V1.001B

Manuel de l’utilisateur

Sommaire

Introduction

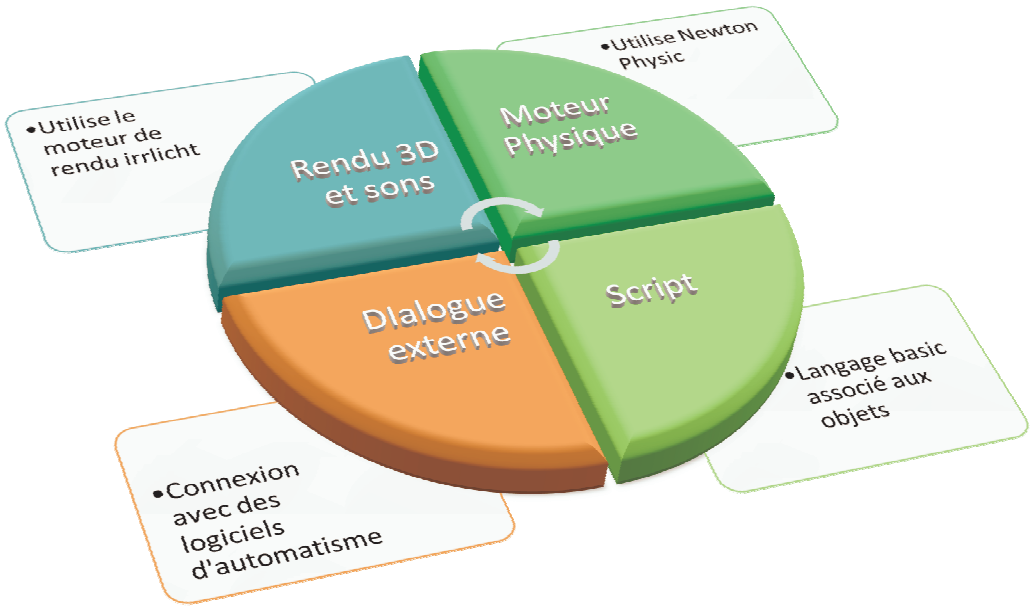
Virtual Universe est un simulateur de monde 3D dédié à l'automatisme et la robotique. En intégrant les dernières technologies de rendu 3D, de son 3D, de simulation physique, et de script, Virtual Universe va vous permettre de créer des simulations ultra réalistes. Virtual Universe peut communiquer avec des ateliers logiciels d'automatisme (AUTOMGEN, UNITY, etc.) afin que les systèmes virtuels puissent être pilotés comme les systèmes réels.

Système nécessaire

Virtual Universe fonctionne sous les systèmes d'exploitation suivants : Windows XP, Windows Vista, Windows 7.

Virtual Universe est compatible avec AUTOMGEN 8.015 ou versions suivantes.

Organisation



Installation

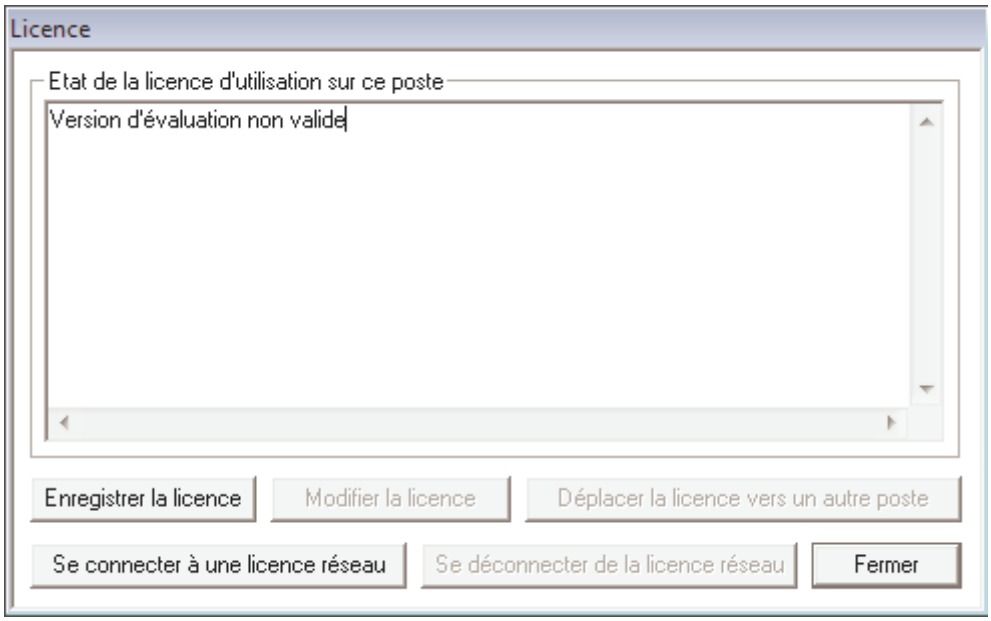
Pour Installer Virtual Universe, lancez simplement l'exécution du package d'installation qui vous a été livré sur CD-ROM ou par téléchargement. Visitez notre site Internet (www.irai.com) pour télécharger les dernières mises à jour de Virtual Universe.

Licence

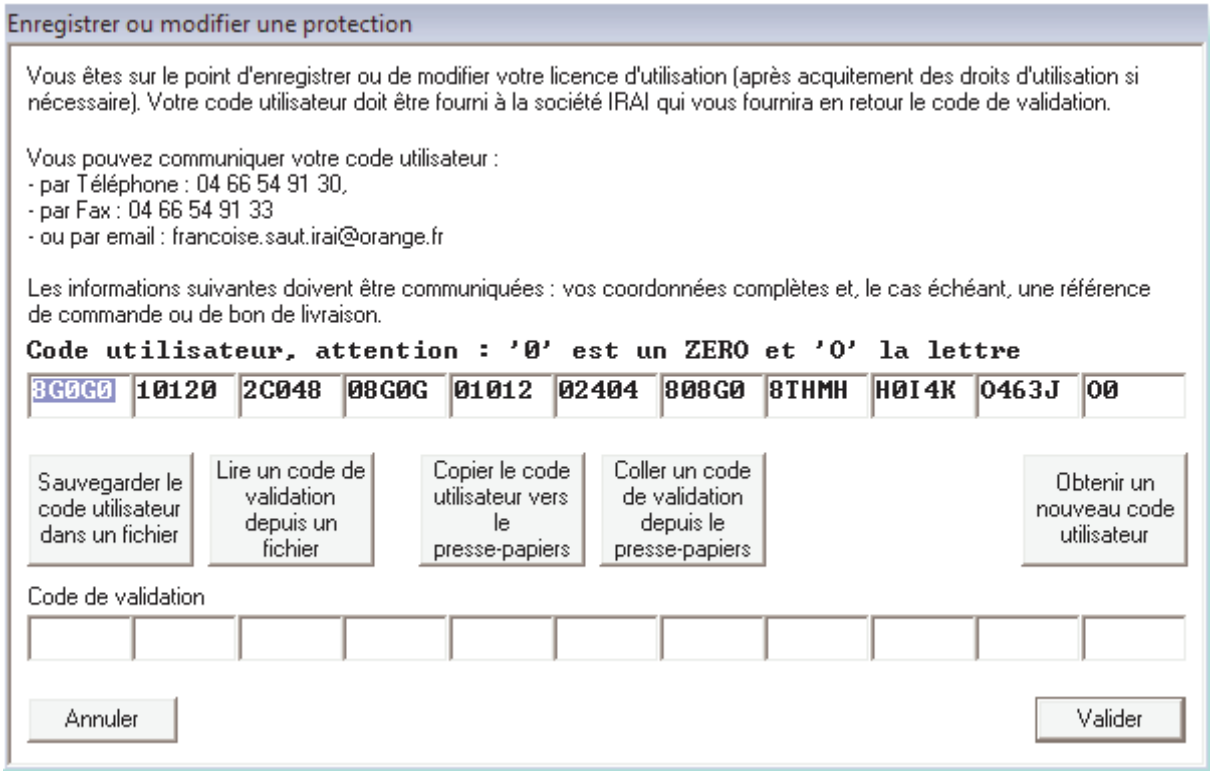
Enregistrer la licence

Virtual Universe fonctionne en version d'évaluation (version limitée à 40 jours d'essai) tant que vous n'avez pas enregistré la licence.

Pour enregistrer la licence, cliquez sur le bouton « Licence » dans la fenêtre de configuration de Virtual Universe.



Cliquez sur le bouton « Enregistrer la licence ».



Envoyez le code utilisateur ainsi généré par email à l'adresse francoise.saut.irai@orange.fr

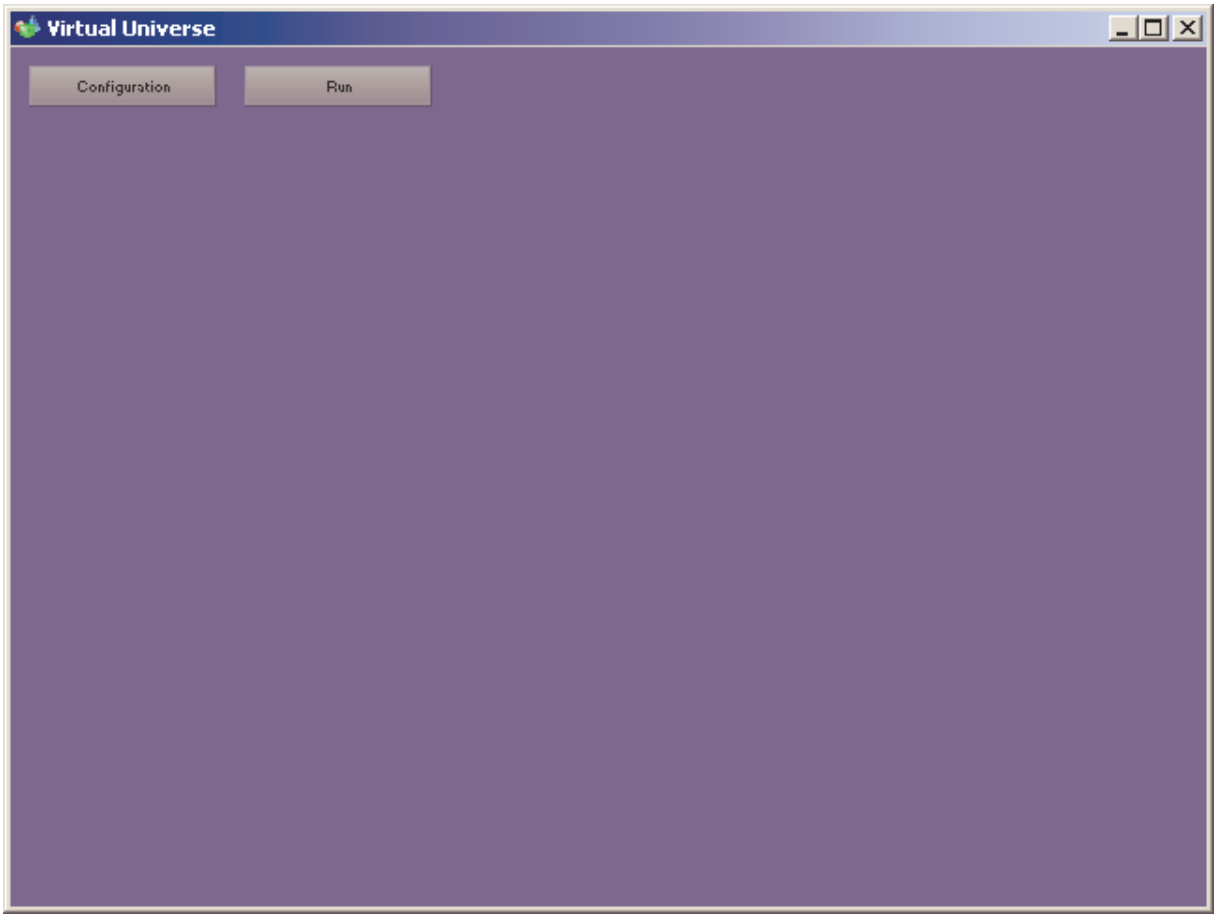
Vous recevrez par email un code de validation que vous saisirez dans les zones « code de validation », puis vous cliquerez sur « Valider » pour valider la licence. Vous disposez de 20 jours entre la génération d'un code utilisateur et la saisie du code de validation.

Installation en réseau

Les fichiers de Virtual Universe peuvent être installés sur un serveur de fichiers. Les licences peuvent également être gérées par un gestionnaire de licences réseau (voir le manuel spécifique du gestionnaire de licences réseau).

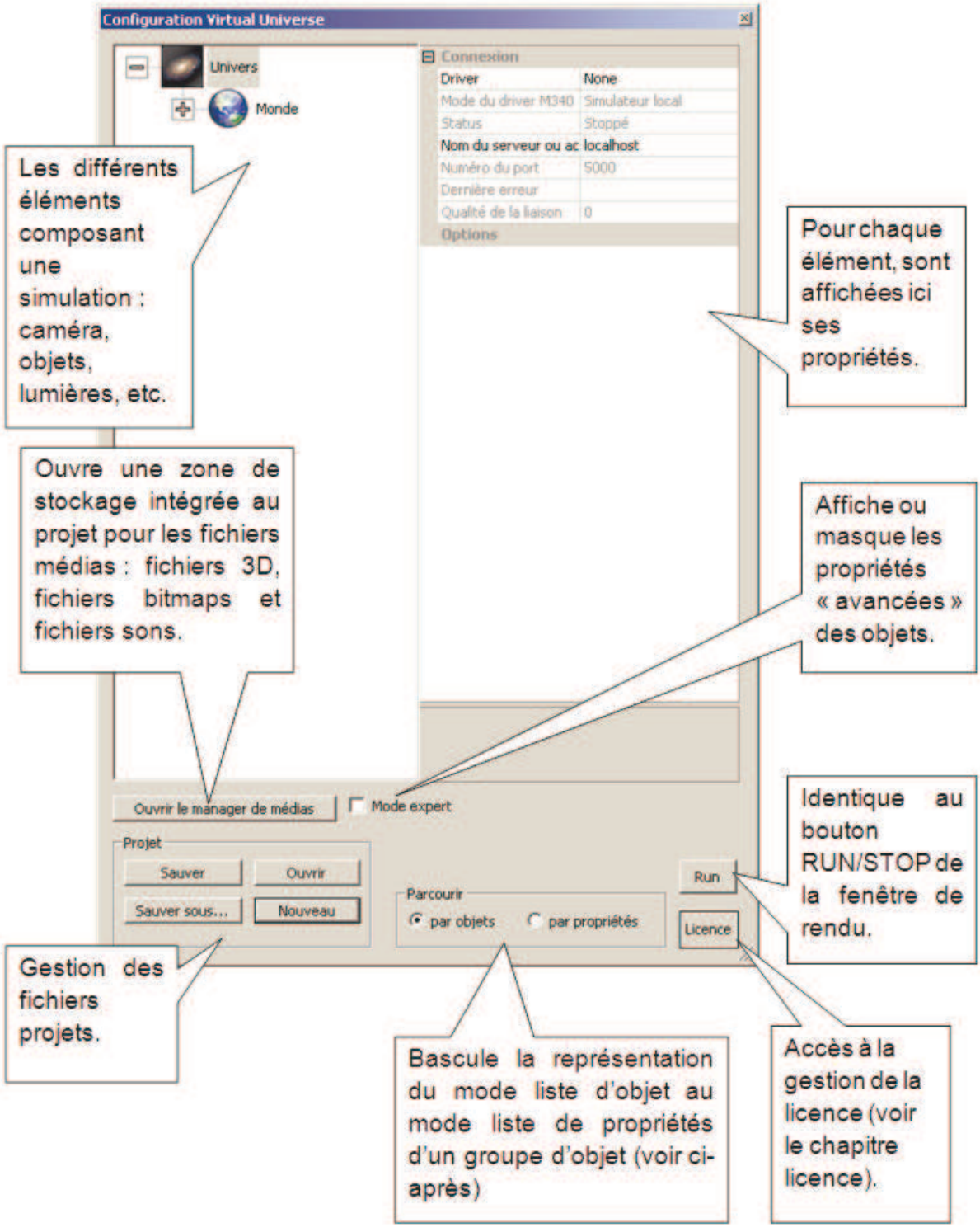
Environnement

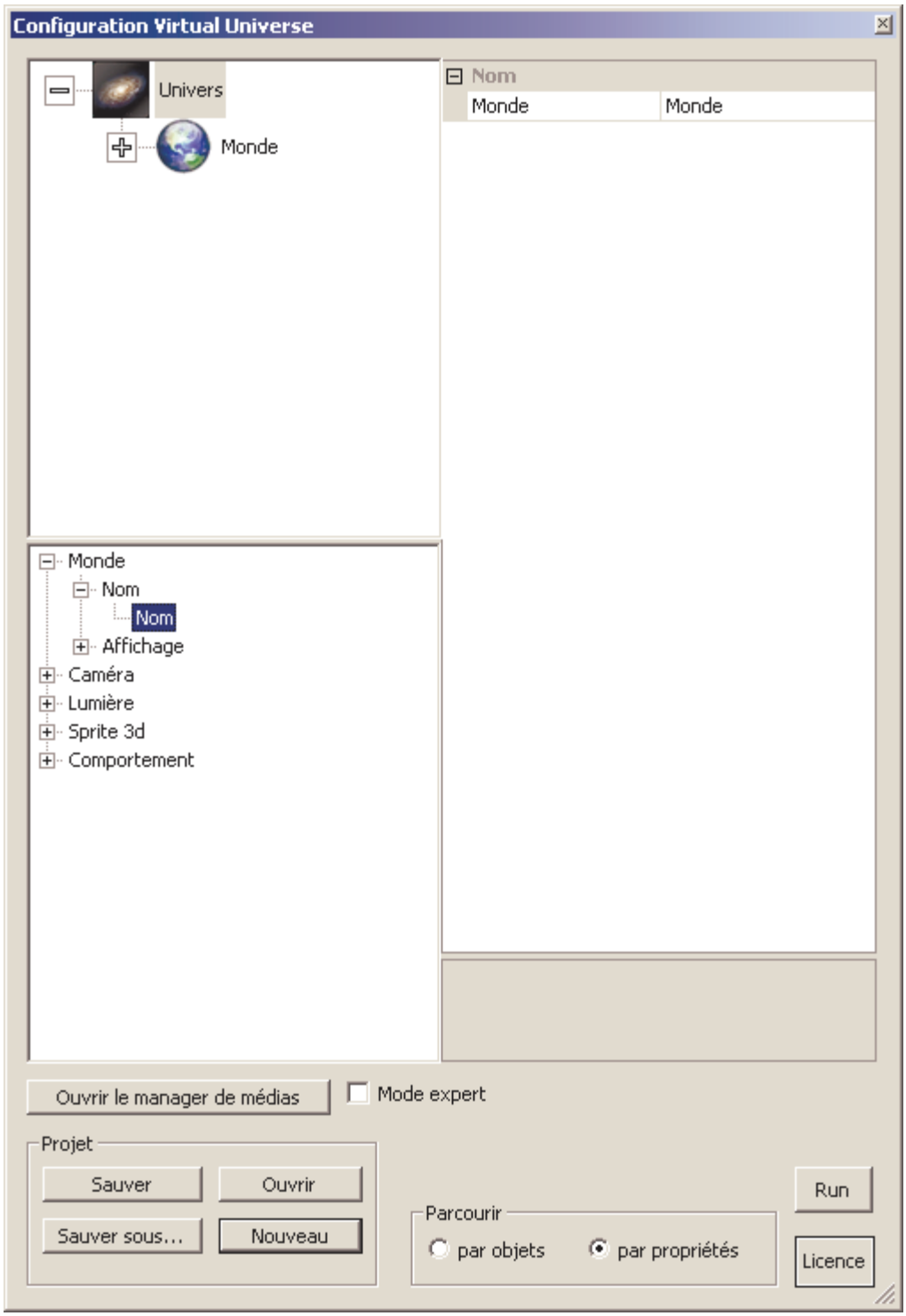
Au lancement de Virtual Universe apparaît la fenêtre de rendu du monde 3D :



Le bouton RUN/STOP permet de lancer ou de stopper la simulation.

