScript syntax
trace(member("scene").getProp("model",1).worldPosition)
// vector(-1333.2097, 0.0000, -211.0973)
```

Voir aussi

`getWorldTransform()`, `position (transformation)`

worldTransform

Syntaxe

```
member(whichMember).model(whichModel).bonesPlayer.bone[index].worldTransform
```

Description

Propriété 3D du modificateur bonesPlayer ; permet d'obtenir la transformation d'un segment spécifique par rapport à l'univers, contrairement à la propriété `transform`, qui renvoie la transformation du segment par rapport au parent. La propriété `worldTransform` n'est utilisable qu'avec les modèles modifiés par bonesPlayer.

Exemple

L'instruction suivante enregistre la transformation d'un segment par rapport à l'univers dans la variable `finalTransform`:

```
--Lingo syntax
finalTransform = member("3D").model("biped").bonesPlayer.bone[3].worldTransform

// JavaScript syntax
finalTransform =
member("3D").getPropRef("model",1).getProp("bonesPlayer").bone[3].worldTransform;
```

Voir aussi

`bone`, `getWorldTransform()`, `transform (propriété)`

wrapTransform

Syntaxe

```
member( whichCastmember ).shader( ShaderName ).wrapTransform
member( whichCastmember ).shader[ ShaderIndex ].wrapTransform
member( whichCastmember ).model[modelName].shader.wrapTransform
member( whichCastmember ).model.shaderlist[ shaderListIndex ].wrapTransform
```

Description

Propriété 3D de matériau standard ; cette propriété donne accès à une transformation qui fait correspondre les coordonnées de texture en fonction de la texture du matériau. Faites pivoter cette transformation pour changer la façon dont la texture est projetée sur la surface d'un modèle. La texture même n'est pas affectée ; la transformation ne modifie que l'orientation de la texture appliquée par le matériau.

Remarque : *Remarque : Cette commande n'a d'effet que lorsque la textureModeList a pour valeur #planar, #spherical ou #cylindrical.*

Exemple

Les instructions suivantes attribuent à la propriété `transformMode` du matériau Mat2 la valeur `#wrapCylindrical`, puis font pivoter cette projection cylindrique de 90 degrés autour de l'axe des x de façon à ce que le placage cylindrique s'enroule autour de l'axe des *y* au lieu de l'axe des *z* :

```
--Lingo syntax
s = member("Scene").shader("shad2")
s.textureMode= #wrapCylindrical
s.wrapTransform.rotate(90.0, 0.0, 0.0)

// JavaScript syntax
var s = member("Scene").getPropRef("shader",1);
s.textureMode = symbol("wrapCylindrical");
s.wrapTransform.rotate(90.0,0.0,0.0);
```

wrapTransformList

Syntaxe

```
member( whichCastmember ).shader( ShaderName ).wrapTransformList[ textureLayerIndex ]
member( whichCastmember ).shader[ shaderListIndex ].wrapTransformList[ textureLayerIndex ]
member( whichCastmember ).model( modelName ).shader.wrapTransformList[ textureLayerIndex ]
member( whichCastmember ).model( modelName ).shaderList[ shaderListIndex ].
wrapTransformList[ textureLayerIndex ]
```

Description

Propriété 3D de matériau standard ; donne accès à une transformation qui modifie les coordonnées de texture d'une couche de texture spécifiée. Faites pivoter cette transformation pour changer la façon dont la texture est projetée sur la surface des modèles. La texture même n'est pas affectée, la transformation ne modifiant que la façon dont le matériau applique la texture.

Remarque : *wrapTransformList[indexDeCoucheDeTexture] n'a d'effet que lorsque textureModeList[`indexDeCoucheDeTexture`] présente la valeur #planar, #spherical ou #cylindrical.*

Exemple

Dans l'exemple suivant, la ligne 2 attribue à la propriété `transformMode` de la troisième couche de texture du matériau Mat2 la valeur `#wrapCylindrical`. La ligne 3 fait pivoter cette projection cylindrique de 90 degrés autour de l'axe des x de façon à ce que le placage cylindrique s'enroule autour de l'axe des *y* au lieu de l'axe des *z*.

```
--Lingo syntax
s = member("Scene").shader("shad2")
s.textureModeList[3] = #wrapCylindrical
s.wrapTransformList[3].rotate(90.0, 0.0, 0.0)

// JavaScript syntax
var s = member("Scene").getPropRef("shader",1);
s.textureModeList[3] = symbol("wrapCylindrical");
s.wrapTransformList[3].rotate(90.0, 0.0, 0.0);
```

Voir aussi

newShader, textureModeList

x (vecteur)

Syntaxe

```
member(whichCastmember).vector.x
member(whichCastmember).vector[1]
```

Description

Propriété 3D ; permet d'obtenir ou de définir le composant *x* d'un vecteur.

Exemple

L'instruction suivante indique le composant *x* d'un vecteur :

```
vec = vector(20, 30, 40)
put vec.x
-- 20.0000
```

xAxis

Syntaxe

```
member(whichCastmember).transform.xAxis
```

Description

Propriété 3D de transformation ; permet d'obtenir, mais non de définir, le vecteur représentant l'axe des *x* canonique de la transformation dans l'espace de transformation.

Exemple

La première ligne de cet exemple définit la transformation du modèle cylindreMod en transformation d'identité. Les deux lignes suivantes indiquent que l'axe des *x* de cylindreMod est le vecteur (1,0000, 0,0000, 0,0000). Cela signifie que l'axe des *x* de cylindreMod est aligné sur l'axe des *x* de l'univers. La ligne suivante fait pivoter cylindreMod de 90 degrés autour de son axe des y. Les axes de cylindreMod sont également pivotés. Les deux dernières lignes indiquent que l'axe des *x* de cylindreMod est à présent vector (0,0000, 0,0000, -1,0000). Cela signifie que l'axe des *x* de cylindreMod est désormais aligné sur l'axe des *z* négatif de l'univers.

```
-- Lingo syntax
member("Engine").model("ModCylinder").transform.identity()
put member("Engine").model("ModCylinder").transform.xAxis
-- vector( 1.0000, 0.0000, 0.0000 )
member("Engine").model("ModCylinder").rotate(0, 90, 0)
put member("Engine").model("ModCylinder").transform.xAxis
-- vector( 0.0000, 0.0000, -1.0000 )

// JavaScript syntax
member("Engine").getPropRef("model",1).transform.identity();
trace(member("Engine").getPropRef("model",1).transform.xAxis)
// vector( 1.0000, 0.0000, 0.0000 )
member("Engine").getPropRef("model",1).rotate(0, 90, 0);
trace(member("Engine").getPropRef("model",1).transform.xAxis)
// vector( 0.0000, 0.0000, -1.0000 )
```

xtra

Syntaxe

```
-- Lingo syntax
_player.xtra[xtraNameOrNum]

// JavaScript syntax
_player.xtra[xtraNameOrNum];
```

Description

Propriété de lecteur ; offre un accès par index ou par nom aux Xtras disponibles pour le lecteur Director. Lecture seule.

L'argument *nomOuNumXtra* peut être une chaîne spécifiant le nom de l'Xtra à laquelle accéder ou un nombre entier indiquant la position d'index de cet Xtra.

Cette propriété a la même fonction que la méthode de haut niveau `xtra()`.

Exemple

L'instruction suivante définit la variable `myXtra` sur l'Xtra Texte en voix :

```
-- Lingo syntax
myXtra = _player.xtra["SpeechXtra"]

// JavaScript syntax
var myXtra = _player.xtra["SpeechXtra"];
```

Voir aussi

Lecteur, xtra()

xtraList (animation)

Syntaxe

```
-- Lingo syntax
_movie.xtraList
```

```
// JavaScript syntax
_movie.xtraList;
```

Description

Propriété d'animation ; affiche une liste linéaire de tous les Xtras ajoutés à l'animation dans la boîte de dialogue Xtras de l'animation. Lecture seule.

Deux propriétés différentes peuvent figurer dans la propriété `xtraList` :

- `#filename` : spécifie le nom de fichier de l'Xtra sur la plate-forme en cours. Il est possible que la liste ne contienne aucune entrée `#filename`, comme lorsque l'Xtra n'existe que sur une seule plate-forme.
- `#packageurl` : spécifie l'emplacement, tel qu'une URL, du package de téléchargement spécifié par `#packagefiles`.
- `#packagefiles` : Ne s'utilise que lorsque l'Xtra sélectionné doit être téléchargé. La valeur de cette propriété est une autre liste contenant la liste de propriétés de chaque fichier du dossier de téléchargement pour la plate-forme utilisée. Les propriétés de cette liste de sous-propriétés sont `#name` et `#version` et contiennent les mêmes informations que la propriété `xtraList (lecteur)`.

Exemple

L'instruction suivante affiche la propriété `xtraList` dans la fenêtre Messages :

```
-- Lingo syntax
put(_movie.xtraList)

// JavaScript syntax
put(_movie.xtraList);
```

Voir aussi

`Animation`, `xtraList (lecteur)`

xtraList (lecteur)

Syntaxe

```
-- Lingo syntax
_player.xtraList

// JavaScript syntax
_player.xtraList;
```

Description

Propriété de lecteur ; affiche une liste linéaire des propriétés de tous les Xtras disponibles et des versions de leurs fichiers. Lecture seule.

Cette propriété est utile lorsque la fonctionnalité d'une animation dépend d'une certaine version d'un Xtra.

Deux types de propriétés peuvent apparaître dans `xtraList` :

- `#filename` : spécifie le nom de fichier de l'Xtra sur la plate-forme en cours. Il est possible que la liste ne contienne aucune entrée `#filename`, comme lorsque l'Xtra n'existe que sur une seule plate-forme.
- `#version` : spécifie le même numéro de version que celui indiqué dans la boîte de dialogue Propriétés (Windows) ou Lire les informations (Mac) lorsque le fichier est sélectionné sur le bureau. Un Xtra ne comporte pas nécessairement de numéro de version.

Exemple

L'instruction suivante affiche dans la fenêtre Messages tous les Xtras disponibles dans le lecteur Director.

```
-- Lingo syntax
trace(_player.xtraList)

// JavaScript syntax
trace(_player.xtraList);
```

Voir aussi

mediaXtraList, Lecteur, scriptingXtraList, toolXtraList, transitionXtraList

y (vecteur)

Syntaxe

```
member(whichCastmember).vector.y
member(whichCastmember).vector[2]
```

Description

Propriété 3D ; permet d'obtenir ou de définir le composant *y* d'un vecteur.

Exemple

L'instruction suivante indique le composant y d'un vecteur :

```
vec = vector(20, 30, 40)
put vec.y
-- 30.0000
```

yAxis

Syntaxe

```
member(whichCastmember).transform.yAxis
```

Description

Propriété 3D de transformation ; permet d'obtenir, mais non de définir, le vecteur représentant l'axe des *y* canonique de la transformation dans l'espace de transformation.

Exemple

La première ligne de cet exemple définit la transformation du modèle cylindreMod en transformation d'identité. Les deux lignes suivantes indiquent que l'axe des y de cylindreMod est `vector(0.0000, 1.0000, 0.0000)`. Cela signifie que l'axe des *y* de cylindreMod est aligné sur l'axe des *y* de l'univers. La ligne suivante fait pivoter cylindreMod de 90 degrés autour de son axe des *x*. Les axes de cylindreMod sont également pivotés. Les deux dernières lignes indiquent que l'axe des *y* de cylindreMod est désormais le vecteur (0,0000, 0,0000, 1,0000). Cela signifie que l'axe des *y*de cylindreMod est désormais aligné sur l'axe des *z* positif de l'univers.

```
member("Engine").model("ModCylinder").transform.identity()
put member("Engine").model("ModCylinder").transform.yAxis
-- vector( 0.0000, 1.0000, 0.0000 )
member("Engine").model("ModCylinder").rotate(90, 0, 0)
put member("Engine").model("ModCylinder").transform.yAxis
-- vector( 0.0000, 0.0000, 1.0000 )
```

yon

Syntaxe

```
member(whichCastmember).camera(whichCamera).yon
```

Description

Propriété 3D ; permet d'obtenir ou de définir la distance, depuis la caméra, définissant l'emplacement de recadrage du frustrum de la vue le long de l'axe des z. Les objets situés à une distance supérieure à celle définie par la propriété `yon` ne sont pas dessinés.

La valeur par défaut de cette propriété est 3,40282346638529e38.

Exemple

L'instruction suivante attribue à la propriété `yon` de la caméra 1 la valeur 50 000 :

```
-- Lingo syntax
member("3d world").camera[1].yon = 50000

// JavaScript syntax
member("3d world").getPropRef("camera",1).yon = 50000
```

Voir aussi

`hither`

z (vecteur)

Syntaxe

```
member(whichCastmember).vector.z
member(whichCastmember).vector[3]
```

Description

Propriété 3D ; permet d'obtenir ou de définir le composant *z* d'un vecteur.

Exemple

L'instruction suivante affiche le composant z d'un vecteur :

```
vec = vector(20, 30, 40)
put vec.z
-- 40.0000
```

zAxis

Syntaxe

```
member(whichCastmember).transform.zAxis
```

Description

Propriété 3D de transformation ; permet d'obtenir, mais non de définir, le vecteur représentant l'axe des *z* canonique de la transformation dans l'espace de transformation.

Exemple

La première ligne de cet exemple définit la transformation du modèle cylindreMod en transformation d'identité. Les deux lignes suivantes indiquent que l'axe des *z* de cylindreMod est le vecteur (0,0000, 0,0000, 1,0000). Cela signifie que l'axe des *z* de cylindreMod est aligné sur l'axe des *z* de l'univers. La ligne suivante fait pivoter cylindreMod de 90 degrés autour de son axe des *y*. Les axes de cylindreMod sont également pivotés. Les deux dernières lignes indiquent que l'axe des *z* de cylindreMod est à présent le vecteur (1,0000, 0,0000, 0,0000). Cela signifie que l'axe des *z* de cylindreMod est désormais aligné sur l'axe des *x* de l'univers.

