



V1.002

User manual

©opyright 2010 IRAI



Contents

Introduction	9
Necessary system	9
Organization	9
Installation.....	10
License.....	10
Registering the license	10
Installation in a network	11
Environment.....	12
Browsing and interactions	15
The different types of objects	16
Universe	16
World.....	16
Camera	16
Light	16
3D Sprites	16
Behaviors	16
Basic concepts.....	17
3D rendering and sounds	17
Physics engine.....	18
Dialogue	19
Script.....	19
RUN/STOP mode	19
Media manager	20
Properties	20
Universe	20
Connection	20
Driver	20
Server name or IP address.....	20

Port	20
Driver M340 mode	20
Options	21
Automatic RUN	21
Variable and state display	21
Wireframe	21
Debug mode for the physics engine	21
World	21
Name	21
Display	21
Window size	21
Editable size	21
Background color	21
Environment light	22
Show the shading	22
Number of images displayed per second (read only)	22
Use the shader	22
Maximum number of images per second	22
Camera	22
Name	22
Position	22
Current position	22
Light	23
Name	23
Position	23
Color, type, etc.	23
3d Sprite	23
Name	23
Drawing	23
Position and size	23

Position and size (current values)	23
Material	23
Material (current values)	23
Browsing	24
Not selectable	24
Physics	24
Use the physics engine	24
Use gravity	24
The user can apply a force to the object	24
Type of body	24
Mass	26
Inertia force	26
Automatically adjust the center of mass	26
Coefficients... ..	26
Speed	26
Penetration	26
Physical joint with the parent	26
Joint	26
Pivot position	27
Action line	27
Limits	27
Joint power	27
Joint strength	27
Joint breaking strength	27
Physical joint with another 3D Sprite	28
2D Sprite	28
Behaviors	29
Name	29
Type, etc.	29
Behavior type	29

Strength	32
Apply to brothers.....	33
Position / Rotation / Color	33
Links.....	33
Initial value	33
Current value, internal current value, conversion of data, write mode	33
Names of other 3D Sprites	33
Use the value of this Behavior.....	33
External link	33
Sounds	34
Minimum distance.....	34
Script	34
Script.....	35
Write a script	36
Specific functions	37
3D Sprite name syntax	37
Access functions to values associated to a 3D Sprite	38
Behavior name syntax	39
Access functions to values associated to Behavior	39
Access functions to values associated to the Universe	40
Other functions	41
Object library.....	43
External links	43
Current value and internal current value	43
Reading a Virtual Universe value from the external software	44
Writing an external software value to Virtual Universe.....	45
Access to the external links of an object group.....	46
Examples	47
Conveyor.....	47

Operation	53
List of AUTOMGEN / AUTOSIM variable references	54
List of UNITY PRO variable references	55
Robot and bottles.....	56
Operation	60
List of AUTOMGEN / AUTOSIM variable references	60
NXT robot.....	61
Operation	66
The two wheels are controlled by motors whose power is controlled by numeric variables. Two numeric values are used to control the motor in each direction.	66
List of AUTOMGEN / AUTOSIM variable references	66
6 axis ABB Robot.....	67
Operation	71
List of AUTOMGEN / AUTOSIM variable references	72
Vacuum robot.....	73
Operation	75
List of AUTOMGEN / AUTOSIM variable references	75
Manipulator with cylinders and suction cup.....	76
Operation	81
List of AUTOMGEN / AUTOSIM variable references	81

Introduction

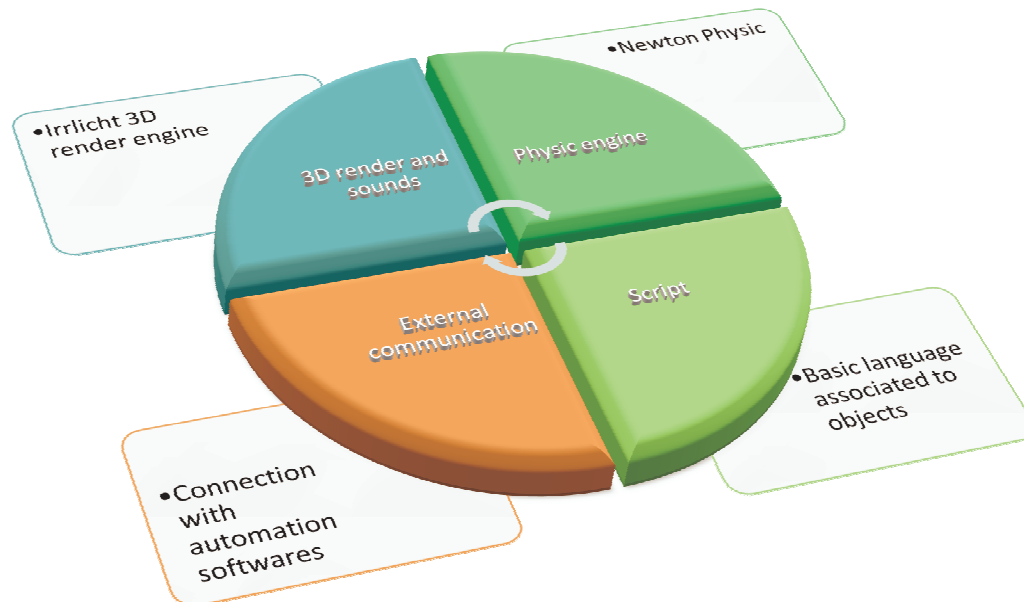
Virtual Universe is a 3D world simulator dedicated to automaton and robotics. By integrating the latest technologies in 3D rendering, 3D sound, physical simulation and script, Virtual Universe can be used to create ultra realistic simulations. Virtual Universe can communicate with automaton software workshops (AUTOMGEN, UNITY, etc.) so that the virtual systems can be controlled like real systems.

Necessary system

Virtual Universe operates with the following operating systems: Windows XP, Windows Vista and Windows 7.

Virtual Universe is compatible with AUTOMGEN 8.015 or later versions.

Organization



Installation

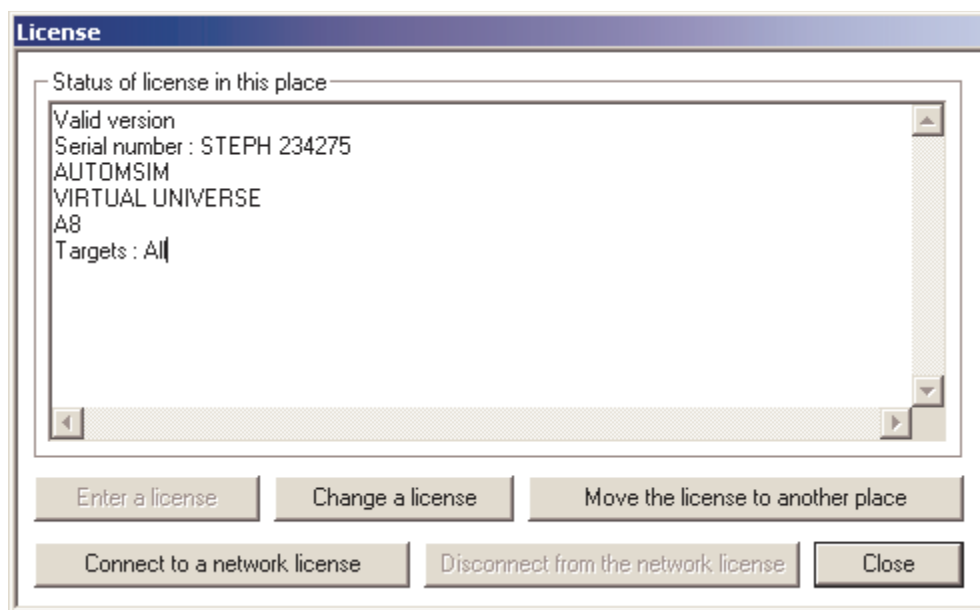
To install Virtual Universe, simply run the execution of the installation package which has been delivered to you on a CD-ROM or by downloading. Visit our website (www.irai.com) to download the latest updates for Virtual Universe.

License

Registering the license

Virtual Universe operates as a demo version (for a 40 day trial) as long as you haven't registered the license.

To register the license, click on the "License" button in the Virtual Universe configuration window.



Click on the "Enter a license" button.

Enter or change a protection

You are about to save or change your user license (after requesting authorization to use the information if necessary).
Your user code must be provided to IRAI which will then send you a validation code.

You can send your user code by Telephone : (33) 4 66 54 91 30,
- by Fax : (33) 4 66 54 91 33
- or by e-mail : francoise.saut.irai@orange.fr

The following information must be provided: your complete address and telephone number and order reference or delivery note if required.

User code, careful : '0' is ZERO and 'O' is the letter

12GE5	29BNU	5189G	2I05Q	ULNTB	7GIGG	G2IG5	6UF9F	ULK8L	HPCH9	C0
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	----

Save the user code in a file

Read a validation code from a file

Copy the user code to the clipboard

Paste a validation code from the clipboard

Obtain a new user code

Validation code

--	--	--	--	--	--	--	--	--	--	--

Cancel

Validate

Send the user code which is then generated by e-mail to the address francoise.saut.irai@orange.fr

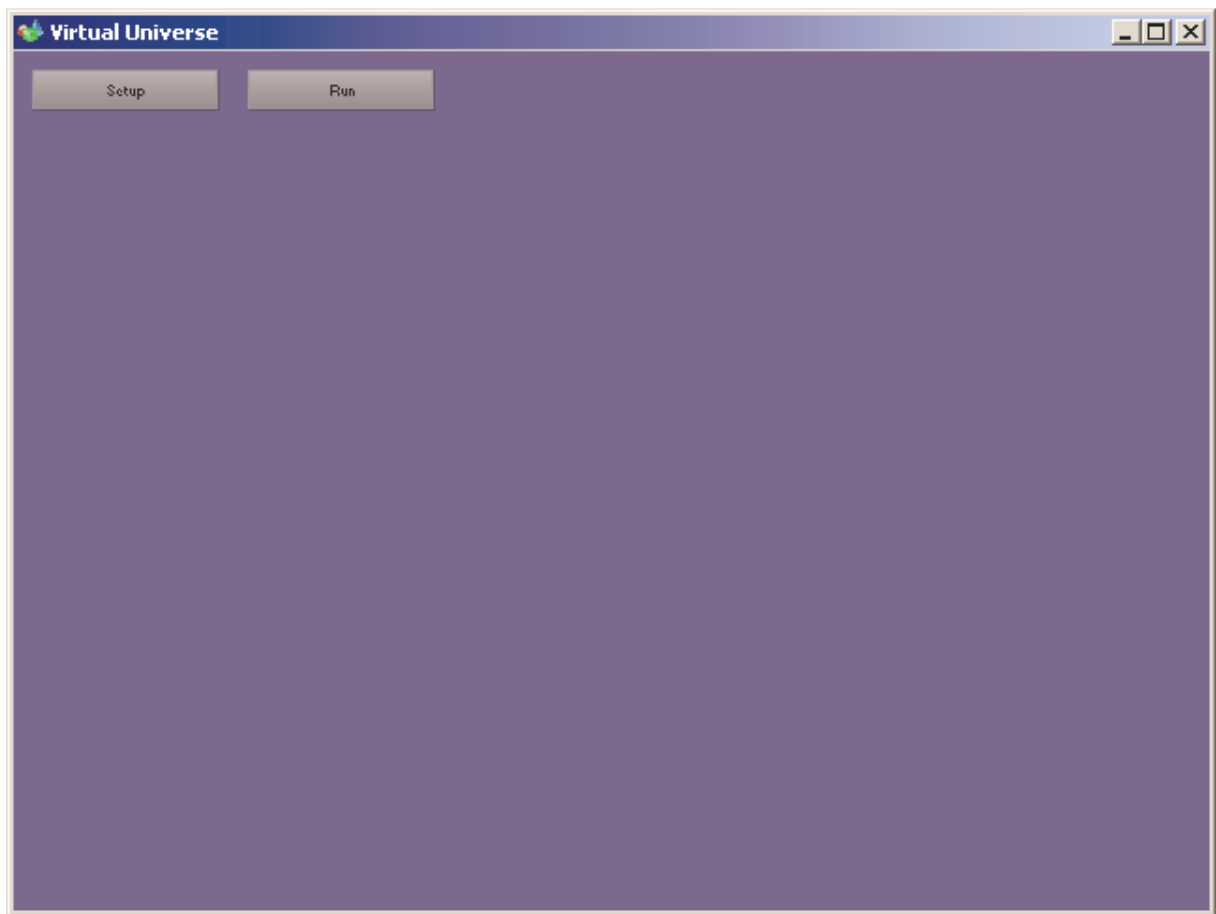
You will receive a validation code by e-mail which you then enter in the “validation code” areas, then click on “Validate” to validate the license. You have 20 days after the user code is generated to enter the validation code.

Installation in a network

The Virtual Universe files can be installed on a file server. The licenses can also be managed by a network license manager (see the specific network license manager).

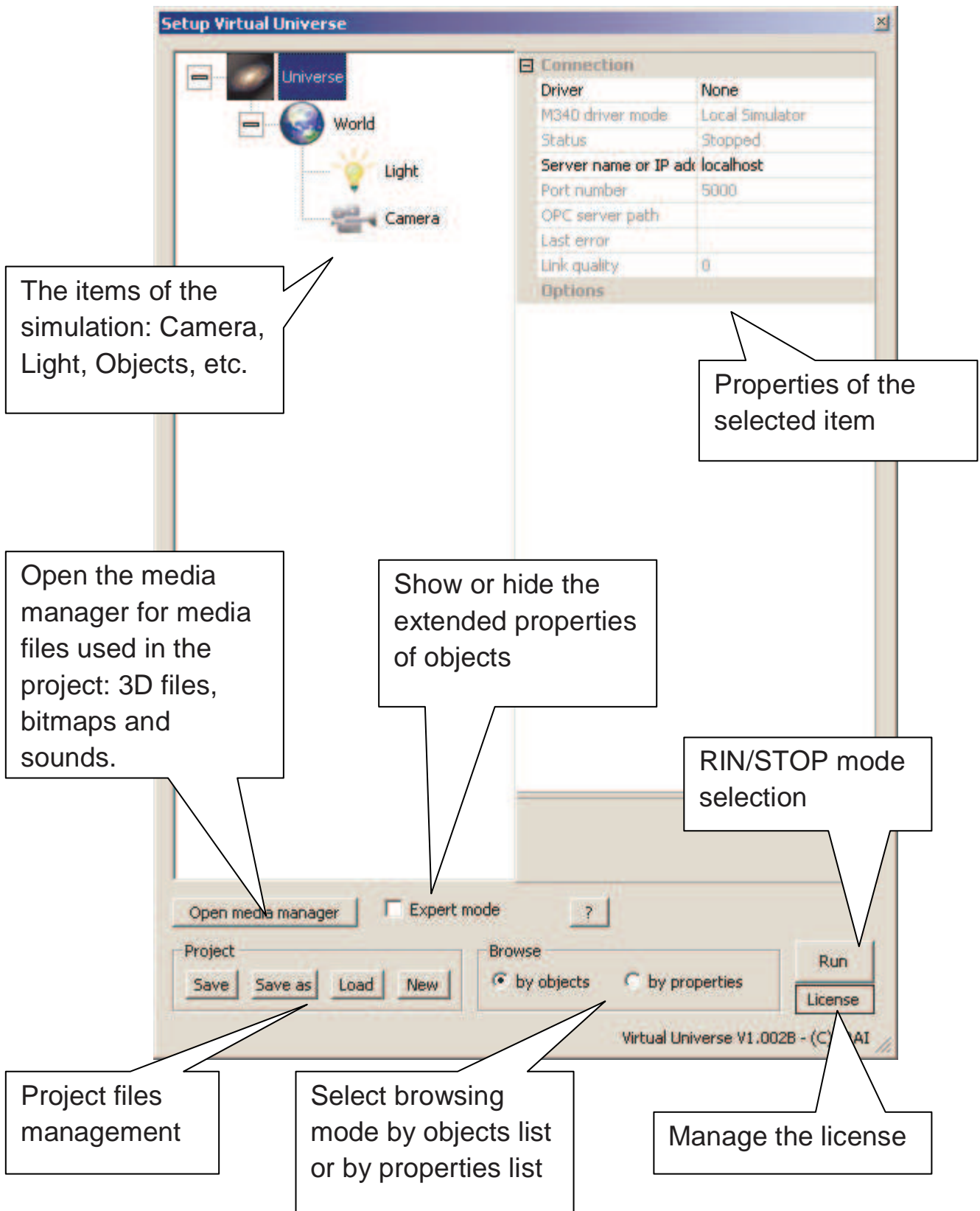
Environment

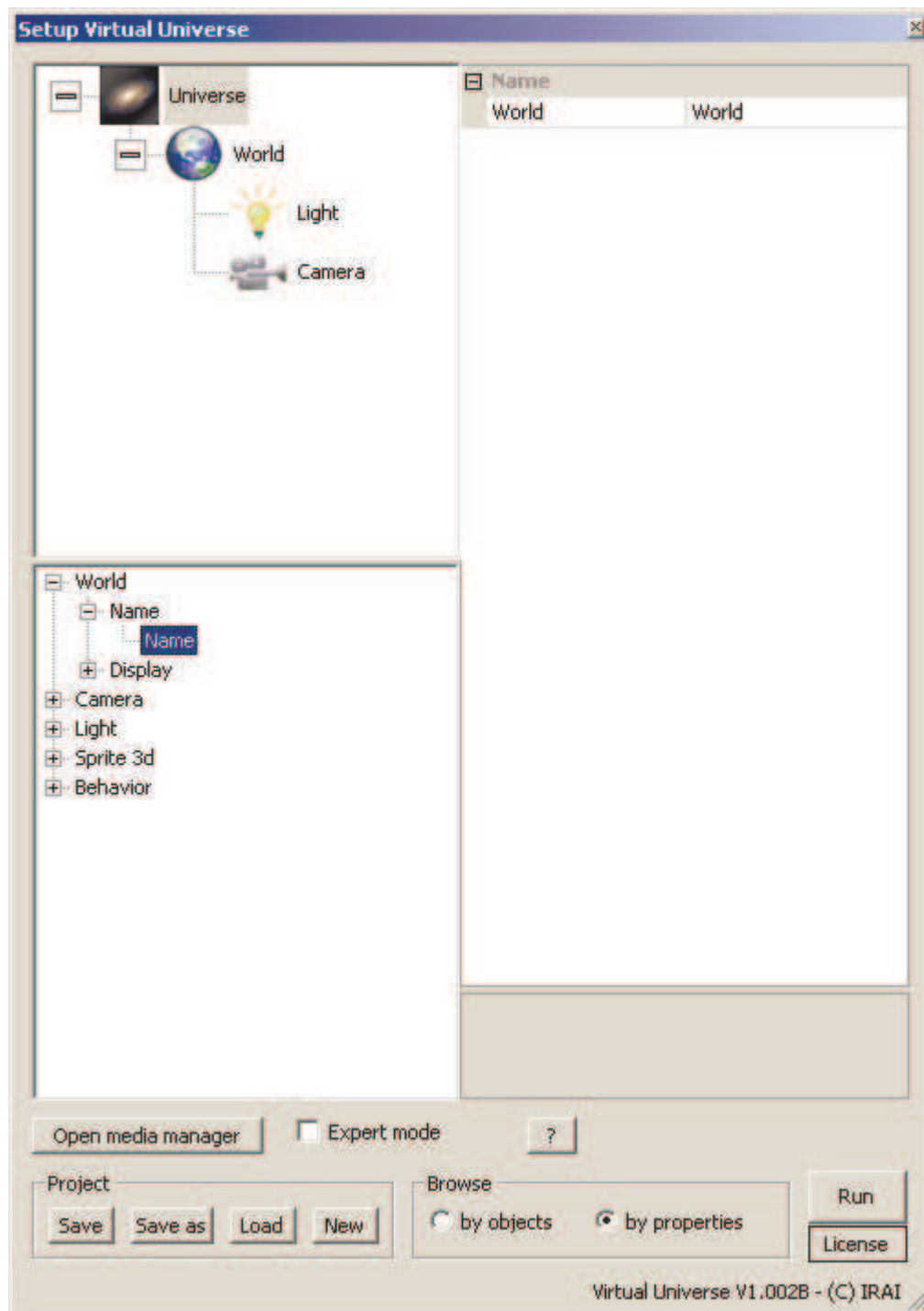
When Virtual Universe is started a 3D world rendering window appears:



The RUN/STOP button is used to run or stop the simulation.