Le bouton CONFIGURATION ouvre ou ferme la fenêtre de configuration :


Configuration Virtual Universe
Univers
Monde
Connexion
Driver
None
Mode du driver M340
Simulateur local
Status
Stoppé
Nom du serveur ou ac
localhost
Numéro du port
5000
Dernière erreur
Qualité de la liaison
0
Options
Les différents éléments composant une simulation : caméra, objets, lumières, etc.
Pour chaque élément, sont affichées ici ses propriétés.
Ouvre une zone de stockage intégrée au projet pour les fichiers médias : fichiers 3D, fichiers bitmaps et fichiers sons.
Affiche ou masque les propriétés « avancées » des objets.
Ouvrir le manager de médias
Mode expert
Identique au bouton RUN/STOP de la fenêtre de rendu.
Projet
Sauver
Ouvrir
Sauver sous...
Nouveau
Parcourir
par objets
par propriétés
Run
Licence
Gestion des fichiers projets.
Bascule la représentation du mode liste d'objet au mode liste de propriétés d'un groupe d'objet (voir ci-après)
Accès à la gestion de la licence (voir le chapitre licence).



La même fenêtre en mode « par propriétés » permet d'obtenir pour un groupe d'objets la liste des valeurs d'une même propriété. Dans ce mode, en haut à gauche doit être sélectionné le parent des objets et en bas à gauche la propriété.

Navigation et interactions

Les commandes suivantes permettent de naviguer et dans le Monde 3D ou d'interagir avec celui-ci :

- Molette de la souris ou touches Haut et Bas du clavier : Zoom
- Bouton droit de la souris enfoncé et déplacement de la souris : orbite autour de l'objet sélectionné.
- Déplacement de la souris dans la fenêtre de rendu : sélection automatique de l'objet survolé pour orbiter autour.
- Clic gauche de la souris sur un objet en mode STOP : sélection de l'objet.
- Bouton gauche de la souris enfoncé sur un objet et déplacement de la souris en mode RUN : interaction avec l'objet sélectionné : pousser, tirer, déplacer.

Les différents types d’objets

Les objets sont organisés de façon hiérarchique enfant/parent.

Univers

C’est l’objet parent de tout projet Virtual Universe, il contient un ou plusieurs Mondes, ses propriétés définissent notamment avec quel logiciel d’automatisme dialoguera Virtual Universe. L’objet Univers est toujours le parent de la hiérarchie.

Monde

C’est un sous ensemble de l’Univers. Ses propriétés définissent entre autre l’aspect de la fenêtre de rendu. Les objets Mondes sont toujours enfants de l’Univers.

Caméra

Les Caméras représentent un point de vue de l’utilisateur dans un monde 3D. Les objets caméras sont enfants des objets Mondes ou Sprites 3d.

Lumière

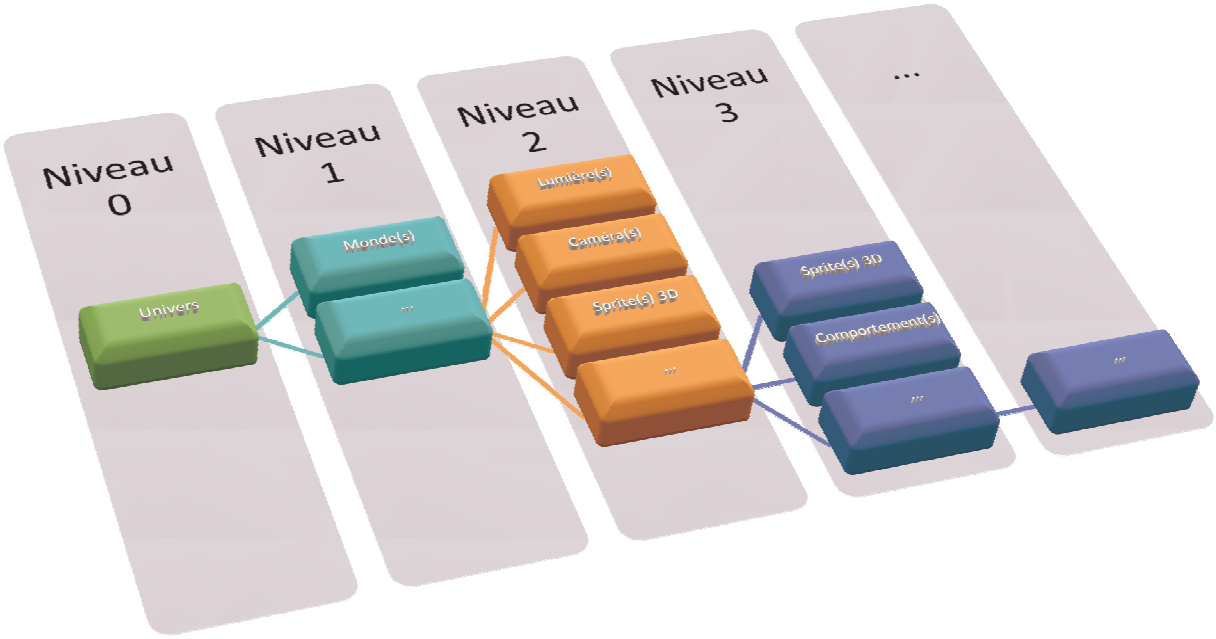
Les Lumières sont nécessaires, comme dans le Monde réel pour pouvoir observer les objets. Les objets Lumières sont enfants des objets Mondes ou Sprites 3d.

Sprites 3D

Ce sont les objets et leurs multiples caractéristiques physiques et visuelles. Les objets Sprites 3D sont enfants des objets Mondes ou Sprites 3D.

Comportements

Associés à un Sprite 3D ou une Lumière, ils vont pouvoir modifier dynamiquement leurs propriétés : modifier leurs positions ou leurs couleurs par exemple ou encore exécuter un script qui pourra agir sur ces objets. Un Comportement pourra agir comme un moteur pour transmettre une force à un Sprite 3D. Les objets Comportements sont enfants des objets Lumières ou Sprites 3D.

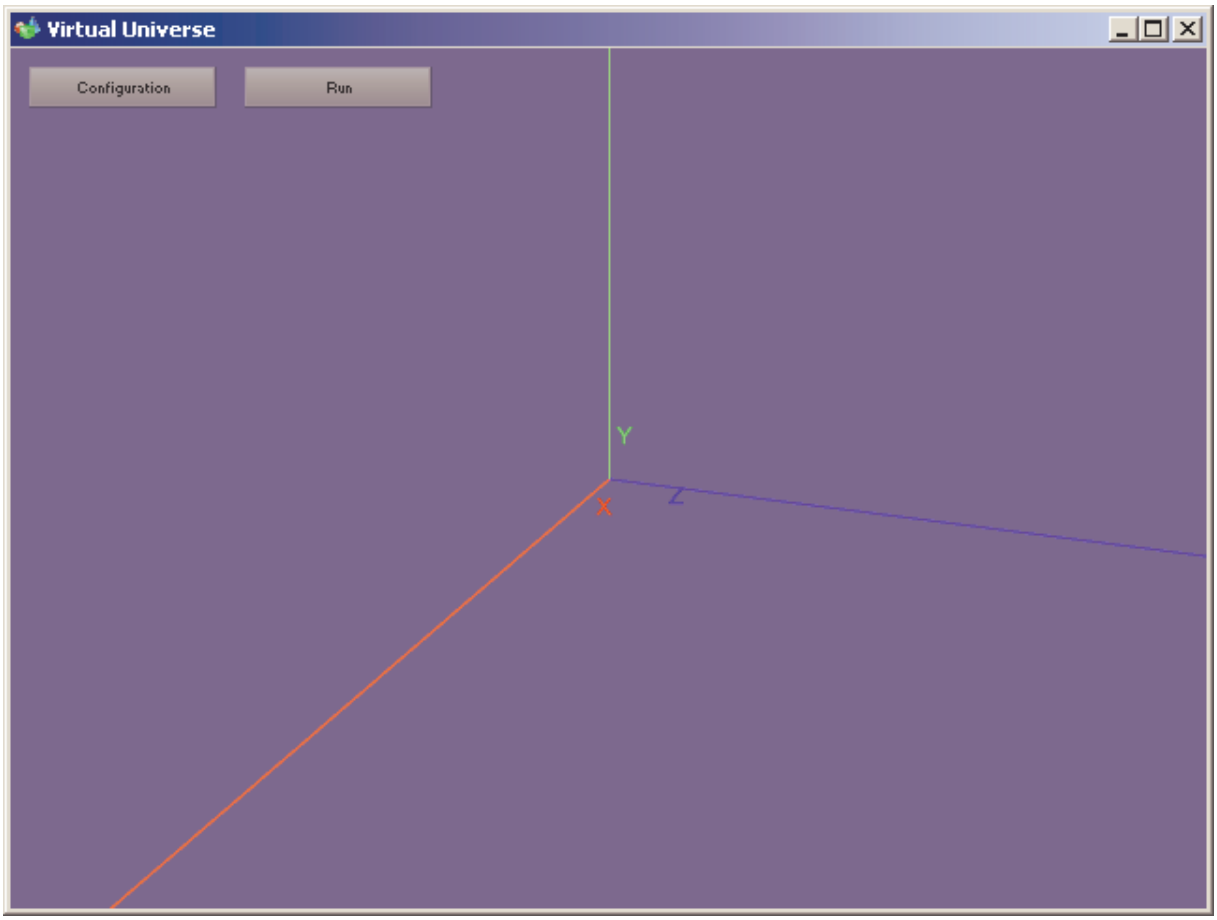


Concepts fondamentaux

Rendu 3D et sons

Le moteur de rendu utilisé par Virtual Universe est Irrlicht qui s'appuie lui-même sur DIRECTX 8 ou 9 ou OPENGL (en fonction de ce qui est disponible sur le PC). Le rôle du moteur de rendu 3D est l'affichage des objets du monde 3D éclairés par les Lumières en fonction du point de vue défini par une caméra. Des coordonnées cartésiennes X/Y/Z régissent le monde 3D.

Un repérage des axes est affiché dans la fenêtre de rendu lorsque la fenêtre de configuration est ouverte.



Les sons 3D augmentent le réalisme des simulations. Les sons sont émis dans le monde virtuel à la position des objets et sont donc perçus en fonction de la position de la caméra.

Moteur physique

Newton Physic Engine est le moteur physique utilisé par Virtual Universe pour la gestion physique des objets : la gravité par exemple, mais aussi beaucoup plus que cela.

Pour tirer pleinement partie du moteur physique, il est important d'être familier avec les concepts de bases de la physique, tels que forces, vitesses, frictions, masse, etc.

Les paramètres du moteur physique sont associés à chaque Sprite 3d. Un Sprite 3d peut être géré ou pas par le moteur physique. Un objet

servant uniquement visuellement par exemple pourra ne pas être géré par le moteur physique.

Dialogue

Le dialogue avec un logiciel externe est un des éléments essentiel permettant de piloter les simulations. Le type du logiciel externe et le paramétrage de la connexion se trouvent dans les propriétés de l'Univers. Les liens sont ensuite définis dans les Comportements. Le type du Comportement déterminera le sens du dialogue (lecture depuis ou écriture vers le logiciel externe).

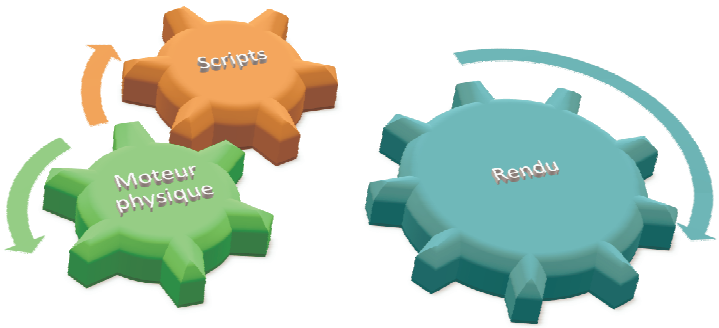
Script

Des scripts écrits en langage basic peuvent être associés à chaque objet par l'intermédiaire d'un Comportement.

Mode RUN/STOP

Virtual Universe peut être en mode STOP (simulation stoppée et initialisée) ou RUN (simulation en cours). En mode RUN, le moteur physique et le dialogue avec le logiciel externe sont activés. Les Comportements et les scripts sont actifs.

En mode RUN, le rendu est effectué aussi vite que possible par rapport aux performances du PC, le moteur physique et les scripts sont appelés toutes les 10ms.



Les objets possèdent un doublon pour certains paramètres (leurs positions par exemple). Le premier jeu de paramètre correspond aux valeurs initiales, le deuxième jeu aux valeurs courantes. En mode STOP, les valeurs initiales sont recopiées dans les valeurs courantes.

Manager de médias

Il permet de stocker les fichiers médias (fichiers 3D, fichiers bitmaps et fichiers sons) utilisés dans un projet. Les objets peuvent utiliser des fichiers se trouvant dans le manager de médias ou en dehors. Les fichiers se trouvant dans le manager de médias seront sauvegardés dans le fichier projet. Cette dernière méthode est conseillée si le projet doit être échangé ou exécuté sur un autre PC.

Propriétés

Univers

Connexion

Driver

Détermine la connexion avec un logiciel externe[1]

Cas de la connexion avec AUTOMGEN ou AUTOSIM

Nom du serveur ou adresse IP

Utilisez « localhost » si le logiciel externe est lancé sur le même PC. Si le logiciel est lancé sur un autre PC du réseau, entrez son adresse IP ou son nom tel que vu sur le réseau.

Port

Doit être le même que celui sélectionné dans les propriétés du projet AUTOMGEN / AUTOSIM dans l'onglet Exécution Connexion TCP/IP, serveur, port.

Cas de la connexion à UNITY (PLC simulateur ou API M340)

[1] Logiciel externe est le terme générique utilisé pour définir le logiciel avec lequel dialogue Virtual Universe

Mode du driver M340

Simulateur local : PLC simulateur lancé sur le même PC, API connecté par USB : un API M340 connecté à un port USB, API ou simulateur sur IP : un API connecté par Ethernet ou un simulateur lancé sur un autre PC connecté au réseau. Dans ce cas, documentez l'adresse IP de l'API ou du PC.

Options

RUN automatique

Provoquera le passage en RUN à l'ouverture du projet.

Affiche les variables et les états

Affichera dans la fenêtre de rendu les noms de variables ainsi que les états pour les Comportements faisant référence à une variable du logiciel externe.

Fil de fer

L'ensemble des Sprites 3D du projet seront affichés en mode « fil de fer » si vrai.

Mode Debug pour le moteur physique

Si vrai, les volumes utilisés par le moteur physique sont affichés dans la fenêtre de rendu (lignes de couleurs jaunes). Ceci est très utile en phase de développement d'un projet utilisant le moteur physique pour visualiser les volumes pris en compte par le moteur physique.

Monde

Nom

Permet de désigner un Monde par son nom.

Affichage

Taille de la fenêtre

Détermine la taille de la fenêtre de rendu en pixels.

Taille modifiable

Si vrai, la taille de la fenêtre peut être modifiée par l'utilisateur.

Couleur du fond
Détermine la couleur affichée comme fond sur la fenêtre de rendu.

Lumière ambiante
Détermine la couleur et l'intensité de la lumière ambiante (lumière illuminant l'ensemble des objets quel que soit leurs positions et leurs orientations).

Montrer les ombres
Si vrai, gère l'affichage des ombres, nécessite que les propriétés des objets relatives aux ombres soit également positionnées. L'affichage des ombres peut ralentir de façon significative le rendu.

Nombre d'images affichées par seconde (lecture seule)
Indique le nombre d'images affichées en une seconde dans la fenêtre de rendu.

Utiliser le shader
Technique d'affichage 3D évolué réservée aux spécialistes.

Nombre maximum d'images par seconde
Si différent de 0, limite le nombre d'images affichées par seconde à la valeur indiquée. Permet de préserver le temps processeur.

Caméra

Nom
Permet de désigner une Caméra par son nom.

Position
Détermine la position initiale de la Caméra par les coordonnées de la cible (ce que regarde la Caméra) ainsi qu'une rotation sur les axes X et Y et un zoom

Position courante
Idem mais pour la position courante.

La position courante peut être recopiée dans la position initiale en cliquant sur les flèches descendantes apparaissant à droite des

éléments de positions initiales et en choisissant « Copier depuis les valeurs courantes ».

Lumière

Nom
Permet de désigner une Lumière par son nom

Position
Permet de définir les coordonnées et la direction de la Lumière (la direction n'est utilisée que pour les Lumières de type Spot et Directionnel.

Couleur, type, etc.
Détermine les caractéristiques de la Lumière.

Sprite 3d

Nom
Permet de désigner un Sprite 3d par son nom

Dessin
Détermine le fichier 3D utilisé pour définir la géométrie du Sprite 3D ainsi que d'éventuels fichiers de texture.

Position et taille
Définit la position, la rotation (ainsi que l'axe) et l'échelle initiale du Sprite 3D. Les rotations sont exprimées en degrés (de -180 à +180 degrés).

Position et taille (valeurs courantes)
Idem mais pour les valeurs courantes, avec en plus : la translation et la rotation relative (par rapport au Sprite 3D parent), ainsi que la position du centre de l'objet et la rotation absolue par rapport au Monde.

Matériel
Ces propriétés regroupent les caractéristiques du matériel utilisé pour afficher l'objet. Ces caractéristiques sont directement liées au moteur de rendu Irrlicht.

Matériel (valeurs courantes)
Idem pour les valeurs courantes.

Navigation

Non sélectionnable

Si vrai, le Sprite 3D n'influe pas sur la navigation lorsqu'il est survolé par le curseur de la souris.

Physique

Regroupe les propriétés de l'objet concernant le moteur physique.

Utilise la physique

Si vrai, le Sprite 3D sera pris en compte par le moteur physique, dans le cas contraire, l'objet sera complètement ignoré par le moteur physique, en d'autres termes, l'objet sera uniquement affiché dans le monde 3D mais n'aura aucune interaction physique avec les autres objets.

Utilise la gravité

Si vrai, le Sprite 3D sera soumis à la gravité. Sa masse devra également être non nulle.

L'utilisateur peut appliquer une force à l'objet

Si vrai, l'utilisateur peut, en mode RUN, agir sur l'objet en maintenant le bouton droit de la souris enfoncé lorsque le curseur est sur le Sprite 3D et en déplaçant le curseur.