```
-- Lingo syntax
member("Engine").model("ModCylinder").transform.identity()
put member("Engine").model("ModCylinder").transform.zAxis
-- vector( 1.0000, 0.0000, 0.0000 )
member("Engine").model("ModCylinder").rotate(0, 90, 0)
put member("Engine").model("ModCylinder").transform.zAxis
-- vector( 0.0000, 0.0000, -1.0000 )

// JavaScript syntax
member("Engine").getPropRef("model",1).transform.identity();
trace(member("Engine").getPropRef("model",1).transform.zAxis)
//vector( 1.0000, 0.0000, 0.0000 )
member("Engine").getPropRef("model",1).rotate(0, 90, 0);
trace(member("Engine").getPropRef("model",1).transform.zAxis)
// vector( 0.0000, 0.0000, -1.0000 )
```

Chapitre 15 : Moteur physique

L'Xtra Physics est un outil haute performance conçu pour aider les développeurs à créer des univers 3D dans lesquels les objets interagissent. L'Xtra exécute des calculs pour déterminer les résultats de collisions, en prenant en compte des propriétés d'objet comme la masse, la vitesse et la rotation. Il permet également d'appliquer des forces et de connecter des objets les uns aux autres avec des contraintes. Les contraintes disponibles sont les joints à 6 degrés de liberté, les joints linéaires, les joints angulaires et les joints ressorts.

En outre, les terrains et les raycastings sont pris en charge. Un terrain est semblable à un plan bosselé qui est infini sur deux dimensions et qui définit une élévation le long de la troisième dimension. Le raycasting est le mécanisme de détection des collisions avec des rayons. Il est possible d'exécuter un raycasting avec tous les types de corps rigides et de terrains.

Avec cet outil Xtra, les développeurs peuvent se concentrer sur le jeu et l'interaction utilisateur, sans être obligés de créer un moteur physique en temps réel à l'aide de scripts Lingo.

L'Xtra Physics (dynamique) est un simulateur de physique de corps rigide totalement intégré à Adobe® Director®. L'Xtra dynamique est pris en charge sur les plates-formes Windows et MAC.

Le moteur physique destiné à Director comprend les fonctions suivantes. Pour plus de détails sur ces fonctions, reportez-vous à la suite du chapitre.

Propriétés de l'univers physique

- angularDamping
- contactTolerance
- friction
- gravity
- IsInitialized
- linearDamping
- restitution
- scalingFactor
- sleepMode
- sleepThreshold
- subSteps
- timeStep
- timeStepMode

angularDamping

Syntaxe

```
world.angularDamping
```

Accès : obtention/définition

Type : valeur à virgule flottante

Valeur par défaut : 0,000

Description

Représente la quantité d'amortissement selon laquelle les forces angulaires sont réduites.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").angularDamping = 10.0

//Javascript syntax
member("PhysicsWorld").angularDamping = 10.0;
```

Voir aussi

linearDamping

contactTolerance

Syntaxe

```
world.contactTolerance
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Profondeur de pénétration entre des corps rigides pour permettre la détection de la collision

Remarque : *Pour obtenir le résultat attendu lors d'une simulation de collision entre des corps, ajustez la valeur de tolérance de contact.*

Exemple

L'instruction suivante définit la profondeur de pénétration entre des corps rigides pour permettre la détection de la collision.

```
--Lingo Syntax
member("PhysicsWorld").contactTolerance = 0.01

//JavaScript Syntax
member("PhysicsWorld").contactTolerance = 0.01;
```

Remarque :

- Dans l'idéal, la tolérance de contact doit représenter 2 % des dimensions du corps rigide.
- Si la tolérance de contact de chaque corps rigide n'est pas spécifiée, la valeur définie pour l'univers est utilisée.
- Des valeurs de tolérance de contact supérieures sont sources d'erreurs numériques et d'instabilité.

friction

Syntaxe

```
world.friction
```

Accès : obtention/définition

Type : valeur à virgule flottante

Valeur 0 – 1

Description

Représente la valeur par défaut du coefficient de friction de tous les corps rigides de l'univers. La propriété friction accepte les valeurs comprises entre 0 et 1.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").friction = 0.5

//Javascript syntax
member("PhysicsWorld").friction = 0.5;
```

Voir aussi

restitution

gravity

Syntaxe

```
world.gravity
```

Accès : obtention/définition

Type : vecteur

Description

Représente l'accélération due à la gravité à laquelle sont soumis tous les corps de l'univers physique.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").gravity = vector(0,-9.81,0)

//Javascript syntax
member("PhysicsWorld").gravity = vector(0,-9.81,0);
```

IsInitialized

Syntaxe

```
world.isInitialized
```

Accès : obtention

Type : valeur booléenne

Description

Etat d'initialisation actuel de l'univers. Si la méthode init() est utilisée avec succès sur l'univers, alors cette propriété est définie sur 1.

Exemple

```
--Lingo Syntax
If not member("PhysicsWorld").isInitalized then
-- Initialize the physics world here
end if
```

```
//JavaScript Syntax
If (member("PhysicsWorld").isInitialized == 0) {
    // Initialize World here
}
```

Voir aussi

init()

linearDamping

Syntaxe

```
world.linearDamping
```

Accès : obtention/définition

Type : valeur à virgule flottante

Valeur par défaut : 0,000

Description

Représente la perte de vitesse linéaire à la fin d'une période.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").linearDamping = 10.0
```

```
//Javascript syntax
member("PhysicsWorld").linearDamping = 10.0;
```

Voir aussi

angularDamping

restitution

Syntaxe

```
world.restitution
```

Accès : obtention/définition

Type : valeur à virgule flottante

Valeur 0 – 1

Description

Représente la valeur par défaut du coefficient de restitution de tous les corps rigides de l'univers. La propriété restitution accepte les valeurs comprises entre 0 et 1.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").restitution = 0.5
```

```
//Javascript syntax
member("PhysicsWorld").restitution = 0.5;
```

Voir aussi

friction

scalingFactor

Syntaxe

```
world.scalingFactor
```

Accès : Obtention

Type : vecteur

Description

Représente la mise à l'échelle utilisée pour la conversion si l'univers 3D et l'univers physique utilisent des unités différentes. scalingFactor est un vecteur qui prend la mise à l'échelle pour la longueur, la masse et le temps.

Remarque : *Vous ne pouvez pas définir la valeur de cette propriété dans Director 11. Cette fonctionnalité sera disponible dans les versions ultérieures.*

Exemple

```
--Lingo Syntax
put member("PhysicsWorld").scalingFactor

//Javascript syntax
put(member("PhysicsWorld").scalingFactor)
```

Voir aussi

`init()`

sleepMode

Syntaxe

```
world.sleepMode
```

Accès : obtention/définition

Valeurs : #energy, #linearvelocity

Valeur par défaut : #energy

Description

Permet d'obtenir ou de définir le mode veille pour les corps rigides.

#energy L'énergie cinétique du corps rigide permet de déterminer l'état de veille de ce dernier.

#linearvelocity La vitesse linéaire du corps rigide permet de déterminer l'état de veille de ce dernier.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").sleepMode = #energy

//Javascript syntax
member("PhysicsWorld").sleepMode = symbol("energy");
```

Voir aussi

`sleepThreshold`

sleepThreshold

Syntaxe

```
world.sleepThreshold
```

Accès : obtention/définition

Type : valeur à virgule flottante

Valeur par défaut : 0,000

Description

Représente le seuil au-dessous duquel le corps rigide devient inactif. Selon la valeur définie pour la propriété sleepMode, cette valeur correspond à l'énergie cinétique ou à la vitesse linéaire du corps rigide. Cette propriété s'applique aux objets dynamiques uniquement.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").sleepThreshold = 10.0

//Javascript syntax
member("PhysicsWorld").sleepThreshold = 10.0;
```

Voir aussi

sleepMode

subSteps

Syntaxe

```
world.subSteps
```

Accès : obtention/définition

Type : nombre entier

Description

Représente le nombre de sous-étapes pour chaque appel de simulation.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").subSteps = 5

//Javascript syntax
member("PhysicsWorld").subSteps = 5;
```

Voir aussi

init(),timeStep,timeStepMode

timeStep

Syntaxe

```
world.timeStep
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Cette propriété est utilisée uniquement en mode de période "equal", où elle indique la période de l'univers physique.

Exemple

```
--Lingo Syntax
member("PhysicsWorld"). timeStep = 0.0167

//Javascript syntax
member("PhysicsWorld"). timeStep = 0.0167;
```

Voir aussi

`init()`, `subSteps`, `timeStepMode`

timeStepMode

Syntaxe

```
world.timeStepMode
```

Accès : obtention/définition

Description

Renvoie le mode de période actuel tel que spécifié dans l'appel init() à l'univers. La valeur peut être #automatic ou #equal.

#equal : progresse pas à pas dans l'univers physique selon la période spécifiée dans le paramètre timeStep de l'appel init.

#automatic : progresse pas à pas dans l'univers physique selon le temps réel écoulé.

Exemple

L'instruction suivante définit le mode de période actuel de l'univers physique sur "equal".

```
--Lingo Syntax
member("PhysicsWorld").timeStepMode = #equal

//JavaScript Syntax
member("PhysicsWorld").timeStepMode = symbol("equal");
```

Voir aussi

`init()`, `timeStep`, `subSteps`

Méthodes de l'univers physique

- destroy()
- getSimulationTime()
- init()
- PauseSimulation()
- ResumeSimulation ()
- simulate()

destroy()

Syntaxe

```
<void>world.destroy()
```

Description

Permet d'arrêter la simulation de l'univers et de libérer toutes les ressources de l'univers. Pour redémarrer la simulation, Init() doit être appelé sur l'univers.

Paramètres

Aucune.

Exemple :

```
--Lingo Syntax
member("PhysicsWorld").destroy()

//Javascript syntax
member("PhysicsWorld").destroy();
```

Remarque : *si destroy() n'est pas appelé et que l'animation est relue dans l'environnement auteur de Director, une erreur survient lors de la création de tout objet physique.*

Voir aussi

```
init()
```

getSimulationTime()

Syntaxe

```
<float>world.getSimulationTime()
```

Description

Renvoie le temps de simulation écoulé de l'univers physique.

Paramètres

Aucune.

Exemple

```
--Lingo Syntax
nFloatTime = member("PhysicsWorld").getSimulationTime()

//Javascript syntax
var nFloatTime = member("PhysicsWorld").getSimulationTime();
```

Voir aussi

```
simulate()
```

init()

Syntaxe

```
world.init(string 3dmembername, vector scalingFactor, symbol timeStepMode, float timeStep,
int substepCount).
world.init(3dmemberref, vector scalingFactor, symbol timeStepMode,float timeStep, int
substepCount).
```

Description

Cette fonction initialise l'univers physique avec l'acteur 3D spécifié. Chaque acteur physique doit être associé à un acteur 3D unique représenté par "3dmembername".

- vous devez terminer la fonction init() avec world.destroy() avant d'appeler une autre fonction init().
- n'utilisez pas la méthode resetworld() entre les méthodes init() et destroy() dans une simulation physique.
- il est déconseillé d'appliquer la physique aux modèles avec une animation d'images-clés ou de lecteur de segments.

Remarque : *Une erreur -2 est renvoyée si l'univers n'est pas initialisé avant l'accès à une méthode ou à une propriété d'un objet physique.*

Paramètres

Paramètre	Description
3dmembername	Obligatoire. Référence de chaîne/d'acteur qui spécifie un acteur 3D unique.
ScalingFactor	Obligatoire. Vecteur qui indique le facteur de mise à l'échelle de l'univers 3d dans la physique.
Timestep	Valeur en millisecondes.
Timestepmode	Requis. #equal : progresse pas à pas dans l'univers physique selon la période spécifiée dans le paramètre timeStep de l'appel init. #automatic : progresse pas à pas dans l'univers physique selon le temps réel écoulé.
subStepcount	Obligatoire. Valeur entière qui est calculée pour une meilleure précision.

Exemple :

```
--Lingo Syntax
member("PhysicsWorld").init("3dmembername",vector(1.0,1.0,1.0),#equal,0.33,5)
member("PhysicsWorld").init(member("3dWorld"),vector(1.0,1.0,1.0),#equal,0.33,5)
member("PhysicsWorld").init("3dmembername",vector(1.0,1.0,1.0),#automatic,0,5)
member("PhysicsWorld").init(member("3dWorld"),vector(1.0,1.0,1.0),#automatic,0,5)

//Javascript syntax
member("PhysicsWorld").init("3dmembername",vector(1.0,1.0,1.0),symbol("equal"),0.167,5);
member("PhysicsWorld").init(member("3dWorld"),vector(1.0,1.0,1.0),symbol("equal"),0.33,
5);
member("PhysicsWorld").init("3dmembername",vector(1.0,1.0,1.0),symbol("automatic"),0,5);
member("PhysicsWorld").init(member("3dWorld"),vector(1.0,1.0,1.0),symbol("automatic"),0,
5);
```

Voir aussi

destroy(),subSteps,timeStep,timeStepMode

PauseSimulation()

Syntaxe

```
world.PauseSimulation()
```

Description

Permet d'interrompre momentanément la simulation physique de cet univers physique. L'appel ResumeSimulation () est sans effet jusqu'à ce que l'appel simulate() soit exécuté.

Paramètres

Aucune.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").PauseSimulation()

//Javascript syntax
member("PhysicsWorld").PauseSimulation();
```

Voir aussi

ResumeSimulation ()

ResumeSimulation ()

Syntaxe

```
<void>world.ResumeSimulation ()
```

Description

Permet de reprendre la simulation de l'univers physique qui a été interrompue momentanément par l'appel world.PauseSimulation().