The SETUP button opens or closes the configuration window:





The same window in “by properties” mode is used to obtain the list of values of the same property for an object group. In this mode, the parent of the objects needs to be selected on the upper left and the property on the lower left.

Browsing and interactions

The following commands are used to browse and in 3D World or to interact with these:

- Mouse wheel or keyboard Up and Down keys: Zoom
- Right mouse button pressed and movement of mouse: orbit around the selected object.
- Movement of the mouse inside the rendering window: automatic selection of the browsed object to orbit around.
- Left click of the mouse on an object in STOP mode: selection of the object.
- Left mouse button pressed on an object and movement of mouse in RUN mode: interaction with the selected object: push, pull, move.

The different types of objects

The objects are organized hierarchically in child/parent.

Universe

This is the parent object of the entire Virtual Universe project, it contains one or more Worlds, its properties set which automaton Virtual Universe will dialogue with. The Universe object is always the parent of the hierarchy.

World

This is a subset of the Universe. Its properties define the rendering window aspect among other things. The World objects are always the children of the Universe.

Camera

The Cameras represent a user's viewpoint in a 3D world. The camera objects are children of the World objects or 3D Sprites.

Light

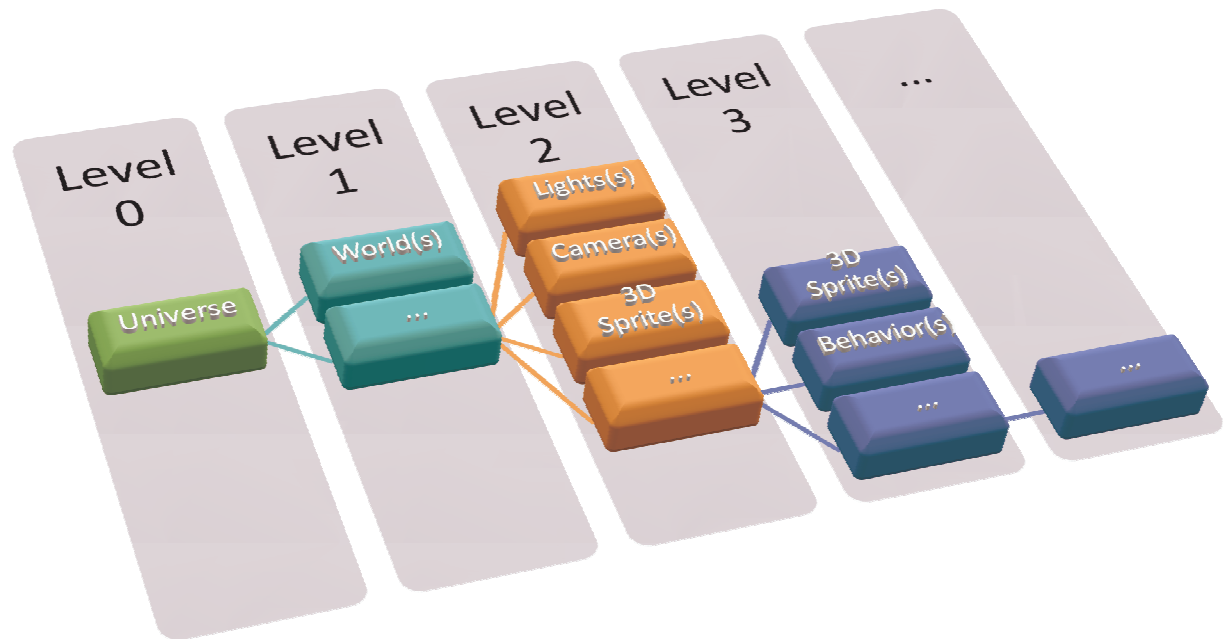
The Lights are necessary, just like in the real World for being able to observe the objects. The Light objects are children of the World objects or 3D Sprites.

3D Sprites

These are the objects and their multiple physical and visual characteristics. 3D Sprite objects are children of the World objects or 3D Sprites.

Behaviors

Associated to a 3D Sprite or a Light, they will dynamically change their properties: for example to change their positions or their colors or even execute a script which can act on these objects. A Behavior can act as an engine to transmit a force to a 3D Sprite. The Behavior objects are children of the Light objects or 3D Sprites.

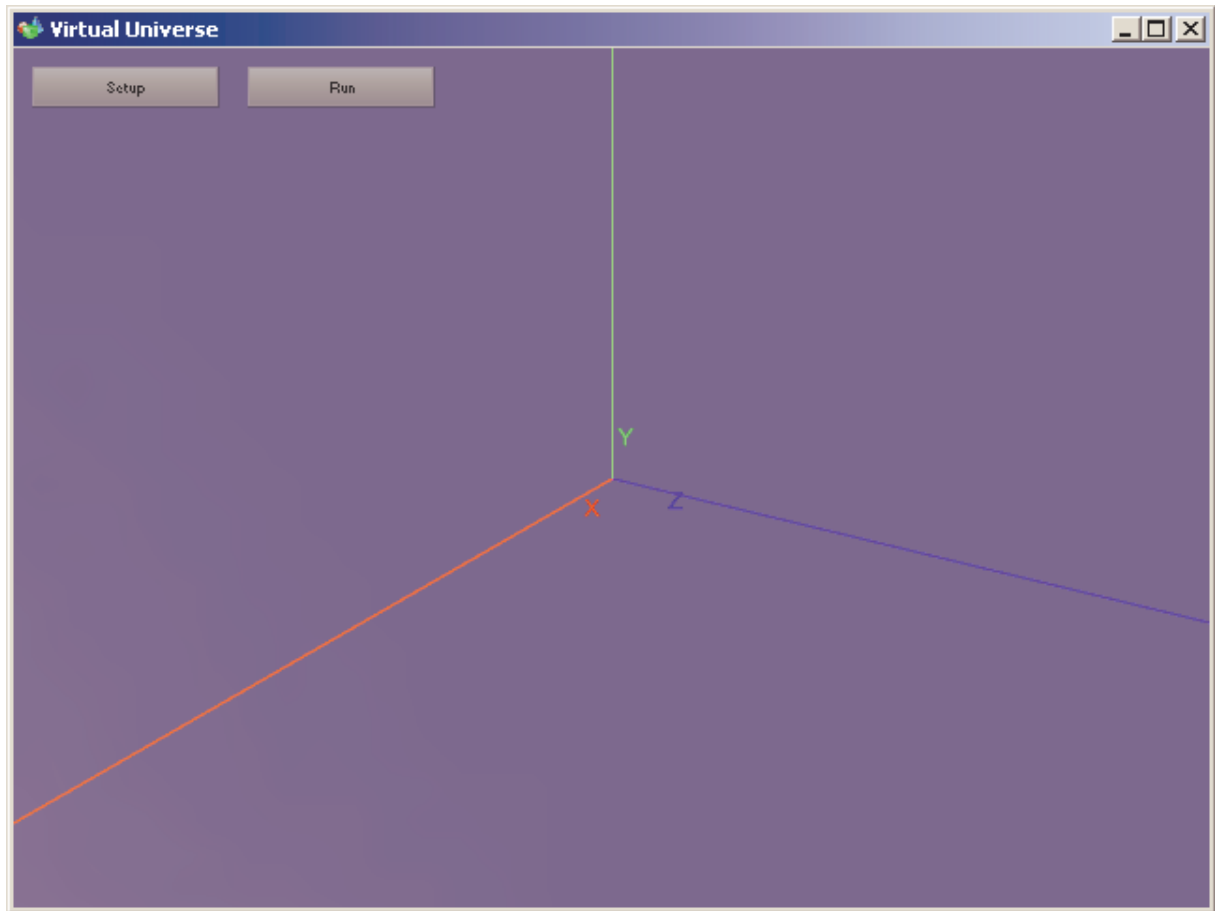


Basic concepts

3D rendering and sounds

The rendering engine used by Virtual Universe is Irrlicht which supports via DIRECTX 8 or 9 or OPENGGL (based on what is available on the PC). The role of the 3D rendering engine is to display the 3D world objects lit by the Lights based on the viewpoint set by a camera. The Cartesian coordinates $X/Y/Z$ govern the 3D world.

An axis identifier is displayed in the rendering window when the configuration window is open.



The 3D sounds increase the realism of the simulations. The sounds are emitted in the virtual world at the position of the objects and are thus perceived based on the camera position.

Physics engine

Newton Physic Engine is the physics engine used by Virtual Universe for physical object management: for example gravity, but also much more than that.

To get the most out of the physics engine, it is important to be familiar with the basic concepts of physics, such as forces, velocities, frictions, mass, etc.

The physics engine parameters are associated to each 3d Sprite. A 3d Sprite can be managed or not by the physics engine. For example, an object only used visually may not be managed by the physics engine.

Dialogue

The dialogue with an external software is one of the essential elements used to control the simulations. The external software type and the connection parameter settings are found in the Universe properties. The links are then set in Behaviors. The Behavior type will determine the dialogue direction (reading from or writing to the external software).

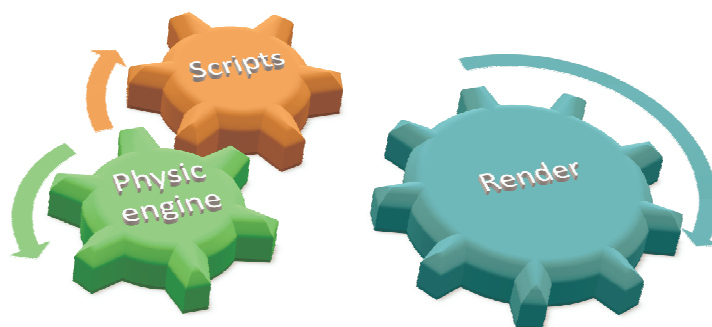
Script

Scripts written in basic language can be associated to any object by a Behavior.

RUN/STOP mode

Virtual Universe can be in STOP mode (simulation stopped and initialized) or RUN (simulation in progress) mode. In RUN mode, the physics engine and dialogue with the external software are enabled. The Behaviors and scripts are enabled.

In RUN mode, rendering is performed as quickly as possible based on the PC performance, the physics engine and the scripts are called every 10 ms.



The objects possess a double entry for certain parameters (for example their positions). The first parameter set corresponds to their initial values, the second set to their current values. In STOP mode, the initial values are recopied in the current values.

Media manager

This is used to store the media files (3D files, bitmap files and sound files) used in a project. The objects can use files found in the media manager or outside it. The files located in the media manager will be saved in the project file. This latter method is recommended if the project needs to be shared or executed on another PC.

Properties

Universe

Connection

Driver

Determines the connection with an external software¹

This is the case of connection with AUTOMGEN or AUTOSIM

Server name or IP address

Use “localhost” if the external software is run on the same PC. If the software is run on another network PC, enter its IP address or its name as seen on the network.

Port

Must be the same as the one selected in the AUTOMGEN / AUTOSIM properties in the TCP/IP Connection Execution tab, server, port.

This is the case of connection to UNITY (simulator PLC or API M340)

Driver M340 mode

Local simulator: Simulator PLC run on the same PC, API connected by USB: an API M340 connected to a USB port, API or simulator on IP : an

¹ External software is the generic term used to define the software with which Virtual Universe dialogues

API connected by Ethernet or a simulator run on another PC connected to the network. In this case, document the IP address of the API or PC.

Options

Automatic RUN

Causes it to go to RUN when the project is opened.

Variable and state display

Displays the variable names and the states for the Behaviors in the rendering window referred to a variable of the external software.

Wireframe

The group of project 3D Sprites will be displayed in “wireframe” mode if true.

Debug mode for the physics engine

If true, the volumes used by the physics engine are displayed in the rendering window (yellow lines). This is very useful in the development phase of a project using the physics engine to display the volumes handled by the physics engine.

World

Name

Used to indicate a World by its name.

Display

Window size

Determines the rendering window size in pixels.

Editable size

If true, the window size can be edited by the user.

Background color

Determines the color displayed for the background on the rendering window.

Environment light

Determines the color and intensity of the environment light (light lighting the group of objects no matter what their positions and their orientations).

Show the shading

If true, this manages display of shading, it requires that the properties of objects relative to shading also be positioned. The display of shading may significantly slow down the rendering.

Number of images displayed per second (read only)

Indicates the number of images displayed in a second in the rendering window.

Use the shader

Evolved 3D displaying technique reserved for specialists.

Maximum number of images per second

If different than 0, this limits the number of images displayed per second to the indicated value. Used to preserve the processor time.

Camera

Name

Used to indicate a Camera by its name.

Position

Determines the initial position of the Camera by the target coordinates (related to the Camera) as well as a rotation on the X and Y axes and a zoom.

Current position

The same as above but for the current position.

The current position can be recopied in the initial position by clicking on the down arrows appearing to the right of the initial position elements and selecting "Copy from current values".

Light

Name

Used to indicate a Light by its name.

Position

Used to define the coordinates and direction of the Light (direction is only used for the Spot and Directional type lights).

Color, type, etc.

Determines the Light characteristics.

3d Sprite

Name

Used to indicate a 3d Sprite by its name

Drawing

Determines the 3D file used to set the 3D Sprite geometry and any texture files.

Position and size

Sets the position, rotation (as well as the axis) and the initial scale of the 3D Sprite. Rotations are expressed in degrees (from – 180 to + 180 degrees).

Position and size (current values)

The same as above but for current values, with more: the translation and relative rotation (based on the parent 3D Sprite), as well as the position of the object center and the absolute rotation based on the World.

Material

These properties group the characteristics of the material used to display the object. These characteristics are directly linked to the Irrlicht rendering engine.

Material (current values)

The same as above for current values.

Browsing

Not selectable

If true, the 3D Sprite does not affect browsing when it is browsed by the mouse cursor.

Physics

Groups the properties of an object concerning the physics engine.

Use the physics engine

If true the 3D Sprite will be handled by the physics engine, if not, the object will be completely ignored by the physics engine, in other words, the object will be only displayed in the 3D world, but it will have no physical interaction with the other objects.

Use gravity

If true the 3D Sprite will be submitted to gravity. Its mass cannot be zero.

The user can apply a force to the object

If true, the user can, in RUN mode, act on the object by keeping the right mouse button pressed when the cursor is on the 3D Sprite and by moving the cursor.

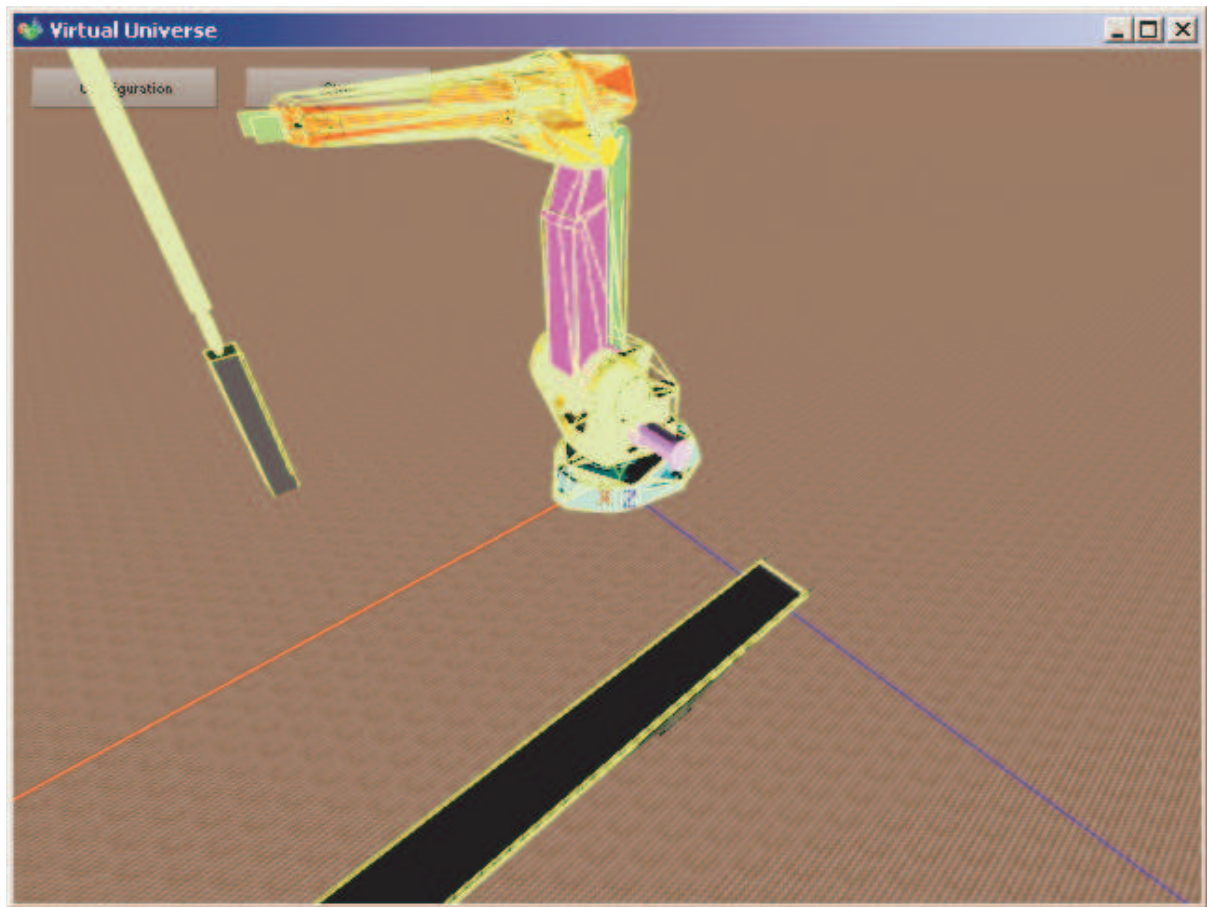
Type of body

Determines the 3D Sprite geometry type for the physics engine

- Any: a convex shape obtained from the 3D Sprite geometry;
- Box: a rectangle parallelepiped;
- Sphere: a sphere;
- Capsule: a capsule.