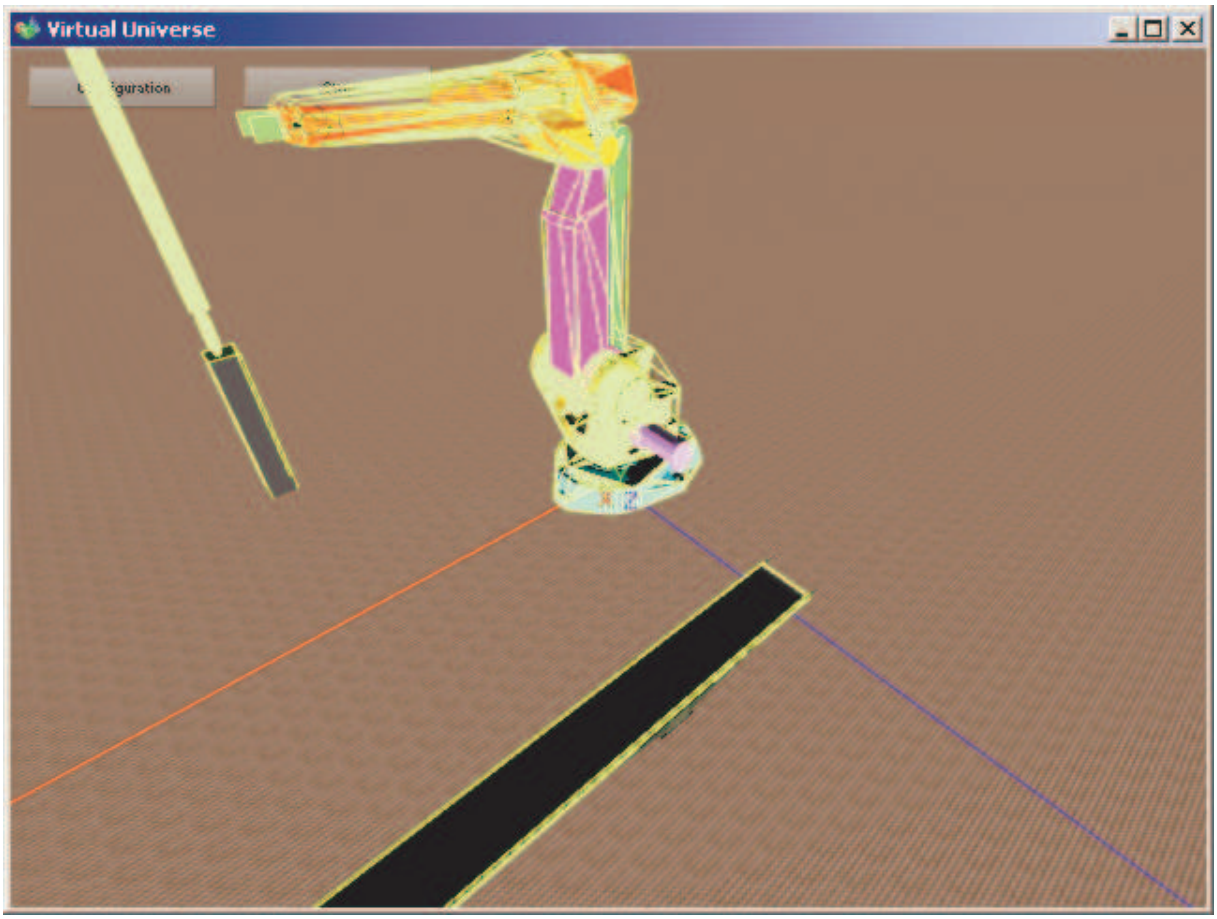
Type de corps

Détermine pour le moteur physique le type de géométrie du Sprite 3D

- Quelconque : une forme convexe déduite de la géométrie du Sprite 3D,
- Boîte : un parallélépipède rectangle,
- Sphère : une sphère,
- Capsule : une capsule.

Attention, le type « Quelconque », s'il est utilisé avec un Sprite 3D complexe (possédant de très nombreuses faces), peut consommer beaucoup de ressources pour la simulation physique. Privilégiez, lorsque ceci est possible un des autres types.

Il est possible et souvent utile dans les phases de mise au point de visualiser les géométries manipulées par le moteur physique en activant l'option « Mode debug pour le moteur physique » dans les propriétés de l'Univers. Exemple :



Les volumes pris en compte par le moteur physique apparaissent en jaune.

Solution dans le cas d'un Sprite 3D complexe nécessitant une forme quelconque :

- définir une forme simplifiée du Sprite 3D (avec moins de facettes), lui donner les attributs « invisible » et « géré par le moteur physique »,
- conserver la forme complexe du Sprite 3D et l'ajouter comme enfant avec les attributs « visible » et « non géré par le moteur physique ».

Cette solution est exploitée dans l'exemple « Convoyeur ».

Solution dans le cas d'un Sprite 3D nécessitant une forme physique concave :

définir plusieurs formes convexes et les lier par des joints.

Masse

La masse de l'objet. Une masse de 0 fige l'objet.

Moment d'inertie

Détermine la quantité d'énergie nécessaire pour faire tourner l'objet sur chacun des axes.

Ajuster automatiquement le centre de masse

Si vrai, le centre de masse de l'objet est automatiquement recalculé en fonction de la géométrie du Sprite 3D. Sinon, le centre de masse est le point de coordonnées 0/0/0 du Sprite 3D.

Coefficients…

Déterminent la friction, l'élasticité et la souplesse des objets. Une valeur de 0 utilise les paramètres par défaut du moteur physique. Le coefficient utilisé par le moteur physique entre un objet A et un objet B est une combinaison (le produit) des coefficients de l'objet A et de l'objet B.

Vitesse

Permet d'accéder aux valeurs de vitesses globales et locales de l'objet. Ces valeurs ne sont disponibles que pour les objets gérés par le moteur physique.

Pénétration

Si vrai, les collisions de l'objet ne sont pas gérées. Dans le cas d'objets liés par des joints (voir ci-après) les collisions sont automatiquement désactivées entre deux objets liés par un joint.

Joint physique avec le parent

Détermine le type de joint entre un Sprite 3D enfant et un son parent. Les deux Sprites 3D doivent être gérés par le moteur physique. Ils peuvent être soumis ou non à la gravité.

Joint

Détermine le type du joint :

- Pivot,

- Glissière
- Fixe.

Position du pivot

Détermine la position x/y/z de la liaison avec l'objet parent pour les liaisons pivot.

Ligne d'action

Détermine la ligne d'action du joint pour les joints Glissière (axe de translation) et Pivot (axe de rotation).

Limites…

Détermine les limites minimales et maximales du joint. Si ces deux valeurs sont égales, alors le joint n'a pas de limite (rotation ou translation sans borne).

Puissance du joint

Détermine la rigidité du joint.

Force du joint

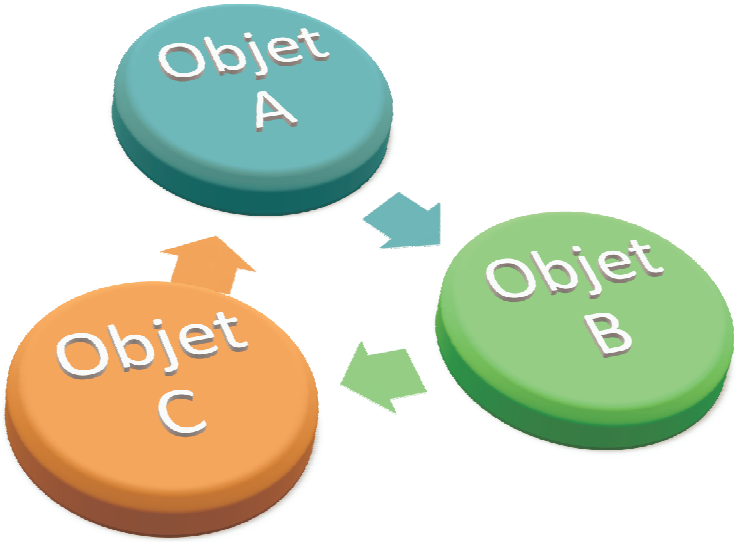
Donne la valeur de la force supportée par le joint.

Force de cassure du joint

Force au-delà de laquelle le joint sera automatiquement détruit. Si 0, cette fonction est désactivée. Ceci permet de simuler la destruction d'une liaison entre deux objets (voir l'exemple « NXT »).

Joint physique avec un autre Sprite 3D

Jeux de paramètres identiques mais le lien est réalisé entre le Sprite 3D et un autre Sprite 3D que le Sprite 3D parent. L'autre Sprite 3D est désigné par son nom. Ce deuxième joint permet de créer des modèles circulaires (voir la simulation « Robot ABB ») :



Sprite 2D

Permet d'afficher un bitmap 2D à la position du Sprite 3D. Voir la gestion de la poussière dans l'exemple du robot aspirateur.

Comportements

Les Comportements sont les éléments qui vont « donner vie » à la simulation. Ils vont aussi définir les liens entre la simulation et les logiciels externes.

Ils sont intimement liés au moteur physique et vont pouvoir communiquer des forces aux Sprites 3D par son intermédiaire et aussi manipuler les données physiques en retour (la vitesse d'un objet par exemple).

Pour une simulation réaliste, des actions par application de forces doivent être privilégiées à des actions modifiant directement la position ou l'orientation des Sprites 3D.

Un Comportement peut aussi être utilisé simplement pour stocker une valeur pendant la simulation. Les scripts pourront accéder à cette valeur en lecture et en écriture. On peut considérer ces Comportements comme « variables globales » de l'application.

Nom

Permet de désigner le Comportement.

Type, etc.

Type de Comportement

Un des types suivants pour les Comportements associés à une Lumière :

- Aucun, le Comportement ne fait rien,
- Ecrit l'intensité de la Lumière,
    - La valeur du Comportement détermine l'intensité lumineuse de la Lumière associée.

Un des types suivants pour les Comportements associés à un Sprite 3D :

- Aucun, le Comportement ne fait rien,
- Applique une force ou un couple,
    - Cet ensemble de types de Comportement permet d'appliquer une force ou un couple à l'objet. Le paramètre Force détermine la direction de la force, le repère peut être global ou local (en fonction du type). La force sera appliquée en

fonction de la valeur courante du Comportement. La force appliquée sera la force paramétrée multipliée par la valeur courante du Comportement.

- Applique une force locale au Sprites 3D en contact,
    - Applique une force à l'ensemble des Sprites 3D en collision avec le Sprite 3D associé. L'application typique est la simulation d'un tapis roulant. Voir l'exemple « Robot ABB » pour une illustration. « Nom(s) des autres Sprites 3D » permet de limiter l'action de ce Comportement à un groupe de Sprites 3D (voir plus loin). Il est conseillé de limiter les tests de collision au strict nécessaire.
- Applique une force ou un couple de frein,
    - Identique mais la force appliquée agira comme frein.
- Définit la force d'attraction du Sprite 3D,
    - Le Sprite 3D attire les autres Sprites 3D. . « Nom(s) des autres Sprites 3D » permet de limiter l'action de ce Comportement à un groupe de Sprites 3D (voir plus loin). « Attraction » permet de définir la force d'attraction, « Distance d'attraction » modifie la zone d'action de cette attraction (infinie si 0). La force d'attraction est aussi proportionnelle au carré de la distance. L'exemple « Manipulateur avec vérins et ventouses » illustre ceci.
- Ecrit la position et la rotation du Sprite 3D,
    - Modifie la position ou l'orientation d'un Sprite 3D. Utilisable par exemple pour ramener un objet en position initiale (voir l'exemple « Convoyeur »).
- Ecrit la position et la rotation des Sprites 3D en collision,
    - Identique mais écrit la position et la rotation de l'ensemble des Sprites 3D en collision avec le Sprite 3D parent. Le paramètre « Nom(s) des autres Sprites 3D » permet de limiter l'action de ce Comportement à un groupe de Sprites 3D (voir plus loin). Il est conseillé de limiter les tests de collision au strict nécessaire. Voir l'exemple « Robot ABB » pour une illustration de ce Comportement.
- Ecrit la couleur ambiante du Sprite 3D,
    - Modifie la couleur ambiante du Sprite 3D parent si la valeur courante du Comportement est différente de 0. La valeur à

appliquer est un des paramètres du Comportement (voir plus loin). Voir l'exemple « Convoyeur » pour une illustration.

- Exécute un script,
    - Exécute un script si la valeur courante du Comportement est différente de 0. Voir le chapitre consacré au Scripts.
- Joue un son,
    - Permet de jouer un fichier son en boucle ou une seule fois. Le son 3D sera perçu comme provenant du Sprite 3D parent. Le son est joué si la valeur courante du Comportement est différente de 0. De plus, la valeur courante du Comportement peut moduler le volume ou la vitesse du son joué. Les exemples illustrent ceci en modulant la vitesse du son joué pour simuler le son des moteurs en fonction de la vitesse de rotation.
- Lecture générique,
    - Le Comportement ne fera que lire une variable du logiciel externe. Cette valeur pourra par exemple être utilisée dans un script.
- Reset,
    - Remet la simulation dans son état initial si la valeur courante du Comportement est différente de 0. L'exemple « Robot et bouteilles » illustre ce type de Comportement.
- Teste la collision avec d'autres Sprites 3D,
    - Permet d'obtenir nombre de collisions entre le Sprite 3D parent et les autres Sprites 3D du Monde courant. Le paramètre « Nom(s) des autres Sprites 3D » permet de limiter l'action de ce Comportement à un groupe de Sprites 3D (voir plus loin). Il est conseillé de limiter les tests de collision au strict nécessaire. Voir l'exemple « Convoyeur » pour une illustration de ce Comportement.
- Teste si le joint est détruit,
    - Permet d'obtenir l'état du joint entre le Sprite 3D associé au Comportement et son parent.
- Obtient la pénétration avec d'autres Sprites 3D,
    - Donne la profondeur de pénétration entre le Sprite 3D associé au Comportement et les autres Sprites 3D. L'utilisation est typiquement le capteur de proximité. Le

paramètre « Nom(s) des autres Sprites 3D » permet de limiter l'action de ce Comportement à un groupe de Sprites 3D (voir plus loin). Il est conseillé de limiter les tests de pénétration au strict nécessaire. Voir l'exemple « NXT » pour une illustration de ce Comportement.

- Obtient des informations sur un Sprite 3D,
  - Permet d'accéder aux valeurs dynamiques d'un Sprite 3D. Le paramètre « Sélectionne l'information à lire… » détermine la valeur.
- Teste la position du joint,
  - Permet de tester si la valeur d'un joint est comprise entre deux bornes. L'utilisation typique est la simulation d'un capteur de position sur un actionneur. « Position min » et « Position max » sont les bornes. L'exemple « Manipulateur avec vérins et ventouses » illustre ceci.
- Ecriture générique,
  - Le Comportement ne fera qu'écrire une variable du logiciel externe. Cette valeur pourra par exemple être calculée dans un script.

Force

Définit la valeur de la force sur chacun des axes pour les types de Comportements concernés.

Appliquer aux frères

Si vrai, le Comportement est appliqué au Sprite 3D concerné et à l'ensemble des frères. Voir l'exemple « Convoyeur » pour une illustration de ce paramètre.

Position / Rotation / Couleur

Valeurs utilisés pour les Comportements les nécessitant.

Liens

Suivant le driver sélectionné dans les propriétés de l'Univers, apparaîtra la définition d'un nom de variable externe spécifique à chaque logiciel externe.

Valeur initiale

Sera recopiée dans la valeur courante au passage en mode RUN de la simulation. Peut être utilisé pour activer de façon permanente un

Comportement. Par exemple, un script pourra être exécuté de façon inconditionnelle dés le lancement de la simulation en mettant cette propriété à 1.

Valeur courante, valeur courante interne, conversion des données, mode d'écriture

Les valeurs du Comportement et le mode de conversion, voir le chapitre « Liens externes » pour plus d'informations.

Noms des autres Sprites 3D

Certains Comportements peuvent utiliser un ensemble de Sprites 3D. Par défaut, si ce paramètre est vide, l'ensemble des Sprites 3D du Monde courant sont concernés. En documentant ce paramètre, la portée du Comportement est réduite aux Sprites 3D dont le nom commence par le texte contenu dans celui-ci. Par exemple, « DUST » réduira les Sprites à ceux dont le nom commence par « DUST ». Voir l'exemple du « Robot aspirateur » pour une illustration de ceci.

Utilise la valeur de ce Comportement

Si non vide, cette zone donne le nom d'un Comportement dont la valeur sera lue et recopiée dans la valeur interne courante. Voir le chapitre « Script » pour plus d'informations sur les conventions de nom pour les Comportements.

Lien externe

Si vrai, le Comportement sera listé dans la liste des liens externes (voir le chapitre « Liens externes »).

Sons

Distance minimale

Permet de modifier le rapport entre le son du volume et la distance de l'objet générant le son et la caméra.

Script

Voir le chapitre suivant

Script

Le concept de Script est un des plus puissants outils de Virtual Universe. Il permet d'intégrer à la simulation des traitements très sophistiqués. Les scripts sont activés par l'intermédiaire des Comportements. Chaque Comportement peut activer un script qui sera une tâche entièrement autonome. Le Script s'exécute tant que la valeur courante du Comportement associé est différente de 0 et que le script n'est pas terminé. Un script se termine si la dernière ligne d'exécution est atteinte ou que le l'instruction END est exécutée. Le langage utilisé est le Basic. Des instructions spécifiques sont utilisables pour accéder en lecture et en écriture aux valeurs associées aux objets.

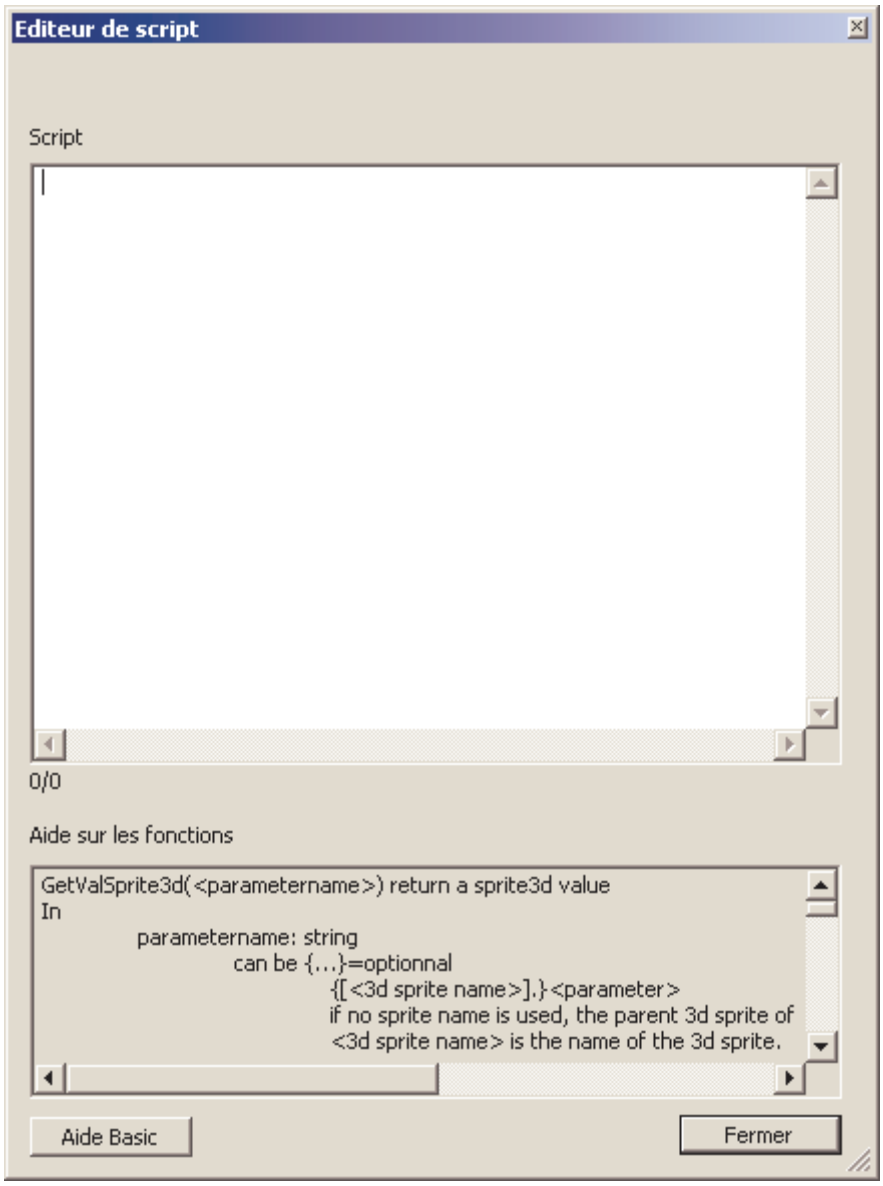
La priorité d'exécution des scripts peut être modifiée dans les propriétés du Comportement associé. La priorité « Normale » correspond à l'exécution d'un élément du script toutes les 10ms., la priorité haute correspond à l'exécution de l'ensemble du script toutes les 10ms. D'autres priorités plus faibles sont également accessibles. Sauf nécessité (ou script court), il est conseillé de ne pas utiliser la priorité haute : plus consommatrice de temps processeur.