Paramètres

Aucune.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").ResumeSimulation ()

//Javascript syntax
member("PhysicsWorld").ResumeSimulation ();
```

Voir aussi

PauseSimulation()

simulate()

Syntaxe

```
world.simulate()
```

Description

L'appel progresse pas à pas dans l'univers physique selon un certain temps. La période en question dépend de la valeur définie pour timeStepMode lors de l'appel init.

timeStepMode peut avoir les deux valeurs suivantes :

#equal : l'appel simulate() progressera pas à pas dans l'univers physique selon le temps spécifié dans le paramètre timeStep de l'appel init.

#automatic : l'appel simulate() progressera pas à pas dans l'univers physique selon le temps réel écoulé.

• toutes les modifications apportées à l'univers physique ne sont reflétées que lorsque la méthode simulate() est appelée. Par exemple, si vous appliquez une impulsion à un corps rigide à l'aide de la méthode applyimpulse(), la valeur de linearVelocity ne change pas tant que vous n'appelez pas la méthode simulate().

• Il est recommandé d'appeler la méthode simulate() dans chaque exitFrame() pour garantir une simulation adéquate.

Remarque : *Une erreur -3 s'affiche en cas d'échec de l'appel de simulation.*

Paramètres

Aucune.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").simulate()

//Javascript syntax
member("PhysicsWorld").simulate();
```

Voir aussi

`init()`, `subSteps`, `timeStep`, `timeStepMode`

Méthodes de gestion des corps rigides

- createRigidBody()
- deleteRigidBody()
- getRigidBody()
- getRigidBodies()
- getSleepingBodies()

Remarque : *Assurez-vous que le nom du corps rigide est différent de celui du terrain. Si vous utilisez le même nom pour le corps rigide et le terrain, une erreur -4 indiquant un paramètre non valide est renvoyée.*

createRigidBody()

Syntaxe

```
<RigidBody> world.createRigidBody(string rigidbodyname, string
3Dmodelname,symbolBodyProxy,symbol bodyType,symbol flipNormals)
```

Description

Cette méthode crée un corps rigide avec la forme de substitution spécifiée.

Le modificateur #meshdeform doit être ajouté au modèle avant la création d'un corps rigide. Une erreur -21 s'affiche si le modificateur meshdeform n'est pas ajouté au modèle 3D.

Paramètres

Paramètre	Description
3dModelName	Obligatoire. Chaîne qui spécifie le nom du modèle 3d sur lequel le corps rigide est créé.
rigidBodyName	Obligatoire. Chaîne qui spécifie le nom du corps rigide.

symbolBodyProxy	Obligatoire. Symbole qui prend les valeurs suivantes.
	`#box`
	`#Sphere`
	`#convexShape`
	`#concaveShape`

bodyType	Requis. #static : si un corps statique ou immuable doit être créé. ***Remarque :*** Lorsque vous déplacez un corps rigide en modifiant sa position, la position du corps rigide change, mais celle du modèle 3D reste inchangée. #dynamic : si un corps déplaçable doit être créé.
flipNormals	Facultatif. Ce paramètre inverse les normales de faces. Pour les corps qui utilisent une substitution #concaveshape, les collisions sont détectées dans la direction positive des normales de faces. La pénétration est observée du côté opposé. Pour inverser ce comportement, utilisez FlipNormals. Ce paramètre fonctionne uniquement avec les substitutions #convexshape et #concaveshape. Il génère une erreur pour les autres substitutions.

Une référence au corps rigide est renvoyée. Si la création échoue, void est renvoyé.

Remarque : *Un corps rigide qui utilise une substitution #concaveshape doit être #static. Une erreur -28 est renvoyée si vous essayez de créer un corps dynamique concave.*

Exemple

```
--Lingo syntax
objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#sphere,#static)
objRigidBody = member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#box,#dynamic)
objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#convexshape,#dynamic)

objRigidBody
=member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#concaveshape,#static,#flipN
ormals) --The last parameter,#flipNormals,is optional.

//JavaScript Syntax
var objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",symbol("sphere"),symbol("stat
ic"));
var objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",symbol("box"),symbol("dynamic
"));
var objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",symbol("convexshape"),symbol(
"dynamic'));
```

```
var objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",symbol("concaveshape"),symbol
("static"),symbol("flipNormals"));//The last parameter, symbol("flipNormals"),is optional.
```

Voir aussi

deleteRigidBody()

deleteRigidBody()

Syntaxe

```
<void>world.deleteRigidBody(string rigidbodyname)
<void>world.deleteRigidBody(RigidBodyref r)
```

Description

Supprime le corps rigide de l'univers physique.

Remarque : *prenez soin de ne pas supprimer le modèle 3D associé au corps rigide avant de supprimer ce dernier.*

Paramètres

L'un des paramètres suivants est obligatoire.

Paramètre	Description
Rigidbodyname	Chaîne qui spécifie le nom du corps rigide.
Rigibodyref	Référence au corps rigide créé à l'aide de "createrigidbody()"

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#sphere,#static)
member("PhysicsWorld").deleteRigidBody("RigidBodyA")
```

ou

```
objRB = member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#sphere,#static)
member("PhysicsWorld").deleteRigidBody(objRB);
```

```
//Javascript Syntax
var objRB = member("PhysicsWorld").createRigidBody("RigidBodyA",
"ModelA",symbol("sphere"),symbol("static"));
member("PhysicsWorld").deleteRigidBody("RigidBodyA");
```

ou

```
var objRB = member("PhysicsWorld").createRigidBody("RigidBodyA",
"ModelA",symbol("sphere"),symbol("static"));
member("PhysicsWorld").deleteRigidBody(objRB);
```

Voir aussi

createRigidBody()

getRigidBody()

Syntaxe

```
<RigidBody>world.getRigidBody(string rigidBodyName)
```

Description

Renvoie le corps rigide spécifié par rigidbodyname. Renvoie void si le corps rigide avec le nom spécifié n'existe pas.

Paramètres

Paramètre	Description
Rigidbodyname	Obligatoire. Chaîne qui spécifie le nom du corps rigide.

Renvoie la référence au corps rigide.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
```

getRigidBodies()

Syntaxe

```
<list of RigidBodies>world.getRigidBodies()
```

Description

Renvoie la liste des corps rigides actuellement présents dans l'univers physique.

Paramètres

Aucun

Renvoie une liste contenant la référence aux corps rigides de l'univers physique.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBodies()

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBodies();
```

getSleepingBodies()

Syntaxe

```
<list of RigidBodies> world.getSleepingBodies()
```

Description

Renvoie la liste des corps rigides en veille.

Paramètres

Aucun

Renvoie une liste contenant la référence aux corps rigides en veille de l'univers physique.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getSleepingBodies()
```

```
//Javascript Syntax
var objRB = member("PhysicsWorld").getSleepingBodies ();
```

Propriétés des corps rigides

- angularDamping
- angularMomemtum
- angularVelocity
- centerOfMass
- contactTolerance
- friction
- isPinned
- isSleeping
- linearDamping
- linearMomemtum
- linearVelocity
- mass
- model
- name
- orientation
- position
- properties
- restitution
- shape
- sleepMode
- sleepThreshold
- type
- userData

angularDamping

Syntaxe

```
r.angularDamping
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Permet d'obtenir/de définir l'amortissement angulaire du corps. Il s'agit de l'équivalent angulaire de l'amortissement linéaire.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.angularDamping
objRB.angularDamping = 0.5

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.angularDamping);
objRB.angularDamping = 0.5;
```

Voir aussi

linearDamping

angularMomemtum

Syntaxe

```
r.angularMomemtum
```

Accès : obtention/définition

Type : vecteur

Description

Permet d'obtenir/de définir l'impulsion angulaire du corps rigide.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.linearVelocity
objRB.angularMomentum = vector(-100,0,0)

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.linearVelocity);
objRB.angularMomentum =  vector(-100,0,0);
```

Voir aussi

linearMomemtum

angularVelocity

Syntaxe

```
r.angularVelocity
```

Accès : obtention/définition

Type : vecteur

Description

Permet d'obtenir/de définir le vecteur de vitesse angulaire du corps rigide.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
```

```
put objRB. angularVelocity
objRB.angularVelocity = vector(-100,0,0)
```

```
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB. angularVelocity);
objRB.angularVelocity =  = vector(-100,0,0);
```

Voir aussi

linearVelocity

centerOfMass

Syntaxe

```
r.centerOfMass
```

Accès : obtention/définition

Type : vecteur

Description

Représente le centre de la masse du corps rigide dans les coordonnées du corps rigide.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.centerOfMass
```

```
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.centerOfMass);
```

Voir aussi

mass

contactTolerance

Syntaxe

```
r.contacttolerance
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Permet de définir la profondeur d'interpénétration du corps rigide lorsqu'il entre en collision avec un autre corps.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.contactTolerance = 0.01
```

```
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.contactTolerance = 0.01;
```

friction

Syntaxe

```
r.friction
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Permet d'obtenir/de définir le coefficient de friction du corps rigide. La propriété friction accepte les valeurs comprises entre 0 et 1.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.friction
objRB.friction= 0.5
```

```
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody ("RigidBodyA");
put(objRB.friction);
objRB.friction= 0.5;
```

Voir aussi

restitution

isPinned

Syntaxe

```
r.ispinned
```

Accès : obtention/définition

Type : valeur booléenne

Description

Permet d'obtenir/de définir l'ancrage du corps. Si cette propriété est définie sur « true », le corps est ancré à sa position dans son orientation actuelle. Les forces externes ne changent pas la position ou l'orientation du corps.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.isPinned
objRB.isPinned = TRUE
```

```
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.isPinned);
objRB.isPinned = true;
```

isSleeping

Syntaxe

```
r.isSleeping
```

Accès : obtention/définition

Type : valeur booléenne

Description

Renvoie l'état d'un corps rigide comme étant en veille ou définit l'état d'un corps rigide sur veille.

Remarque : *L'état de repos d'un corps rigide dépend de sa forme. Par exemple, la mise en veille d'un corps rigide créé à l'aide d'une substitution de boîte prend davantage de temps que celle d'une sphère.*

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.isSleeping

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody ("RigidBodyA");
put(objRB. isSleeping)
```

linearDamping

Syntaxe

```
r.linearDamping
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Permet d'obtenir/de définir la valeur actuelle d'amortissement linéaire du corps. Il s'agit de la quantité de ralentissement du corps alors qu'il se déplace de manière linéaire dans l'univers.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.linearDamping
objRB.linearDamping = 0.5

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.linearDamping);
objRB.linearDamping = 0.5;
```

Voir aussi

angularDamping

linearMomemtum

Syntaxe

```
r.linearMomemtum
```

Accès : obtention/définition

Type : vecteur

Description

Permet d'obtenir/de définir la quantité de mouvement linéaire du corps rigide.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.linearMomentum
objRB.linearMomentum = vector(-100,0,0)
```

```
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.linearMomentum);
objRB.linearMomentum =  vector(-100,0,0);
```

Voir aussi

angularMomemtum

linearVelocity

Syntaxe

```
r.linearVelocity
```

Accès : obtention/définition

Type : vecteur

Description

Permet d'obtenir/de définir le vecteur de vitesse linéaire du corps rigide.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.linearVelocity
objRB.linearVelocity = vector(-100,0,0)
```

```
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB. linearVelocity);
objRB.linearVelocity = vector(-100,0,0);
```

Voir aussi

angularVelocity

mass

Syntaxe

```
r.mass
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Permet de définir ou d'obtenir la masse du corps rigide. Par défaut, la valeur de la masse est zéro. Définissez une valeur supérieure à zéro pour cette propriété. Si vous ne spécifiez aucune valeur, la simulation comporte des inexactitudes.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.mass
objRB.mass = objRB.mass + 20

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.mass);
objRB.mass = objRB.mass + 20;
```

Voir aussi

centerOfMass

model

Syntaxe

```
r.model
```

Accès : Obtention

Description

Renvoie le modèle 3D associé à ce corps rigide.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objModel = objRB.model
put objModel.name

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
var objModel = objRB.model
put(objModel.name)
```

name

Syntaxe

```
r.name
```

Accès : Obtention

Type : chaîne

Description

Renvoie le nom du corps rigide.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.name

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.name);
```

orientation

Syntaxe

```
r.orientation
```

Accès : obtention/définition

Type : list

Description

Permet d'obtenir/de définir l'orientation du corps rigide dans l'univers physique. La propriété orientation est une liste avec le vecteur de direction et l'angle entre les axes du corps rigide et l'axe de l'univers.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.orientation = [vector(10,0,0),45]

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.orientation = list(vector(10,0,0),45);
```

Remarque : *le fait de changer la position ou l'orientation d'un corps rigide statique ne change pas la position du modèle. La position ou l'orientation du modèle doit être changée explicitement.*

position

Syntaxe

```
r.position
```

Accès : obtention/définition

Type : vecteur

Description

Permet d'obtenir/de définir la position du corps rigide dans l'univers physique. La position représente un vecteur (x,y,z) dans les coordonnées de l'univers.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.position = vector(0,0,0)

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.position = vector(0,0,0);
```

Remarque : *le fait de changer la position ou l'orientation d'un corps rigide statique ne change pas la position du modèle. La position ou l'orientation du modèle doit être changée explicitement.*

properties

Syntaxe