Attention, the “Any” type if used with a complex 3D Sprite (possessing very numerous faces), may use a lot of resources for the physical simulation. So, if possible use one of the other types.

It is possible and often useful during the set up phases to display the geometries handled by the physics engine by enabling the “Debug mode for the physics engine” option in the Universe properties. Example:



The volumes handled by the physics engine appear in yellow.

Solution in the case of a complex 3D Sprite requiring an any shape:

- set a simplified 3D Sprite shape (with fewer facets), give it the “invisible” and “managed by physics engine” attributes;
- preserve the complex 3D Sprite shape and add it as child with the “visible” and “not managed by physics engine” attributes.

This solution is used in the “Conveyor” example.

Solution in the case of a 3D Sprite requiring a concave physical shape:

set several convex shapes and link them by joints.

Mass

The mass of the object. A mass of 0 freezes the object.

Inertia force

Determines the amount of energy needed to turn the object on each of the axes.

Automatically adjust the center of mass

If true, the object's center of mass is automatically recalculated based on the 3D Sprite geometry. If not, the center of mass is the point of the 3D Sprite's coordinates 0/0/0.

Coefficients...

Determine the friction, elasticity and suppleness of objects. A value of 0 uses the physics engine default parameters. The coefficient used by the physics engine between an object A and an object B is the combination (product) of the coefficients of object A and object B.

Speed

Used to access the total and local speed values of the object. These values are only available for objects managed by the physics engine.

Penetration

If true, the collisions of the object are not managed. For objects linked by joints (see below) the collisions are automatically deactivated between two objects linked by a joint.

Physical joint with the parent

Determines the type of joint between a child 3D Sprite and its parent. The two 3D Sprites must be managed by the physics engine. They can be submitted to gravity or not.

Joint

Determines the joint type:

- Pivot;
- Sliding

- Fixed.

Pivot position

Determines the x/y/z position of the link with the parent object for pivot links.

Action line

Determines the joint action line for Sliding (translation axis) and Pivot (rotation axis) joints.

Limits...

Determines the joint minimum and maximum limits. If these two values are equal, then the joint has no limits (rotation or translation without limits).

Joint power

Determines the joint rigidity.

Joint strength

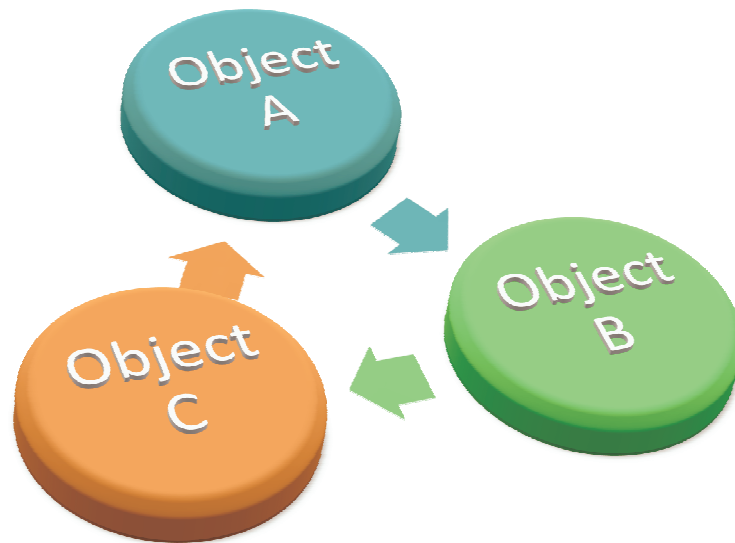
Gives the value of the strength supported by the joint.

Joint breaking strength

Strength beyond which the joint will be automatically destroyed. If 0, this function is disabled. This is used to simulate the destruction of a link between two objects (see the “NXT” example).

Physical joint with another 3D Sprite

Set of identical parameters but the link is created between the 3D Sprite and another 3D Sprite rather than the parent 3D Sprite. The other 3D Sprite is indicated by its name. This second joint is used to create circular models (see the “ABB Robot” simulation):



2D Sprite

Used to display a 2D bitmap at the 3D Sprite position. See the dust management in the vacuum robot example.

Behaviors

Behaviors are the elements which “give life” to the simulation. They also define the links between the simulation and external software.

They are closely tied to the physics engine and will be able to communicate strengths to the 3D Sprites and also manipulate the return physical data (for example the speed of an object).

For a realistic simulation, actions by application of strengths should have priority over actions directly changing the position or orientation of the 3D Sprites.

A Behavior can also be used simply for storing a value during simulation. The scripts will be able to access this value in read and write. Behaviors can be considered like “global variables” for the application.

Name

Used to indicate the Behavior.

Type, etc.

Behavior type

One of the following types for Behaviors associated to a Light:

- None, the behavior does nothing;
- Writes the intensity of the Light;
 - o The Behavior value determines the light intensity of the associated Light.

One of the following types for Behaviors associated to a 3D Sprite:

- None, the behavior does nothing;
- Applies a force or torque;
 - o All of these Behavior types are used to apply a strength or torque to the object. The Strength parameters determines the strength direction, the identification can be global or local (based on type). The strength will applied based on the current value of the Behavior. The strength applied will be the

parameterized strength multiplied by the current value of the Behavior.

- Applies a local force to the 3D Sprites in contact;
 - o Applies a strength to all of the 3D Sprites in collision with the associated 3D Sprite. The typical application is the simulation of a conveyor belt. See the “ABB Robot” example for an illustration. « Name/s of other 3D Sprites » is used to limit the action of this Behavior to a group of 3D Sprites (see below). Collision tests should only be used when strictly necessary.
- Applies a braking force or braking torque;
 - o Identical but the applied strength will act like a brake.
- Defines the attraction force of the 3D Sprite;
 - o The 3D Sprite attracts other 3D Sprites. . “Name/s of other 3D Sprites” is used to limit the action of this Behavior to a group of 3D Sprites (see below). “Attraction” is used to set the attraction force, “Attraction distance” changes the action area of this attraction (infinite if 0). The attraction strength is also proportional to the square of the distance. This is illustrated in the “Manipulator with cylinders and suctions” example.
- Writes the 3D Sprite position and rotation;
 - o Modifies the position or orientation of a 3D Sprite. For example, useful for taking an object to the initial position (see the “Conveyor” example).
- Writes the 3D Sprite position and rotation in collision;
 - o Identical but writes the position and rotations of all the 3D Sprites in collision with the parent 3D Sprite. The “Name/s of other 3D Sprites” parameter is used to limit the action of this Behavior to a group of 3D Sprites (see below). Collision tests should only be used when strictly necessary. See the “ABB Robot” example for an illustration of this Behavior.

- Write the 3D Sprite environment color;
 - Changes the parent 3D Sprite environment color if the current Behavior value is different from 0. The value to apply is one of the Behavior parameters (see below). See the “Conveyor” example for an illustration.
- Execute a script;
 - Executes a script if the current Behavior value is different from 0. Read the chapter on Scripts.
- Play a sound;
 - Used to play a sound file in a loop or just once. The 3D sound will be perceived as if coming from the parent 3D Sprite. The sound is played if the current Behavior value is different from 0. In addition, the current behavior value can modulate the volume of speed of the played sound. The examples illustrate this by modulating the played sound speed to simulate the sound of engines based on the rotation speed.
- Generic reading;
 - The Behavior will only read an external software variable. For example, this value can be used in a script.
- Reset;
 - Reset the simulation to its initial state if the current Behavior value is different from 0. The “Robot and bottles” example illustrates this type of Behavior.
- Collision test with other 3D Sprites;
 - Used to obtain the number of collisions between the parent 3D Sprite and the other Sprites of the current World. The “Name/s of other 3D Sprites” parameter is used to limit the action of this Behavior to a group of 3D Sprites (see below). Collision tests should only be used when strictly necessary.

See the “Conveyor” example for an illustration of this Behavior.

- Test if the joint is destroyed;
 - Used to obtain the state of the joint between the 3D Sprite associated to a Behavior and its parent.
- Obtain penetration with other 3D Sprites;
 - Gives the penetration depth between the 3D Sprite associated to the Behavior and the other 3D Sprites. This use is typically the proximity sensor. The “Name/s of other 3D Sprites” parameter is used to limit the action of this Behavior to a group of 3D Sprites (see below). Penetration tests should only be used when strictly necessary. See the “NXT” example for an illustration of this Behavior.
- Obtain information on a 3D Sprite;
 - Used to access the dynamic values of a 3D Sprite. The “Select information to read” parameter determines the value.
- Joint position test;
 - Used to test whether a joint value is between two limits. The typical use is the simulation of a position sensor on an actuator. “Min position” and “Max position” are the limits. This is illustrated in the “Manipulator with cylinders and suctions” example.
- Generic writing;
 - The Behavior will only write an external software variable. For example, this value can be calculated in a script.

Strength

Defines the strength value on each of the axes for the involved Behavior types.

Apply to brothers

If true, the behavior is applied to the concerned 3D Sprite and all the brothers. See the “Conveyor” example for an illustration of this parameter.

Position / Rotation / Color

Values used for the Behaviors which need them.

Links

Based on the selected driver in the Universe properties, the definition of an external variable name will appear specific to each external software.

Initial value

This will be recopied in the current value when going to simulation RUN mode. It can be used to permanently enable a Behavior. For example, a script can be unconditionally executed from simulation running by putting this property to 1.

Current value, internal current value, conversion of data, write mode

The Behavior values and the conversion mode, see the “External links” chapter for more information.

Names of other 3D Sprites

Certain Behaviors can use a group of 3D Sprites. By default, if this parameter is blank, all of the current World 3D Sprites are concerned. By documenting this parameter, the range of the Behavior is limited to the 3D Sprites whose name starts with the text contained in it. For example, “DUST” will limit the Sprites to those whose name starts with “DUST”. See the “Vacuum robot” example for an illustration of this.

Use the value of this Behavior

If not blank, this area gives the name to a Behavior whose value will be read and recopied in the current internal value. See the “Script” chapter for more information on the name conventions for Behaviors.

External link

If true, the Behavior will be listed in the list of external links (see the “External links” chapter).

Sounds

Minimum distance

Used to edit the ratio between the volume sound and the distance of the object generating the sound and the camera.

Script

See the following chapter

Script

The concept of Script is one of the most powerful tools of Virtual Universe. It is used to integrate very sophisticated treatments to the simulation. Scripts are activated by Behaviors. Each Behavior can activate a script which will be a completely autonomous task. The Script is executed when the current associated Behavior value is different from 0 and the script has not ended. The script ends if the last execution line is reached or the END instruction is executed. Basic language is used. Specific instructions can be used to access the values associated to objects in read or write.

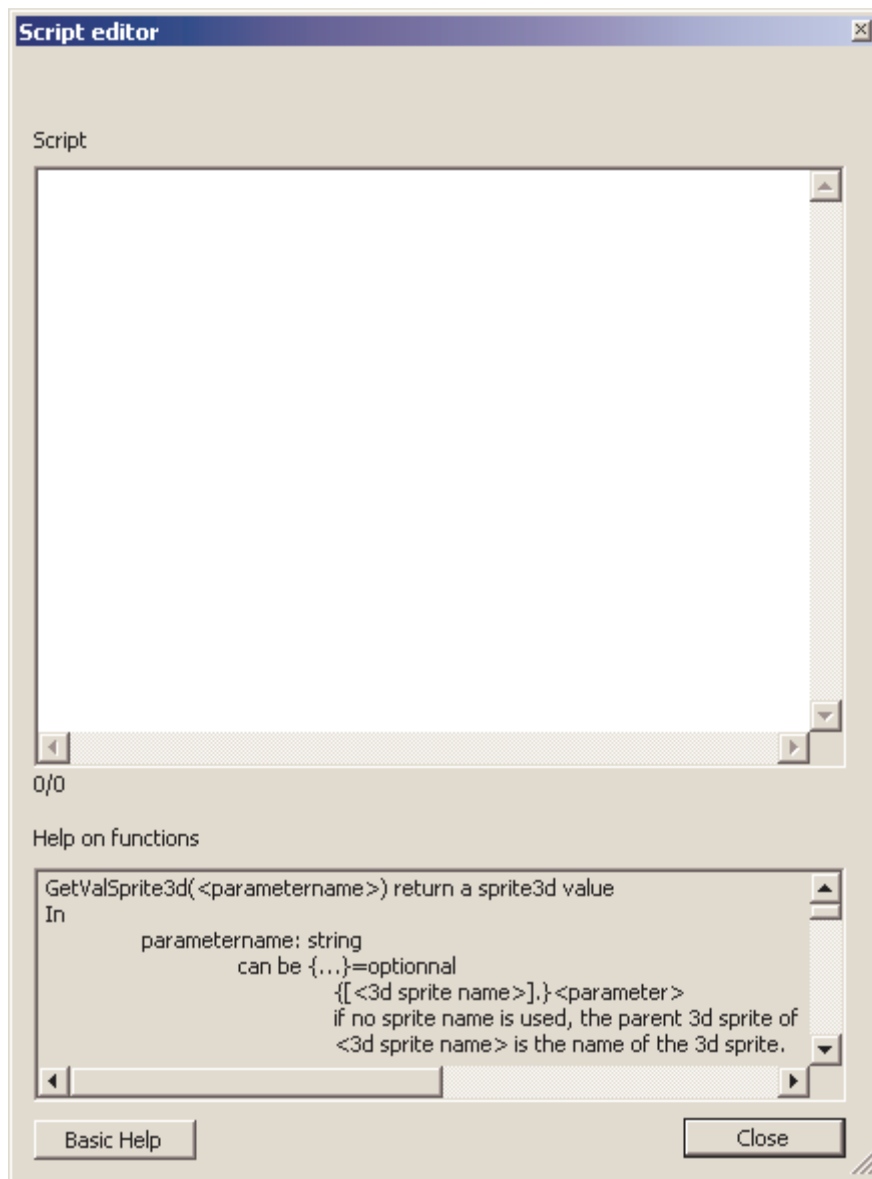
The script execution priorities can be edited in the properties of the associated Behavior. The “Normal” priority corresponds to executing a script element every 10 ms., the high priority corresponds to executing all the scripts every 10 ms. Other lower priorities are also accessible. High priority should not be used unless necessary (or a short script): it uses up more processor time.

The Script is based on BeeBasic software.

For more information, see the help file `basic_api.chm` located in the Virtual Universe installation directory, or click on the “Basic Help” button.

Write a script

Scripts are written in the Behaviors “Script” parameter. A script editing window opens.



An editing area as well as a help button on the specific functions is displayed. The “Script Error” Behavior element is used to obtain an error possibly encountered in script analysis (the script in question is not executed in this case but the simulation can still go to RUN mode). If an error is detected, the line number is displayed so that the error can be found in the editor (the line and column numbers are displayed at the bottom of the editing area).

The “Script Output” Behavior element displays the outputs generated by the PRINT basic function. These outputs are also displayed in the

rendering window at the location of the 3D Sprite associated to the Behavior.

Specific functions

The specific functions are used to access the Virtual Universe values associated to objects in read or write.

3D Sprite name syntax

The name for reference to 3D Sprites must comply with the following syntax:

- a name without path: it will search for the first 3D Sprite whose name starts with this text in all the current World 3D Sprites
- `..\<name>` : a named 3D Sprite, brother of the parent 3D Sprite of the Behavior;
- `<name 1>**\<name2>` : a 3D Sprite named name2, descendant of a 3D Sprite named name1.
- `<path\name>` : a 3D Sprite corresponding to the path.

These names are not case sensitive.

Examples:

`my sprite`: indicates the first 3D Sprite whose name starts with the text “my sprite”.

`..\another sprite`: indicates the 3D Sprite named “another sprite”, brother of the parent 3D Sprite of the Behavior;

`robot1**\level3` : indicates the 3D Sprite named “level3” descendant of the robot 1 3D Sprite.

`..\..\one more sprite`: indicates the 3D Sprite named “one more sprite”, a brother of the parent 3D Sprite parent of the Behavior;

Note: making reference to relative names (using relative paths) rather than to absolute names makes it possible to have easy to duplicate objects without having to change the links. The “ABB Robot” example illustrated here.

Access functions to values associated to a 3D Sprite

GetValSprite3d(<parameter>) returns a value associated to a 3D Sprite

<parameter> indicates the parameter. It can indicate a 3D Sprite by its name. If this is not the case, the parent 3D Sprite of the Behavior is used.

The syntax is [<3D Sprite name>].<parameter name>

Examples:

POSX: position X of parent 3D Sprite

[BOX3].SPEEDZ speed on axis Z of the 3D Sprite named BOX3

List of possible parameters:

POSX, POXY, POSZ : absolute position in the 3D World.

ROTX, ROTY, ROTZ : absolute rotation in the 3D World.

RELPOSX, RELPOSY, RELPOSZ : relative position based on the parent 3D Sprite. Only valid if the object is managed by the physics engine and is linked to a parent 3D Sprite by a joint.

RELROTX, RELROTY, RELROTZ : relative rotation based on the parent 3D Sprite. Only valid if the object is managed by the physics engine and is linked to a parent 3D Sprite by a joint.

SCALEX, SCALEY, SCALEZ : scale.

FORCEX, FORCEY, FORCEZ : applied strength.

TORQUEx, TORQUEY, TORQUEZ : applied torque.

FORCEBRAKEX, FORCEBRAKEY, FORCEBRAKEZ : applied brake strength.

TORQUEBRAKEX, TORQUEBRAKEY, TORQUEBRAKEZ : applied brake torque.

SPEEDX, SPEEDY, SPEEDZ : Speed

ROTSPEEDX, ROTSPEDY, ROTSPEEDZ : Angular speed

RELSPEEDX, RELSPEEDY, RELSPEEDZ : Relative speed

RELROTSPEEDX, RELROTSPEEDY, RELROTSPEEDZ : Relative angular speed (at parent)

TRANSPARENCY : Transparency (from 0=opaque to 1=invisible)

SetValSprite3d(<parameter>,<value>) edits a value associated to a 3D Sprite

<parameter> is identical to GetValSprite3d with more:

JOINTMIN, JOINTMAX : minimum and maximum value of the joint with the parent

JOINTMIN2, JOINTMAX2 : minimum and maximum value of the second joint

Behavior name syntax

The name for reference to Behaviors must comply with the following syntax:

- a name without path: it will search for the first Behavior whose name starts with this text in all the current World 3D Behaviors
- ..\<name> : a named Behavior brother of the current behavior;
- <sprite name>\<Behavior name> : a named Behavior child of a 3D Sprite. The 3D Sprite name must meet the criteria defined in the “3D Sprite name syntax” chapter

Access functions to values associated to Behavior

GetBehavior(<parameter>) returns a value associated to a Behavior

<parameter> can be a Behavior name or a Behavior name and value type.

The syntax is:

[<Behavior name>].<value type>

Or

[<Behavior name>]

If the value type is omitted, the current value is referenced.

The possible value type is “internalvalue” to access the current internal value.

Examples:

[MY BEHAVIOR] : current value of the Behavior names “MY BEHAVIOR”.

[MY DOG].internalvalue : current internal value of the Behavior named “MY DOG”.

[robot1**\level2\position] : current value of the Behavior named “position” child of the 3D Sprite named “level2” descendant of the 3D Sprite named “robot1”.