Le Script est basé sur le logiciel BeeBasic.

Pour plus d'information, consultez le fichier d'aide basic_api.chm se trouvant dans le répertoire d'installation de Virtual Universe, ou cliquez sur le bouton « Aide Basic ».

Ecrire un script

Les scripts sont écrits dans le paramètre « Script » des Comportements. Une fenêtre d'édition de script s'ouvre.



Une zone d'édition ainsi qu'une aide sur les fonctions spécifiques est affichée. L'élément « Erreur de Script » du Comportement permet d'obtenir une éventuelle erreur rencontrée dans l'analyse du script (le script en question n'est pas exécuté dans ce cas mais la simulation peut quand même passer en mode RUN). Si une erreur est détectée, le numéro de ligne est affiché permettant de localiser l'erreur dans l'éditeur (les numéros de lignes et de colonnes sont affichés en bas de la zone d'édition).

L'élément « Sortie de Script » du Comportement affiche les sorties générées par la fonction basic PRINT. Ces sorties sont également affichées dans la fenêtre de rendu à l'emplacement du Sprite 3D associé au Comportement.

Fonctions spécifiques

Les fonctions spécifiques permettent d'accéder en lecture et en écriture aux valeurs associées aux objets de Virtual Universe.

Syntaxe des noms de Sprites 3D

Le nom pour la référence aux Sprites 3D doit respecter la syntaxe suivante :

- un nom sans chemin d'accès : cherchera le premier Sprite 3D dont le nom commence par ce texte dans l'ensemble des Sprites 3D du Monde courant
- ..\<nom> : un Sprite 3D nommé, frère du Sprite 3D parent du Comportement,
- <nom 1>**\<nom2> : un Sprite 3D nommé nom2, descendant d'un Sprite 3D nommé nom1.
- <chemin d'accès\nom> : un Sprite 3D correspondant au chemin d'accès.

Il n'y a pas de différenciation majuscules/minuscules dans les noms.

Exemples :

mon sprite : désigne le premier Sprite 3D dont le nom commence par le texte « mon sprite ».

..\un autre sprite : désigne le Sprite 3D nommé « un autre sprite » frère du Sprite 3D parent du Comportement.

robot1**\level3 : désigne le Sprite 3D nommé « level3 » descendant du Sprites 3D robot1.

..\..\encore un sprite : désigne le Sprite 3D nommé « encore un sprite » frère du parent du Sprite 3D parent du Comportement.

Remarque : faire référence à des noms relatifs (utilisant des chemins relatifs) plutôt qu'à des noms absolus permet d'avoir des objets

facilement duplicables sans nécessité de modifier les liens. L'exemple « Robot ABB » illustre ceci.

Fonctions d'accès aux valeurs associées à un Sprite 3D

GetValSprite3d(<paramètre>) retourne une valeur associée à un Sprite 3D

<paramètre> désigne le paramètre. Il peut désigner un Sprite 3D par son nom. Si ce n'est pas le cas, c'est le Sprite 3D parent du Comportement qui est utilisé.

La syntaxe est [<nom du Sprite 3D>].<nom du paramètre>

Exemples:

POSX: position X du Sprite 3D parent

[BOX3].SPEEDZ vitesse sur l'axe Z du Sprite 3D nommé BOX3

Liste des paramètres possibles :

POSX, POXY, POSZ : position absolue dans le Monde 3D. ROTX, ROTY, ROTZ : rotation absolue dans le Monde 3D. RELPOSX, RELPOSY, RELPOSZ : position relative par rapport au Sprite 3D parent. Valide uniquement si l'objet est géré par le moteur physique et est lié à un Sprite 3D parent par un joint. RELROTX, RELROTY, RELROTZ : rotation relative par rapport au Sprite 3D parent. Valide uniquement si l'objet est géré par le moteur physique et est lié à un Sprite 3D parent par un joint.

SCALEX,SCALEY, SCALEZ : échelle.

FORCEX, FORCEY, FORCEZ : force appliquée.

TORQUEX, TORQUEY, TORQUEZ : couple appliqué.

FORCEBRAKEX, FORCEBRAKEY, FORCEBRAKEZ : force frein appliquée.

TORQUEBRAKEX, TORQUEBRAKEY, TORQUEBRAKEZ : couple frein appliqué.

SPEEDX, SPEEDY, SPEEDZ : Vitesse

ROTSPEEDX, ROTSPEEDY, ROTSPEEDZ : Vitesse angulaire

RELSPEEDX, RELSPEEDY, RELSPEEDZ : Vitesse relative

RELROTSPEEDX, RELROTSPEEDY, RELROTSPEEDZ : Vitesse angulaire relative (au parent)

TRANSPARENCY : Transparence (de 0=opaque à 1=invisible)

SetValSprite3d(<paramètre>,<valeur>) modifie une valeur associée à un Sprite 3D

<paramètre> est identique à GetValSprite3d avec en plus :

JOINTMIN, JOINTMAX : valeur minimale et maximale du joint avec le parent
JOINTMIN2, JOINTMAX2 : valeur minimale et maximale du deuxième joint

Syntaxe des noms de Comportements

Le nom pour la référence aux Comportements doit respecter la syntaxe suivante :

- un nom sans chemin d'accès : cherchera le premier Comportement dont le nom commence par ce texte dans l'ensemble Comportements du Monde courant
- ..\<nom> : un Comportement nommé frère du Comportement courant,
- <nom de sprite>\<nom de Comportement> : un Comportement nommé enfant d'un Sprite 3D. Le nom du Sprite 3D doit répondre aux critères définis au chapitre « Syntaxe des noms de Sprites 3D »

Fonctions d'accès aux valeurs associées à un Comportement

GetBehavior(<paramètre>) retourne une valeur associée à un Comportement

<paramètre> peut être un nom de Comportement ou un nom de Comportement et le type de valeur.

La syntaxe est :

[<nom du Comportement>].<type de valeur>

Ou

[<nom du Comportement>]

Si le type de valeur est omis, c'est la valeur courante qui est référencée.

Le type de valeur possible est « internalvalue » pour accéder à la valeur courante interne.

Exemples :

[MON COMPORTEMENT] : valeur courante du Comportement nommé « MON COMPORTEMENT ».

[MON CHIEN].internalvalue : valeur courante interne du Comportement nommé « MON CHIEN ».

[robot1**\level2\position] : valeur courante du Comportement nommé « position » enfant du Sprite 3D nommé « level2 » descendant du Sprite 3D nommé « robot1 ».

[..\..\..\request].internalvalue : valeur courante interne du grand parent du parent du Comportement.

SetBehavior(<paramètre>,<valeur>) écrit la valeur d'un Comportement
<paramètre> est identique à GetBehavior.

Fonctions d'accès aux valeurs associées à l'Univers

GetUniverse(<paramètre>) retourne une valeur associée à l'Univers
<paramètre> peut être :

- RUNNINGDURING : retourne la durée en ms depuis le dernier passage en RUN de la simulation
- MOUSEBUTTONS : retourne l'état des boutons de la souris : bit 0 pour le bouton gauche, bit 1 pour le bouton droit, bit 2 pour le bouton du milieu.

- MOUSEX, MOUSEY : retourne la position du curseur de la souris relative au coin supérieur gauche de la fenêtre de rendu.

SetUniverse(<paramètre>,<valeur>) écrit une valeur associée à l'Univers
<parameter> peut être

PLEASEQUIT : force la terminaison de Virtual Universe

Autres fonctions

GetFirstSprite3D(<name>) retourne le premier numéro d'un Sprite 3D
Le nom doit respecter la syntaxe des noms de Sprites 3D. La valeur numérique retournée peut directement être passée en paramètre aux fonctions d'accès aux valeurs associées aux Sprites 3D sous la forme « #numéro ». Si la valeur retournée est inférieure à 0, aucun Sprite 3D n'a été trouvé. Voir l'exemple « Robot Aspirateur » pour une illustration.

GetNextSprite3D(<numéro>,<name>) retourne le numéro du Sprite 3D suivant.
<numéro> est la valeur retournée par GetFirstSPrite3d.

Si la valeur retournée est inférieure à 0, aucun Sprite 3D n'a été trouvé.

Rand() : retourne une valeur aléatoire comprise entre 0 et 1
Voir l'exemple « Robot aspirateur ».

ComputeIK(<ndof>,<x>,<y>,<z>,<a>,<b>,<c>,<tx>,<ty>,<tz>,<b1>,<b2>,<b3>,<b4>,<b5,<b6>) calcul la résolution cinématique inverse d'un robot

Le Comportement associé doit être enfant du Sprite 3D composant le dernier élément du robot.

Voir l'exemple « Robot ABB ».

<ndof> : nombre de degrés de liberté (doit être 6),

<x,y,z> : position à atteindre,

<a,b,c> : angle souhaité pour le dernier élément,

<tx,ty,tz> : déplacement de l'outil,

<b1> à <b6> : nom de 6 Comportements qui recevront les valeurs pour chacun des 6 axes.

La valeur de retour est :

0 : pas d’erreur, les valeurs ont été calculées,

-7 : la position ne peut pas être atteinte,

Autre valeur < à 0 : erreur.

Bibliothèque d’objets

Il est possible d’importer et d’exporter des objets « complexes » composées de Sprites 3D, de Lumières et de Comportements.

L’importation d’objets complexes est réalisée en cliquant sur un Monde ou un Sprite 3D avec le bouton droit de la souris et en choisissant « Importer ». Des exemples d’objets se trouvent dans le sous-répertoire « library » du répertoire d’installation de Virtual Universe.

L’exportation est réalisée en cliquant avec le bouton droit de la souris sur un Sprite 3D et en choisissant « Exporter ». L’ensemble des éléments « Enfants » sont exportés.

Liens externes

Les liens externes permettent le pilotage des simulations créés dans Virtual Universe par un logiciel externe (AUTOMGEN par exemple).

Le type de la connexion est défini dans les propriétés de l’Univers.

Les échanges sont activés dés que Virtual Universe est en mode RUN et que le logiciel externe est capable d’effectuer ces échanges.

L’état de la connexion est affiché dans les propriétés de l’Univers.

Un lien est établi entre le logiciel externe et un Comportement.

Suivant le logiciel externe sélectionné, un nom de variable spécifique peut être documenté dans chaque Comportement.

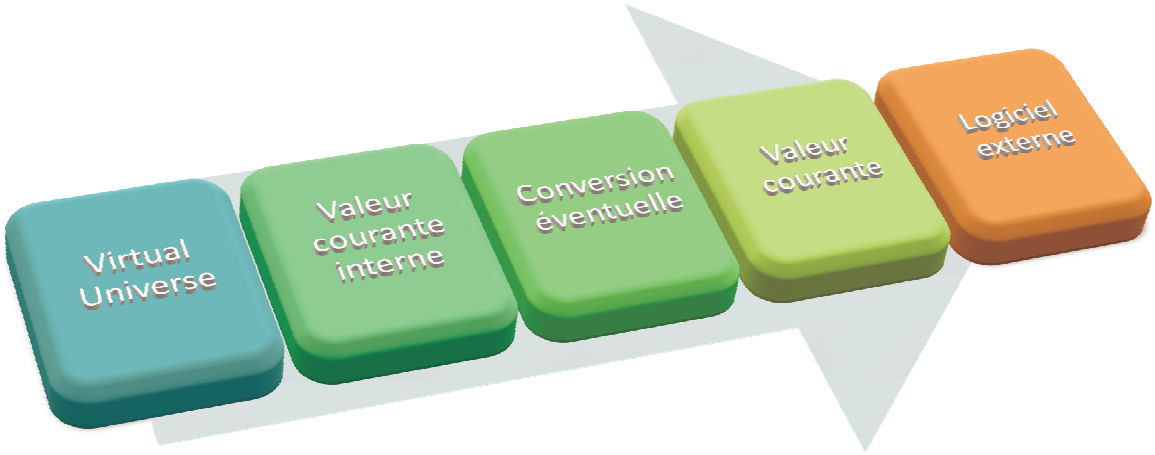
Valeur courante et valeur courante interne

Chaque Comportement possède deux états : un état courant et un état courant interne. Ces deux états sont utilisés d’une manière différente et inverse suivant le sens de l’information : du logiciel externe vers Virtual Universe ou de Virtual Universe vers le logiciel externe.

Lecture d’une valeur de Virtual Universe depuis le logiciel externe

Cette action peut être décrite comme « lecture d’une entrée » du point de vue du logiciel externe.

Le cheminement des données est le suivant :



Le type de conversion peut être une simple recopie de la valeur ou une inversion (pour les variables booléennes complémentées).

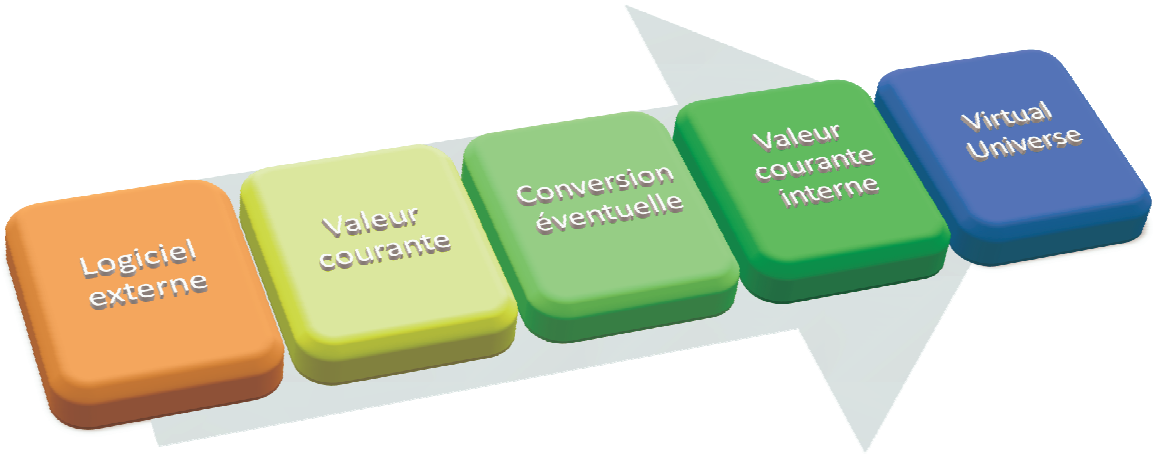
En complément, un « mode d’écriture vers le logiciel externe » peut être spécifié. Trois modes sont disponibles :

- « Normal » : la valeur est écrite à chaque échange,
- « Seulement sur changement » : la valeur n’est écrite vers le logiciel externe que si elle a changée (l’écriture vers certains logiciels externes peut être consommatrice de ressources, cette option à pour but d’amoindrir l’impact des écritures en terme de ressources) ;
- « Sécurisé » : la valeur est écrite à chaque changement. Chaque écriture est vérifiée (lecture de la valeur après écriture). Ce mode garantit qu’un changement fugitif d’état (typiquement un capteur vu vrai pendant une durée très courte – inférieur au temps d’échange des données entre Virtual Universe et le logiciel externe – sera « vu » par le logiciel externe. Ceci est exploité dans l’exemple « Convoyeur ».

Ecriture d’une valeur du logiciel externe vers Virtual Universe

Cette action peut être décrite comme « écriture d’une sortie » du point de vue du logiciel externe.

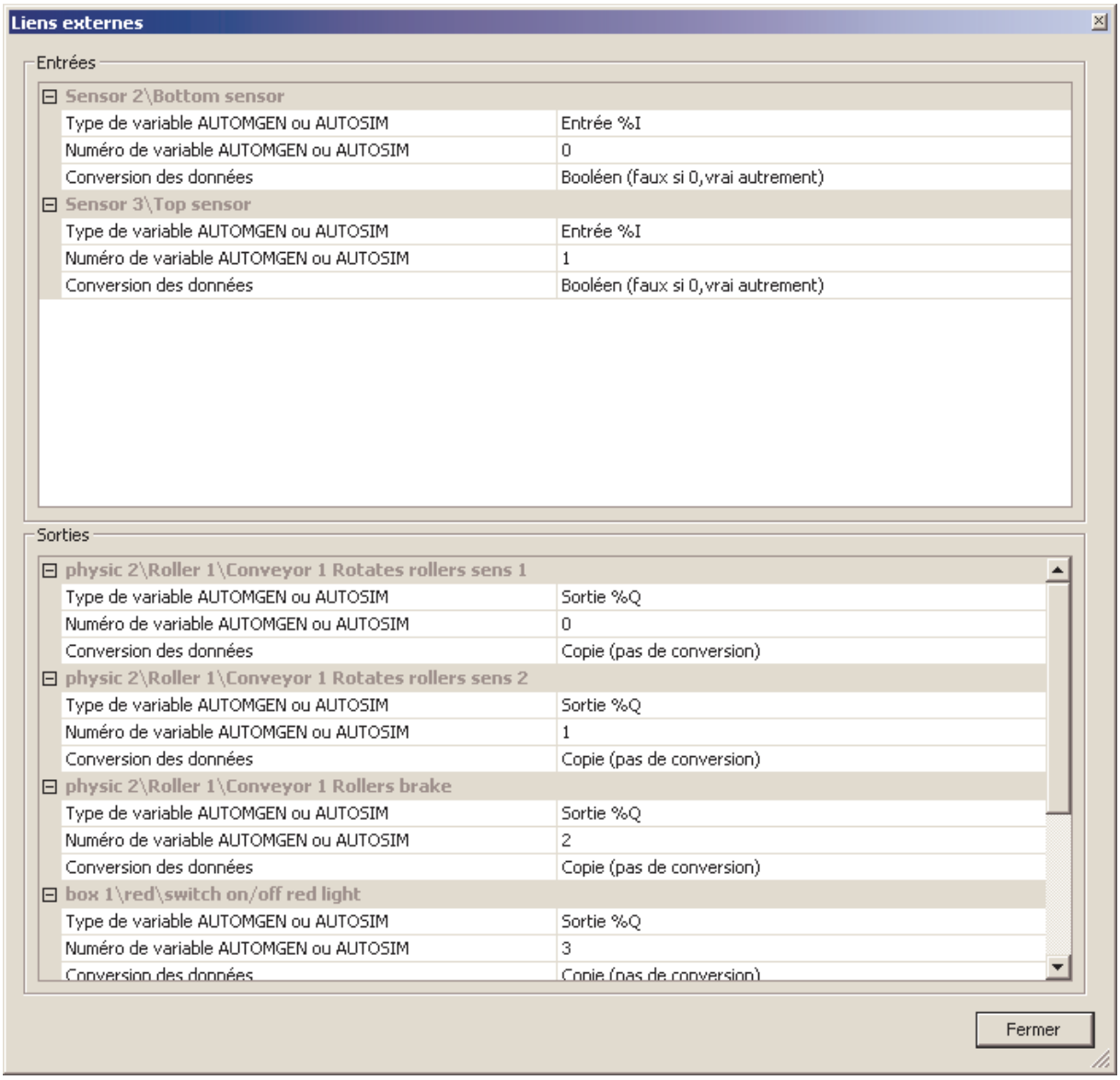
Le cheminement des données est le suivant :



Le type de conversion peut être une simple recopie de la valeur ou une inversion (pour les variables booléennes complémentées).