```
r.properties
```

Accès : Obtention

Type : liste de propriétés

Description

Renvoie une liste de propriétés qui représente les propriétés de ce corps rigide. Par exemple, le rayon est renvoyé dans le cas d'une sphère et, dans le cas d'une boîte, la liste des propriétés contient les dimensions de la boîte. Cette propriété renvoie une liste de propriétés vide pour #concaveshape et #convexshape.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.properties

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.properties);
```

restitution

Syntaxe

```
r.restitution
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Permet d'obtenir/de définir le coefficient de restitution du corps rigide. La propriété restitution accepte les valeurs comprises entre 0 et 1.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.restitution
objRB.restitution = 0.5

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.restitution);
objRB.restitution = 0.5;
```

shape

Syntaxe

```
r.shape
```

Accès : Obtention

Valeurs : #sphere, #box, #convexshape ou #concaveshape

Description

Renvoie la forme du corps rigide.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
 put objRB.shape

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.shape);
```

sleepMode

Syntaxe

```
r.sleepMode
```

Accès : obtention/définition

Description

Permet d'obtenir ou de définir le mode veille d'un corps rigide. La valeur peut être #energy ou #linearvelocity. La valeur par défaut est #energy.

#energy L'énergie cinétique du corps rigide permet de déterminer l'état de veille de ce dernier.

#linearvelocity La vitesse linéaire du corps rigide permet de déterminer l'état de veille de ce dernier.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.sleepMode = #energy

//Javascript syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.sleepMode = symbol("energy");
```

Voir aussi

sleepThreshold

sleepThreshold

Syntaxe

```
r.sleepThreshold
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Représente le seuil au-dessous duquel le corps rigide devient inactif. Selon la valeur définie pour la propriété sleepMode, cette valeur correspond à l'énergie cinétique ou à la vitesse linéaire du corps rigide. Cette propriété s'applique aux objets dynamiques uniquement. La valeur par défaut est 0,000.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.sleepMode = #energy
objRB.sleepThreshold = 10.0
```

```
//Javascript syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.sleepMode = symbol("energy");
objRB.sleepThreshold = 10.0;
```

Voir aussi

sleepMode

type

Syntaxe

```
r.type
```

Accès : Obtention

Description

Renvoie le type du corps rigide. La valeur renvoyée peut être #static ou #dynamic.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.type
```

```
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody ("RigidBodyA");
put(objRB.type);
```

userData

Syntaxe

```
r.userData
```

Accès : obtention/définition

Type : objet

Description

Permet à l'utilisateur d'obtenir/de définir toutes les données définies par l'utilisateur et de les associer à ce corps rigide.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
--Assigning Integer Value
objRB.userData = 0.5
--Assigning String Value
objRB.userData = "My User data Value"
--Assigning Reference Value
objRB.userData = member("PhysicsWorld").getRigidBody("RigidBodyB")
```

```
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
--Assigning Integer Value
objRB.userData = 0.5;
--Assigning String Value
objRB.userData = "My User data Value";
--Assigning Reference Value
objRB.userData = member("PhysicsWorld").getRigidBody("RigidBodyB");
```

Méthodes des corps rigides

applyForce()

Syntaxe

```
<int>r.applyForce(vector vForce, vector vposition)
```

Description

Applique la force donnée à la position spécifiée par la position du vecteur dans les coordonnées du corps rigide. Si la position n'est pas spécifiée, alors la force est appliquée au centre de la masse du corps rigide.

Paramètres

Paramètre	Description
vForce	Requis. Valeur vectorielle qui spécifie la magnitude et la direction de la force appliquée.
vPosition	Facultatif. Vecteur qui spécifie le point où la force est appliquée.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.applyForce(vector(-100,0,0),vector(0,0,0))
```

```
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.applyForce(vector(-100,0,0),vector(0,0,0));
```

applyLinearImpulse()

Syntaxe

```
< int >r.applyLinearImpulse(vector vImpulse, vector vposition)
```

Description

Applique l'impulsion linéaire donnée à la position spécifiée par la position du vecteur dans les coordonnées du corps rigide. Si la position n'est pas spécifiée, alors l'impulsion est appliquée au centre de la masse du corps rigide.

Paramètres

Paramètre	Description
vImpulse	Requis. Valeur vectorielle qui spécifie la magnitude et la direction de la force appliquée.
vPosition	Facultatif. Vecteur qui spécifie le point où la force est appliquée.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.applyLinearImpulse(vector(-100,0,0),vector(0,0,0))

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.applyLinearImpulse(vector(-100,0,0),vector(0,0,0));
```

applyAngularImpulse()

Syntaxe

```
< int >r.applyAngularImpulse(vector vImpulse)
```

Description :

Applique l'impulsion angulaire donnée au centre de la masse.

Paramètres

Paramètre	Description
vImpulse	Obligatoire. Vecteur qui spécifie l'impulsion angulaire appliquée.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.applyAngularImpulse(vector(-100,100,0))

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.applyAngularImpulse (vector(-100,0,0));
```

applyTorque()

Syntaxe

```
< int >r.applyTorque(vector vTorque)
```

Description

Applique le couple spécifié au corps rigide au centre de sa masse.

Paramètres

Paramètre	Description
vTorque	Obligatoire. Vecteur qui spécifie le couple appliqué.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.applyTorque(vector(-100,100,0))

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.applyTorque(vector(-100,0,0));
```

attemptMoveTo()

Syntaxe

```
< int >r.attemptMoveTo(vector vposition)
```

Description

Cet appel tente de déplacer le corps rigide jusqu'à la position spécifiée s'il n'y a pas de collision. Si une collision est détectée à la position spécifiée, alors l'objet n'est pas déplacé du tout.

Paramètres

Paramètre	Description
vPosition	Obligatoire. Vecteur qui spécifie le point où l'objet doit être déplacé.

Exemple

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.attemptMoveTo(vector(-100,0,0))

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody ("RigidBodyA");
objRB.atemptMoveTo(vector(-100,0,0));
```

Méthodes des contraintes

- ConstraintDesc
- createAngularJoint()
- createLinearJoint()
- createSpring()
- deleteConstraint()
- deleteSpring()
- getConstraint()
- getConstraints()
- getSpring()
- getSprings()

ConstraintDesc

Syntaxe

```
ConstraintDesc(sname,ObjectA,ObjectB,vPointA,vPointB,fStiffness,fDamping)
```

vPointA et vPointB doivent se situer sur les corps rigides ou à l'intérieur de ces derniers. Ils sont spécifiés dans les coordonnées de l'objet. Le comportement n'est pas défini si les points ne sont pas situés sur les corps.

Description

Un objet descripteur de contrainte doit être créé avant la création d'un ressort, d'un joint angulaire ou d'un joint linéaire.

Remarque : *Une erreur -5 s'affiche lorsque vous tentez de créer une fonction ConstraintDesc portant un nom déjà existant.*

Paramètres

Toutes les contraintes sont créées à l'aide d'un descripteur de contrainte. Ce descripteur présente les contenus suivants :

Paramètre	Description
string Name	Nom de la contrainte
ObjectA	Corps rigide A auquel la contrainte est connectée.
ObjectB	Corps rigide B auquel la contrainte est connectée.
vector PoCA	Point auquel la contrainte est attachée au corps rigide A.
vector PoCB	Point auquel la contrainte est attachée au corps rigide B.
float Stiffness	Valeur de la rigidité de la contrainte
float damping	Valeur de l'amortissement de la contrainte

Exemple

```
--Lingo Syntax
-- Spring constraint Descriptor between two rigid bodies' b1 and b2.
SpringDesc =
ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
-- Linear Joint constraint Descriptor between a rigid body b1 and point in world.
LJointDesc =
ConstraintDesc("LJoinDesc",b1,void,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)

//JavaScript Syntax
//Spring constraint Descriptor between two rigid bodies' b1 and b2.
var SpringDesc =
ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
```

Remarque :

- pour créer une contrainte entre un point dans l'univers et un corps rigide, définissez la valeur de ObjectB sur VOID.
- Il est impossible de créer un descripteur de contrainte entre un point et un corps rigide statique, entre deux corps rigides statiques ou entre deux-points dans l'univers.

Voir aussi

createAngularJoint(),createLinearJoint(),createSpring(),createD6Joint

createAngularJoint()

Syntaxe

```
<AngularJoint> world.createAngularJoint(ConstraintDesc desc , float length)
```

Description

Crée un joint angulaire. Un joint angulaire peut se déplacer librement dans la direction angulaire. Il est limité dans la direction linéaire. La longueur à garder limitée est spécifiée dans le second paramètre.

Paramètres

Paramètre	Description
ConstraintDesc	Requis. Référence à l'objet descripteur de la contrainte.
restlength	Obligatoire. Valeur à virgule flottante qui spécifie la longueur de repos du joint.

Exemple

```
--Lingo Syntax
-- Angular Joint constraint Descriptor between two rigid bodies' b1 and b2.
AJointDesc =
ConstraintDesc("AJointDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
--Create a angular joint with a rest length of 5.
member("PhysicsWorld").createAngularJoint(AJointDesc, 5.0)
```

```
//JavaScript Syntax
//Angular Joint constraint Descriptor between two rigid bodies' b1 and b2.
var AJointDesc =
ConstraintDesc("AJointDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
//Create a angular Joint with a rest length of 5.
member("PhysicsWorld").createAngularJoint (AJointDesc,5.0);
```

Voir aussi

ConstraintDesc,createLinearJoint(),createSpring(),createD6Joint

createLinearJoint()

Syntaxe

```
<LinearJoint> world.createLinearJoint(ConstraintDesc desc , list orientation)
```

Description :

Crée un joint linéaire. Un joint linéaire peut se déplacer librement dans la direction linéaire. Il est limité dans la direction angulaire. Les angles à garder limités sont spécifiés dans le second paramètre.

Paramètres

Paramètre	Description
ConstraintDesc	Requis. Référence à l'objet descripteur de la contrainte.
Orientation	Requis. Valeur de liste spécifiant l'orientation du joint.

Exemple

```
--Lingo Syntax
-- Angular Joint constraint Descriptor between two rigid bodies' b1 and b2.
LJointDesc =
ConstraintDesc("LJointDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
```

```
--Create a linear joint with angular constraint in x,y axis at an angle of 45.
member("PhysicsWorld").createLinearJoint(LJointDesc,[vector(1,1,0),45])

//JavaScript Syntax
//Linear Joint constraint Descriptor between two rigid bodies' b1 and b2.
var LJointDesc =
ConstraintDesc("LJointDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
//Create a linear joint with angular constraint in x,y axis at an angle of 45.
member("PhysicsWorld").createLinearJoint (LJointDesc, list(vector(1,1,0),45));
```

Voir aussi

createAngularJoint(),ConstraintDesc,createSpring(),createD6Joint

createSpring()

Syntaxe

```
<Spring> world.createSpring( ConstraintDesc desc, symbol forceExertionMode , float
restLength)
```

Description

Crée un ressort avec les détails spécifiés dans le descripteur de la contrainte.

Paramètres

Paramètre	Description
ConstraintDesc	Requis. Référence à l'objet descripteur de la contrainte.
forceexertionmode	Prend les valeurs de symbole suivantes : #kDuringCompression : une force de rétablissement est appliquée uniquement lorsque la longueur réelle est inférieure à la longueur de repos entre objectA et objectB. #kDuringExpansion : une force de rétablissement est appliquée uniquement lorsque la longueur réelle est supérieure à la longueur de repos entre objectA et objectB. #kBoth : une force de rétablissement est appliquée dans les deux cas ci-dessus.
restlength	Obligatoire. Valeur à virgule flottante qui spécifie la longueur de repos du ressort.

Exemple

```
--Lingo Syntax
-- Spring constraint Descriptor between two rigid bodies' b1 and b2.
SpringDesc =
ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
--Create a spring which has expansion and compression with a rest length of 5.
member("PhysicsWorld").createSpring(SpringDesc,#kboth,5.0)

//JavaScript Syntax
//Spring constraint Descriptor between two rigid bodies' b1 and b2.
SpringDesc =
ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
//Create a spring which has expansion and compression with a rest length of 5.
member("PhysicsWorld").createSpring(SpringDesc,symbol("kboth"),5.0);
```

Voir aussi

createAngularJoint(),createLinearJoint(),ConstraintDesc,createD6Joint

deleteConstraint()

Syntaxe

```
<int>deleteConstraint(string constraintname)
<int>deleteConstraint(Constraint constraintref)
```

Description

Supprime la contrainte spécifiée en utilisant le nom de la contrainte.

Paramètres

Paramètre	Description
ConstraintName	Obligatoire. Nom de chaîne de la contrainte.

ou

Paramètre	Description
Constraintref	Obligatoire. Référence à l'objet contrainte.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").deleteConstraint("ConstraintDesc")

//JavaScript Syntax
member("PhysicsWorld").deleteConstraint("ConstraintDesc");
```

deleteSpring()

Syntaxe

```
<int>deleteSpring(string springname)
<int>deleteSpring(Spring spring)
```

Description

Supprime la chaîne spécifiée en utilisant le nom du ressort.

Paramètres

Paramètre	Description
Springname	Requis. Valeur de chaîne qui spécifie le nom du ressort créé.

Exemple

```
--Lingo Syntax
-- Spring constraint descriptor between two rigid bodies' b1 and b2.
SpringDesc =
ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
--Create a spring which has expansion and compression with a rest length of 5.
member("PhysicsWorld").createSpring(SpringDesc,#kboth,5.0)
--Deletes the spring
member("PhysicsWorld").deleteSpring("SpringDesc")

//JavaScript Syntax
//Spring constraint descriptor between two rigid bodies' b1 and b2.
SpringDesc =
ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
```

```
//Create a spring which has expansion and compression with a rest length of 5.
member("PhysicsWorld").createSpring(SpringDesc,symbol("kboth"),5.0);
//Deletes the spring
member("PhysicsWorld").deleteSpring("SpringDesc");
```

getConstraint()

Syntaxe

```
<AngularJoint / LinearJoint>getConstraint(string constraintname)
```

Description

Renvoie la contrainte portant le nom spécifié.