[..\..\request].internalvalue : current internal value of the grandparent of the parent of the Behavior.

SetBehavior(<parameter>,<value>) writes the value of a Behavior
<parameter > is identical to GetBehavior.

Access functions to values associated to the Universe

GetUniverse(<parameter>) returns a value associated to a Universe
<parameter> may be:

- RUNNINGDURING : returns the duration in ms since the last switch to RUN of the simulation
- MOUSEBUTTONS : returns the state of the mouse buttons : bit 0 for the left button, bit 1 for the right button and bit 2 for the middle button.
- MOUSEX, MOUSEY : returns the position of the mouse cursor related to the upper left corner of the rendering window.

SetUniverse(<parameter>,<value>) writes a value associated to the Universe

<parameter> may be

PLEASEQUIT : forces termination of Virtual Universe

Other functions

GetFirstSprite3D(<name>) returns the first number of a 3D Sprite

The name must comply with the 3D Sprite name syntax. The numeric value returned may be directly passed as a parameter to the access functions to the values associated to 3D Sprites in the form of “#number”. If the returned value is less than 0, no 3D Sprite was found. See the “Vacuum Robot” example for an illustration.

GetNextSprite3D(<number>,<name>) returns the number of the next 3D Sprite.
<number> is the value returned by GetFirstSPrite3d.

If the returned value is less than 0, no 3D Sprite was found.

Rand() : returns a random value between 0 and 1

See the “Vacuum Robot” example.

ComputeIK(<ndof>,<x>,<y>,<z>,<a>,,<c>,<tx>,<ty>,<tz>,<b1>,<b2>,<b3>,<b4>,<b5>,<b6>) calculates the inverse kinematic resolution of a robot

The associated Behavior must be the child of 3D Sprite composing the last element of the robot.

See the “ABB Robot” example.

<ndof> : number of degrees of freedom (must be 6) ;

<x,y,z> : position to reach ;

<a,b,c> : desired angle for the last element;

<tx,ty,tz> : tool movement ;

<b1> to <b6> : name of the 6 Behaviors which will receive the values for each of the 6 axes.

The return value is:

0 : no error, the values were calculated;

-7 : the position cannot be reached;

Another value < than 0 : error.

Object library

It is possible to import and export “complex” objects composed of 3D Sprites, Lights and Behaviors.

Importing of complex objects is implemented by clicking on a World or on a 3D Sprite with the right button of the mouse and selecting “Import”. Examples of objects are located in the “library” sub-directory of the Virtual Universe installation directory.

Exporting is implemented by clicking on a 3D Sprite with the right button of the mouse and selecting “Export”. All of the “Children” elements are exported.

External links

The external links are used to control simulations created in Virtual Universe by an external software (for example AUTOMGEN).

The connection type is set in the Universe properties.

Exchanges are enabled when Virtual Universe is in RUN mode and the external software is capable of performing these exchanges.

The connection state is displayed in the Universe properties.

A link is established between the external software and a Behavior.

Based on the selected external software, a specific variable name can be documented in each Behavior.

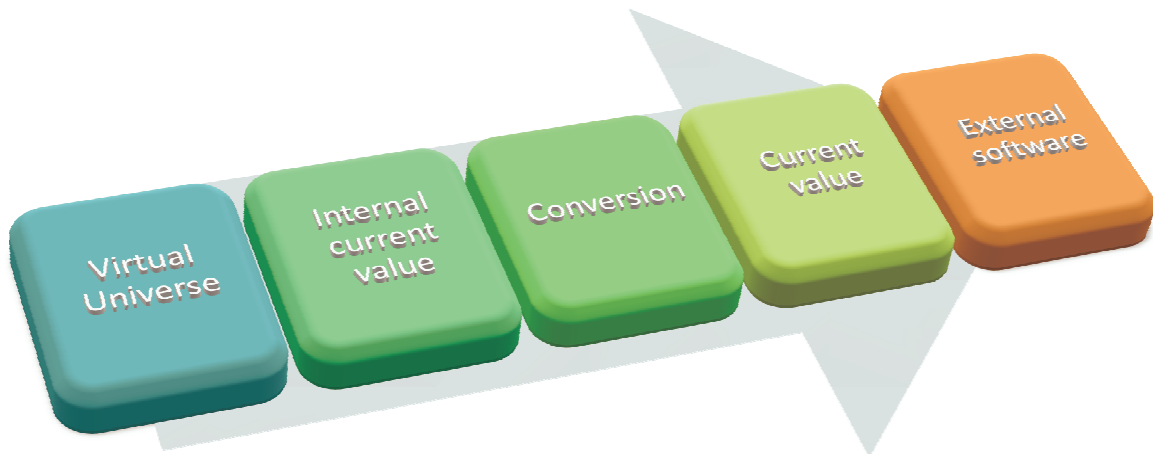
Current value and internal current value

Each Behavior possesses two states : a current state and an internal current state. These two states are used differently and inversely based on the information direction: external software towards Virtual Universe or Virtual Universe towards the external software.

Reading a Virtual Universe value from the external software

This action can be described as “reading an input” from the external software viewpoint.

The data path is as follows:



The conversion type may be simple recopying of the value or an inversion (for complemented Boolean variables).

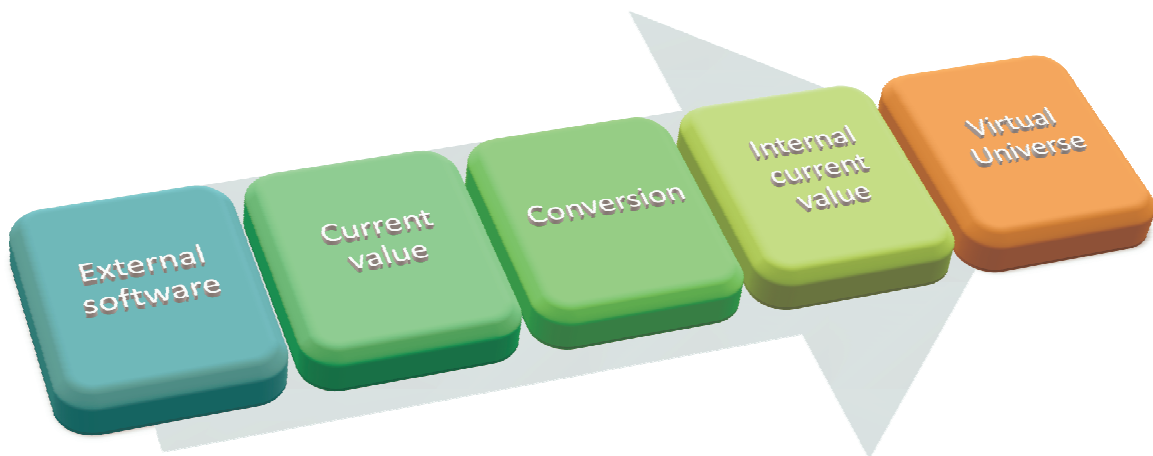
In complement a “write mode to external software” can be specified. Three modes are available:

- « Normal »: the value is written with each exchange;
- “Only when changed”: the value is only written to the external software if it has changed (writing to certain external software may use resources, the aim of this option is to mitigate the impact of writing in terms of resources);
- « Safe »: the value is written with each change; Each writing is checked (reading of the value after writing). This mode guarantees that a fugitive state change (normally a true view sensor during a very short duration – less than the data exchange time between the Virtual Universe and external software – will be “seen” by the external software. This is used in the “Conveyor” example.

Writing an external software value to Virtual Universe

This action can be described as “writing an output” from the external software viewpoint.

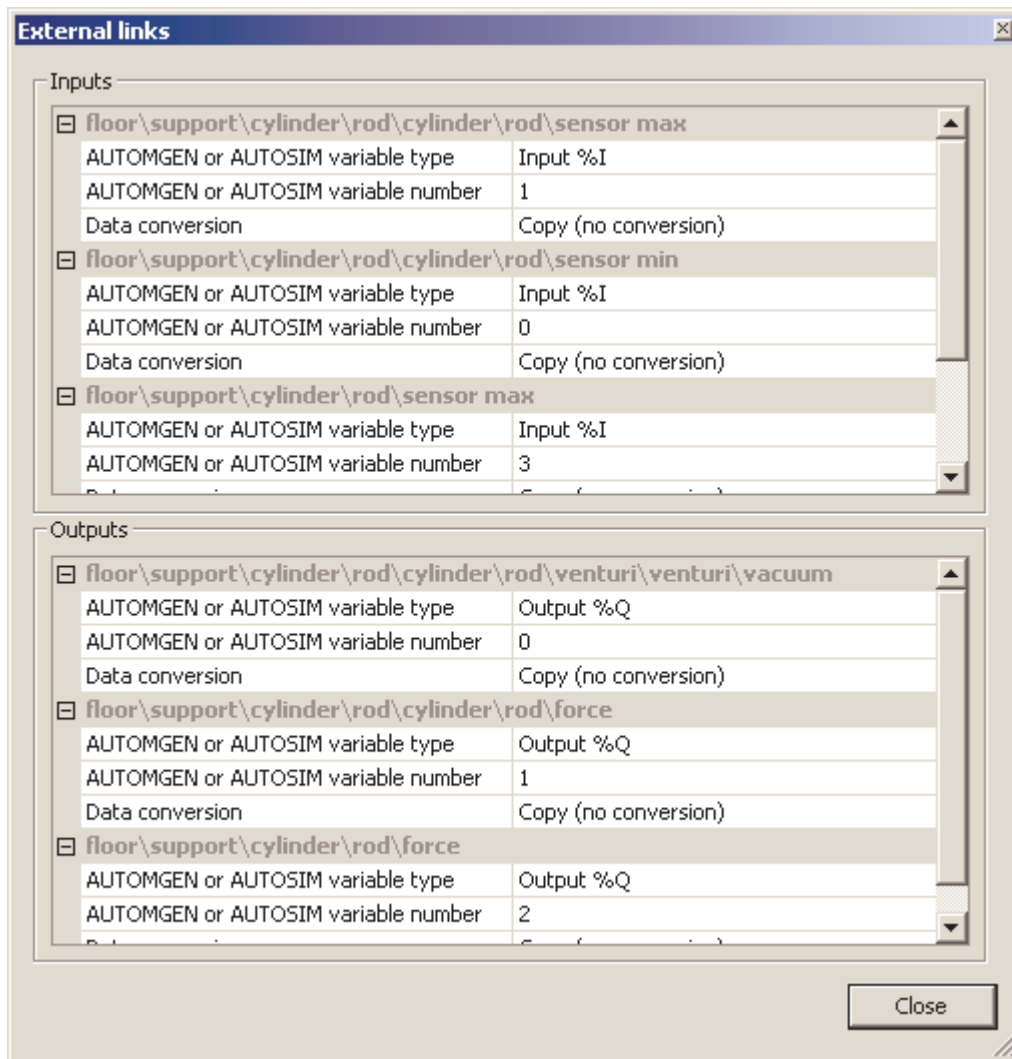
The data path is as follows:



The conversion type may be simple recopying of the value or an inversion (for complemented Boolean variables).

Access to the external links of an object group

It is possible to easily access all of the inputs and outputs associated to an object group by clicking with the right button of the mouse on the parent (click on the World to have all of the links of objects found in the World) and selecting “External links”. Examples:

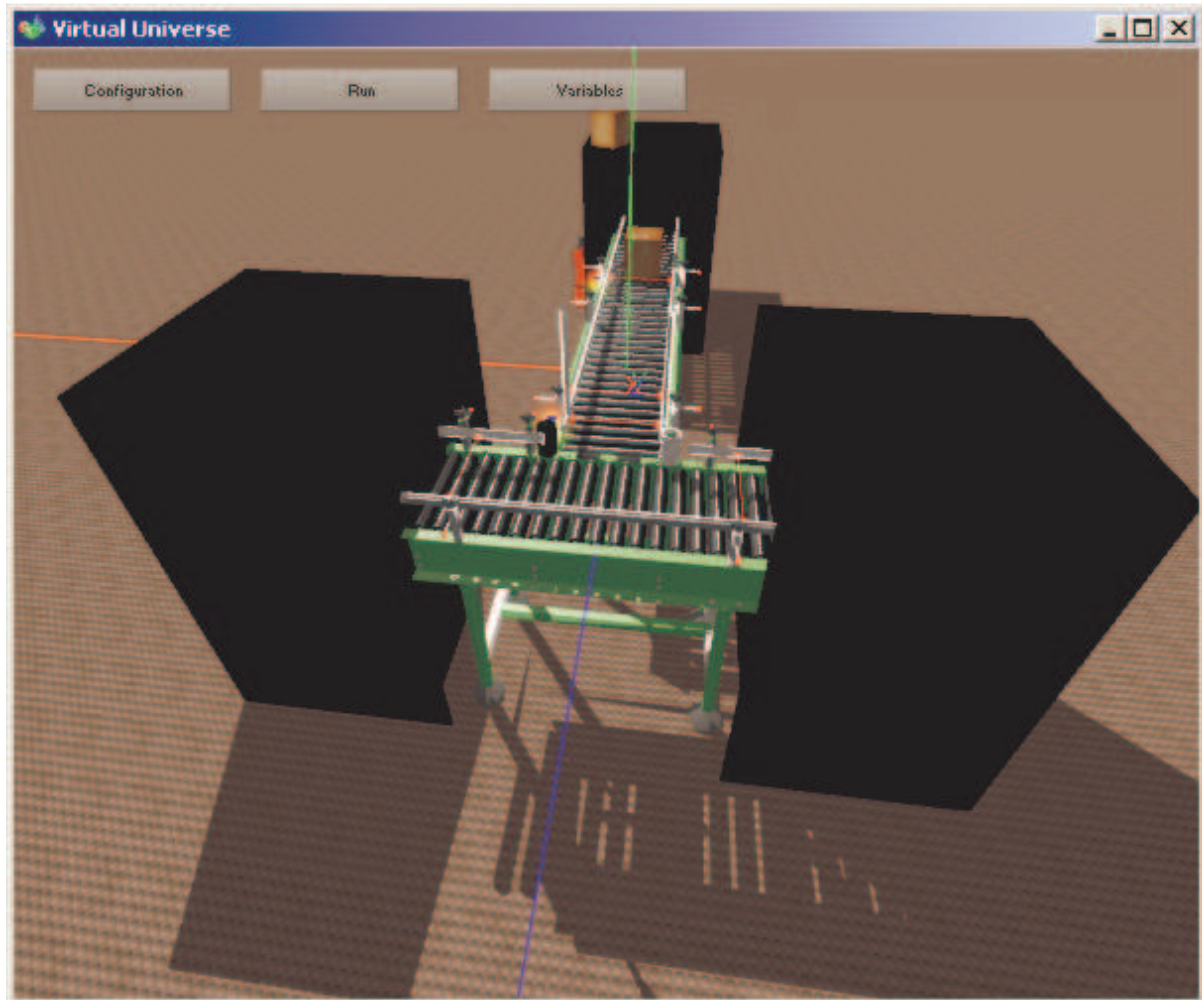


The typical application of this is to edit the attribution of inputs and outputs of an object after importing or duplication. The variables associated to inputs and outputs depend on the type of driver (type of external software) selected in the Universe properties.

Examples

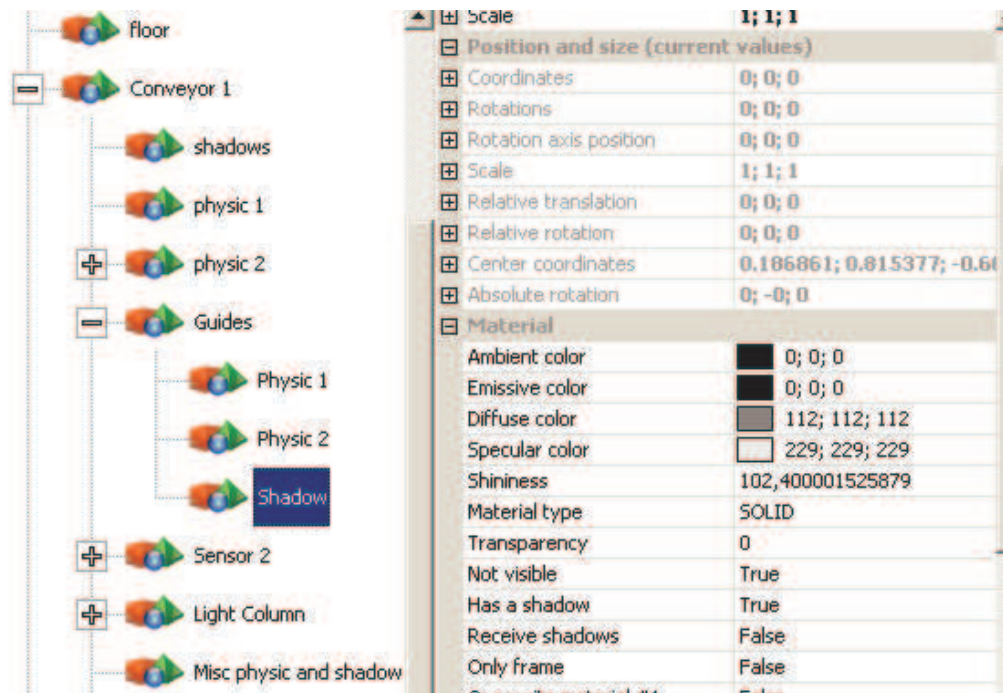
Conveyor

The conveyor project is located in the “samples\conveyor” sub-directory of the Virtual Universe installation directory. It is accompanied by an .AGN project for the AUTOMGEN or AUTOSIM software and an .XEF project for the Unity Pro software.

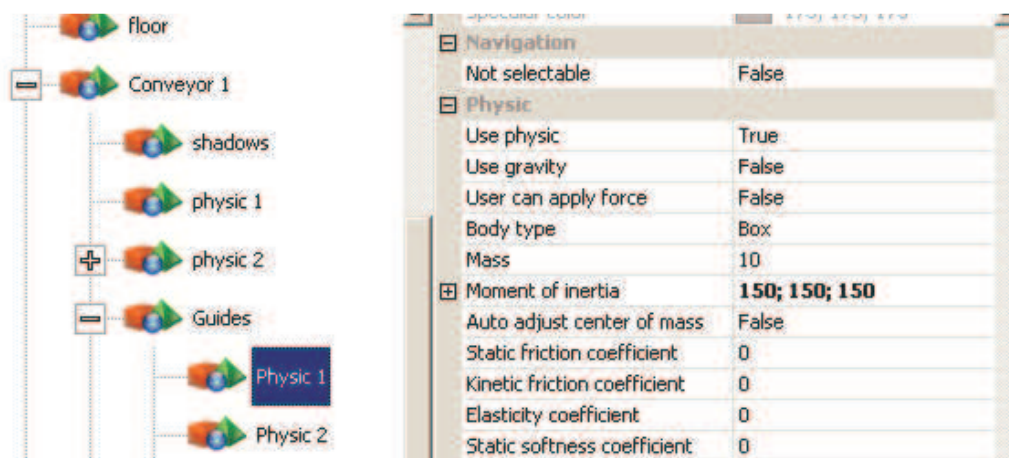


This example illustrates the following functionalities in particular:

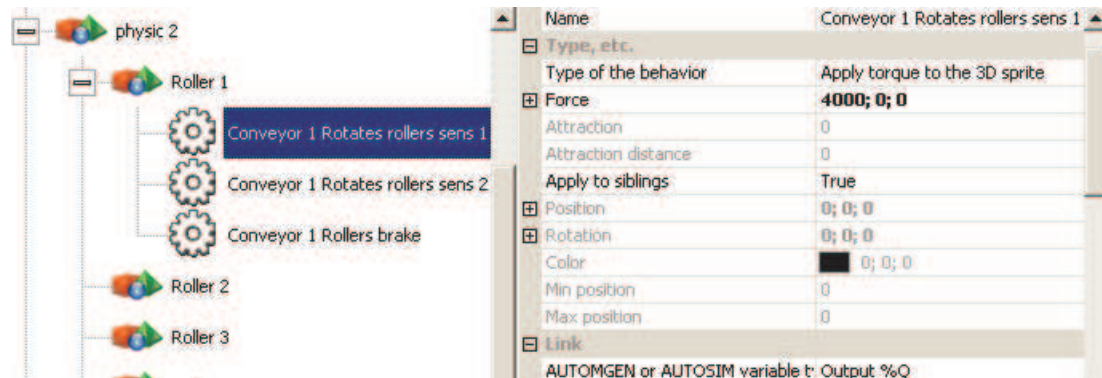
- Shadingmanagement : here “lightened” (with fewer faces) 3D Sprites have been used to render the shading in order not to slow down the rendering too much.