Accès aux liens externes d’un groupe d’objets

Il est possible d’accéder facilement à la totalité des entrées sorties associées à un groupe d’objets en cliquant avec le bouton droit de la souris sur le parent (cliquez sur le Monde pour avoir la totalité des liens des objets se trouvant dans un Monde) et en sélectionnant « Liens externes ». Exemples :

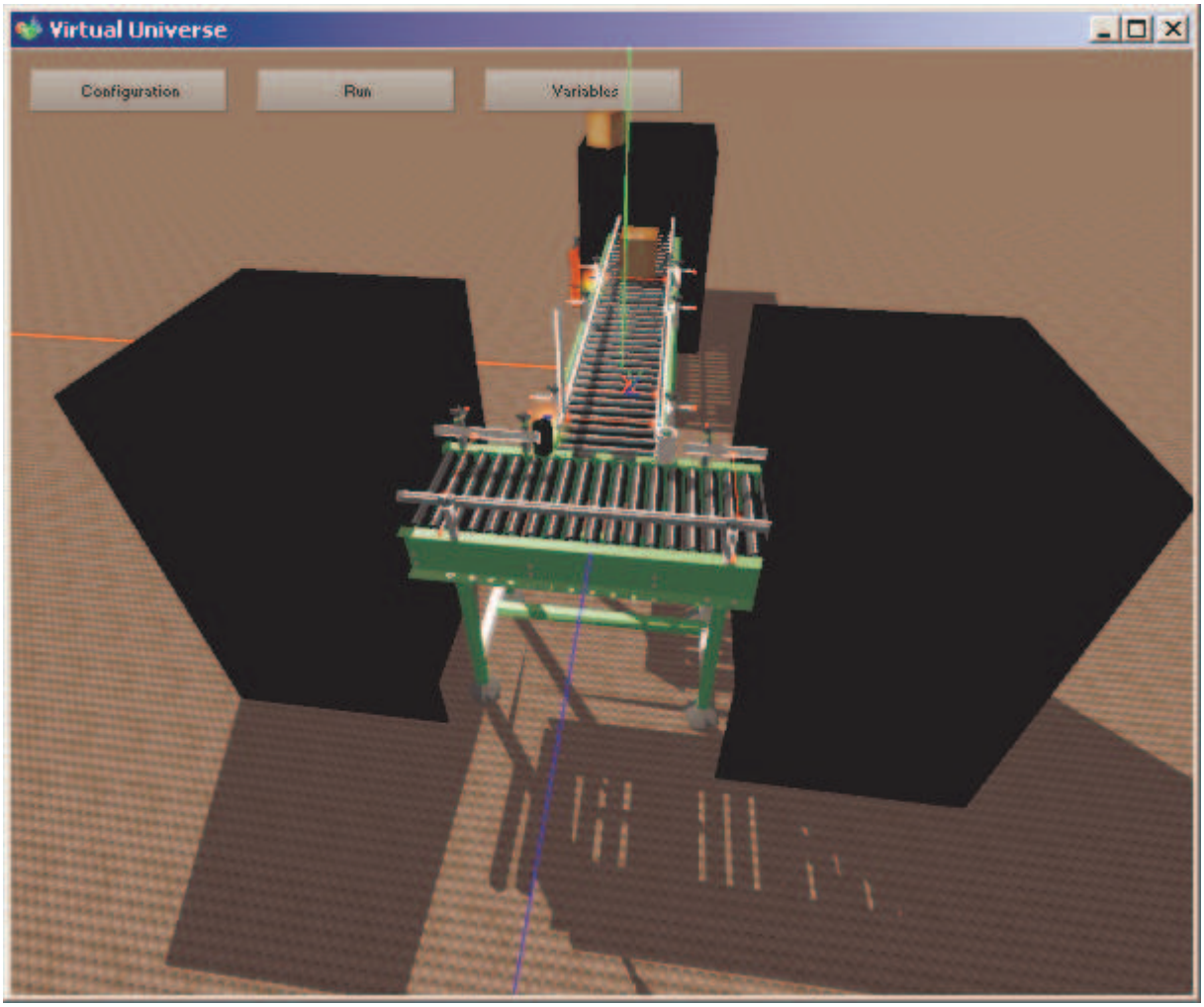


L’application typique de ceci est de modifier l’attribution des entrées sorties d’un objet après importation ou duplication. Les variables associés aux entrées sorties dépendent du type de driver (type de logiciel externe) sélectionné dans les propriétés de l’Univers.

Exemples

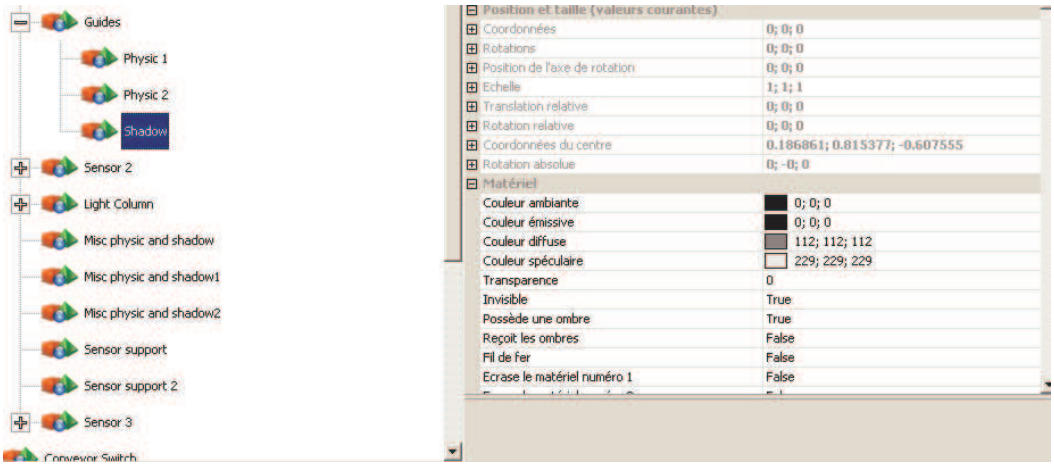
Convoyeur

Le projet convoyeur se trouve dans le sous-répertoire « samples\conveyor » du répertoire d'installation de Virtual Universe. Il est accompagné d'un projet .AGN pour les logiciel AUTOMGEN ou AUTOSIM et d'un projet .XEF pour le logiciel Unity Pro.

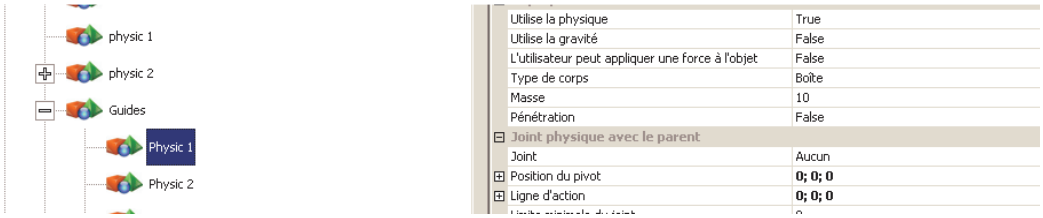


Cet exemple illustre notamment les fonctionnalités suivantes :

- Gestion des ombres : ici des Sprites 3D « allégés » (avec moins de faces) ont été utilisés pour le rendu des ombres afin de ne pas trop ralentir le rendu.

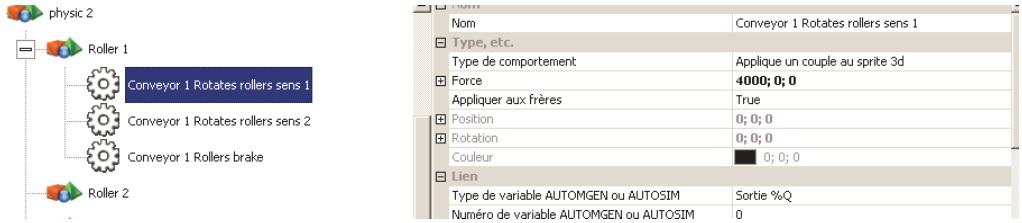


- Gestion physique : le même principe a été utilisé pour la gestion « physique ».

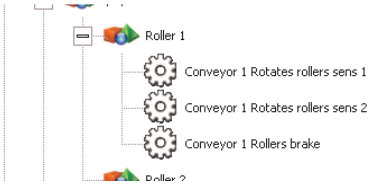


-

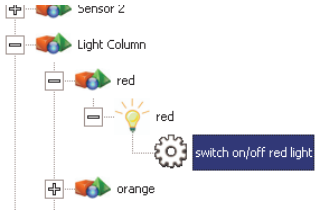
- Applications d’un Comportement à un groupe d’objets : le couple appliqué aux rouleaux de chaque convoyeur est généré par un seul Comportement avec l’attribut « appliquer aux objets frères ».



- Simulation marche avant, marche arrière et frein pour les convoyeurs.

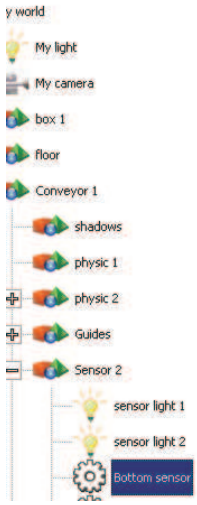


- Simulation d’une colonne lumineuse.



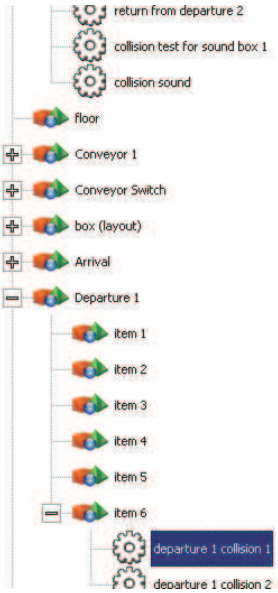
Type de variable AUTOMGEN ou AUTOSIM	Sortie %Q
Numéro de variable AUTOMGEN ou AUTOSIM	3
Valeur initiale	0
Valeur courante	0
Conversion des données	Copie (pas de conversion)
Valeur courante interne	0
Nom(s) des autres sprites 3d	
Mode d'écriture	Normal
Utilise la valeur de ce comportement	
Nom de variable M340	%m103
Valeur externe	0

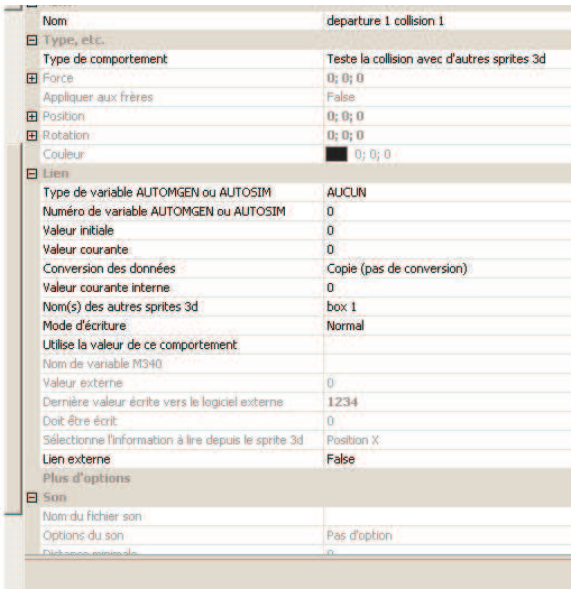
- La gestion des capteurs a été réalisée avec des Comportements « Test de collision ». La gestion de la « furtivité » de l’information provenant des capteurs a été traitée avec un mode d’écriture « sécurisé » pour que le logiciel externe puisse « voir » l’information de façon certaine.
-

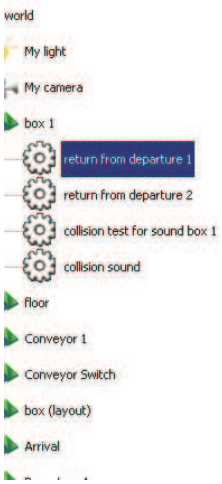


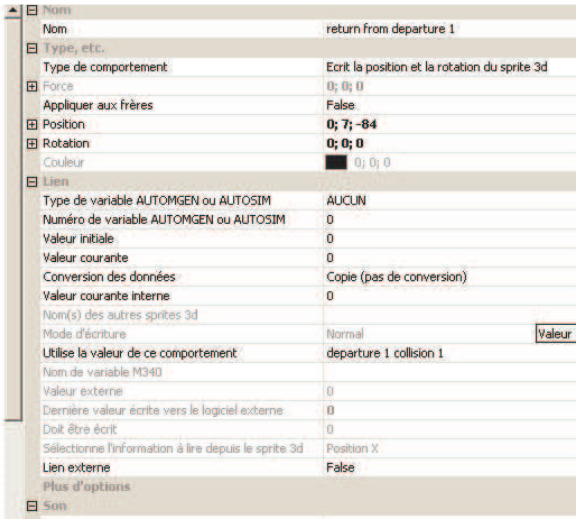
Nom	Bottom sensor
Type, etc.	
Type de comportement	Teste la collision avec d'autres sprites 3d
Force	0; 0; 0
Appliquer aux frères	False
Position	0; 0; 0
Rotation	0; 0; 0
Couleur	0; 0; 0
Lien	
Type de variable AUTOMGEN ou AUTOSIM	Entrée %I
Numéro de variable AUTOMGEN ou AUTOSIM	0
Valeur initiale	0
Valeur courante	0
Conversion des données	Booléen (Faux si 0, vrai autrement)
Valeur courante interne	0
Nom(s) des autres sprites 3d	box
Mode d'écriture	Sécurisé
Utilise la valeur de ce comportement	
Nom de variable M340	%m0
Valeur externe	0
Dernière valeur écrite vers le logiciel externe	1234
Doit être écrit	0
Sélectionne l'information à lire depuis le sprite 3d	Position X
Lien externe	True
Plus d'options	
Son	
Nom du fichier son	

- Le retour des boîtes à la position de départ est géré par des couples de Comportements « Test de collision » et « Ecriture de la position ».

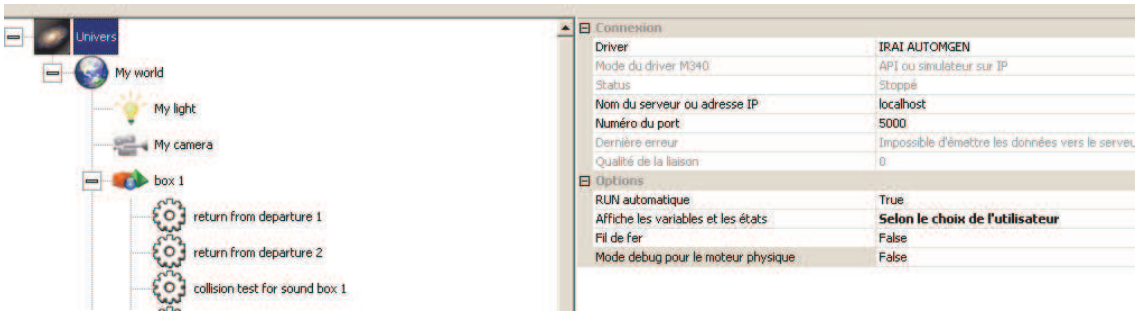




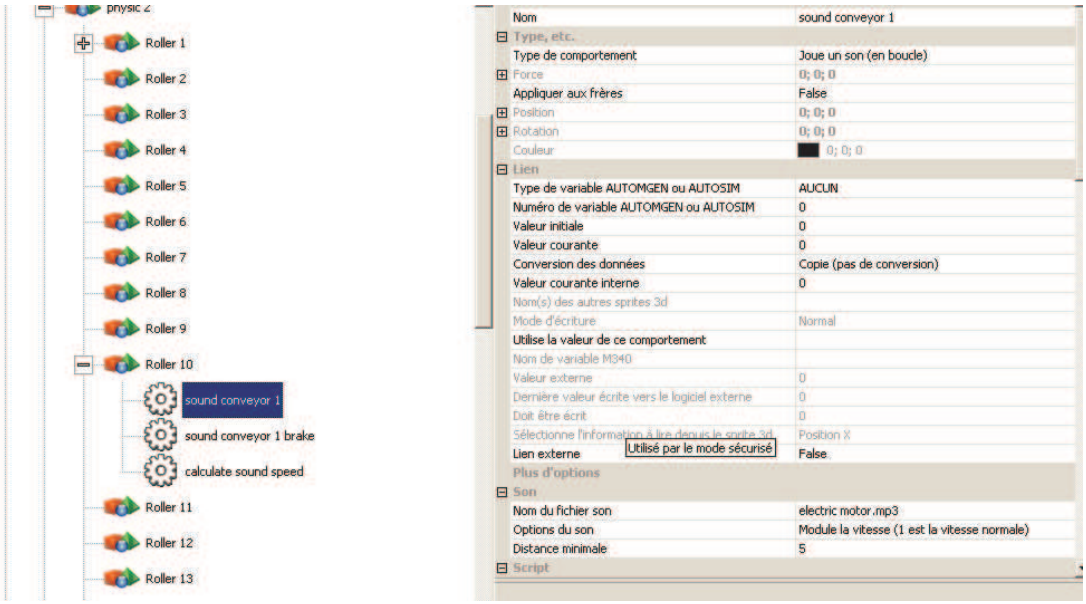




- Les noms et états des variables sont affichables sur la fenêtre de rendu (choix de l'utilisateur sélectionné dans les propriétés de l'Univers).



- Modulation de la vitesse des sons en fonction de la vitesse de rotation des rouleaux. Calcul réalisé dans un script.



Fonctionnement

Les boîtes sont de deux tailles différentes, le but est de les évacuer vers deux destinations différentes en fonction de leur taille. Deux capteurs (un capteur bas et un capteur haut) permettent d'identifier la taille des boîtes. Capteur bas seul = petite boîte, capteur bas et capteur haut = grande boîte.