Paramètres

Paramètre	Description
ConstraintName	Obligatoire. Nom de chaîne de la contrainte.

Renvoie la référence au joint.

Exemple

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("ConstraintDesc")

//JavaScript Syntax
var objJoint = member("PhysicsWorld").getConstraint("ConstraintDesc");
```

Voir aussi

ConstraintDesc

getConstraints()

Syntaxe

```
<list of Constraints> world.getConstraints(symbol constraintType)
```

Description

Renvoie toutes les contraintes du type spécifié. La valeur peut être #linearjoint, #angularjoint ou #d6joint.

Paramètres

Paramètre	Description
constraintType	Facultatif. Symbole qui spécifie le type de contrainte. S'il n'est pas indiqué, cette fonction renvoie toutes les contraintes, à l'exception des ressorts, dans l'univers physique.

Exemple

```
--Lingo Syntax
--Returns all the constraints in the world.
lstJoint = member("PhysicsWorld").getConstraints()
--Returns all the Linear joint constraints in the world.
lstJoint = member("PhysicsWorld").getConstraints(#linearjoint)

//JavaScript Syntax
var lstJoint = member("PhysicsWorld").getConstraints(symbol("angularjoint"));
```

Voir aussi

ConstraintDesc

getSpring()

Syntaxe

```
<Spring > world.getSpring(string springname)
```

Description

Renvoie l'objet ressort portant le nom spécifié.

Paramètres

Paramètre	Description
Springname	Requis. Valeur de chaîne qui spécifie le nom du ressort créé.

Exemple

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringName")

//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringName");
```

Voir aussi

ConstraintDesc

getSprings()

Syntaxe

```
<list of Springs> world.getSprings();
```

Description

Renvoie la liste de tous les ressorts présents dans l'univers physique.

Fonctions de collision : par défaut, les collisions entre tous les corps dans l'univers sont activées. Ces fonctions peuvent être utilisées pour activer/désactiver les collisions entre des paires de corps.

Paramètres

Aucune.

Exemple

```
--Lingo Syntax
lSprings= member("PhysicsWorld").getSprings()

//JavaScript Syntax
var lSprings = member("PhysicsWorld").getSprings();
```

Propriétés des contraintes

- constraintType
- damping

- name
- objectA
- objectB
- pointA
- pointB
- properties
- stiffness

constraintType

Syntaxe

```
c.constraintType
```

Accès : Obtention

Description

Renvoie le type de la contrainte. La valeur peut être #linearjoint, #angularjoint ou #6djoint.

Exemple

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.constraintType
```

```
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.constraintType);
```

damping

Syntaxe

```
c.damping
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Renvoie l'amortissement de la contrainte.

Exemple

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getSpring("lJoint01")
put objJoint.damping
```

```
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getSpring("lJoint01");
put(objJoint.damping);
```

name

Syntaxe

```
c.name
```

Accès : Obtention

Type : chaîne

Description

Renvoie le nom de la contrainte.

Exemple

```
--Lingo Syntax
objConstraint = member("PhysicsWorld").getConstraint("ConstraintA")
put objConstraint.name

//Javascript Syntax
var objConstraint = member("PhysicsWorld").getConstraint("ConstraintA");
put(objConstraint.name);
```

objectA

Syntaxe

```
s.objectA
```

Accès : Obtention

Type : RigidBody

Description

Renvoie le nom du corps rigide « A » attaché au joint.

Exemple

```
--Lingo Syntax
--Change one end of the Joint to already created rigidbody - rigidbodyX
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.objectA

//JavaScript Syntax.
--Change one end of the Joint to already created rigidbody - rigidbodyX
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.objectA);
```

objectB

Syntaxe

```
s.objectB
```

Accès : Obtention

Type : RigidBody

Description

Renvoie le nom du corps rigide « B » attaché au ressort.

Exemple

```
--Lingo Syntax
--Change one end of the Joint to already created rigidbody - rigidbodyX
objJoint = member("PhysicsWorld").getConstraint ("lJoint01")
put objJoint.objectB
--Change one end of the Joint to a worldPoint
```

```
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.objectB

//JavaScript Syntax.
--Change one end of the joint to already created rigidbody - rigidbodyX
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.objectB);
```

pointA

Syntaxe

```
s.pointA
```

Accès : Obtention

Type : vecteur

Description

Renvoie le point auquel le joint est attaché à l'objet A.

Exemple

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.pointA

//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.pointA);
```

pointB

Syntaxe

```
s.pointB
```

Accès : Obtention

Type : vecteur

Description

Renvoie le point auquel le joint est attaché à l'objet B.

Exemple

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.pointB

//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.pointB);
```

properties

Syntaxe

```
c.properties
```

Accès : obtention/définition

Type : liste de propriétés

Description

Renvoie une liste de propriétés qui représente les propriétés du joint. Par exemple, le joint linéaire renvoie l'orientation et le joint angulaire renvoie la longueur.

Exemple

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.properties
```

```
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.properties);
```

stiffness

Syntaxe

```
s.stiffness
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Renvoie la rigidité de la contrainte.

Exemple

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getSpring("lJoint01")
put objJoint.stiffness
```

```
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getSpring("lJoint01");
put(objJoint.stiffness);
```

Propriétés des ressorts

- damping
- flags
- name
- objectA
- objectB
- pointA
- pointB
- restLength
- stiffness

damping

Syntaxe

```
s.damping
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Représente l'amortissement du ressort.

Exemple

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringDesc")
put objSpring.damping
```

```
//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringDesc");
put(objSpring.damping);
```

flags

Syntaxe

```
s.flags
```

Accès : obtention/définition

Description

Représente les forces de compression/d'expansion appliquées au ressort. Renvoie #kDuringCompression, #kDuringExpansion ou #kBoth.

Exemple

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringDesc")
put objSpring.flags
```

```
//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringDesc");
put(objSpring.flags);
```

name

Syntaxe

```
s.name
```

Accès : Obtention

Type : chaîne

Description

Renvoie le nom du ressort.

Exemple

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringA")
put objSpring.name

//Javascript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringA");
put(objSpring.name);
```

objectA

Syntaxe

```
s.objectA
```

Accès : obtention/définition

Type : RigidBody

Description

Renvoie le nom du corps rigide « A » attaché au ressort.

Exemple

```
--Lingo Syntax
--Change one end of the spring to already created rigidbody - rigidbodyX
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.objectA
objSpring.objectA = rigidbodyX
--Change one end of the spring to a worldPoint
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.objectA
objSpring.objectA = void

//JavaScript Syntax.
--Change one end of the spring to already created rigidbody - rigidbodyX
objSpring = member("PhysicsWorld").getSpring("Spring01");
put(objSpring.objectA);
objSpring.objectA = rigidbodyX;
```

objectB

Syntaxe

```
s.objectB
```

Accès : obtention/définition

Type : RigidBody

Description

Renvoie le nom du corps rigide « B » attaché au ressort.

Exemple

```
--Lingo Syntax
--Change one end of the spring to already created rigidbody - rigidbodyX
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.objectB
objSpring.objectB = rigidbodyX
```

```
--Change one end of the spring to a worldPoint
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.objectB
objSpring.objectB = void

//JavaScript Syntax.
--Change one end of the spring to already created rigidbody - rigidbodyX
objSpring = member("PhysicsWorld").getSpring("Spring01");
put(objSpring.objectB);
objSpring.objectB = rigidbodyX;
```

pointA

Syntaxe

```
s.pointA
```

Accès : obtention/définition

Type : vecteur

Description

Renvoie le point auquel le ressort est attaché à l'objet A.

Exemple

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.pointA

//JavaScript Syntax.
objSpring = member("PhysicsWorld").getSpring("Spring01");
put(objSpring.pointA);
```

pointB

Syntaxe

```
s.pointB
```

Accès : obtention/définition

Type : vecteur

Description

Renvoie le point auquel le ressort est attaché à l'objet B.

Exemple

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.pointB

//JavaScript Syntax.
objSpring = member("PhysicsWorld").getSpring("Spring01");
put(objSpring.pointB);
```

restLength

Syntaxe

```
s.restLength
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Représente la longueur du ressort au repos.

Exemple

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringDesc")
put objSpring.restLength

//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringDesc");
put(objSpring.restLength);
```

stiffness

Syntaxe

```
s.stiffNess
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Représente la rigidité du ressort.

Exemple

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringDesc")
put objSpring.stiffness

//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringDesc");
put(objSpring.stiffness);
```

Méthodes de collision et de rappel de collision

- disableCollision()
- disableCollisionCallback()
- enableCollision()
- enableCollisionCallback()
- getCollisionCallbackDisabledPairs()
- getCollisionDisabledPairs()
- registerCollisionCallback()
- removeCollisionCallback()

disableCollision()

Syntaxe

```
<int>world .disableCollision(RigidBody a, RigidBody b)
<int>world .disableCollision(string rigidbodyname1, string rigidbodyname2)
```

Description

Si A et B ont la valeur void, la collision entre tous les corps rigides de l'univers est désactivée.

Si B a la valeur void, la collision du corps A avec tous les corps est désactivée.

Si A et B sont spécifiés, la collision entre ces deux corps est désactivée si elle avait été activée précédemment.

Remarque : *disableCollision( ) désactive tous les rappels. Toutefois, le fait d'utiliser enableCollision( ) n'active pas automatiquement les rappels. Vous devez appeler enableCollisionCallback( ) séparément pour activer les rappels.*

Paramètres

Dépend du nombre de corps rigides pour lesquels la collision doit être désactivée.

Exemple

```
--Lingo Syntax
--All collisions are disabled.
member("PhysicsWorld").disableCollision()
--Collisions disabled for rigid body A
member("PhysicsWorld").disableCollision(rigidbodyA)
--Collisions disabled for rigidbody A and rigidbody B
member("PhysicsWorld").disableCollision(rigidbodyA,rigidbodyB)
```

```
//JavaScript Syntax
//All collisions are disabled.
member("PhysicsWorld").disableCollision();
//Collisions disabled for rigid body A
member("PhysicsWorld").disableCollision(rigidbodyA);
//Collisions disabled for rigidbody A and rigidbody B
member("PhysicsWorld").disableCollision(rigidbodyA,rigidbodyB);
```

Voir aussi

enableCollision()

disableCollisionCallback()

Syntaxe

```
<int>world .disableCollisionCallback(RigidBody a, RigidBody b)
<int>world . disableCollisionCallback (string rigidbodyname1, string rigidbodyname2)
```

Description

Si aucun paramètre n'est transmis, les rapports de rappel de collision pour tous les corps rigides créés avant cet appel sont désactivés.

Si des paramètres sont transmis à cette méthode, les rappels de collision sont désactivés uniquement pour le ou les corps rigides spécifiés.

Paramètres

Dépend du nombre de corps rigides pour lesquels les rappels de collision doivent être désactivés.

Exemple

```
--Lingo Syntax
--All collisions callbacks are disabled for rigid bodies created before this call.
member("PhysicsWorld").disableCollisionCallback()
--Collisions callback disabled for rigid body A with all other bodies.
member("PhysicsWorld").disableCollisionCallback(rigidbodyA)
--Collisions  callback disabled between rigidbody A and rigidbody B.
member("PhysicsWorld").disableCollisionCallback(rigidbodyA,rigidbodyB)

//JavaScript Syntax
//All collision callbacks are disabled.
member("PhysicsWorld").disableCollisionCallback();
//Collision callbacks disabled for rigid body A
member("PhysicsWorld").disableCollisionCallback(rigidbodyA);
//Collision callbacks disabled between rigidbody A and rigidbody B
member("PhysicsWorld").disableCollisionCallback(rigidbodyA,rigidbodyB);
```

Voir aussi

enableCollisionCallback(),getCollisionCallbackDisabledPairs(),getCollisionDisabledPairs(), registerCollisionCallback(),removeCollisionCallback()

enableCollision()

Syntaxe

```
<int>world .enableCollision(RigidBody a, RigidBody b)
<int>world .enableCollision(string rigidbodyname1, string rigidbodyname2)
```

Description

Si aucun paramètre n'est transmis, toutes les collisions sont activées. Sinon, les collisions sont activées uniquement pour le ou les corps rigides spécifiés.

Paramètres

Dépend du nombre de corps rigides pour lesquels la collision doit être activée.

Exemple

```
--Lingo Syntax
--All collisions are enabled.
member("PhysicsWorld").enableCollision()
--Collisions enabled for rigid body A with all other bodies.
member("PhysicsWorld").enableCollision(rigidbodyA)
--Collisions enabled between rigidbody A and rigidbody B.
member("PhysicsWorld").enableCollision(rigidbodyA,rigidbodyB)

//JavaScript Syntax
//All collisions are enabled.
member("PhysicsWorld").enableCollision();
//Collisions enabled for rigid body A
member("PhysicsWorld").enableCollision(rigidbodyA);
//Collisions enabled for rigidbody A and rigidbody B
member("PhysicsWorld").enableCollision(rigidbodyA,rigidbodyB);
```

Voir aussi

disableCollision()

enableCollisionCallback()

Syntaxe

```
<int>world .enableCollisionCallback(RigidBody a, RigidBody b)
<int>world . enableCollisionCallback(string rigidbodyname1, string rigidbodyname2)
```

Description

Si aucun paramètre n'est transmis, les rapports de rappel de collision pour tous les corps rigides créés avant cet appel sont activés.

Si des paramètres sont transmis à cette méthode, les rappels de collision sont activés uniquement pour le ou les corps rigides spécifiés.