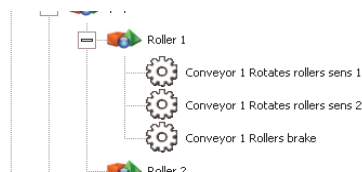
- Physical management : the same principle has been used for “physical” management.



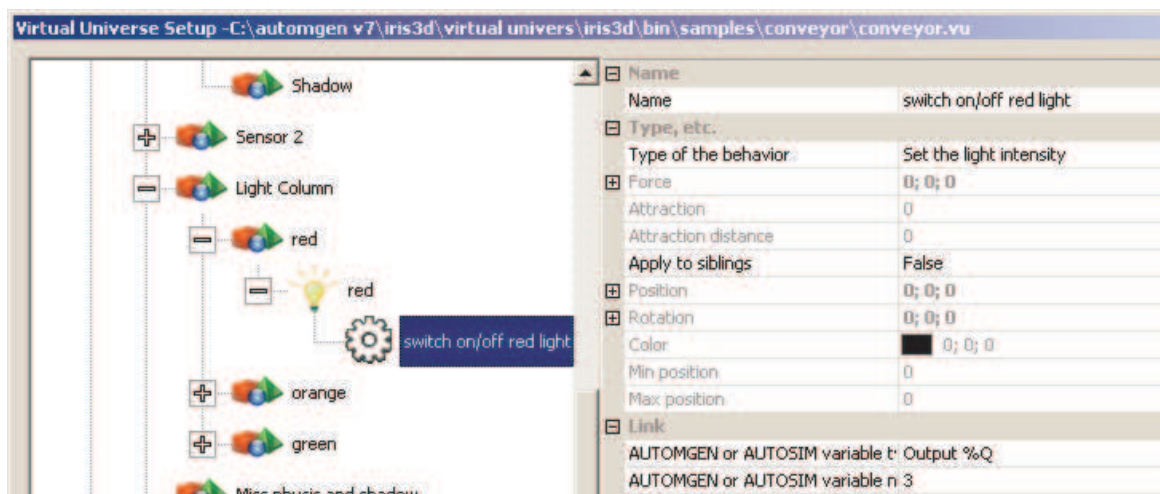
- Applications of a Behavior to an object group: the torque applied to the rollers of each conveyor is generated by a sole Behavior with the “apply to siblings” attribute.



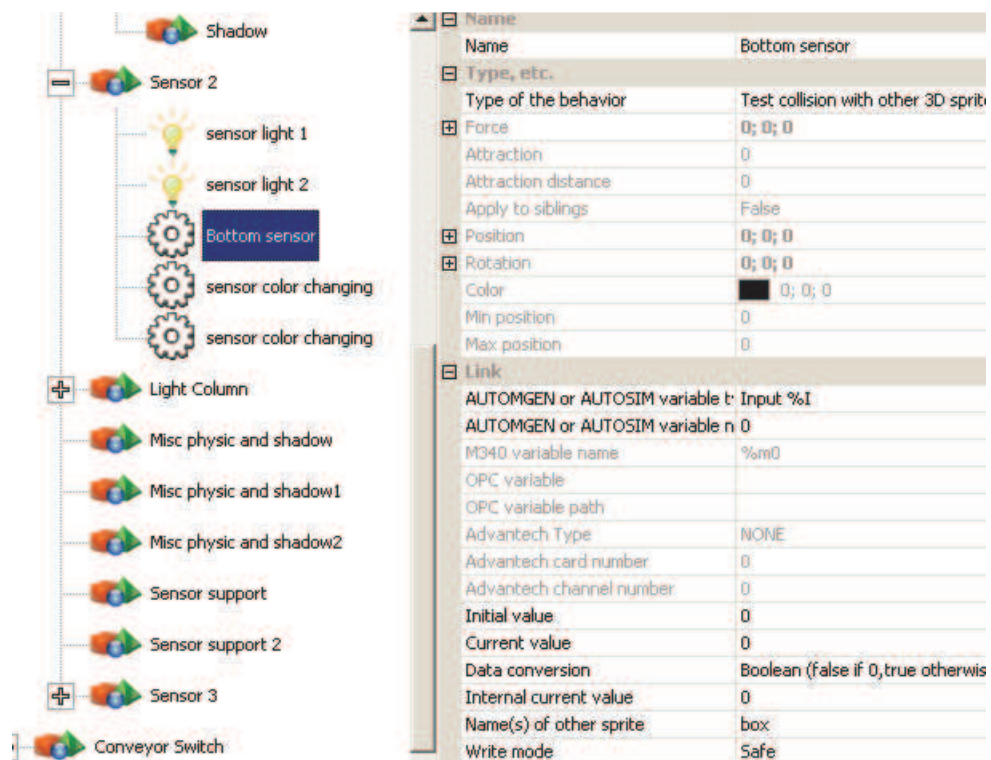
- Forward run, backward run and brake simulation for the conveyors.



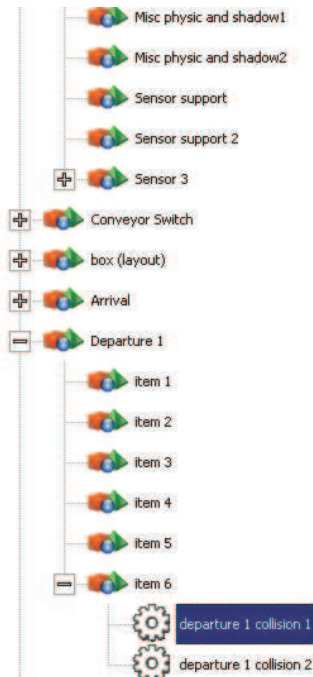
- Simulation of a light column.



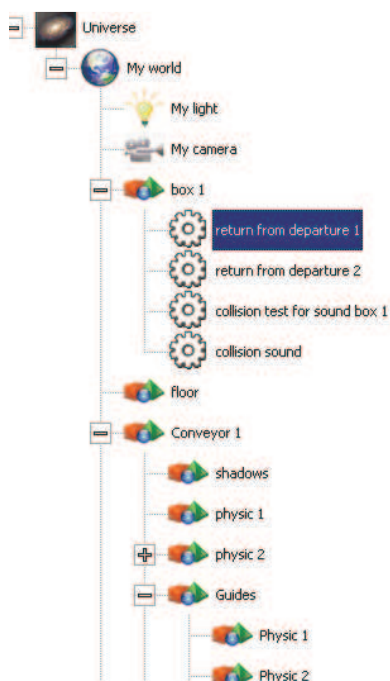
- Management of the sensors has been implemented with “Collision test” Behaviors. Management of the “stealth” of information from the sensors has been treated with a “safe” write mode so that the external software can “see” the information in a certain manner.



- Return of the boxes to the start position is managed by the pair of Behaviors “Collision test” and “Writing of position”.

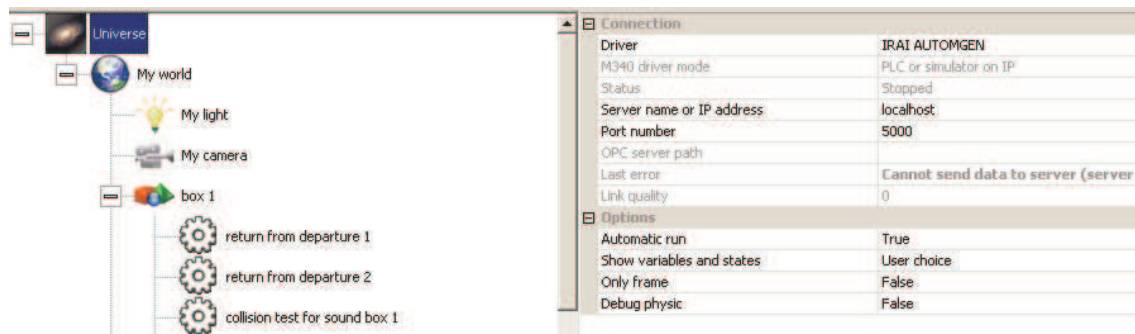


Name	departure 1 collision 1
Type, etc.	
Type of the behavior	Test collision with other 3D s
Force	0; 0; 0
Attraction	0
Attraction distance	0
Apply to siblings	False
Position	0; 0; 0
Rotation	0; 0; 0
Color	0; 0; 0
Min position	0
Max position	0
Link	
AUTOMGEN or AUTOSIM variable type	NONE
AUTOMGEN or AUTOSIM variable number	0
M340 variable name	
OPC variable	
OPC variable path	
Advantech Type	NONE
Advantech card number	0
Advantech channel number	0
Initial value	0
Current value	0
Data conversion	Copy (no conversion)
Internal current value	0
Name(s) of other sprite	box 1
Write mode	Normal
Get value from this behavior	
External value	0
Last value written to the external software	1234
Need to be written	0

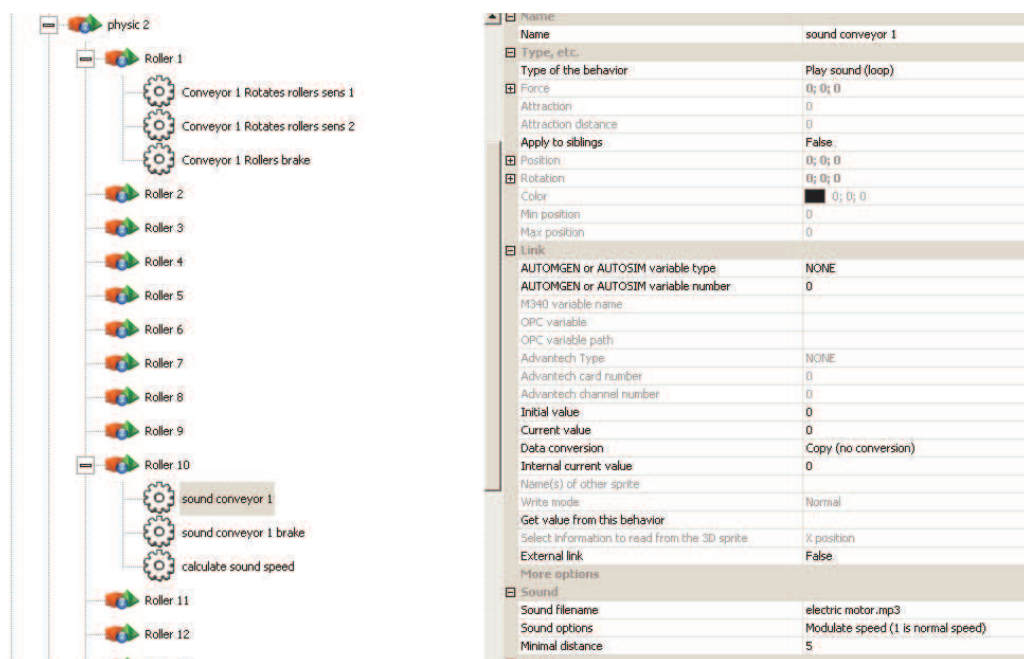


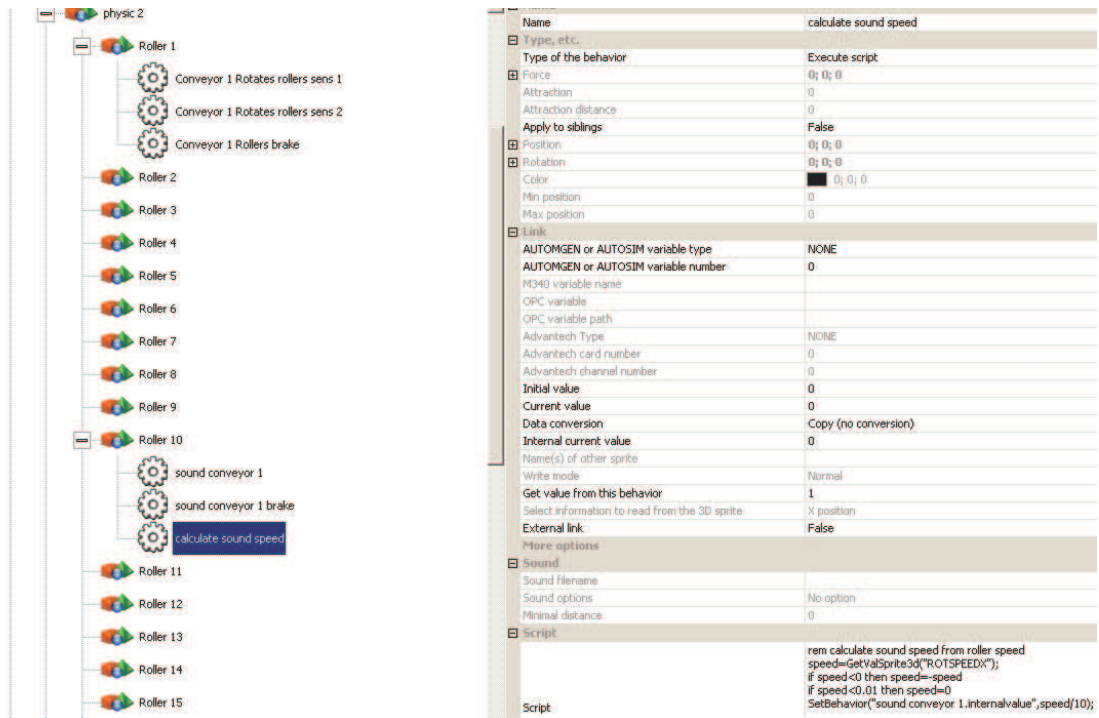
Name	return from departure 1
Type, etc.	
Type of the behavior	Set the 3D sprite position and rotation
Force	0; 0; 0
Attraction	0
Attraction distance	0
Apply to siblings	False
Position	0; 7; -84
Rotation	0; 0; 0
Color	0; 0; 0
Min position	0
Max position	0
Link	
AUTOMGEN or AUTOSIM variable type	NONE
AUTOMGEN or AUTOSIM variable number	0
M340 variable name	
OPC variable	
OPC variable path	
Advantech Type	NONE
Advantech card number	0
Advantech channel number	0
Initial value	0
Current value	0
Data conversion	Copy (no conversion)
Internal current value	0
Name(s) of other sprite	
Write mode	Normal
Get value from this behavior	departure 1 collision 1
External value	0
Last value written to the external software	0
Need to be written	0

- The names and states of the variables are displayed in the rendering window (choice of the user selected in the Universe properties).



- Modulation of the sound speed based on the roller rotation speed. Calculation implemented in a script.





Operation

The boxes are two different sizes, the aim is to empty them towards two different destinations based on their size. Two sensors (a lower sensor and an upper sensor) are used to identify the box size. Only lower sensor = small box, lower and upper sensor = large box.






List of AUTOMGEN / AUTOSIM variable references
















Symbol	Variable	Comments
before switch low sensor	%i0	lower sensor before the switch conveyor
before switch high sensor	%i1	upper sensor before the switch conveyor
departure sensor backward	%i2	departure sensor behind switch conveyor
departure sensor forward	%i3	departure sensor before switch conveyor
arrival sensor	%i4	conveyor arrival sensor
middle conveyor forward	%q0	middle conveyor motor forward
middle conveyor backward	%q1	middle conveyor motor backward
middle conveyor brake	%q2	middle conveyor brake
middle conveyor red light	%q3	middle conveyor light column red light
middle conveyor orange light	%q4	middle conveyor light column orange light
middle conveyor green light	%q5	middle conveyor light column green light
switch conveyor forward	%q6	switch conveyor motor forward
switch conveyor backward	%q7	switch conveyor motor backward
switch conveyor brake	%q8	switch conveyor brake
arrival conveyor forward	%q9	arrival conveyor motor forward
arrival conveyor backward	%q10	arrival conveyor motor backward
arrival conveyor brake	%q11	conveyor motor brake
arrival conveyor orange light	%q12	arrival conveyor light column orange light
arrival conveyor red light	%q13	arrival conveyor light column red light
arrival conveyor green light	%q14	arrival conveyor light column green light

Input

Output

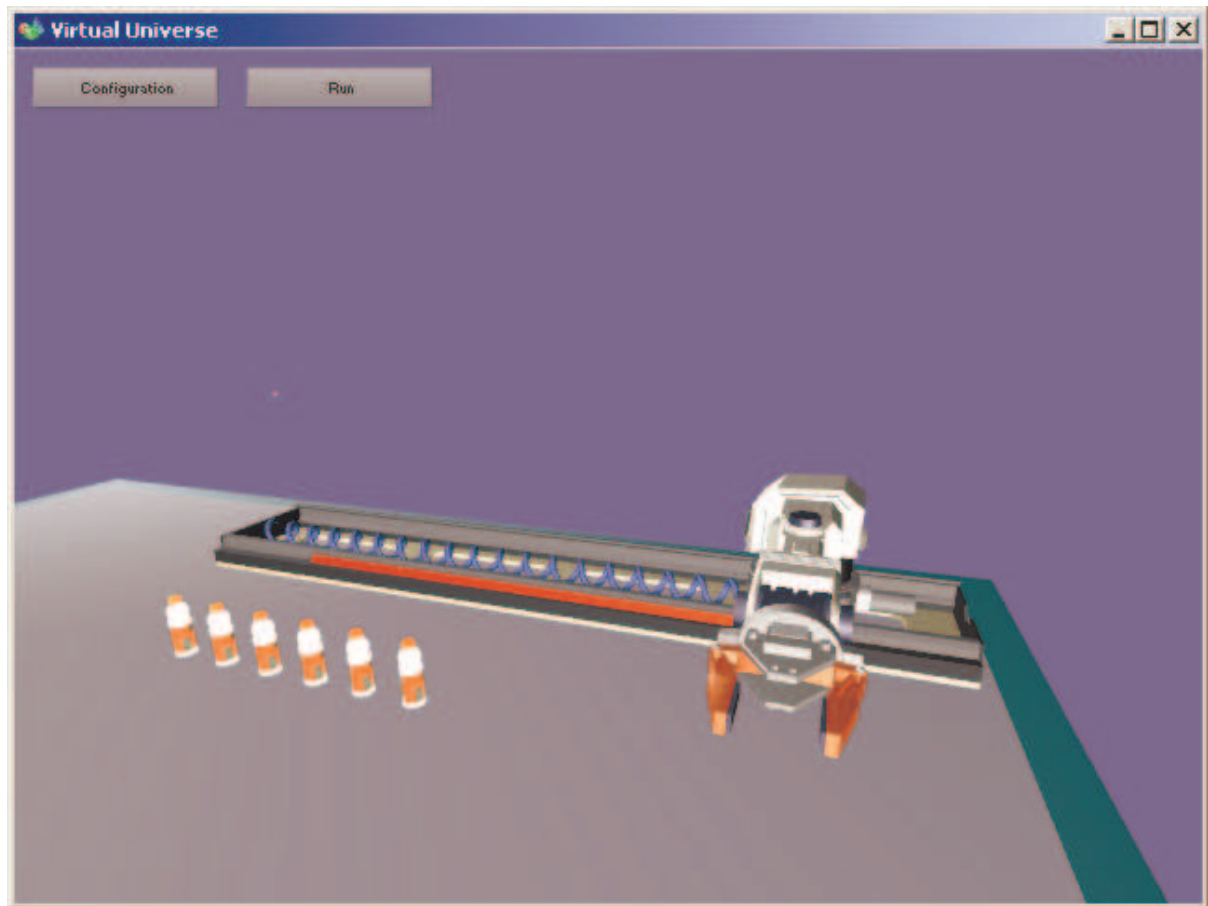
List of UNITY PRO variable references

..  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

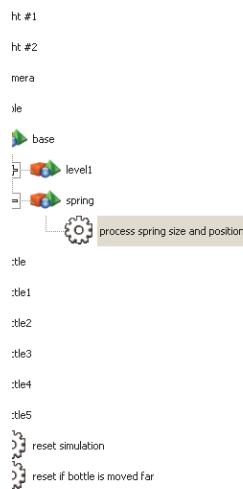
Robot and bottles

This project is located in the “samples\robot and bottles” sub-directory of the Virtual Universe installation directory. It is accompanied by an .AGN file.