Liste des références de variables AUTOMGEN / AUTOSIM

Symbole	Variable	Commentaire
before switch low sensor	%i0	capteur bas avant le convoyeur de tri
before switch high sensor	%i1	capteur haut avant le convoyeur de tri
departure sensor backward	%i2	capteur départ arrière convoyeur de tri
departure sensor forward	%i3	capteur départ avant convoyeur de tri
arrival sensor	%i4	capteur convoyeur d'arrivée
middle conveyor forward	%q0	moteur convoyeur du milieu en avant
middle conveyor backward	%q1	moteur convoyeur du milieu en arrière
middle conveyor brake	%q2	frein convoyeur du milieu
middle conveyor red light	%q3	lumière rouge colonne lumineuse convoyeur du milieu
middle conveyor orange light	%q4	lumière orange colonne lumineuse convoyeur du milieu
middle conveyor green light	%q5	lumière verte colonne lumineuse convoyeur du milieu
switch conveyor forward	%q6	moteur convoyeur de tri en avant
switch conveyor backward	%q7	moteur convoyeur de tri en arrière
switch conveyor brake	%q8	frein convoyeur de tri
arrival conveyor forward	%q9	moteur convoyeur d'arrivée en avant
arrival conveyor backward	%q10	moteur convoyeur d'arrivée en arrière
arrival conveyor brake	%q11	frein moteur convoyeur
arrival conveyor orange light	%q12	lumière orange colonne lumineuse convoyeur d'arrivée
arrival conveyor red light	%q13	lumière rouge colonne lumineuse convoyeur d'arrivée
arrival conveyor green light	%q14	lumière verte colonne lumineuse convoyeur d'arrivée

Entrée

Sortie

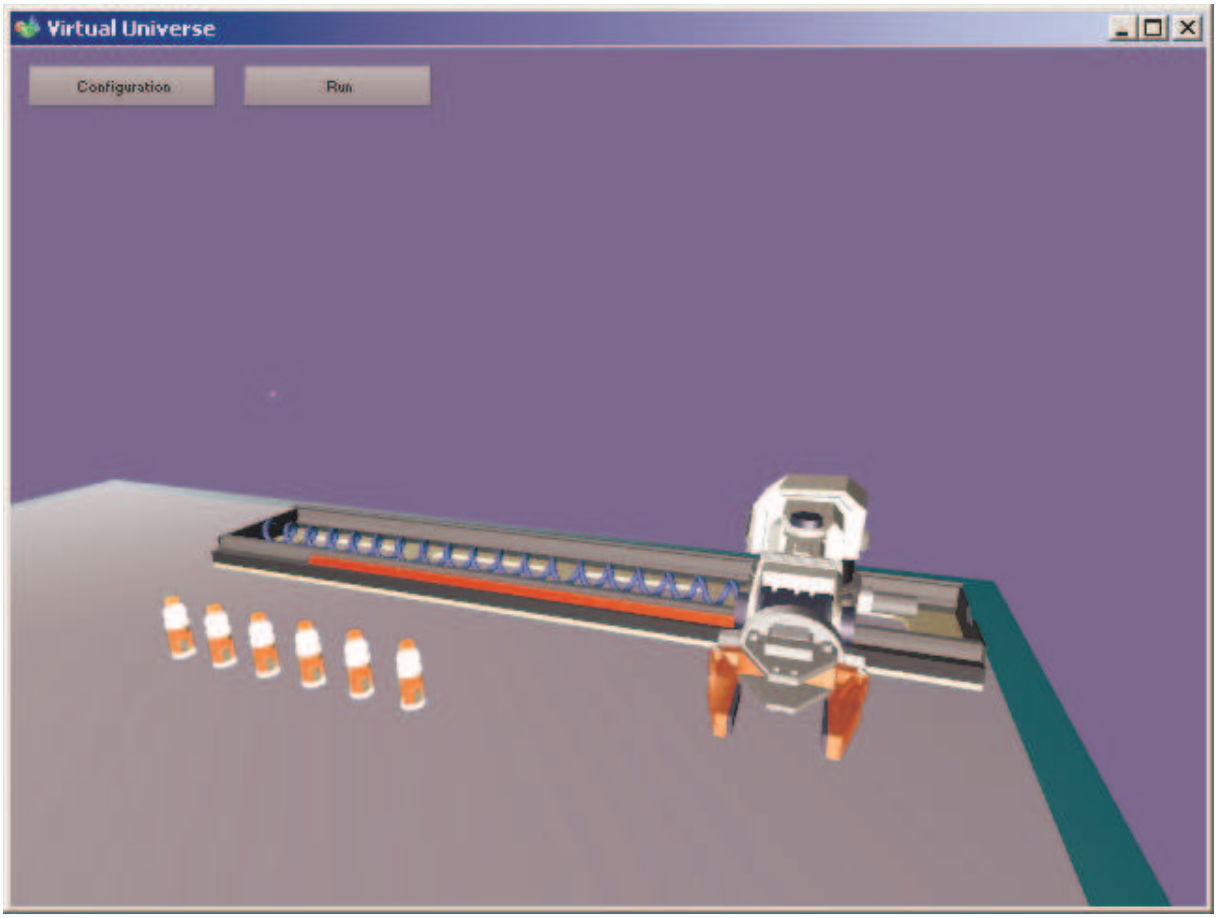
Liste des références de variables UNITY PRO

before_switch_low_sensor	EBOOL	%I0.1.0
before_switch_high_sensor	EBOOL	%I0.1.1
departure_sensor_backward	EBOOL	%I0.1.2
departure_sensor_forward	EBOOL	%I0.1.3
arrival_sensor	EBOOL	%I0.1.4

middle_conveyor_forward	EBOOL	%Q0.2.0
middle_conveyor_backward	EBOOL	%Q0.2.1
middle_conveyor_brake	EBOOL	%Q0.2.2
middle_conveyor_red_light	EBOOL	%Q0.2.3
middle_conveyor_orange_light	EBOOL	%Q0.2.4
middle_conveyor_green_light	EBOOL	%Q0.2.5
switch_conveyor_forward	EBOOL	%Q0.2.6
switch_conveyor_backward	EBOOL	%Q0.2.7
switch_conveyor_brake	EBOOL	%Q0.2.8
arrival_conveyor_forward	EBOOL	%Q0.2.9
arrival_conveyor_backward	EBOOL	%Q0.2.10
arrival_conveyor_brake	EBOOL	%Q0.2.11
arrival_conveyor_red_light	EBOOL	%Q0.2.12
arrival_conveyor_orange_light	EBOOL	%Q0.2.13
arrival_conveyor_green_light	EBOOL	%Q0.2.14

Robot et bouteilles

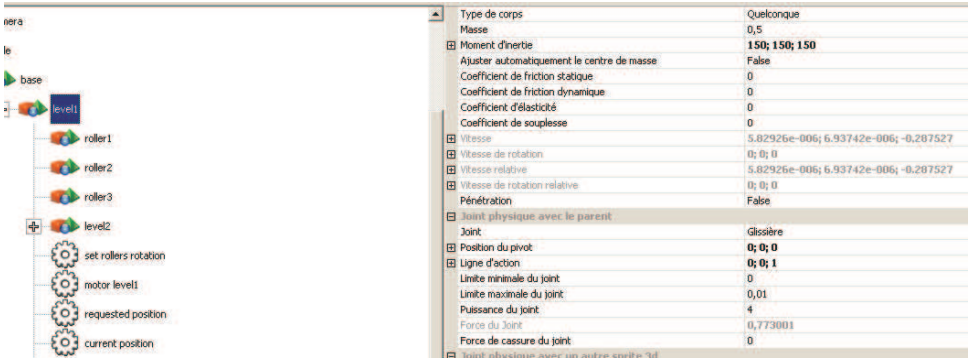
Ce projet se trouve dans le sous-répertoire « samples\robot and bottles » du répertoire d'installation de Virtual Universe. Il est accompagné d'un fichier .AGN.



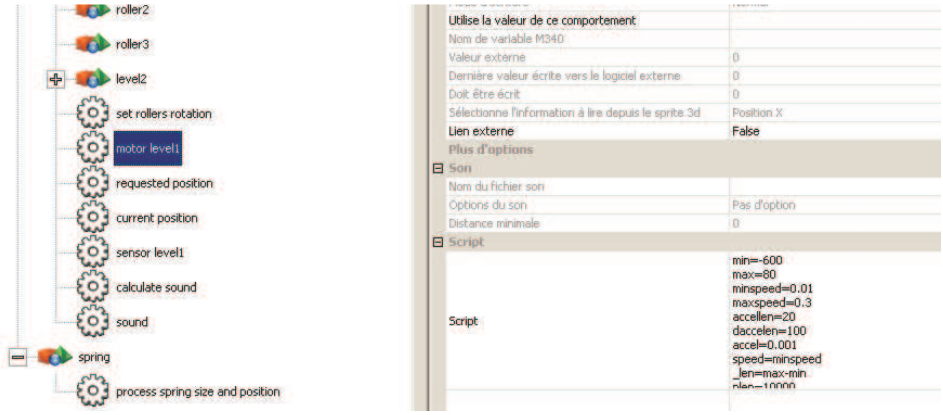
Ce projet illustre notamment les fonctionnalités suivantes :

- Modification de l'échelle et de la position pour la simulation du ressort,

- Définition de joints,

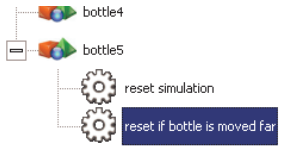


- Simulation des moteurs du robot par modification des limites des joints. Pilotage par variables numériques (une variable associée à chaque axe) géré par un script pour chaque moteur.

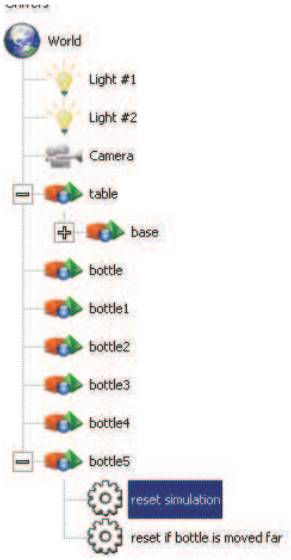


- Préhension : rien de particulier à faire, la fermeture de la pince est suffisante.

- RAZ de la simulation après le déplacement de la dernière bouteille.



Dernière valeur écrite vers le logiciel externe	0
Doit être écrit	0
Sélectionne l'information à lire depuis le sprite 3d	Position X
Lien externe	False
Plus d'options	
Son	
Nom du fichier son	
Options du son	Pas d'option
Distance minimale	0
Script	
Script	if getvalsprite3d("POSZ")>500 then setbehavior("reset simulatio.internalvalue",1) endif



Nom	reset simulation
Type, etc.	
Type de comportement	Reset (STOP puis RUN)
Force	0; 0; 0
Appliquer aux frères	False
Position	0; 0; 0
Rotation	0; 0; 0
Couleur	0; 0; 0
Lien	
Type de variable AUTOMGEN ou AUTOSIM	AUCUN
Numéro de variable AUTOMGEN ou AUTOSIM	0
Valeur initiale	0
Valeur courante	0
Conversion des données	Copie (pas de conversion)
Valeur courante interne	0
Nom(s) des autres sprites 3d	
Mode d'écriture	Normal
Utilise la valeur de ce comportement	
Nom de variable M340	
Valeur externe	0
Dernière valeur écrite vers le logiciel externe	0
Doit être écrit	0
Sélectionne l'information à lire depuis le sprite 3d	Position X
Lien externe	False
Plus d'options	
Son	
Nom du fichier son	
Options du son	Pas d'option
Distance minimale	0

Fonctionnement

Pour chacun des axes, deux variables numériques sont utilisées. Une donne la position courante, l'autre permet de définir la position à atteindre. Si la position courante est proche de la position demandée, alors le mouvement a été effectué.

Liste des références de variables AUTOMGEN / AUTOSIM

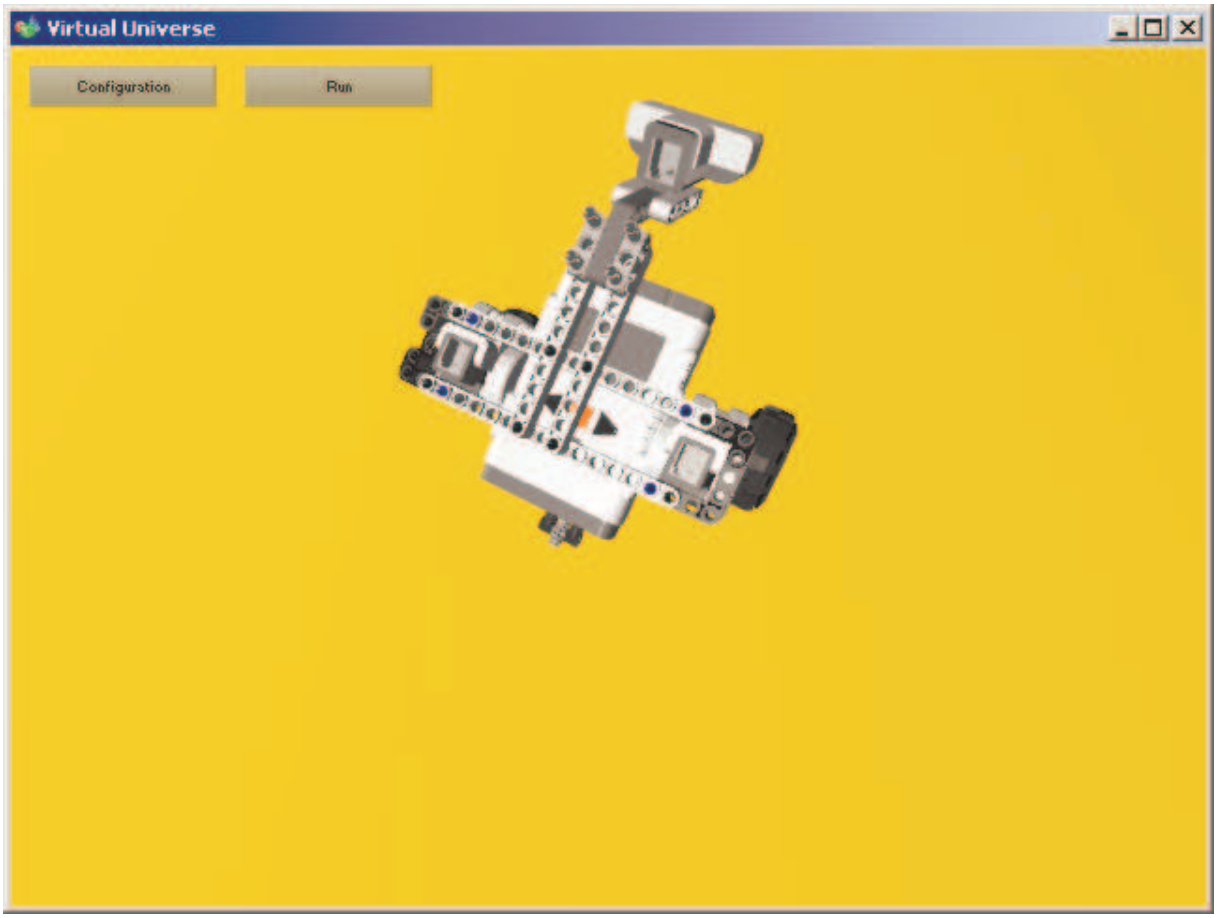
Symbole	Variable	Commentaire
request level 1	%mw200	Position demandée pour l'axe 1
position level 1	%mw201	Position courante pour l'axe 1
…		
request level 5	%mw208	Position demandé pour l'axe 5
position level 5	%mw209	Position courante pour l'axe 5
request finger 1	%mw210	Position demandée pour le doigt 1 de la pince
position finger 1	%mw211	Position courante pour le doigt 1 de la pince
request finger 2	%mw212	Position demandée pour le doigt 2 de la pince
position finger 2	%mw213	Position courante pour le doigt 2 de la pince

Entrée

Sortie

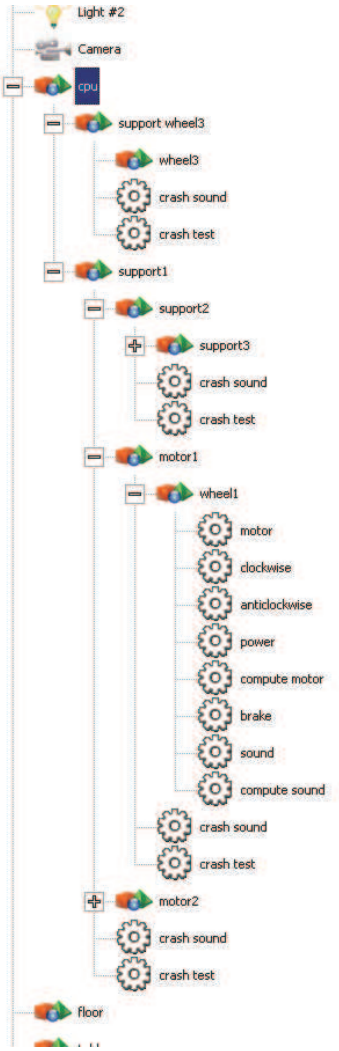
Robot NXT

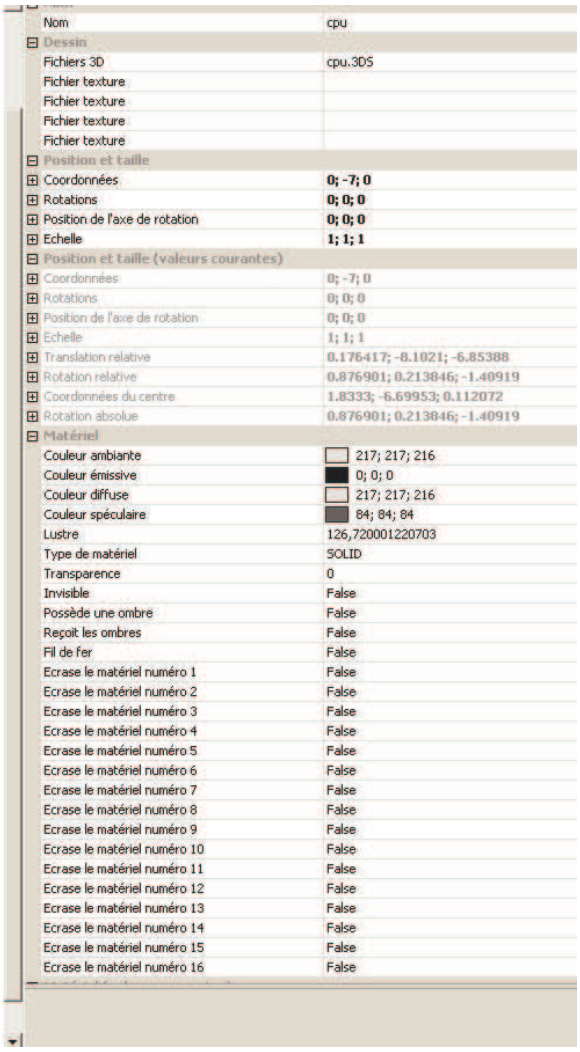
Ce projet se trouve dans le sous-répertoire « samples\nxt » du répertoire d'installation de Virtual Universe. Il est accompagné d'un fichier .AGN.