Paramètres

Dépend du nombre de corps rigides pour lesquels les rappels de collision doivent être activés.

Exemple

```
--Lingo Syntax
--All collisions callbacks are enabled for rigid bodies created before this call.
member("PhysicsWorld").enableCollisioncallback()
--Collisions callback enabled for rigid body A with all other bodies.
member("PhysicsWorld").enableCollisioncallback(rigidbodyA)
--Collisions  callback enabled between rigidbody A and rigidbody B.
member("PhysicsWorld").enableCollisioncallback(rigidbodyA,rigidbodyB)

//JavaScript Syntax
//All collision callbacks are enabled.
member("PhysicsWorld").enableCollisioncallback();
//Collision callbacks enabled for rigid body A
member("PhysicsWorld").enableCollisioncallback(rigidbodyA);
//Collision callbacks enabled for rigidbody A and rigidbody B
member("PhysicsWorld").enableCollisioncallback(rigidbodyA,rigidbodyB);
```

Voir aussi

disableCollisionCallback(),getCollisionDisabledPairs(),getCollisionCallbackDisabledPairs()
,registerCollisionCallback(),removeCollisionCallback()

getCollisionCallbackDisabledPairs()

Syntaxe

```
<[[RigidBody,RigidBody]…]>world .getCollisionCallbackDisabledPairs()
```

Description

Renvoie la liste de toutes les paires de corps rigides pour lesquelles les appels de collision ont été désactivés.

Paramètres

Aucune.

Exemple

```
--Lingo Syntax
lstPairs = member("PhysicsWorld").getCollisionCallbackDisabledPairs()
lstPair1 = lstPairs[1]

//JavaScript Syntax
var lstPairs = member("PhysicsWorld").getCollisionCallbackDisabledPairs();
var lstPair1 = lstPairs(0);
```

Voir aussi

enableCollisionCallback(),disableCollisionCallback(),getCollisionDisabledPairs(),registerCollisionCallback(),removeCollisionCallback()

getCollisionDisabledPairs()

Syntaxe

```
<[[RigidBody,RigidBody]…]>world .getCollisionDisabledPairs()
```

Description

Renvoie la liste de toutes les paires de corps rigides pour lesquelles la collision a été désactivée.

Paramètres

Aucune.

Exemple

```
--Lingo Syntax
lstPairs = member("PhysicsWorld").getCollisionDisabledPairs()
lstPair1 = lstPairs[1]

//JavaScript Syntax
var lstPairs = member("PhysicsWorld").getCollisionDisabledPairs();
var lstPair1 = lstPairs(0);
```

Voir aussi

enableCollisionCallback(),disableCollisionCallback(),getCollisionCallbackDisabledPairs(),registerCollisionCallback(),removeCollisionCallback()

registerCollisionCallback()

Syntaxe

```
<int>world .registerCollisionCallback( #collisionCallback,<script reference>)
```

Description

Cette fonction enregistre la fonction de rappel devant être appelée pour signaler les collisions. Il ne peut y avoir qu'un seul gestionnaire de rappel de collision pour la scène physique. Les informations relatives aux collisions sont transmises à cette fonction dans une liste de rapport de collision, qui présente les informations suivantes :

Paramètres

Paramètre	Description
CollisionCallback	Obligatoire. Spécifie le nom du gestionnaire de rappel de collision.
Référence de script	Facultatif. Il s'agit d'une instance de script de comportement ou parent. Si ce paramètre n'est pas renseigné, le système suppose que la fonction de rappel est définie dans un script d'animation.

Structure d'un rapport de collision :

ObjectA	Objet A impliqué dans la collision
ObjectB	Objet B impliqué dans la collision
list (vector<valeur à virgule flottante,valeur à virgule flottante,valeur à virgule flottante>)	Points de contact entre l'objet A et l'objet B
list (vector<valeur à virgule flottante,valeur à virgule flottante,valeur à virgule flottante>)	Normales de contact pour chacun des points ci-dessus

Remarque : *à l'intérieur du gestionnaire de rappel de collision, si un script engendre des erreurs d'exécution (comme une propriété introuvable), les messages d'erreur ne sont pas affichés et l'exécution du gestionnaire est annulée à l'endroit de l'erreur. Toutes les instructions suivantes ne sont pas exécutées.*
Evitez d'apporter des modifications à la scène 3D dans le gestionnaire de rappel de collision, comme resetworld(), deleteModel(), deleterigidBody(), etc.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").registercollisionCallback(#collisioncallback)
--Movie Script
On collisionCallback collisionReport
CollisionA = collisionReport[1]
R1 = CollisionA.objectA
R2 = CollisionA.objectB
lstPoints = CollisionA.contactPoints
lstNormals = CollisionA.contactNormals
end
```

```
//JavaScript Syntax
member("PhysicsWorld").registerCallback(symbol("collisioncallback"));
//Movie Script
function collisionCallback(collisionreport)
{
var CollisionA = collisionReport(0);
var R1 = CollisionA.objectA;
var R2 = CollisionA.objectB;
var lstPoints = CollisionA.contactPoints;
var lstNormals = CollisionA.contactNormals;
}
```

Voir aussi

enableCollisionCallback(),disableCollisionCallback(),getCollisionDisabledPairs(),getCollisionDisabledPairs(),removeCollisionCallback()

removeCollisionCallback()

Syntaxe

```
<int>world .removeCollisionCallback()
```

Description :

Cette fonction supprime l'appel enregistré précédemment.

Paramètres

Aucune.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").removeCollisionCallback()

//JavaScript Syntax
member("PhysicsWorld").removeCollisionCallback();
```

Voir aussi

enableCollisionCallback(),disableCollisionCallback(),getCollisionDisabledPairs(),getCollisionCallbackDisabledPairs(),registerCollisionCallback()

Méthodes du terrain

- createTerrain
- createTerrainDesc
- deleteTerrain
- getTerrain
- getTerrains

Remarque : *Assurez-vous que le nom du corps rigide est différent de celui du terrain. Si vous utilisez le même nom pour le corps rigide et le terrain, une erreur -4 indiquant un paramètre non valide est renvoyée.*

createTerrain

Syntaxe

```
createTerrain(terrainName, terrainDesc, position, orientation, rowScale, columnScale, heightScale)
```

Description

Cette méthode crée le terrain sous la forme d'un corps rigide statique en utilisant les informations de correspondance de hauteurs fournies.

Remarque : *Pour optimiser les performances lors de la création d'une maille et d'une correspondance de hauteurs en utilisant une image de correspondance de hauteurs, divisez la maille en parties plus petites et reconstituez-la.*

Paramètres

Paramètre	Description
terrainName	Obligatoire. Chaîne indiquant le nom du terrain.
terrainDesc	Obligatoire. Référence au descripteur du terrain.
position	Obligatoire. Vecteur qui définit la position du terrain.
orientation	Obligatoire. Vecteur qui définit l'orientation du terrain.

Paramètre (Suite)	Description (Suite)
rowScale	Obligatoire. Valeur à virgule flottante qui indique le facteur d'échelle de ligne.
columnScale	Obligatoire. Valeur à virgule flottante qui indique le facteur d'échelle de colonne.
heightScale	Obligatoire. Valeur à virgule flottante qui indique le facteur d'échelle de hauteur.

Exemple

```
--Lingo Syntax
objTerrain =
member("PhysicsWorld").createTerrain("myterrain",terrainDesc,position,orientation,1,1,1)

//JavaScript Syntax
var objTerrain=
member("PhysicsWorld").createTerrain("myterrain",terrainDesc,position,orientation,1,1,1);
```

Voir aussi

createTerrainDesc,terrainDesc

createTerrainDesc

Syntaxe

```
createTerrainDesc(elevationMatrix,friction,restitution)
```

Description

Un objet descripteur de terrain doit être créé avant la création d'un terrain.

Paramètres

Paramètre	Description	Valeurs :
ElevationMatrix	Obligatoire. Matrice Lingo qui représente les informations de correspondance des hauteurs du terrain.	
Friction	Obligatoire. Valeur à virgule flottante qui représente le coefficient de friction.	0 – 1
Restitution	Obligatoire. Valeur à virgule flottante qui représente le coefficient de restitution.	0 – 1

Exemple

```
--Lingo Syntax
tDesc = member("PhysicsWorld").createTerrainDesc(myMatrix,0.4,0.5)

//JavaScript Syntax
//Spring constraint Descriptor between two rigid bodies' b1 and b2.
var tDesc = member("PhysicsWorld").createTerrainDesc(myMatrix,0.4,0.5);
```

Voir aussi

createTerrain,terrainDesc

deleteTerrain

Syntaxe

```
world.deleteterrain(String terrainName)
world.deleteterrain(terrain refTerrain)
```

Description

Cette méthode supprime le terrain de la scène physique. Elle peut prendre le nom ou la référence du terrain comme paramètre.

Paramètres

Paramètre	Description
terrainName	Obligatoire. Valeur de chaîne qui spécifie le nom du terrain créé.

ou

Paramètre	Description
refTerrain	Obligatoire. Référence à l'objet terrain créé.

Exemple

```
--Lingo Syntax
member("PhysicsWorld").deleteTerrain("myterrain")

//JavaScript Syntax
member("PhysicsWorld").deleteTerrain("myTerrain");
```

getTerrain

Syntaxe

```
<terrain refTerrain> world.getterrain(String "myterrain")
```

Description

Cette méthode renvoie l'objet terrain du nom spécifié.

Paramètres

Paramètres	Description
MonTerrain	Obligatoire. Chaîne indiquant le nom du terrain.

Exemple

```
--Lingo Syntax
objTerrain = member("PhysicsWorld").getTerrain("myTerrain")

//JavaScript Syntax
var objTerrain = member("PhysicsWorld").getTerrain("myTerrain");
```

Voir aussi

createTerrainDesc,terrainDesc

getTerrains

Syntaxe

```
<list lstTerrain> world.getterrains()
```

Description

Cette méthode renvoie la liste d'objets terrain présents dans l'univers physique.

Paramètres

Aucun

Exemple

```
--Lingo Syntax
lstTerrain = member("PhysicsWorld").getTerrains()

//JavaScript Syntax
var lstTerrain = member("PhysicsWorld").getTerrains();
```

Propriétés du terrain

- columnScale
- contactTolerance
- heightScale
- name
- orientation
- position
- rowScale
- terrainDesc

columnScale

Syntaxe

```
t.columnScale
```

Accès : Obtention

Description

Représente le facteur d'échelle de colonne du terrain.

Exemple

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.columnScale

//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.columnScale);
```

contactTolerance

Syntaxe

```
t.contactTolerance
```

Accès : obtention/définition

Type : valeur à virgule flottante

Description

Profondeur de pénétration entre le terrain et un autre corps rigide/terrain pour que la collision soit détectée.

Exemple

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.contactTolerance

//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.contactTolerance);
```

heightScale

Syntaxe

```
t.heightScale
```

Accès : Obtention

Description

Représente le facteur d'échelle de hauteur du terrain.

Exemple

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.heightScale

//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.heightScale);
```

name

Syntaxe

```
t.name
```

Accès : Obtention

Type : chaîne

Description

Renvoie le nom du terrain.

Exemple

```
--Lingo Syntax
put objTerrain.name

//Javascript Syntax
put(objTerrain.name);
```

orientation

Syntaxe

```
t.orientation
```

Accès : Obtention

Description

Liste qui représente l'orientation du terrain.

Exemple

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.orientation

//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.orientation);
```

position

Syntaxe

```
t.position
```

Accès : Obtention

Description

Vecteur qui représente la position du terrain.

Exemple

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.position

//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.position);
```

rowScale

Syntaxe

```
t.rowScale
```

Accès : Obtention

Description : représente le facteur d'échelle de ligne du terrain.

Exemple

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.rowScale

//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.rowScale);
```

terrainDesc

Syntaxe

```
t.terrainDesc
```

Accès : Obtention

Description

Représente l'objet descripteur du terrain.