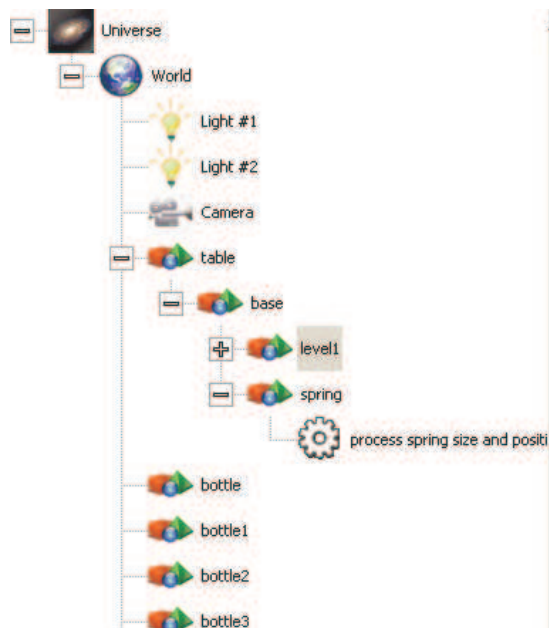
This project illustrates the following functionalities in particular:

- Modification of the scale and position for spring simulation;



Name	process spring size and position
Type, etc.	
Type of the behavior	Execute script
Force	0; 0; 0
Attraction	0
Attraction distance	0
Apply to siblings	False
Position	0; 0; 0
Rotation	0; 0; 0
Color	0; 0; 0
Min position	0
Max position	0
Link	
AUTOMGEN or AUTOSIM variable type	NONE
AUTOMGEN or AUTOSIM variable number	0
M340 variable name	
OPC variable	
OPC variable path	
Advantech Type	NONE
Advantech card number	0
Advantech channel number	0
Initial value	1
Current value	1
Data conversion	Copy (no conversion)
Internal current value	1
Name(s) of other sprite	
Write mode	Normal
Get value from this behavior	
Select information to read from the 3D sprite	X position
External link	False
More options	
Sound	
Sound filename	
Sound options	No option
Minimal distance	0
Script	<pre> level1pos=getvalsprite3d("T..level1,RELPOSZ") setvalsprite3d("SCALEZ",((level1pos+600)*0.001846)+0.1) setvalsprite3d("POSZ",((level1pos+600)*0.66)-320) </pre>
Script	

- Setting of joints;



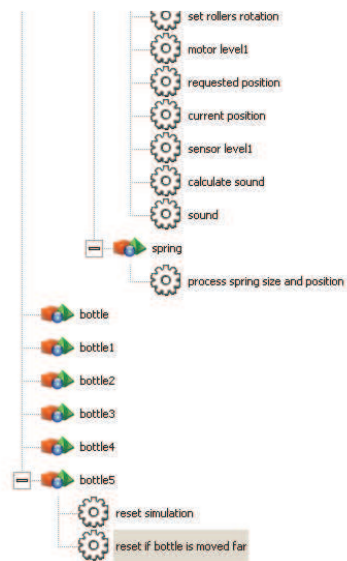
Name	level1
Drawing	
Position and size	
Position and size (current values)	
Material	
Material (current values)	
Navigation	
Physic	
Physic joint with parent	
Joint	Slider
Pivot joint position	0; 0; 0
Line of action	0; 0; 1
Joint min limit	0
Joint max limit	0,01
Joint power	4
Break joint force	0
Physic joint with an other 3D sprite	
Name of the 3D sprite	None
Joint	None
Pivot joint position	0; 0; 0
Line of action	0; 0; 0
Joint min limit	0
Joint max limit	0
Joint power	0
Break joint force	0

- Simulation of robot motors by modification of joint limits. Controlling by numeric variables (a variable associated to each axis) managed by a script for each motor.

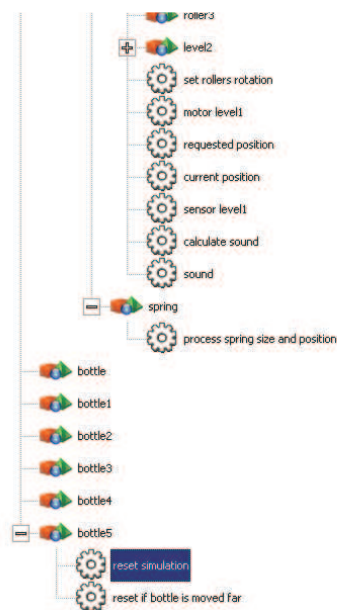


- Gripping: nothing in particular to do, closing of the clamp is sufficient.

- Simulation reset after movement of the last bottle.



Name	reset if bottle is moved far
Type, etc.	
Type of the behavior	Execute script
Force	0; 0; 0
Attraction	0
Attraction distance	0
Apply to siblings	False
Position	0; 0; 0
Rotation	0; 0; 0
Color	0; 0; 0
Min position	0
Max position	0
Link	
More options	
Sound	
Script	<pre> if getvalsprite3d("POSZ")>500 then setbehavior("reset simulation.intervalvalue",1) endif </pre>
Script	
Sound	Parameters regarding sound playing. This is 3D sound depending of the position of the parent.



Name	reset simulation
Type, etc.	
Type of the behavior	Reset (STOP then RUN)
Force	0; 0; 0
Attraction	0
Attraction distance	0
Apply to siblings	False
Position	0; 0; 0
Rotation	0; 0; 0
Color	0; 0; 0
Min position	0
Max position	0
Link	
AUTOMGEN or AUTOSIM variable type	NONE
AUTOMGEN or AUTOSIM variable number	0
M340 variable name	
OPC variable	
OPC variable path	
Advantech Type	NONE
Advantech card number	0
Advantech channel number	0
Initial value	0
Current value	0
Data conversion	Copy (no conversion)
Internal current value	0
Name(s) of other sprite	
Write mode	Normal
Get value from this behavior	
Select information to read from the 3D sprite	X position
External link	False
More options	
Sound	

Operation

Two numeric values are used for each of the axes. One gives the current position, the other is used to set the position to reach. If the current position is near the requested position, then the movement has been performed.

List of AUTOMGEN / AUTOSIM variable references

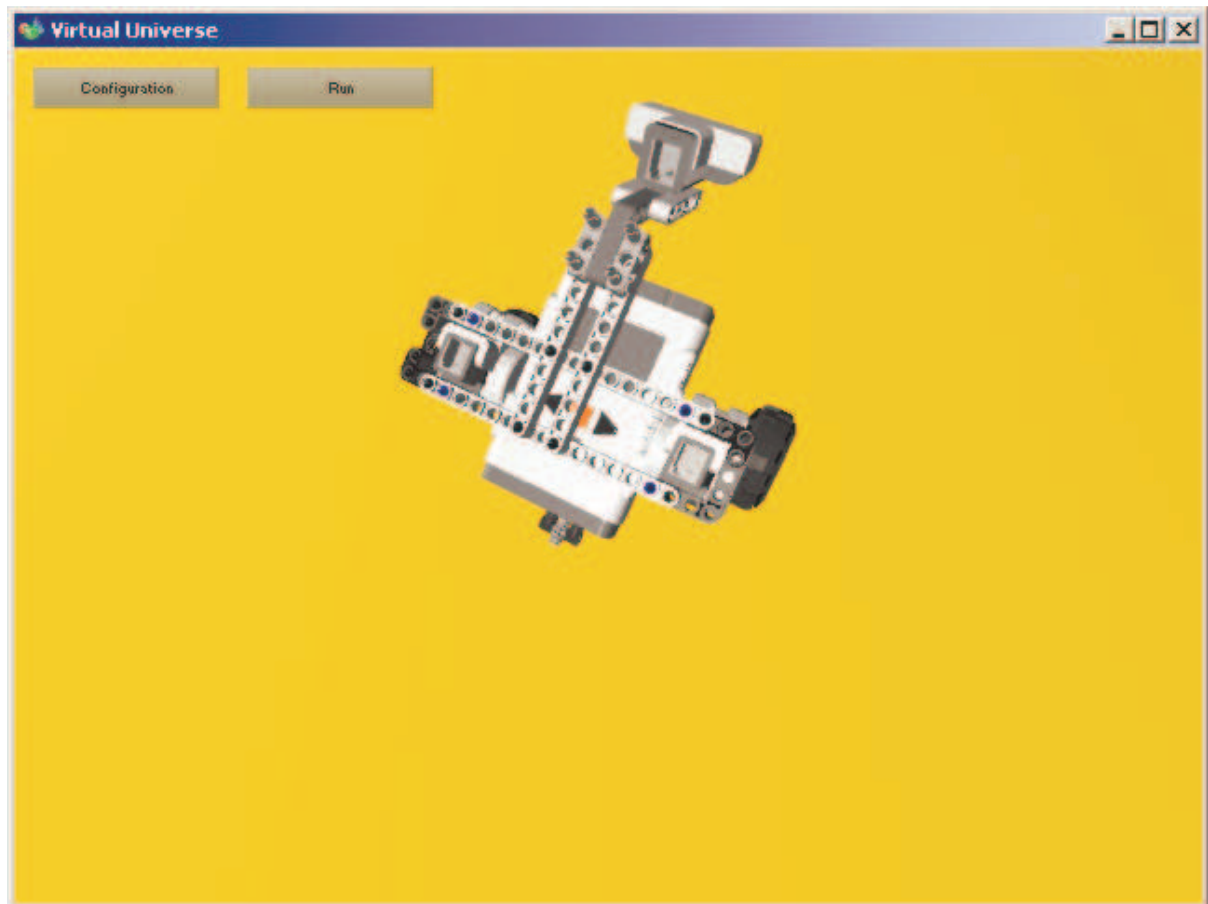
Symbol	Variable	Comments
request level 1	%mw200	Position requested for axis 1
position level 1	%mw201	Current position for axis 1
...		
request level 5	%mw208	Position requested for axis 5
position level 5	%mw209	Current position for axis 5
request finger 1	%mw210	Position requested for clamp finger 1
position finger 1	%mw211	Current position for clamp finger 1
request finger 2	%mw212	Position requested for clamp finger 2
position finger 2	%mw213	Current position for clamp finger 2

Input

Output

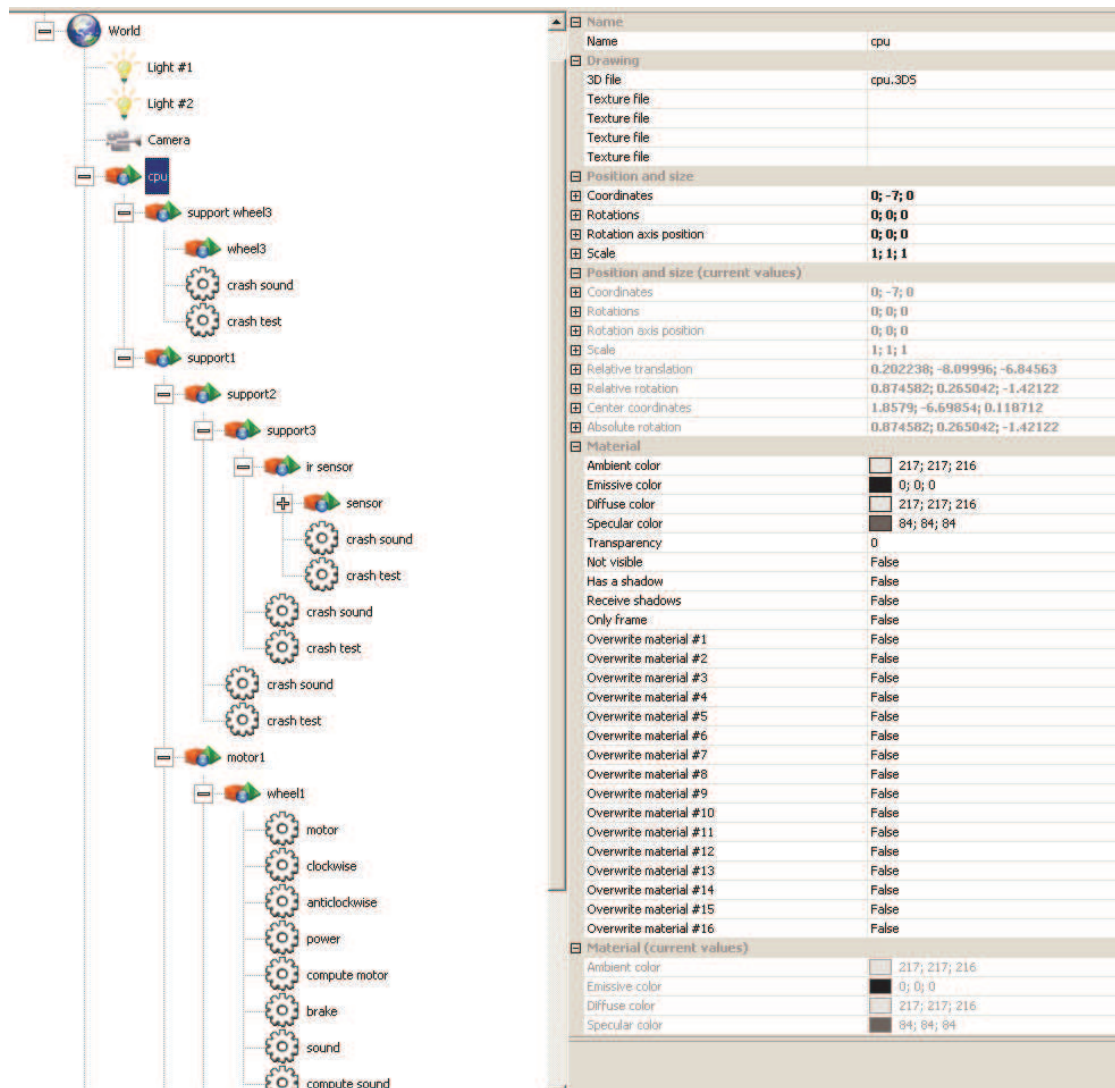
NXT robot

This project is located in the “samples\nxt” sub-directory of the Virtual Universe installation directory. It is accompanied by an .AGN file.