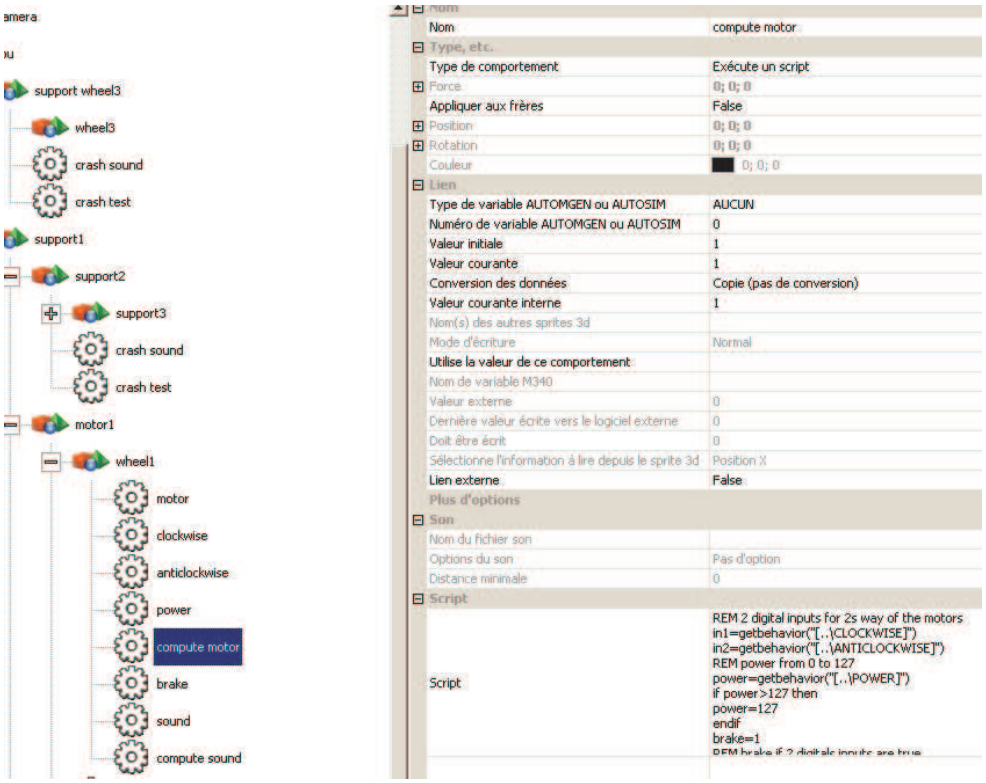
Ce projet illustre notamment les fonctionnalités suivantes :

- Simulation d'un robot mobile,

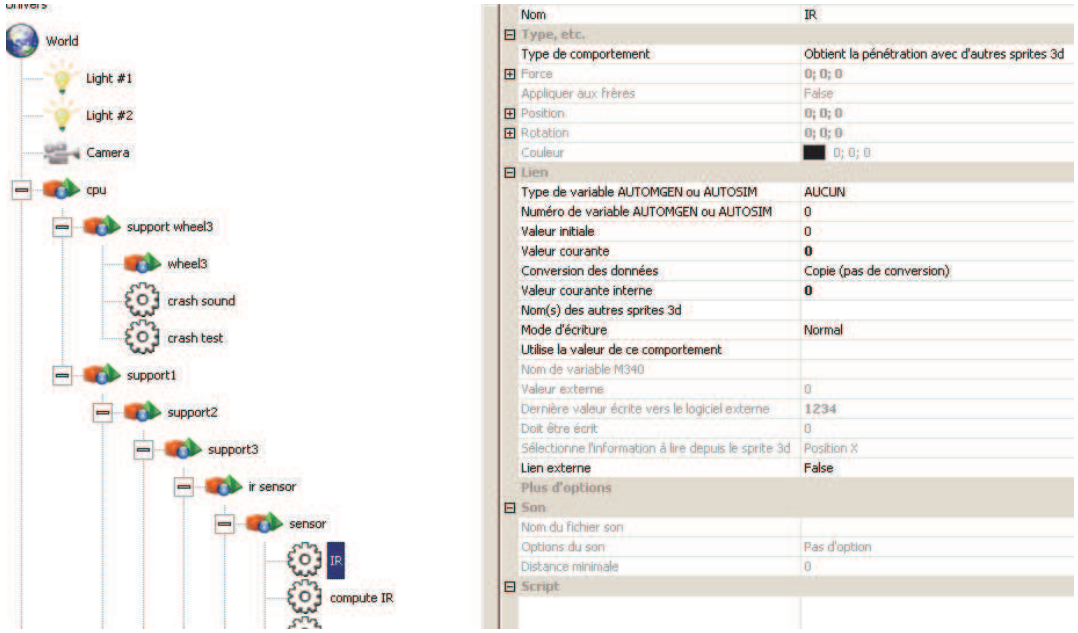


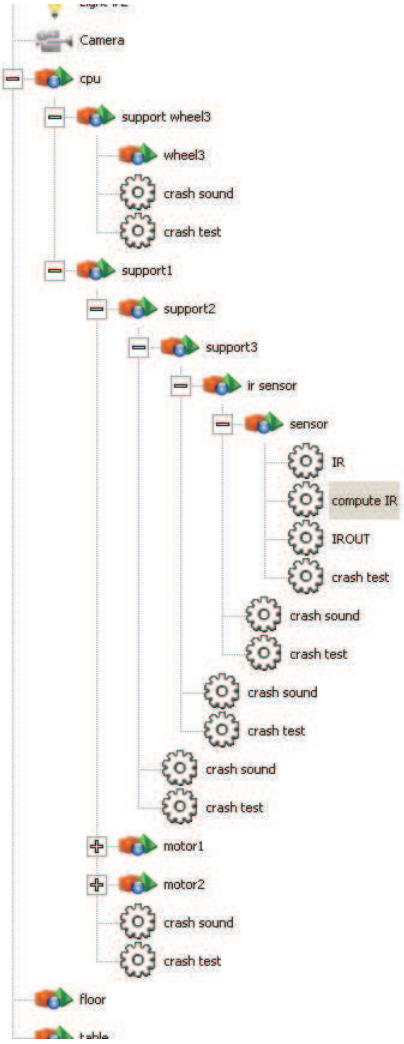


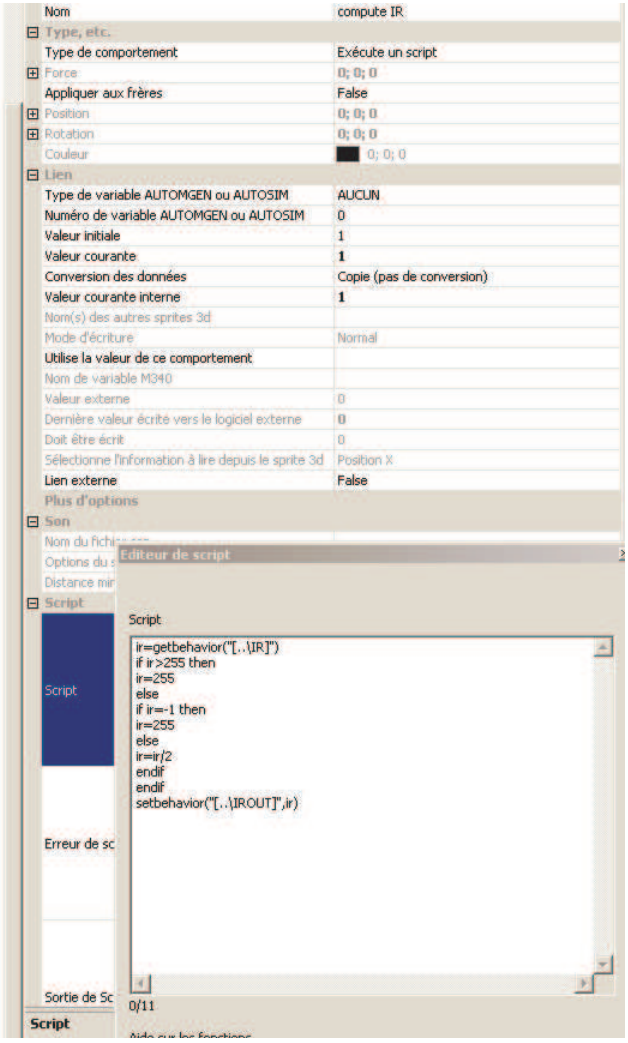
- Simulation de 2 moteurs à vitesse variable modulation de vitesse et frein piloté par 2 variables booléennes et une variable numérique.



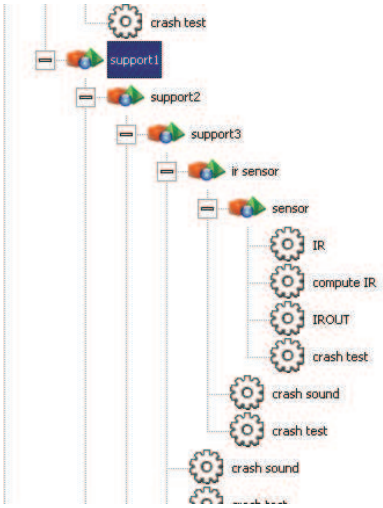
- Capteur de proximité analogique simulé par un test de pénétration.

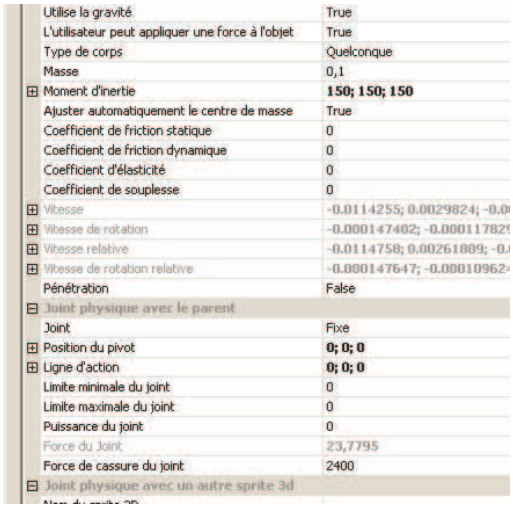






- Destruction de joints.





Fonctionnement

Les deux roues sont pilotées par des moteurs dont la puissance est pilotée par des variables numériques. Deux variables numériques permettent de piloter le moteur dans chaque sens.

Liste des références de variables AUTOMGEN / AUTOSIM

Symbole	Variable	Commentaire
motor#1 power	%mw200	Puissance moteur 1
motor#2 power	%mw201	Puissance moteur 2
sensor	%mw203	Capteur de proximité
motor#1 direction 1	%q0	Direction 1 moteur 1
motor#1 direction 2	%q1	Direction 2 moteur 1
motor#2 direction 1	%q2	Direction 1 moteur 2
motor#2 direction 2	%q3	Direction 2 moteur 2

Entrée

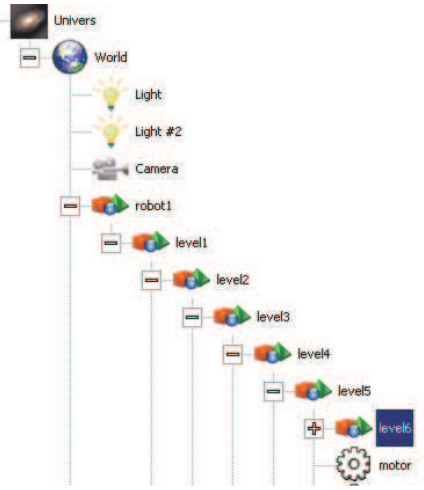
Sortie

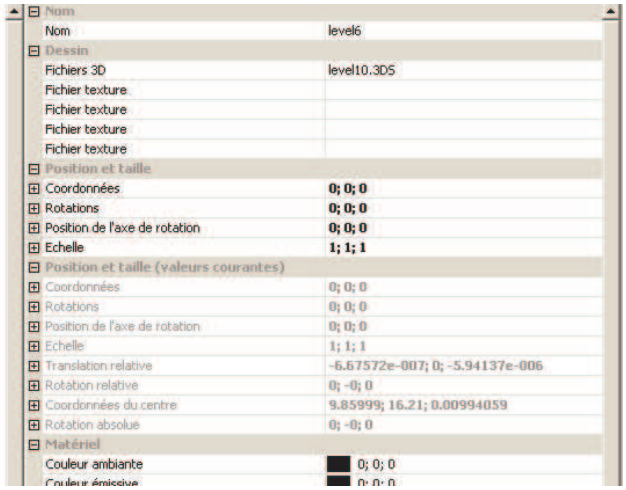
Robot ABB 6 axes

Ce projet se trouve dans le sous-répertoire « samples\abb robot » du répertoire d'installation de Virtual Universe. Il est accompagné d'un fichier .AGN.

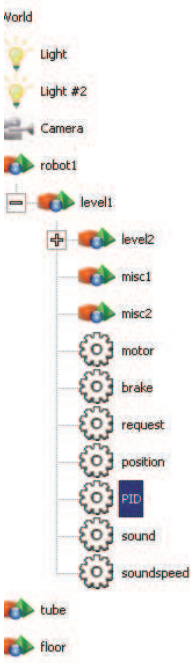
Ce projet illustre notamment les fonctionnalités suivantes :

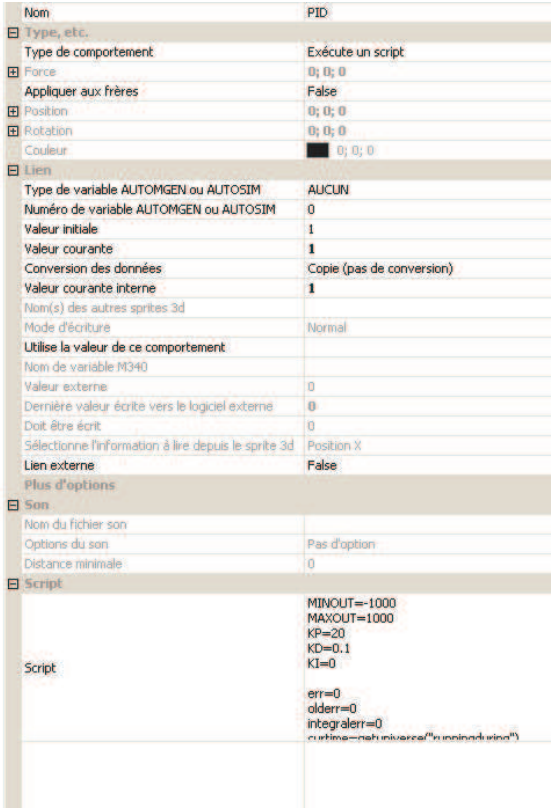
- Simulation d'un robot 6 axes.





- Simulation d’un moteur par PID.

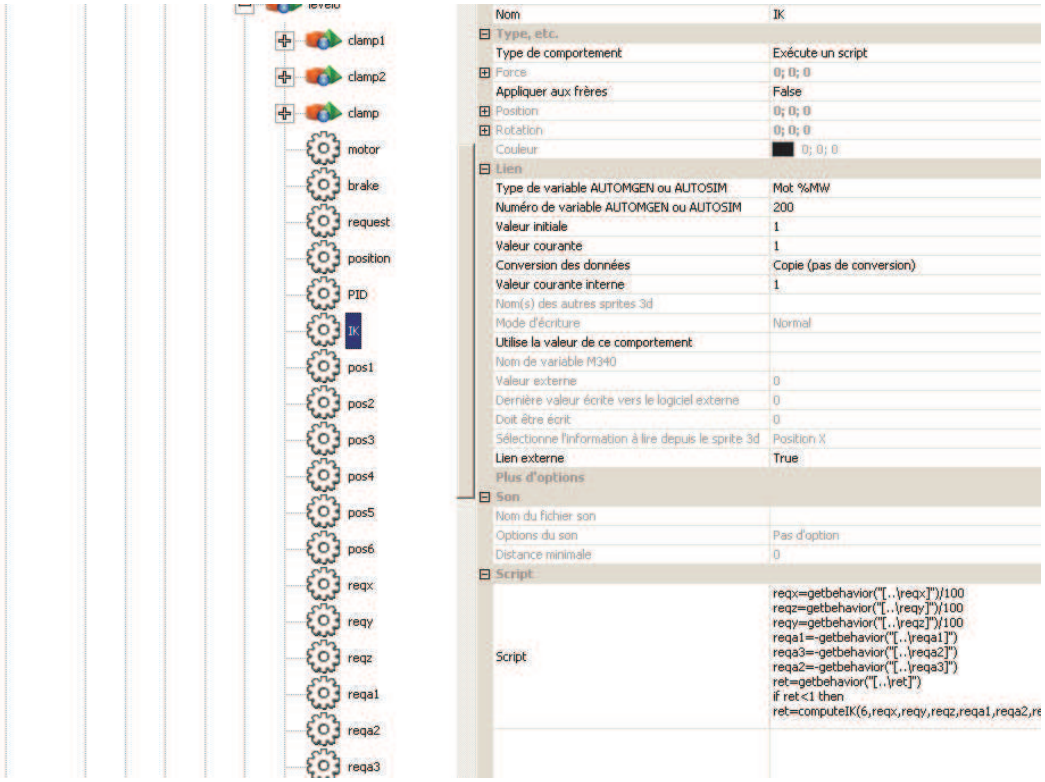




Script

```
MINOUT=-1000
MAXOUT=1000
KP=20
KD=0.1
KI=0

err=0
olderr=0
integralerr=0
curtime=getuniverse("runningduring")

myloop:
dt=getuniverse("runningduring")-curtime
if dr=0 then dt=1
curtime=curtime+dt
request=getbehavior("[..\request]")
rem SetValSprite3d("jointmin",request)
rem SetValSprite3d("jointmin",request+0.0001)

current=getbehavior("[..\position].internalvalue")
olderr=err

err=request-current
P=KP*err
integralerr=integralerr+err*dt
I=KI*integralerr
D=KD*(err-olederr)/dt
out=P+I+D
if out>MAXOUT then out=MAXOUT
if out<MINOUT then out=MINOUT
setbehavior("[..\motor].internalvalue",out)

rem print "P=";P;" I=";I;" D=";D;" request=";request;"position=";current;" dt=";dt;" curtime=";curtime

goto myloop
```

- Calcul de la cinématique inverse du robot.



Script

```
reqx=getbehavior("[..\reqx]")/100
reqz=getbehavior("[..\reqy]")/100
reqy=getbehavior("[..\reqz]")/100
reqa1=-getbehavior("[..\reqa1]")
reqa3=-getbehavior("[..\reqa2]")
reqa2=-getbehavior("[..\reqa3]")
ret=getbehavior("[..\ret]")
if ret<1 then
ret=computeIK(6,reqx,reqy,reqz,reqa1,reqa2,reqa3,0,0,0,"..\pos1","..\pos2","..\pos3","..\pos4","..\pos5","..\pos6")
if ret=0 then
setbehavior("[..\ret]",1)
pos1=getbehavior("[..\pos1]")
pos2=getbehavior("[..\pos2]")
pos3=getbehavior("[..\pos3]")
pos4=getbehavior("[..\pos4]")
pos5=getbehavior("[..\pos5]")
pos6=getbehavior("[..\pos6]")
setbehavior("[..\..\..\..\..\..\..\request]",pos1)
setbehavior("[..\..\..\..\..\..\request]",pos2)
setbehavior("[..\..\..\..\..\request]",pos3)
setbehavior("[..\..\..\..\request]",pos4)
setbehavior("[..\..\..\request]",pos5)
setbehavior("[..\request]",pos6)
else
setbehavior("[..\ret]",ret)
endif
setbehavior("[..\IK].internalvalue",0)
endif
```

- Simulation d’un tapis roulant.

Fonctionnement

Six variables numériques permettent de choisir la destination et l’orientation de la pince du robot. La précision de la position de destination est également donnée par une variable numérique. Plus la précision demandée est importante, plus le temps mis pour la fin du mouvement sera importante, à l’inverse, une précision moindre permettra d’enchaîner plus rapidement les mouvements au détriment de la précision. Une variable numérique de commande permet de lancer le mouvement, une variable numérique de compte rendu permet de savoir si le mouvement a été effectué. Le système de coordonnées est celui réellement utilisé par le robot ABB réel.