Les attributs du descripteur du terrain sont les suivants :

- terraindesc.elevationmatrix
- terraindesc.friction
- terraindesc.restitution
- terraindesc.numrows
- terraindesc.numcolumns

Remarque : *numrows et numcolumns correspondent aux lignes et aux colonnes de la matrice d'élévation.*

Exemple

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.terrainDesc

//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.terrainDesc);
```

Voir aussi

createTerrainDesc,createTerrain

Méthodes du joint à 6 degrés de liberté

createD6Joint

Il est important de respecter l'ordre dans lequel les corps rigides sont fournis en tant que valeurs d'entrée à un joint à 6 degrés de liberté, ainsi que l'axe du joint défini et les valeurs du disque. Par exemple, si l'ordre est rb1,rb2 et que la position du disque est vector(10, 0, 0), la direction de déplacement du corps rigide pour atteindre la position est exactement l'inverse de la direction de déplacement si l'ordre était rb2,rb1.

createD6Joint

Syntaxe

```
<d6joint> world.createD6Joint(jointName, RigidBody rb1 ,Rigidbody rb2, vector globalanchor)
```

Description

Cette méthode crée un joint à 6 degrés de liberté entre deux corps rigides ou entre un point de l'univers et un corps rigide.

Remarque : *Les joints à 6 degrés de liberté sont également un type de contrainte comme les joints linéaires et angulaires. Le type de contrainte du joint à 6 degrés de liberté est #d6joint.*

Utilisez deleteConstraint() pour supprimer le joint à 6 degrés de liberté.

L'ensemble supplémentaire de propriétés spécifiques au joint à 6 degrés de liberté est mentionné dans la section sur les propriétés du joint à 6 degrés de liberté.

Paramètres

Paramètre	Description
jointName	Obligatoire. Chaîne qui indique le nom du joint.
rb1	Obligatoire. Référence au corps rigide qui participe au joint. Si un joint doit être créé entre un point et un corps rigide, alors ce paramètre doit être vide.
rb2	Obligatoire. Référence au corps rigide qui participe au joint.
globalAnchor	Obligatoire. Vecteur qui indique le point d'ancrage du joint.

Exemple

```
--Lingo Syntax
--Creates a joint between rigid bodies rb1 and rb2.
ObjJoint = member("PhysicsWorld").createD6Joint("myJoint",rb1,rb2,vector(1,1,1))
-- Creates a joint between the anchor point and rb2.
objJoint = member("PhysicsWorld").createD6Joint("myJoint",void,rb2,vector(1,1,1))

//JavaScript Syntax
//Creates a joint between rigid bodies rb1 and rb2.
var objJoint = member("PhysicsWorld").createD6Joint("myJoint",rb1,rb2,vector(1,1,1));
```

Voir aussi

createAngularJoint(), createLinearJoint(), createSpring(), ConstraintDesc, globalAnchor

Propriétés 6 degrés de liberté

- axisDrive
- axisMotion
- biNormalDrive
- biNormalMotion
- constraintType
- driveAngularVelocity
- driveLinearVelocity
- driveOrientation
- drivePosition
- globalAnchor
- linearLimit
- localAnchorA
- localAnchorB
- localAxisA
- localAxisB
- localNormalA
- localNormalB
- name

- normalDrive
- normalMotion
- objectA
- objectB
- swing1Limit
- swing1Motion
- swing2Limit
- swing2Motion
- swingDrive
- twistDrive
- twistLimit
- twistMotion

axisDrive

Syntaxe

```
d6joint.axisDrive = [type, spring, damping, forceLimit]
```

Accès : obtention/définition

Description

Valeur de liste qui indique l'entraînement linéaire du joint, qui entraînera le joint jusqu'à la position spécifiée le long de l'axe du joint.

La liste contient les attributs suivants :

type #position ou #velocity

stiffness Le ressort à appliquer pendant l'entraînement (> 0)

damping L'amortissement du ressort (> 0)

forceLimit Le force ou le couple d'entraînement jusqu'à la position ou la vitesse spécifiée (> 0)

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.axisDrive = [#position,100,0.1,100]

//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.axisDrive = [#position,100,0.1,100];
```

Voir aussi

biNormalDrive,normalDrive,swingDrive,twistDrive,driveAngularVelocity,driveLinearVelocity,driveOrientation,drivePosition

axisMotion

Syntaxe

```
D6joint.axisMotion
```

Accès : obtention/définition

Description

Symbole qui définit le degré de liberté linéaire le long de l'axe du joint.

Cette propriété accepte les valeurs suivantes :

#Locked Aucun mouvement n'est possible le long de ce degré de liberté.

#Limited Un mouvement limité est possible le long de ce degré de liberté.

#Free Aucune limite de mouvement ne s'applique le long de ce degré de liberté.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.axisMotion
```

```
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.axisMotion);
```

Voir aussi

biNormalMotion,normalMotion,swing1Motion,swing2Motion,twistMotion

biNormalDrive

Syntaxe

```
d6joint.binormalDrive = [type, spring, damping, forceLimit]
```

Accès : obtention/définition

Description

Valeur de liste qui indique l'entraînement linéaire du joint, qui entraînera le joint jusqu'à la position spécifiée le long de l'axe binormal du joint.

La liste contient les attributs suivants :

type #position ou #velocity

stiffness Le ressort à appliquer pendant l'entraînement (> 0)

damping L'amortissement du ressort (> 0)

forceLimit Le force ou le couple d'entraînement jusqu'à la position ou la vitesse spécifiée (> 0)

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.binormalDrive = [#position,100,0.1,100]
```

```
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.binormalDrive = [#position,100,0.1,100];
```

Voir aussi

normalDrive, swingDrive, twistDrive, driveAngularVelocity, driveLinearVelocity, driveOrientation, drivePosition, axisDrive

biNormalMotion

Syntaxe

```
D6joint.binormalMotion
```

Accès : obtention/définition

Description

Symbole qui définit le degré de liberté linéaire le long de l'axe binormal du joint.

Cette propriété accepte les valeurs suivantes :

#Locked Aucun mouvement n'est possible le long de ce degré de liberté.

#Limited Un mouvement limité est possible le long de ce degré de liberté.

#Free Aucune limite de mouvement ne s'applique le long de ce degré de liberté.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.biNormalMotion
```

```
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.biNormalMotion);
```

Voir aussi

normalMotion, swing1Motion, swing2Motion, twistMotion, axisMotion

constraintType

Syntaxe

```
d6joint.constraintType
```

Accès : Obtention

Description

Les joints à 6 degrés de liberté sont un type de contrainte comme les joints linéaires et angulaires. Le type de contrainte du joint à 6 degrés de liberté est #d6joint.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.constraintType
```

```
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
put(objJoint.constraintType);
```

driveAngularVelocity

Syntaxe

```
d6joint.driveAngularVelocity = vector velocity
```

Accès : obtention/définition

Description

Valeur vectorielle qui indique la vitesse angulaire prévue lorsque le type d'entraînement pour swingDrive ou twistDrive est spécifié comme étant #velocity.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.driveAngularVelocity = vector(20,10,0)

//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.driveAngularVelocity = vector(20,10,0);
```

Voir aussi

normalDrive, swingDrive, twistDrive, driveLinearVelocity, driveOrientation, drivePosition, axisDrive, biNormalDrive

driveLinearVelocity

Syntaxe

```
d6joint.driveLinearVelocity = vector velocity
```

Accès : obtention/définition

Description

Valeur vectorielle qui spécifie la vitesse linéaire prévue lorsque le type d'entraînement pour axisDrive, normalDrive ou binormalDrive est #velocity.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.driveLinearVelocity = vector(20,10,0)

//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.driveLinearVelocity = vector(20,10,0);
```

Voir aussi

normalDrive, swingDrive, twistDrive, driveAngularVelocity, driveOrientation, drivePosition, axisDrive, biNormalDrive

driveOrientation

Syntaxe

```
d6joint.driveOrientation = list orientation
```

Accès : obtention/définition

Description

Valeur de liste qui indique l'orientation de l'objectif lorsque le type d'entraînement pour swingDrive ou twistDrive est spécifié comme étant #position.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.driveOrientation = [vector(20,10,0),45]

//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.driveOrientation = [vector(20,10,0),45];
```

Voir aussi

normalDrive, swingDrive, twistDrive, driveLinearVelocity, driveAngularVelocity, drivePosition, axisDrive, biNormalDrive

drivePosition

Syntaxe

```
d6joint.drivePosition = vector position
```

Accès : obtention/définition

Description

Valeur vectorielle qui indique l'orientation de l'objectif lorsque le type d'entraînement pour axisDrive, normalDrive et binormalDrive est spécifié comme étant #position

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.drivePosition = vector(20,10,0)

//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.drivePosition = vector(20,10,0);
```

Voir aussi

normalDrive, swingDrive, twistDrive, driveLinearVelocity, driveAngularVelocity, driveOrientation, axisDrive, biNormalDrive

globalAnchor

Syntaxe

```
d6joint.globalAnchor = vector position
```

Accès : obtention/définition

Description

Valeur vectorielle qui indique le point d'ancrage du joint.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.globalAnchor = vector(20,10,0)
```

```
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.globalAnchor = vector(20,10,0);
```

Voir aussi

localAnchorA,localAnchorB

linearLimit

Syntaxe

```
D6joint.linearlimit = [limitvalue, stiffness, damping, restitution]
```

Accès : obtention/définition

Description

Liste qui indique le comportement du joint lorsque le mouvement linéaire est limité. Le corps rigide oscille lorsqu'il atteint la valeur limite. Ceci s'applique à tous les mouvements linéaires.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.linearLimit =  [2, 100, 0.01, 0]
```

```
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.linearLimit = [2, 100, 0.01, 0];
```

Remarque : *définissez d'abord des valeurs limites, puis les paramètres de mouvement [#free,#limited, etc.] pour que les limites prennent effet.*

Voir aussi

swing1Limit,swing2Limit,twistLimit

localAnchorA

Syntaxe

```
D6joint.localAnchorA
```

Accès : obtention/définition

Description

Point d'attache du joint dans l'espace de objectA.

Les valeurs des propriétés localAxisA, localAxisB, localNormalA, localNormalB, localAnchorA et localAnchorB sont disponibles uniquement une fois que vous avez défini les valeurs explicitement pour ces propriétés. Si la valeur n'est pas définie, un vecteur void est renvoyé.

Remarque : *Pour un joint stable, indiquez des valeurs cohérentes pour localAnchorA et localAnchorB afin que, dans l'espace de l'univers, ces deux propriétés correspondent au même point.*

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.localAnchorA = vector(0,5,0)
```

```
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint");
objJoint.localAnchorA = vector(0,5,0);
```

Voir aussi

localAnchorB,globalAnchor

localAnchorB

Syntaxe

```
D6joint.localAnchorB
```

Accès : obtention/définition

Description

Point d'attache du joint dans l'espace de objectB.

Les valeurs des propriétés localAxisA, localAxisB, localNormalA, localNormalB, localAnchorA et localAnchorB sont disponibles uniquement une fois que vous avez défini les valeurs explicitement pour ces propriétés. Si la valeur n'est pas définie, un vecteur void est renvoyé.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.localAnchorB = vector(0,-5,0)
```

```
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint");
objJoint.localAnchorB = vector(0,-5,0);
```

Voir aussi

localAnchorA,globalAnchor

localAxisA

Syntaxe

```
D6joint.localAxisA
```

Accès : obtention/définition

Description

Il s'agit de l'axe principal du joint en terme d'espace de coordonnées local de objectA du joint à 6 degrés de liberté. localAxisA et localNormalA doivent être à angle droit l'un par rapport à l'autre.

Quand un corps rigide est statique, les axes locaux définis pour le corps sont considérés comme globaux, car ils ne peuvent pas être orientés le long des axes du joint.

Les valeurs des propriétés localAxisA, localAxisB, localNormalA, localNormalB, localAnchorA et localAnchorB sont disponibles uniquement une fois que vous avez défini les valeurs explicitement pour ces propriétés. Si la valeur n'est pas définie, un vecteur void est renvoyé.

Remarque :

- L'axe principal du joint est l'axe autour duquel la torsion se produit et le long duquel le mouvement de l'axe se produira.

• L'axe normal du joint est l'axe autour duquel le déplacement 1 se produit et le long duquel le mouvement normal se produira.

• L'axe binormal du joint est l'axe autour duquel le déplacement 2 se produit et le long duquel le mouvement binormal se produira.

Pour changer les axes du joint à tout moment, définissez les propriétés suivantes de manière appropriée :

• localAxisA

• localNormalA

• localAxisB

• localNormalB

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
--If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0)
objJoint.localNormalA = vector(1,0,0)
-- If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1)
objJoint.localNormalB = vector(1,0,0)
```

```
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
//If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0);
objJoint.localNormalA = vector(1,0,0);
// If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1);
objJoint.localNormalB = vector(1,0,0);
```

Voir aussi

localAxisB, localNormalA, localNormalB

localAxisB

Syntaxe

```
D6joint.localAxisB
```

Accès : obtention/définition

Description

Il s'agit de l'axe principal du joint en terme d'espace de coordonnées local de objectB du joint à 6 degrés de liberté. localAxisB et localNormalB doivent être à angle droit l'un par rapport à l'autre. Il s'agit de l'axe autour duquel la rotation de torsion et le long duquel le mouvement de l'axe sont définis.

Les valeurs des propriétés localAxisA, localAxisB, localNormalA, localNormalB, localAnchorA et localAnchorB sont disponibles uniquement une fois que vous avez défini les valeurs explicitement pour ces propriétés. Si la valeur n'est pas définie, un vecteur void est renvoyé.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
--If ObjectA's orientation is vector(0,0,0)
```

```
objJoint.localAxisA = vector(0,1,0)
objJoint.localNormalA = vector(1,0,0)
-- If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1)
objJoint.localNormalB = vector(1,0,0)

//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
//If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0);
objJoint.localNormalA = vector(1,0,0);
// If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1);
objJoint.localNormalB = vector(1,0,0);
```

Voir aussi

localAxisA,localNormalA,localNormalB

localNormalA

Syntaxe

```
D6joint.localNormalA
```

Accès : obtention/définition

Description

Il s'agit de l'axe normal du joint en terme d'espace de coordonnées local de objectA du joint à 6 degrés de liberté. localAxisA et localNormalA doivent être à angle droit l'un par rapport à l'autre. Il s'agit de l'axe autour duquel le déplacement 1 et le long duquel le mouvement normal sont définis.

Quand un corps rigide est statique, les axes locaux définis pour le corps sont considérés comme globaux, car ils ne peuvent pas être orientés le long des axes du joint.