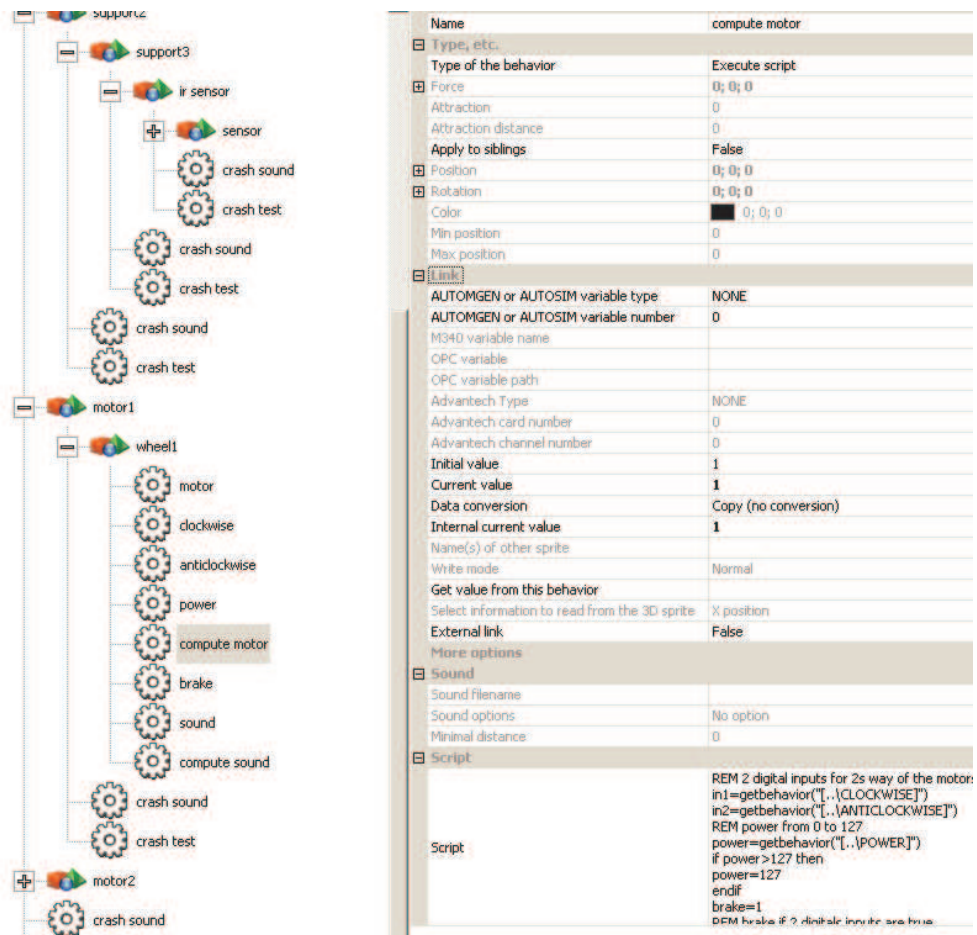


This project illustrates the following functionalities in particular:

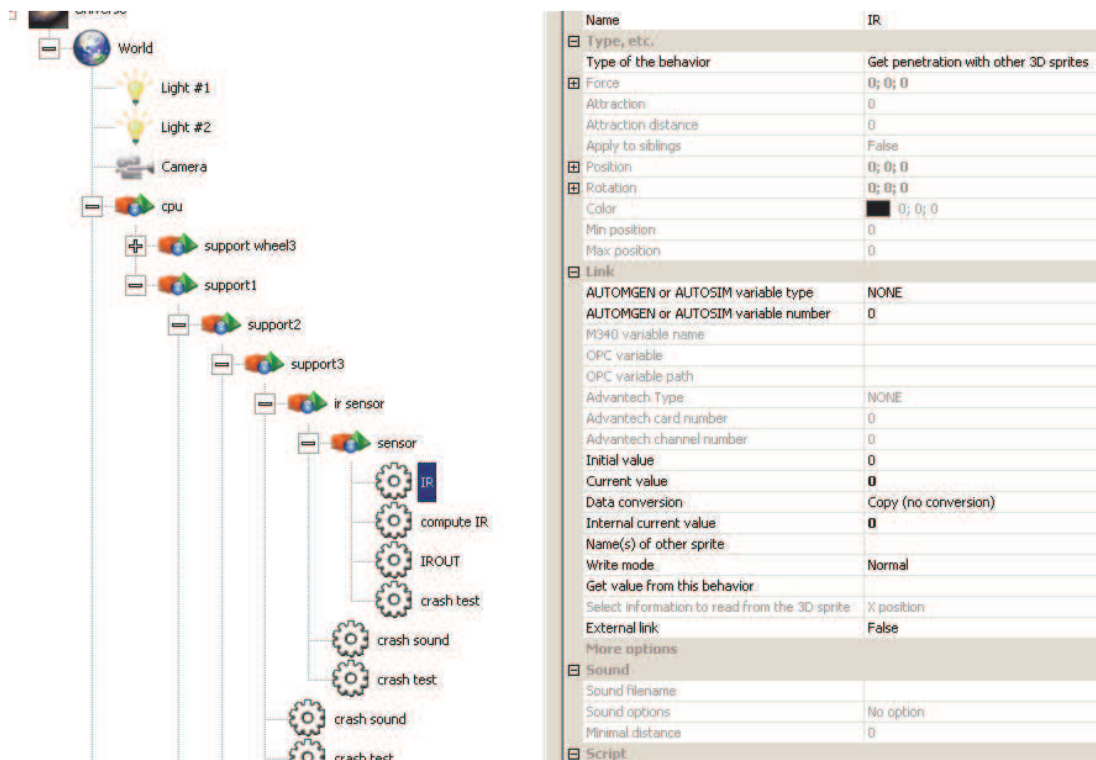
- Simulation of a mobile robot;



- Simulation of 2 variable speed motors modulation of the speed and brake controlled by 2 Boolean variables and a numeric variable.



- Analog proximity sensor simulated by a penetration test.



Name

Name compute IR

Type, etc.

Type of the behavior Execute script

Force 0; 0; 0

Attraction 0

Attraction distance 0

Apply to siblings False

Position 0; 0; 0

Rotation 0; 0; 0

Color 0; 0; 0

Min position 0

Max position 0

Link

AUTOMGEN or AUTOSIM variable type NONE

AUTOMGEN or AUTOSIM variable number 0

M340 variable name

OPC variable

OPC variable path

Advantech Type NONE

Advantech card number 0

Advantech channel number 0

Initial value 1

Current value 1

Data conversion Copy (no conversion)

Internal current value 1

Name(s) of other sprite

Write mode Normal

Get value from this behavior

Select information to read from the 3D sprite X position

External link False

More options

Sound

Sound filename

Sound options No option

Minimal distance 0

Script editor

Script

```

ir=getbehavior("I..IR")
if ir > 255 then
ir=255
else
if ir == -1 then
ir=255
else
ir=ir/2
endif
endif
setbehavior("I..IROUT",ir)

```

Script Basic

- Destruction of joints.

Name support1

Drawing

3D file support1.3DS

Texture file

Texture file

Texture file

Texture file

Position and size

Position and size (current values)

Material

Material (current values)

Navigation

Physic

Physic joint with parent

Joint Fix

Pivot joint position 0; 0; 0

Line of action 0; 0; 0

Joint min limit 0

Joint max limit 0

Joint power 0

Break joint force 2400

Physic joint with an other 3D sprite

Operation

The two wheels are controlled by motors whose power is controlled by numeric variables. Two boolean variables are used to control the motor in each direction.

List of AUTOMGEN / AUTOSIM variable references

Symbol	Variable	Comments
motor#1 power	%mw200	Motor 1 power
motor#2 power	%mw201	Motor 2 power
sensor	%mw203	Proximity sensor
motor#1 direction 1	%q0	Motor 1 Direction 1
motor#1 direction 2	%q1	Motor 1 Direction 2
motor#2 direction 1	%q2	Motor 2 Direction 1
motor#2 direction 2	%q3	Motor 2 Direction 2

Input

Output

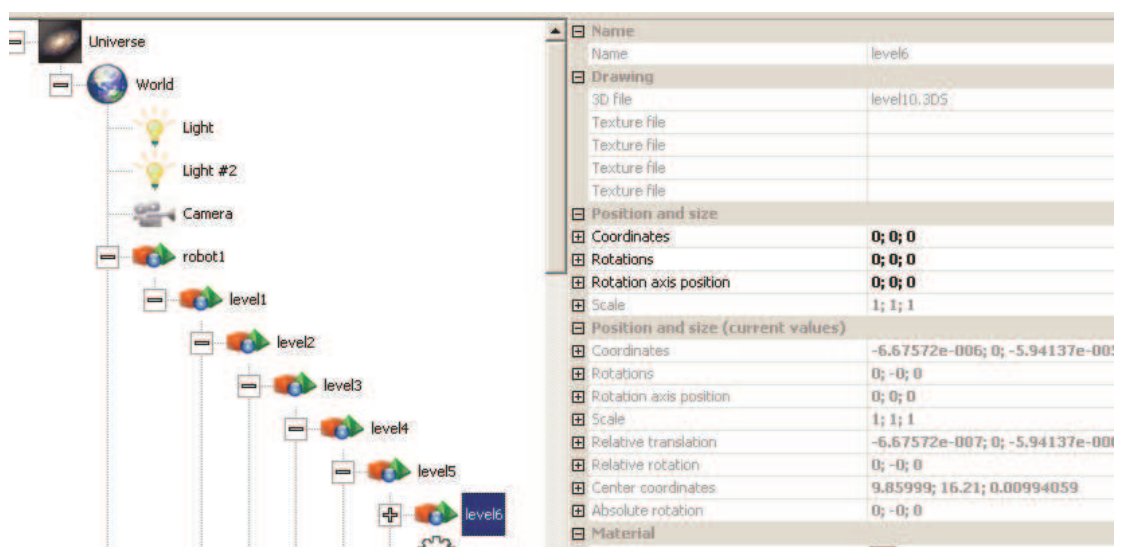
6 axis ABB Robot

This project is located in the “samples\abb robot” sub-directory of the Virtual Universe installation directory. It is accompanied by an .AGN file.

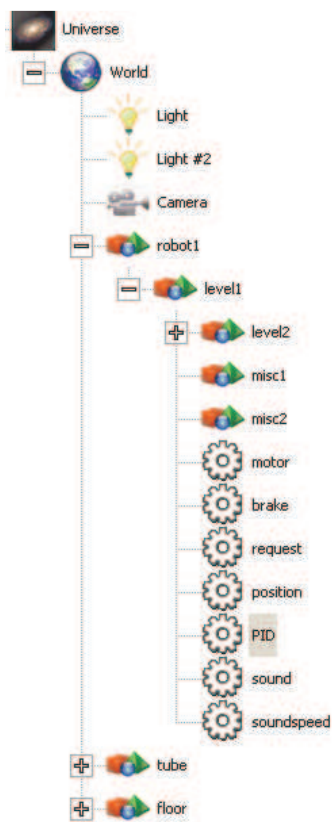


This project illustrates the following functionalities in particular:

- Simulation of a 6 axis robot.



- Simulation of a motor by PID.



Name	
Name	PID
Type, etc.	
Type of the behavior	Execute script
Force	0; 0; 0
Attraction	0
Attraction distance	0
Apply to siblings	False
Position	0; 0; 0
Rotation	0; 0; 0
Color	0; 0; 0
Min position	0
Max position	0
Link	
AUTOMGEN or AUTOSIM variable type	NONE
AUTOMGEN or AUTOSIM variable number	0
M340 variable name	
OPC variable	
OPC variable path	
Advantech Type	NONE
Advantech card number	0
Advantech channel number	0
Initial value	1
Current value	1
Data conversion	Copy (no conversion)
Internal current value	1
Name(s) of other sprite	
Write mode	Normal
Get value from this behavior:	
Select information to read from the 3D spr	X position
External link	False
More options	
Sound	
Sound filename	
Sound options	No option
Minimal distance	0

Script editor

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 dt=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-olderr)/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
```

- Inverse kinematic calculation of the robot.

The screenshot displays a software interface for configuring a robot. On the left, a vertical list of components is shown, including 'level6', 'clamp1', 'clamp2', 'clamp', 'motor', 'brake', 'request', 'position', 'PID', 'IK', 'pos1', 'pos2', 'pos3', 'pos4', 'pos5', 'pos6', 'reqx', 'reqy', and 'reqz'. The 'IK' component is highlighted. On the right, a detailed configuration panel for the 'IK' component is visible. It includes fields for 'Name' (IK), 'Type of the behavior' (Execute script), 'Force' (0; 0; 0), 'Attraction' (0), 'Attraction distance' (0), 'Apply to siblings' (False), 'Position' (0; 0; 0), 'Rotation' (0; 0; 0), 'Color' (0; 0; 0), 'Min position' (0), 'Max position' (0), 'Link' (AUTOMGEN or AUTOSIM variable type: Word %MW, AUTOMGEN or AUTOSIM variable number: 200), 'M340 variable name', 'OPC variable', 'OPC variable path', 'Advantech Type' (NONE), 'Advantech card number' (0), 'Advantech channel number' (0), 'Initial value' (1), 'Current value' (1), 'Data conversion' (Copy (no conversion)), 'Internal current value' (1), 'Name(s) of other sprite', 'Write mode' (Normal), 'Get value from this behavior' (Select information to read from the 3D sprite: X position), 'External link' (True), 'More options', 'Sound' (Sound filename, Sound options: No option, Minimal distance: 0).

Script editor

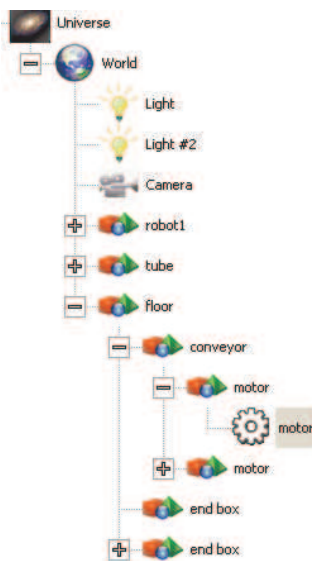
```

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].intervalvalue",0)
endif

```


- Simulation of a conveyor;



Name	motor
Type, etc.	
Type of the behavior	Apply local force to the collide 3D sprites
Force	0; 0; 0.001
Attraction	0
Attraction distance	0
Apply to siblings	False
Position	0; 0; 0
Rotation	0; 0; 0
Color	0; 0; 0
Min position	0
Max position	0
Link	
AUTOMGEN or AUTOSIM variable type	NONE
AUTOMGEN or AUTOSIM variable number	0
MS40 variable name	
OPC variable	
OPC variable path	
Advantech Type	NONE
Advantech card number	0
Advantech channel number	0
Initial value	1
Current value	1
Data conversion	Copy (no conversion)
Internal current value	1
Name(s) of other sprite	bottle
Write mode	Normal
Get value from this behavior	
Select information to read from the 3D sprite	X position
External link	False
More options	

Operation

Six numeric variables are used to select the destination and orientation of the robot clamp. The precision of the destination position is also given by a numeric variable. The higher the requested precision, the longer the time to end the movement, inversely, less precision makes it possible to chain the movements faster to the detriment of precision. A control numeric variable is used to run the movement, a response numeric variable is used to know if the movement has been performed. The coordinate system is the one really used by the real ABB robot.

List of AUTOMGEN / AUTOSIM variable references

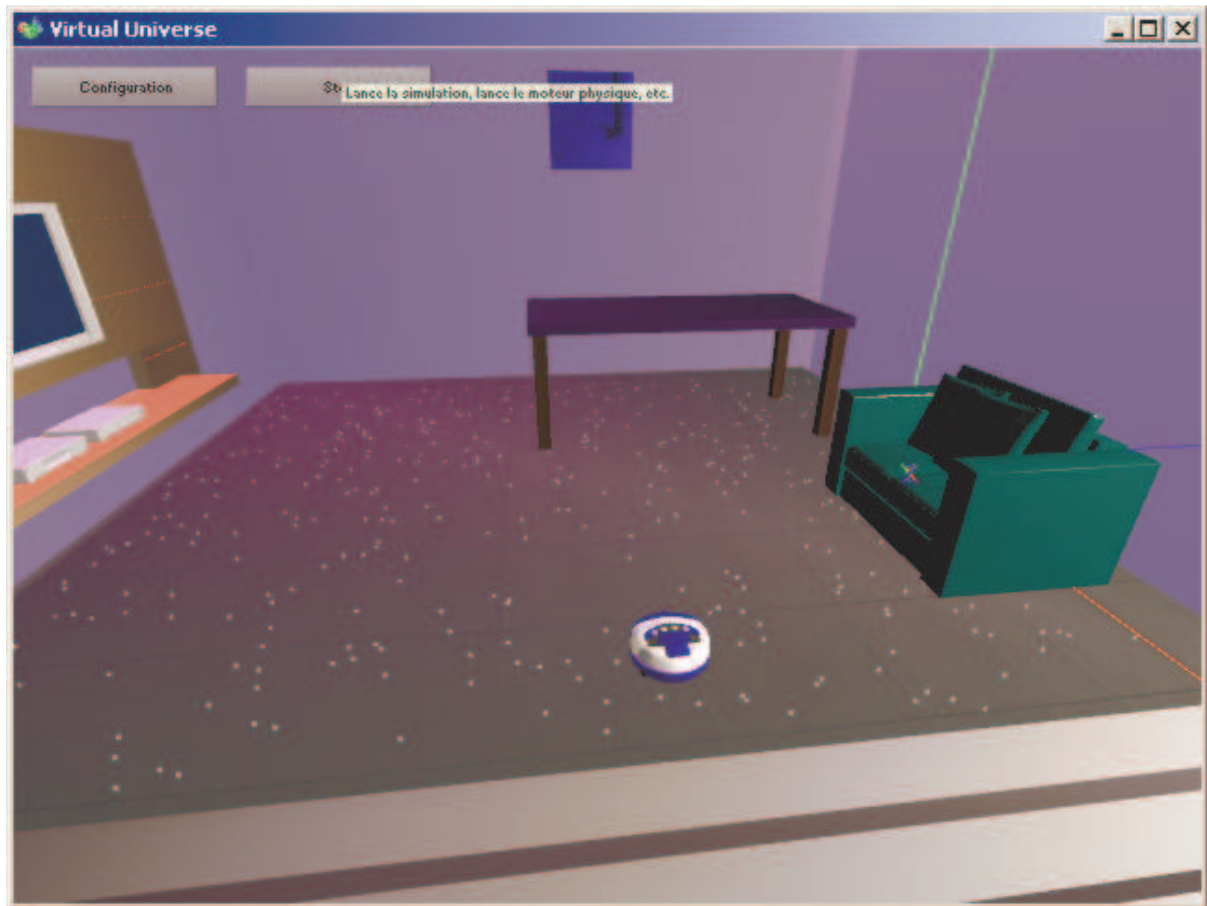
Symbol	Variable	Comments
xrobot1	%mf200	Destination X coordinates
yrobot1	%mf201	Destination Y coordinates
zrobot1	%mf202	Destination Z coordinates
arobot1	%mf203	Destination A angle (in degrees)
brobot1	%mf204	Destination B angle (in degrees)
crobot1	%mf205	Destination C angle (in degrees)
deltarobot1	%mf206	Requested precision
cmdrobot1	%mw200	Command: going from 0 to 1 runs the movement
statrobot1	%mw201	Result: 1=movement performed; <0=error (-7=position impossible to reach)
close clamp	%q0	Closes the clamp

Input

Output

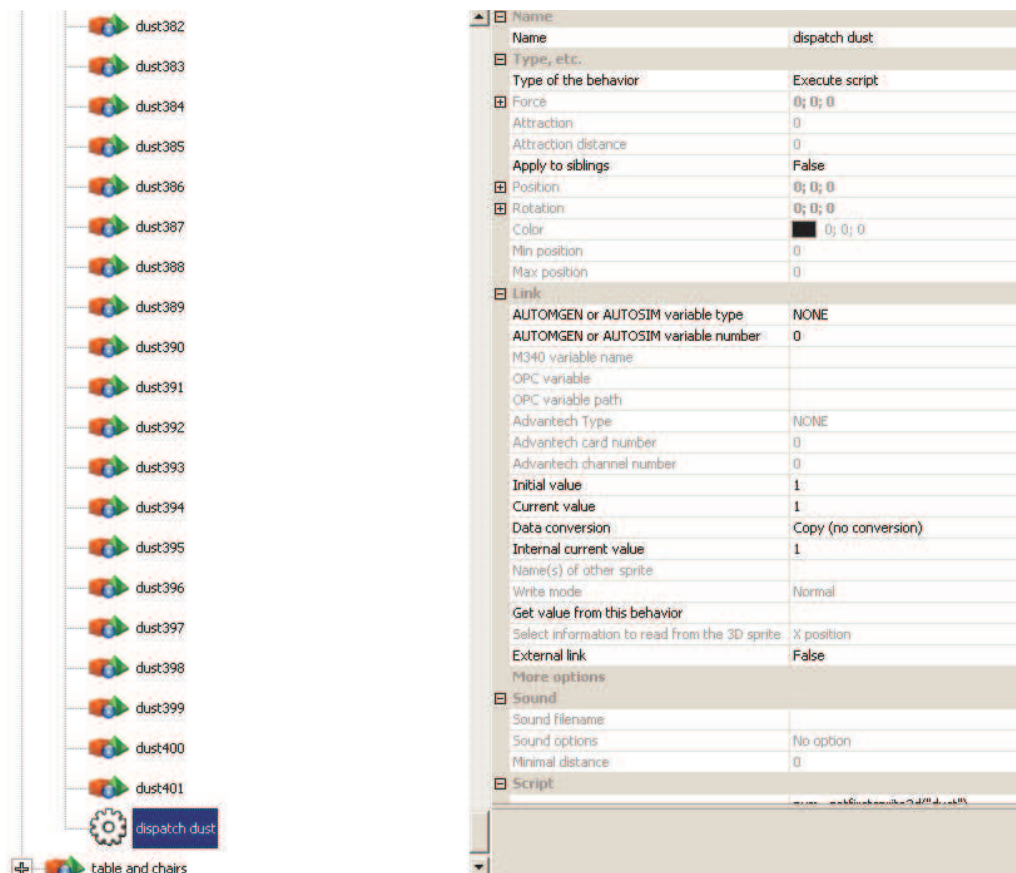
Vacuum robot

This project is located in the “samples\vacuum robot” sub-directory of the Virtual Universe installation directory. It is accompanied by an .AGN file.



This project illustrates the following functionalities in particular:

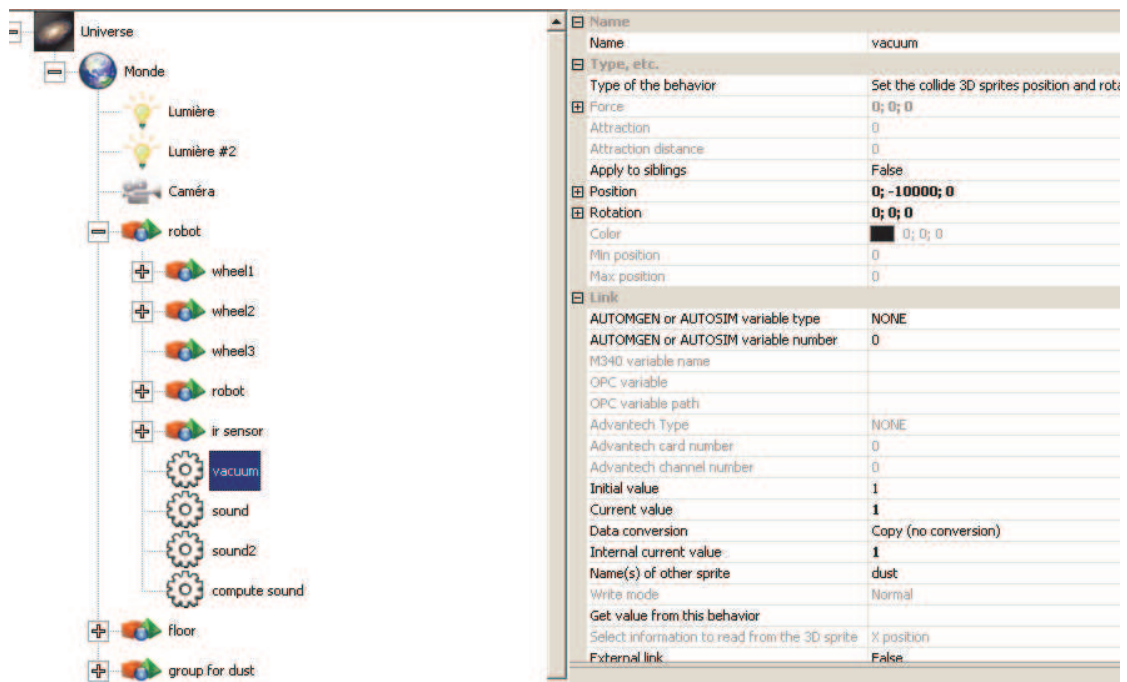
- Random positioning of objects;



Script