Liste des références de variables AUTOMGEN / AUTOSIM

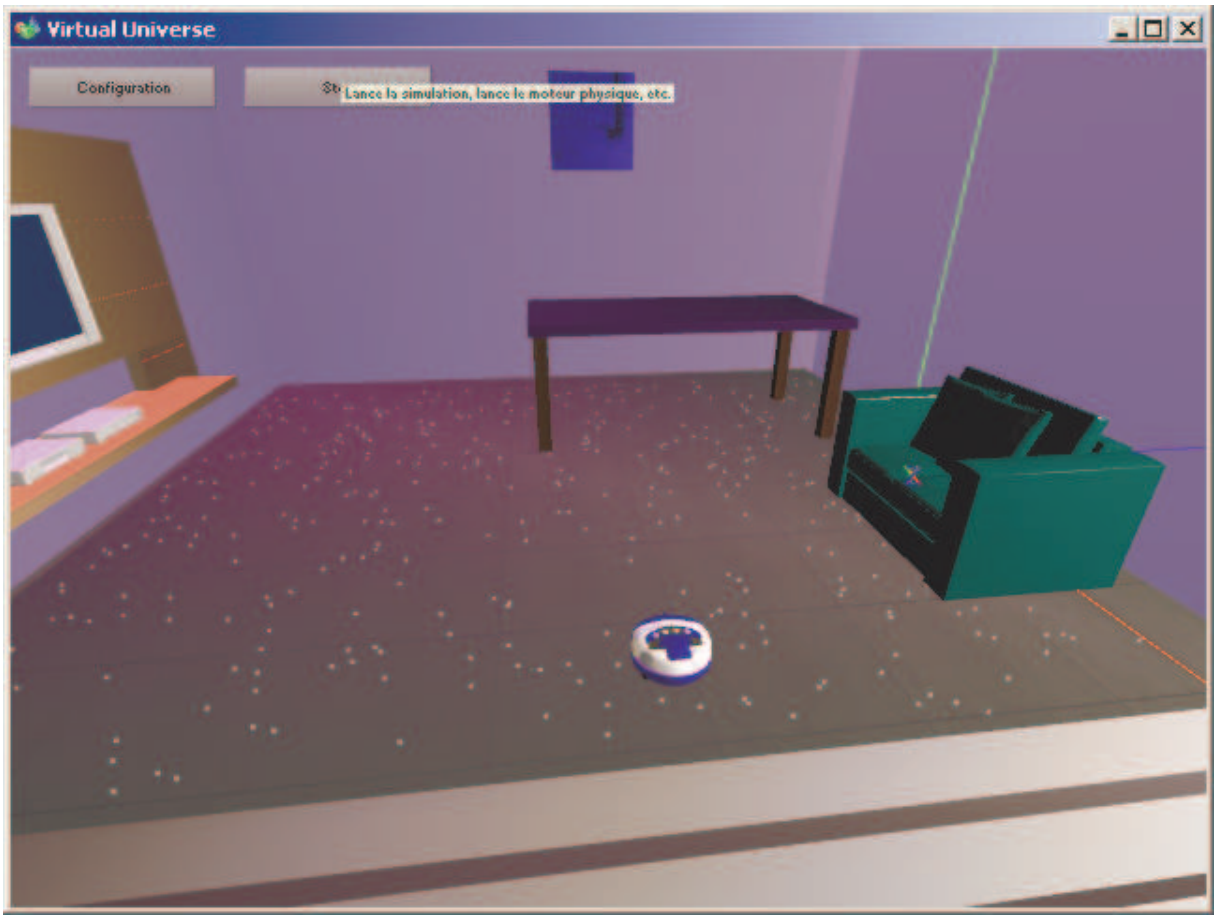
Symbole	Variable	Commentaire
xrobot1	%mf200	Coordonnées X de destination
yrobot1	%mf201	Coordonnées Y de destination
zrobot1	%mf202	Coordonnées Z de destination
arobot1	%mf203	Angle A de destination (en degrés)
brobot1	%mf204	Angle B de destination (en degrés)
crobot1	%mf205	Angle C de destination (en degrés)
deltarobot1	%mf206	Précision demandée
cmdrobot1	%mw200	Commande : le passage de 0 à 1 lance le mouvement
statrobot1	%mw201	Résultat : 1=mouvement effectué, <0=erreur (-7=position impossible à atteindre)
close clamp	%q0	Fermeture de la pince

Entrée

Sortie

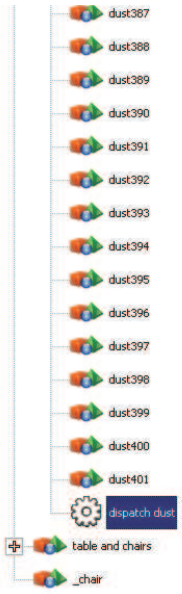
Robot aspirateur

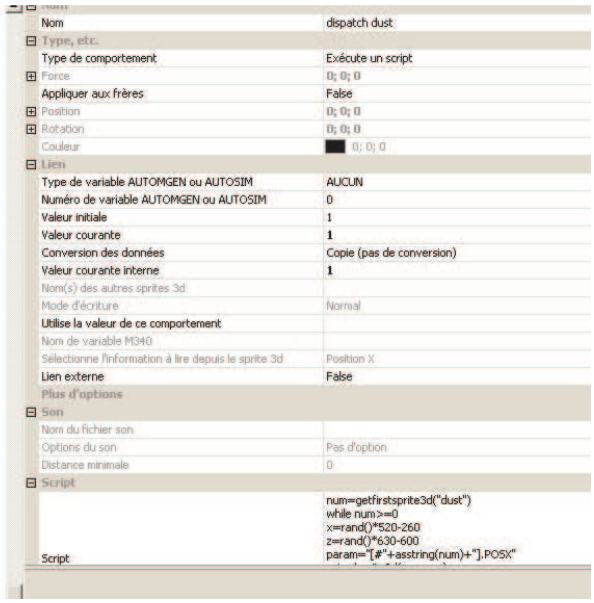
Ce projet se trouve dans le sous-répertoire « samples\vacuum robot » du répertoire d'installation de Virtual Universe. Il est accompagné d'un fichier .AGN.

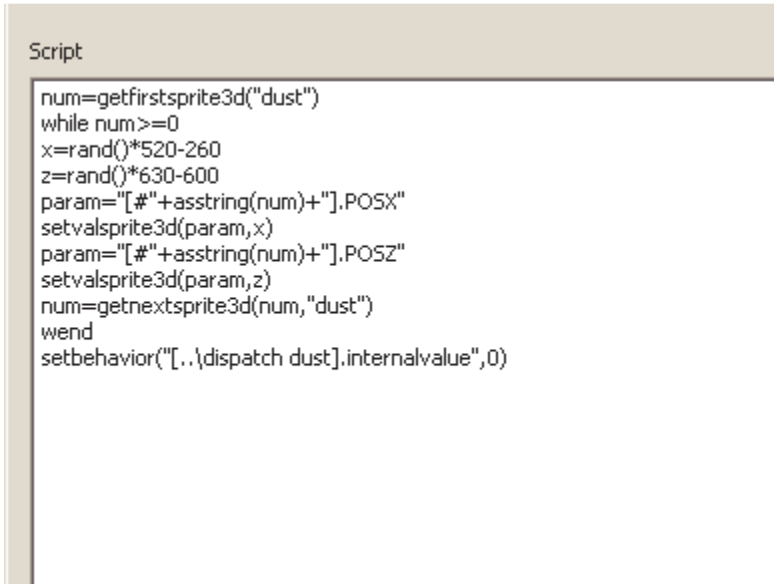


Ce projet illustre notamment les fonctionnalités suivantes :

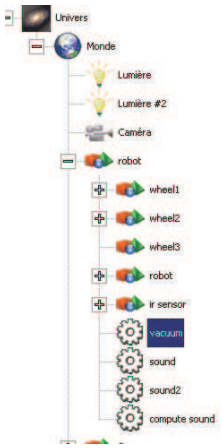
- Positionnement aléatoire d’objets,

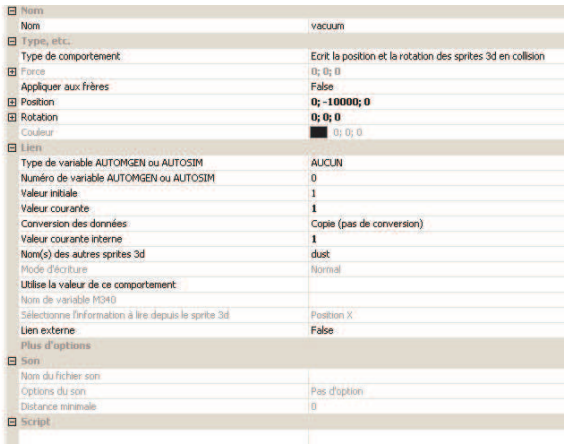






- Simulation d'une aspiration.





Fonctionnement

Le robot est piloté par deux moteurs eux-mêmes pilotés en tout ou rien par 2 sorties chacun (une sortie pour chaque sens de marche). Un capteur de contact détecte les collisions avec les objets, un capteur de proximité permet d'obtenir l'information de l'absence de sens devant le robot.

Liste des références de variables AUTOMGEN / AUTOSIM

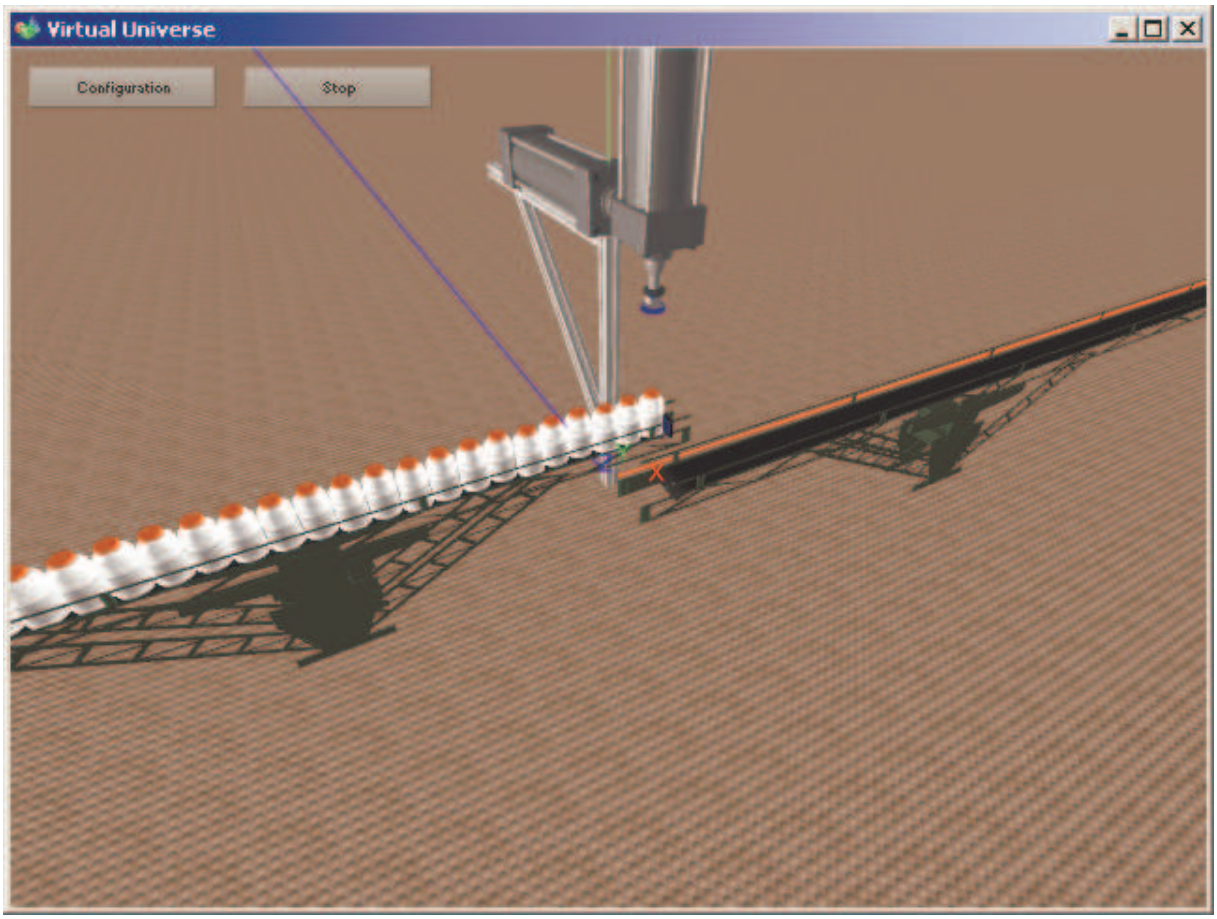
Symbole	Variable	Commentaire
forward motor 1	%q0	Moteur 1 en avant
backward motor 1	%q1	Moteur 1 en arrière
forward motor 2	%q2	Moteur 2 en avant
backward motor 2	%q3	Moteur 2 en arrière
collision	%i0	Capteur collision
ir sensor	%i1	Capteur absence de sol devant le robot

Entrée

Sortie

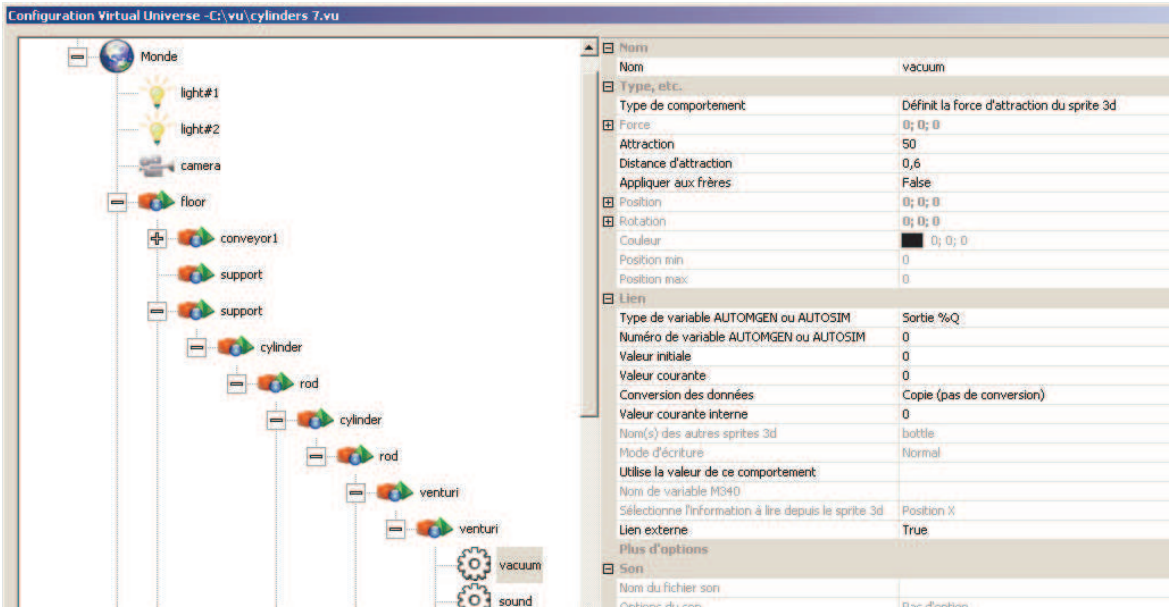
Manipulateur avec vérins et ventouse

Ce projet se trouve dans le sous-répertoire « samples\cylinders » du répertoire d'installation de Virtual Universe. Il est accompagné d'un fichier .AGN.

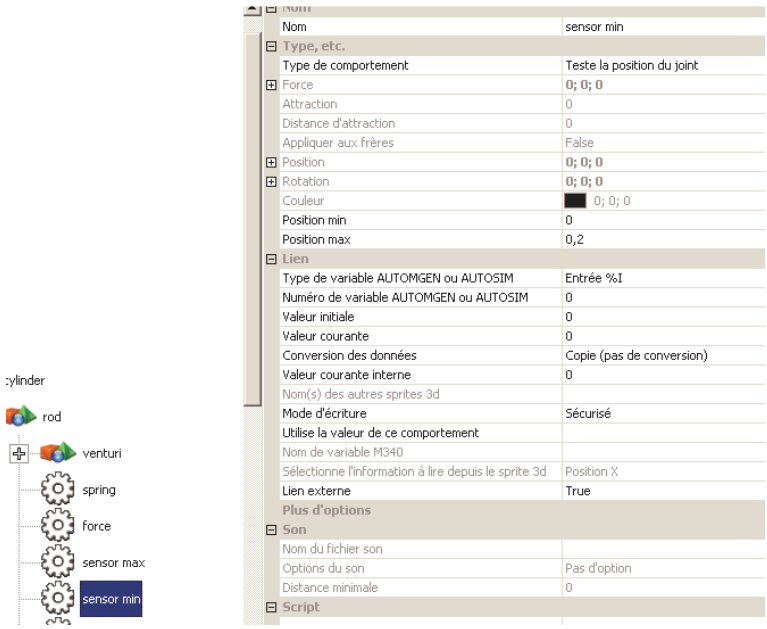


Ce projet illustre notamment les fonctionnalités suivantes :

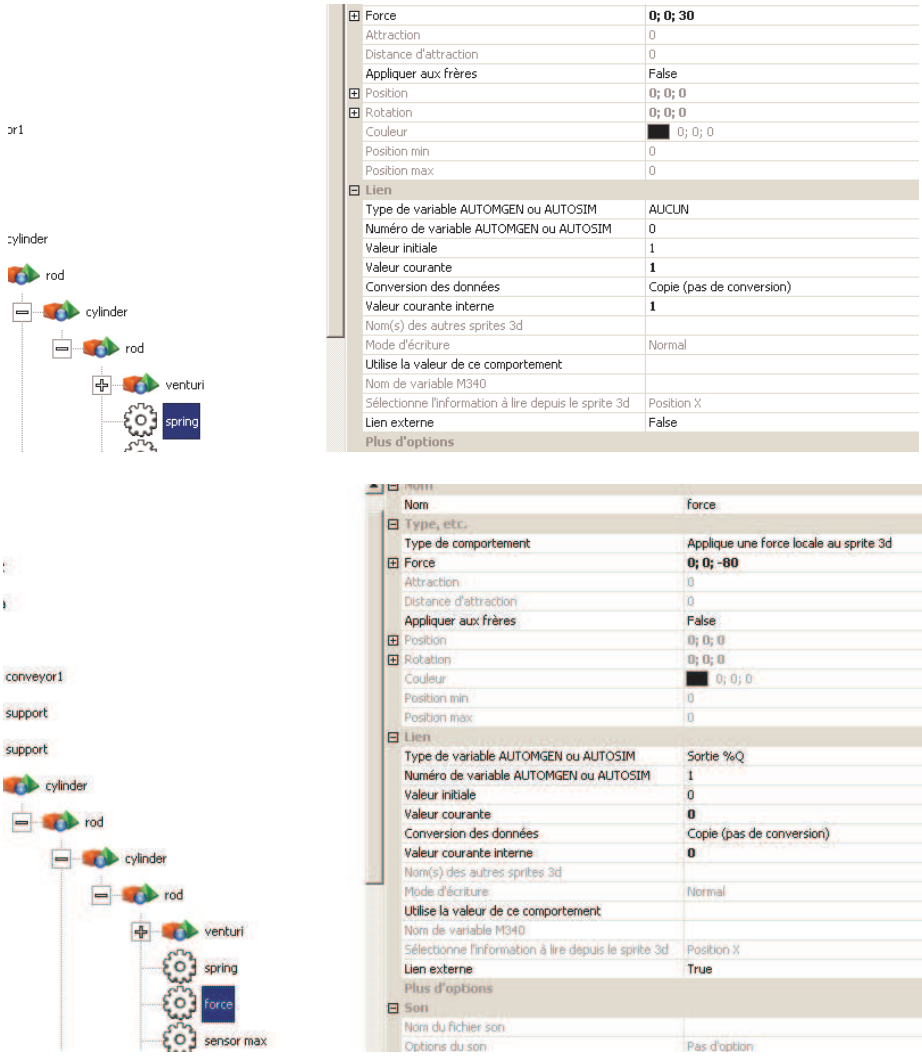
- Simulation de la ventouse,



- Capteurs de positions pour les vérins,



- Simulation de vérin avec retour de la tige par ressort,



Fonctionnement

Le bras est composé de deux vérins simple effet. Une ventouse permet la préhension des bouteilles. Deux convoyeurs gèrent l'arrivée et le départ des bouteilles.

Liste des références de variables AUTOMGEN / AUTOSIM

Symbole	Variable	Commentaire
vaccum	%q0	Active l'aspiration
go down	%q1	Fait sortir le vérin vertical
go out	%q2	Fait sortir le vérin horizontal
top	%i0	Vérin vertical rentré
bottom	%i1	Vérin vertical sorti
in	%i2	Vérin horizontal rentré
out	%i3	Vérin horizontal sorti

Entrée

Sortie