Les valeurs des propriétés localAxisA, localAxisB, localNormalA, localNormalB, localAnchorA et localAnchorB sont disponibles uniquement une fois que vous avez défini les valeurs explicitement pour ces propriétés. Si la valeur n'est pas définie, un vecteur void est renvoyé.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
--If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0)
objJoint.localNormalA = vector(1,0,0)
-- If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1)
objJoint.localNormalB = vector(1,0,0)

//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
//If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0);
objJoint.localNormalA = vector(1,0,0);
// If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1);
objJoint.localNormalB = vector(1,0,0);
```

Voir aussi

localAxisB,localAxisA,localNormalB

localNormalB

Syntaxe

```
D6joint.localNormalB
```

Accès : obtention/définition

Description

Il s'agit de l'axe normal du joint en terme d'espace de coordonnées local de objectB du joint à 6 degrés de liberté. localAxisB et localNormalB doivent être à angle droit l'un par rapport à l'autre. Il s'agit de l'axe autour duquel le déplacement 1 et le long duquel le mouvement normal sont définis.

Les valeurs des propriétés localAxisA, localAxisB, localNormalA, localNormalB, localAnchorA et localAnchorB sont disponibles uniquement une fois que vous avez défini les valeurs explicitement pour ces propriétés. Si la valeur n'est pas définie, un vecteur void est renvoyé.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
--If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0)
objJoint.localNormalA = vector(1,0,0)
-- If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1)
objJoint.localNormalB = vector(1,0,0)
```

```
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
//If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0);
objJoint.localNormalA = vector(1,0,0);
// If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1);
objJoint.localNormalB = vector(1,0,0);
```

Voir aussi

localAxisA, localAxisB, localNormalA

name

Syntaxe

```
d6joint.name
```

Accès : Obtention

Type : chaîne

Description

Renvoie le nom du joint à 6 degrés de liberté.

Exemple

```
--Lingo Syntax
objConstraint = member("PhysicsWorld").getConstraint("d6joint")
put objConstraint.name
```

```
//Javascript Syntax
var objConstraint = member("PhysicsWorld").getConstraint("d6joint");
put(objConstraint.name);
```

normalDrive

Syntaxe

```
d6joint.normalDrive = [type, spring, damping, forceLimit]
```

Accès : obtention/définition

Description

Valeur de liste qui indique l'entraînement linéaire du joint, qui entraînera le joint jusqu'à la position spécifiée le long de l'axe normal du joint.

La liste contient les attributs suivants :

type #position ou #velocity

Remarque : *Si vous choisissez position, la valeur de forcelimit est ignorée et seule la valeur de spring est prise en compte. De même, si vous choisissez velocity, la valeur de spring est ignorée et seule la valeur de forcelimit est prise en compte.*

stiffness Le ressort à appliquer pendant l'entraînement (> 0)

damping L'amortissement du ressort (> 0)

forceLimit Le force ou le couple d'entraînement jusqu'à la position ou la vitesse spécifiée (> 0)

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.normalDrive = [#position,100,0.1,100]
```

```
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.normalDrive = [#position,100,0.1,100];
```

Voir aussi

swingDrive, twistDrive, driveAngularVelocity, driveLinearVelocity, driveOrientation, drivePosition, axisDrive, biNormalDrive

normalMotion

Syntaxe

```
D6joint.normalMotion
```

Accès : obtention/définition

Description

Symbole qui définit le degré de liberté linéaire le long de l'axe normal du joint.

Cette propriété accepte les valeurs suivantes :

#Locked Aucun mouvement n'est possible le long de ce degré de liberté.

#Limited Un mouvement limité est possible le long de ce degré de liberté.

#Free Aucune limite de mouvement ne s'applique le long de ce degré de liberté.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.normalMotion

//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.normalMotion);
```

Voir aussi

swing1Motion, swing2Motion, twistMotion, axisMotion, biNormalMotion

objectA

Syntaxe

```
d6joint.objectA
```

Accès : Obtention

Type : RigidBody

Description

Renvoie le nom du corps rigide « A » attaché au joint.

Exemple

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("Joint01")
put objJoint.objectA

//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("Joint01");
put(objJoint.objectA);
```

Voir aussi

objectB

objectB

Syntaxe

```
d6joint.objectB
```

Accès : Obtention

Type : RigidBody

Description

Renvoie le nom du corps rigide « B » attaché au ressort.

Exemple

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("Joint01")
put objJoint.objectB

//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("Joint01");
put(objJoint.objectB);
```

Voir aussi

objectA

swing1Limit

Syntaxe

```
D6joint.swing1limit = [limitvalue, stiffness, damping, restitution]
```

Accès : obtention/définition

Description

Liste qui indique le comportement du joint lorsque le mouvement angulaire autour de l'axe normal du joint est limité. Le corps rigide oscille lorsqu'il atteint la valeur limite.

La valeur limite peut être comprise entre 3,14 et -3,14 (pi).

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.swing1limit =  [3.14*0.5, 100, 0.01, 0]

//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.swing1limit = [3.14*0.5, 100, 0.01, 0];
```

Remarque : *pour que les limites prennent effet, définissez d'abord les valeurs limites, puis les paramètres de mouvement [#free,#limited, etc.].*

Voir aussi

swing2Limit,twistLimit,linearLimit

swing1Motion

Syntaxe

```
D6joint.swing1Motion
```

Accès : obtention/définition

Description

Symbole qui définit le degré de liberté angulaire autour de l'axe normal du joint.

Cette propriété accepte les valeurs suivantes :

#Locked Aucun mouvement n'est possible le long de ce degré de liberté.

#Limited Le mouvement le long de ce degré de liberté est limité.

#Free Aucune limite de mouvement ne s'applique le long de ce degré de liberté.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.swing1Motion

//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.swing1Motion);
```

Voir aussi

normalMotion,swing2Motion,twistMotion,axisMotion,biNormalMotion

swing2Limit

Syntaxe

```
D6joint.swing2limit = [limitvalue, stiffness, damping, restitution]
```

Accès : obtention/définition

Description

Liste qui indique le comportement du joint lorsque le mouvement angulaire autour de l'axe binormal du joint est limité. Le corps rigide oscille lorsqu'il atteint la valeur limite.

La valeur limite peut être comprise entre 3,14 et -3,14 (pi).

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.swing2limit =  [3.14*0.5, 100, 0.01, 0]

//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.swing2limit = [3.14*0.5, 100, 0.01, 0];
```

Remarque : *pour que les limites prennent effet, définissez d'abord les valeurs limites, puis les paramètres de mouvement [#free,#limited, etc.].*

Voir aussi

swing1Limit,twistLimit,linearLimit

swing2Motion

Syntaxe

```
D6joint.swing2Motion
```

Accès : obtention/définition

Description

Symbole qui définit le degré de liberté angulaire autour de l'axe binormal du joint.

Cette propriété accepte les valeurs suivantes :

#Locked Aucun mouvement n'est possible le long de ce degré de liberté.

#Limited Le mouvement le long de ce degré de liberté est limité.

#Free Aucune limite de mouvement ne s'applique le long de ce degré de liberté.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.swing2Motion

//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.swing2Motion);
```

Voir aussi

normalMotion,swing1Motion,twistMotion,axisMotion,biNormalMotion

swingDrive

Syntaxe

```
d6joint.swingDrive = [type, spring, damping, forceLimit]
```

Accès : obtention/définition

Description

Valeur de liste qui indique l'entraînement angulaire du joint, qui entraînera le joint jusqu'à l'orientation spécifiée autour de l'axe normal et de l'axe binormal du joint.

La liste contient les attributs suivants :

type #position ou #velocity

stiffness Le ressort à appliquer pendant l'entraînement (> 0)

damping L'amortissement du ressort (> 0)

forceLimit Le force ou le couple d'entraînement jusqu'à la position ou la vitesse spécifiée (> 0)

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.swingDrive = [#position,100,0.1,100]

//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.swingDrive = [#position,100,0.1,100];
```

Voir aussi

normalDrive,twistDrive,driveAngularVelocity,driveLinearVelocity,driveOrientation,drivePosition,axisDrive,biNormalDrive

twistDrive

Syntaxe

```
d6joint.twistDrive = [type, spring, damping, forceLimit]
```

Accès : obtention/définition

Description

Valeur de liste qui indique l'entraînement angulaire du joint, qui entraînera le joint jusqu'à l'orientation spécifiée autour de l'axe du joint.

La liste contient les attributs suivants :

type #position ou #velocity

stiffness Le ressort à appliquer pendant l'entraînement (> 0)

damping L'amortissement du ressort (> 0)

forceLimit Le force ou le couple d'entraînement jusqu'à la position ou la vitesse spécifiée (> 0)

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.twistDrive = [#position,100,0.1,100]

//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.twistDrive = [#position,100,0.1,100];
```

Voir aussi

normalDrive, swingDrive, driveAngularVelocity, driveLinearVelocity, driveOrientation, drivePosition, axisDrive, biNormalDrive

twistLimit

Syntaxe

```
D6joint.twistLimit = [limitvalue, stiffness, damping, restitution]
```

Accès : obtention/définition

Description

Liste qui indique le comportement du joint lorsque le mouvement angulaire le long de l'axe du joint est limité. Le corps rigide oscille lorsqu'il atteint la valeur limite.

La valeur limite peut être comprise entre 3,14 et -3,14 (pi).

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.twistLimit =  [3.14*0.5, 100, 0.01, 0]

//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.twistLimit = [3.14*0.5, 100, 0.01, 0];
```

Remarque : *pour que les limites prennent effet, définissez d'abord les valeurs limites, puis les paramètres de mouvement [#free,#limited, etc.].*

Voir aussi

swing2Limit, swing1Limit, linearLimit

twistMotion

Syntaxe

```
D6joint.twistMotion
```

Accès : obtention/définition

Description

Symbole qui définit le degré de liberté angulaire autour de l'axe principal du joint.

Cette propriété accepte les valeurs suivantes :

#Locked Aucun mouvement n'est possible le long de ce degré de liberté.

#Limited Le mouvement le long de ce degré de liberté est limité.

#Free Aucune limite de mouvement ne s'applique le long de ce degré de liberté.

Exemple

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.twistMotion

//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.twistMotion);
```

Voir aussi

normalMotion, swing2Motion, swing1Motion, axisMotion, biNormalMotion

Méthodes du raycasting

- rayCastAll
- rayCastClosest

rayCastAll

Syntaxe

```
<list <list>> world.rayCastAll(vector origin, vector direction)
```

Description

Cette méthode renvoie les références de tous les corps rigides ou terrains trouvés le long du rayon à partir de l'origine spécifiée et de la direction spécifiée. Elle renvoie également le point de contact, la normale de contact et la distance à partir de l'origine du rayon.

Paramètres

Paramètre	Description
origin	Requis. Vecteur qui indique l'origine du rayon.
direction	Obligatoire. Vecteur qui indique la direction du rayon.

Cette méthode renvoie une liste contenant une autre liste qui présente les informations suivantes :

- référence de corps rigide / terrain ;
- point de contact ;
- normale de contact ;
- distance du corps rigide ou du terrain, à partir de l'origine du rayon.

Exemple

```
--Lingo Syntax
lstraycast = member("PhysicsWorld").rayCastAll (vector(10,0,0),vector(0,0,1))
repeat with i = 1 to lstraycast.count
    raycstEntry = lstraycast[i]
    put "Name:" & raycstEntry[1].name
    put "Contact Point:" & raycstEntry[2]
    put "Contact Normal:" & raycstEntry[3]
```

```
    put "Distance:" & raycstEntry[4]
end repeat

//JavaScript Syntax
var lstraycast = member("PhysicsWorld").rayCastAll (vector(10,0,0),vector(0,0,1));

for(i = 1; i <= lstraycast.count ; i++)
{
    raycstEntry = lstraycast[i];
    put("Name:" + raycstEntry[1].name);
    put"Contact Point:" & raycstEntry[2]);
    put("Contact Normal:" & raycstEntry[3]);
    put("Distance:" & raycstEntry[4]);
}
```

Voir aussi

rayCastClosest

rayCastClosest

Syntaxe

```
<list> world.rayCastClosest(vector origin, vector direction)
```

Description

Cette méthode renvoie la référence du corps rigide ou du terrain le plus proche trouvé le long du rayon à partir de l'origine spécifiée et de la direction spécifiée. Elle renvoie également le point de contact, la normale de contact et la distance à partir de l'origine du rayon.

Paramètres

Paramètre	Description
origin	Requis. Vecteur qui indique l'origine du rayon.
direction	Obligatoire. Vecteur qui indique la direction du rayon.

Cette méthode renvoie une liste contenant les informations suivantes :

- référence de corps rigide / terrain ;
- point de contact ;
- normale de contact ;
- distance du corps rigide ou du terrain, à partir de l'origine du rayon.

Exemple

```
--Lingo Syntax
lstraycast = member("PhysicsWorld").rayCastClosest(vector(10,0,0),vector(0,0,1))
put "Name:"  & lstraycast[1].name
put "Contact Point:"  & lstraycast[2]
put "Contact Normal:"  & lstraycast[3]
put "Distance:"  & lstraycast[4]

//JavaScript Syntax
var lstraycast = member("PhysicsWorld").rayCastClosest(vector(10,0,0),vector(0,0,1));
put("Name:"  + lstraycast[0].name);
put"Contact Point:"  & lstraycast[1]);
put("Contact Normal:"  & lstraycast[2]);
put("Distance:"  & lstraycast[3]);
```

Voir aussi

`rayCastAll`

Codes d'erreur

Les codes d'erreur suivants et leur origine sont décrits dans le tableau suivant :

Code d'erreur	Origine de l'erreur
-1	Echec inconnu Une erreur -1 peut survenir lorsque vous créez un corps rigide si la caméra et le modèle sont séparés par une grande distance. Ceci est dû au fait que le décompte des mailles devient nul en raison de l'optimisation.
-2	Univers non initialisé
-3	Echec de la simulation
-4	Paramètre non valide
-5	Contrainte en double
-6	Contrainte non valide / La contrainte n'existe pas.
-8	Corps rigide non valide / Le corps rigide n'existe pas.
-11	Elément introuvable. Certains objets physiques qui ont été supprimés sont en cours d'utilisation.
-18	Modèle 3D non valide
-19	Erreur 3D
-20	Mémoire insuffisante
-21	La déformation de maille n'a pas été ajoutée pour le modèle.
-28	Quand un corps rigide avec une substitution de forme concave devient #dynamic.

Index

B

C

G

M

N

O

Q

R

S