```
num=getfirstsprite3d("dust")
while num>=0
x=rand()*520-260
z=rand()*630-600
param="#" + asstring(num) + ".POSX"
setvalsprite3d(param,x)
param="#" + asstring(num) + ".POSZ"
setvalsprite3d(param,z)
num=getnextsprite3d(num,"dust")
wend
setbehavior("../dispatch dust".internalvalue,0)
```

- Simulation of a vacuum.



Operation

The robot is controlled by two motors, in turn with binary control by 2 outputs each (one output for each running direction). A contact sensor detects collisions with objects, a proximity sensor is used to obtain information on the absence of floor in front of the robot.

List of AUTOMGEN / AUTOSIM variable references

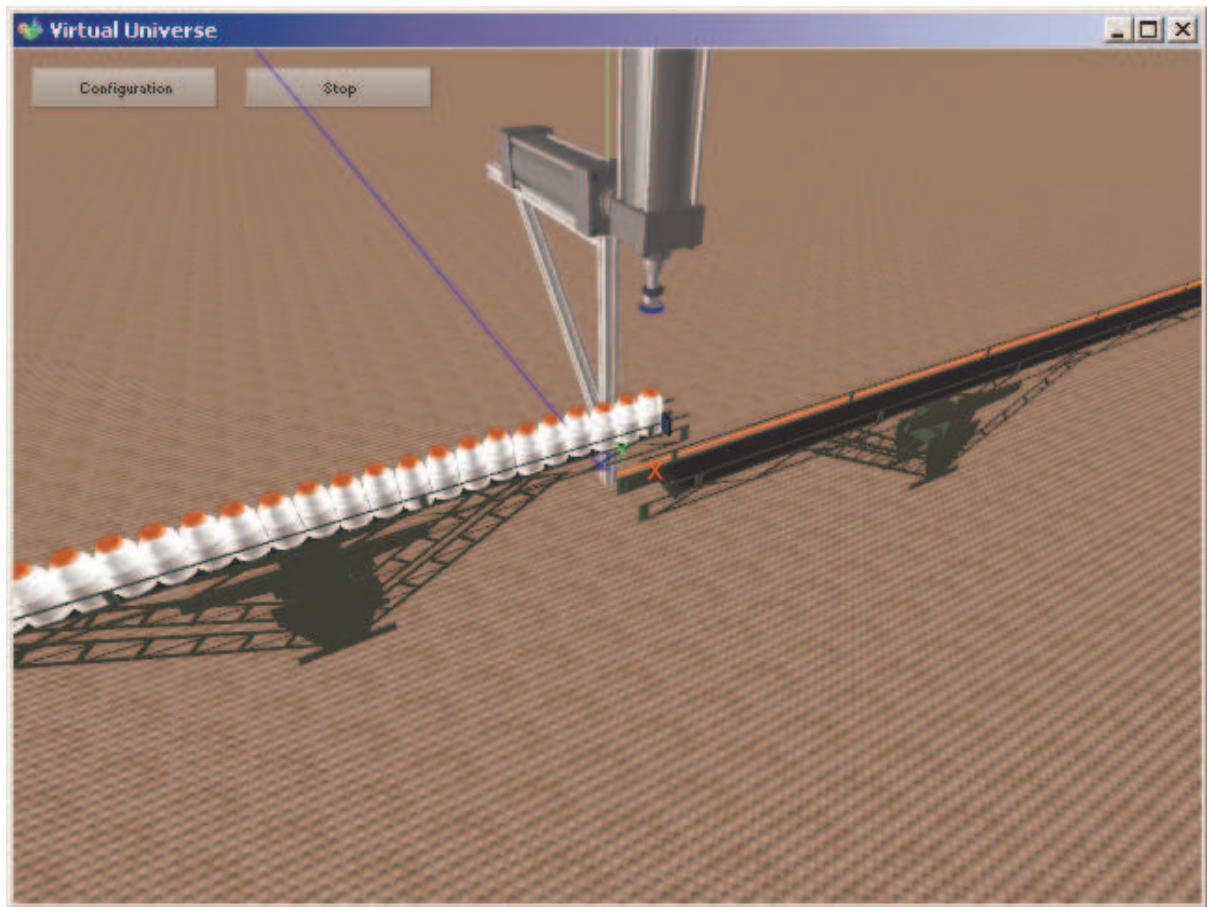
Symbol	Variable	Comments
forward motor 1	%q0	Motor 1 forward
backward motor 1	%q1	Motor 1 backward
forward motor 2	%q2	Motor 2 forward
backward motor 2	%q3	Motor 2 backward
collision	%i0	Collision sensor
ir sensor	%i1	No floor in front of robot sensor

Input

Output

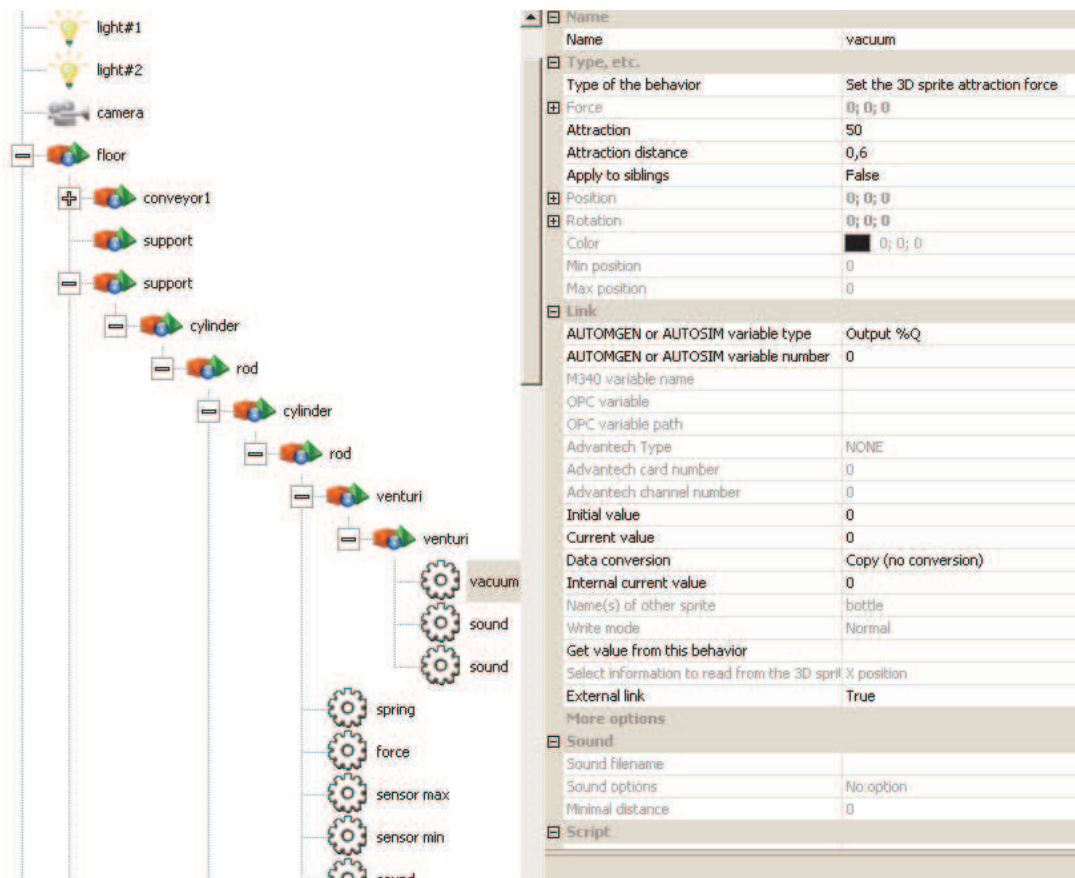
Manipulator with cylinders and suction cup

This project is located in the “samples\cylinders” sub-directory of the Virtual Universe installation directory. It is accompanied by an .AGN file.



This project illustrates the following functionalities in particular:

- Simulation of the suction cup;



- Position sensors for the cylinders;

The image displays a 3D simulation environment on the left and a properties panel for a 'sensor max' behavior on the right.

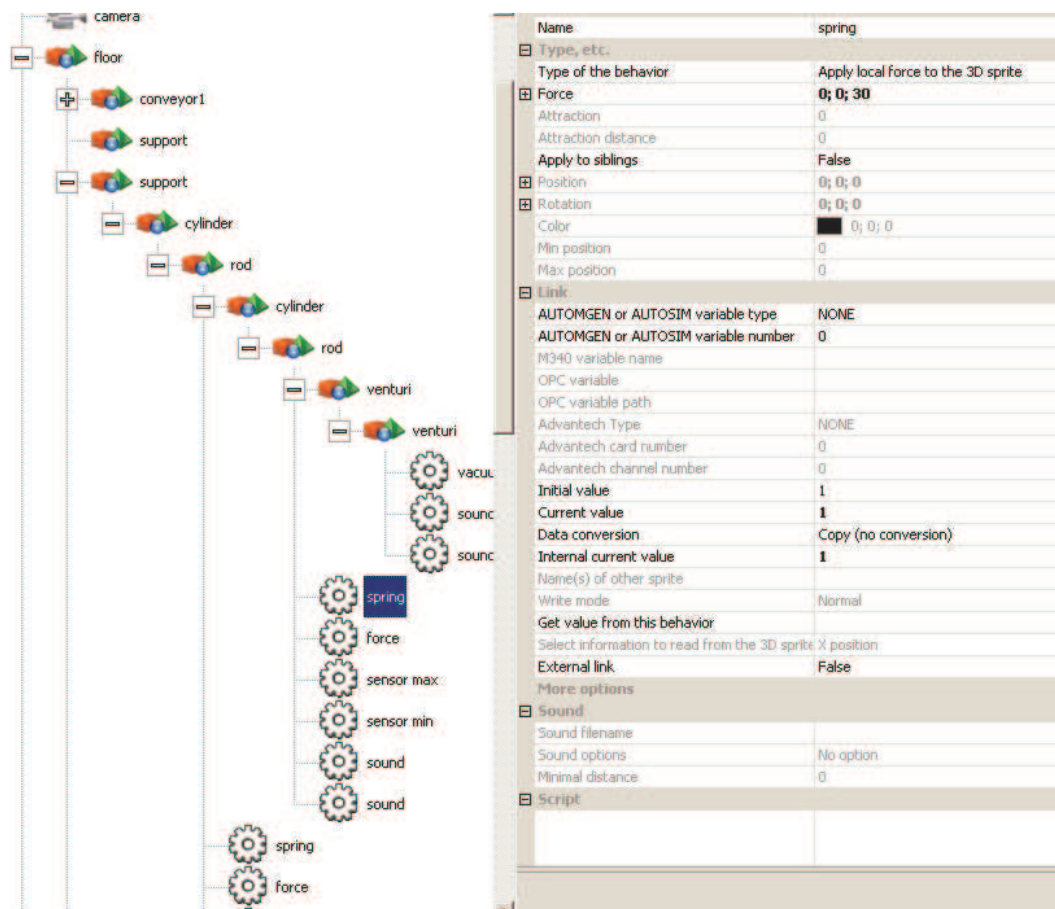
3D Environment:

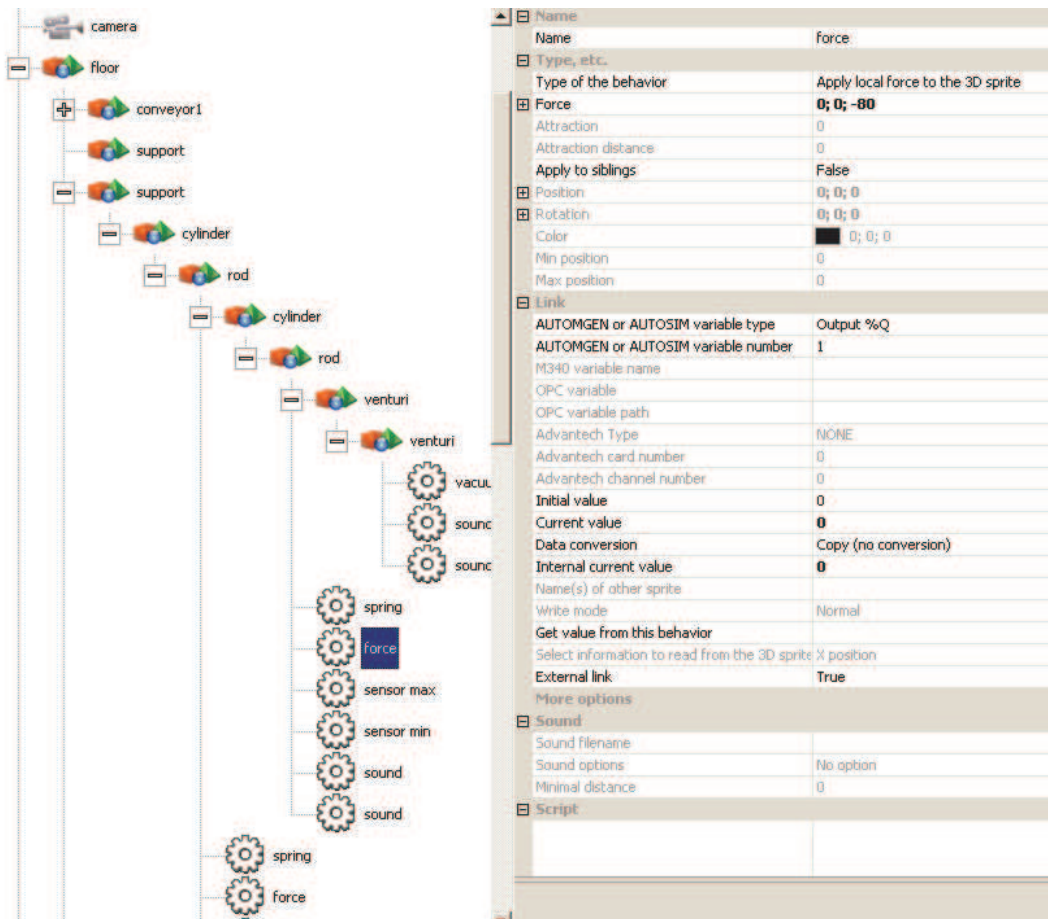
- camera**: A camera object at the top left.
- floor**: A flat ground plane.
- conveyor1**: A conveyor belt object.
- support**: Two support structures.
- cylinder**: A vertical cylinder.
- rod**: A horizontal rod.
- venturi**: Two venturi flow objects.
- vacul**: A vacuum object.
- sounc**: Three sound objects.
- spring**: A spring object.
- force**: A force object.
- sensor max**: A sensor object, highlighted in blue.
- sensor min**: A sensor object.
- sound**: Two additional sound objects.

Properties Panel (sensor max):

Name	
Name	sensor max
Type, etc.	
Type of the behavior	Test joint position
Force	0; 0; 0
Attraction	0
Attraction distance	0
Apply to siblings	False
Position	0; 0; 0
Rotation	0; 0; 0
Color	0; 0; 0
Min position	-1,6
Max position	-1,4
Link	
AUTOMGEN or AUTOSIM variable type	Input %I
AUTOMGEN or AUTOSIM variable number	1
M340 variable name	
OPC variable	
OPC variable path	
Advantech Type	NONE
Advantech card number	0
Advantech channel number	0
Initial value	0
Current value	0
Data conversion	Copy (no conversion)
Internal current value	0
Name(s) of other sprite	
Write mode	Safe
Get value from this behavior	
Select information to read from the 3D sprite	X position
External link	True
More options	
Sound	
Sound filename	
Sound options	No option
Minimal distance	0
Script	

- Simulation of the cylinder with rod return via spring;





Operation

The arm is composed by two single acting cylinders. A suction cup is used to grip the bottles. Two conveyors manage the arrival and departure of the bottles.

List of AUTOMGEN / AUTOSIM variable references

Symbol	Variable	Comments
vacuum	%q0	Starts the vacuum
go down	%q1	Extracts the vertical cylinder
go out	%q2	Extracts the horizontal cylinder
top	%i0	Vertical cylinder retracted
bottom	%i1	Vertical cylinder extracted
in	%i2	Horizontal cylinder retracted
out	%i3	Horizontal cylinder extracted

Input

Output