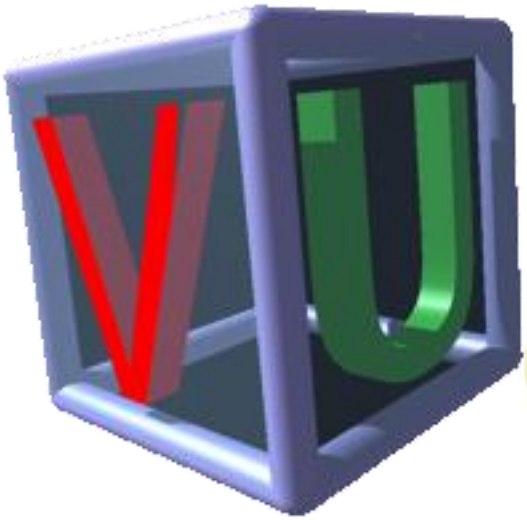


Virtual Universe Pro V2

User manual

Table of Contents

Introduction

VIRTUAL UNIVERSE PRO (Industry package) is an innovative 3D modeling and emulation software enabling to quickly create an interactive 3D emulation of automated systems (or « virtual machines »), by reusing and leveraging 3D CAD models. With the VIRTUAL UNIVERSE PRO emulators, designers of industrial equipments and automated systems can experiment with their products in a realistic, interactive virtual 3D environment and emulate equipment behavior in real time. By connecting the 3D emulators with external controllers such as PLCs (Programmable Logic Controllers) or with embedded virtual controllers, VIRTUAL UNIVERSE PRO reproduces the real world working conditions of a product or machine, in a fully virtual environment.

Installation

Hardware requirements

VIRTUAL UNIVERSE PRO runs on the following Windows operating systems: Windows XP, Windows Vista, and Windows 7. For Windows XP, it is necessary to use DirectX 9 (or a newer version of Direct X).

VIRTUAL UNIVERSE PRO is a native Windows 32-bit software and is Windows 64 bit compatible.

For taking full advantage of its performance (especillay during simulation), it is strongly recommended to use VIRTUAL UNIVERSE PRO on a modern computer, with a good graphics card and a powerful processor:

- The refresh rate of 3D rendering is directly linked to the graphics card performance.
- The accuracy of the simulation (realism of the simulated phenomena) mostly depends on the computer CPU performance.

Software installation

For installing VIRTUAL UNIVERSE PRO on a computer, simply launch the execution of the installation package that was shipped on CD-ROM or downloaded from Internet. Visit our website www.irai.com to download the latest versions of VIRTUAL UNIVERSE PRO.

If an older version of VIRTUAL UNIVERSE PRO is already present on your computer, the installation package will install the latest version of VIRTUAL UNIVERSE PRO, without altering the old version. Removing the old version is made with the "Add or Remove Programs" tool in the Windows Control Panel. If new 3D resources were added in the VIRTUAL UNIVERSE PRO Resource Library, they will be retained.

By default, VIRTUAL UNIVERSE PRO operates the Newton physics simulation engine, which is included in the installation package.

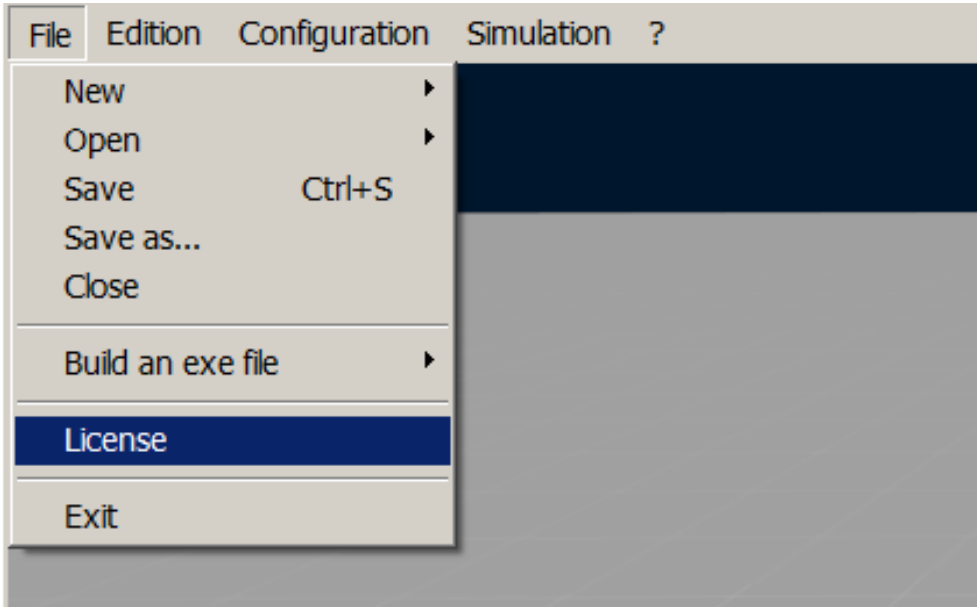
VIRTUAL UNIVERSE PRO is also capable of running the simulation engine Physx from NVIDIA. For that, it is necessary to install the latest version of NVIDIA Physx engine. It can be downloaded from NVIDIA's website.

Licenses registration

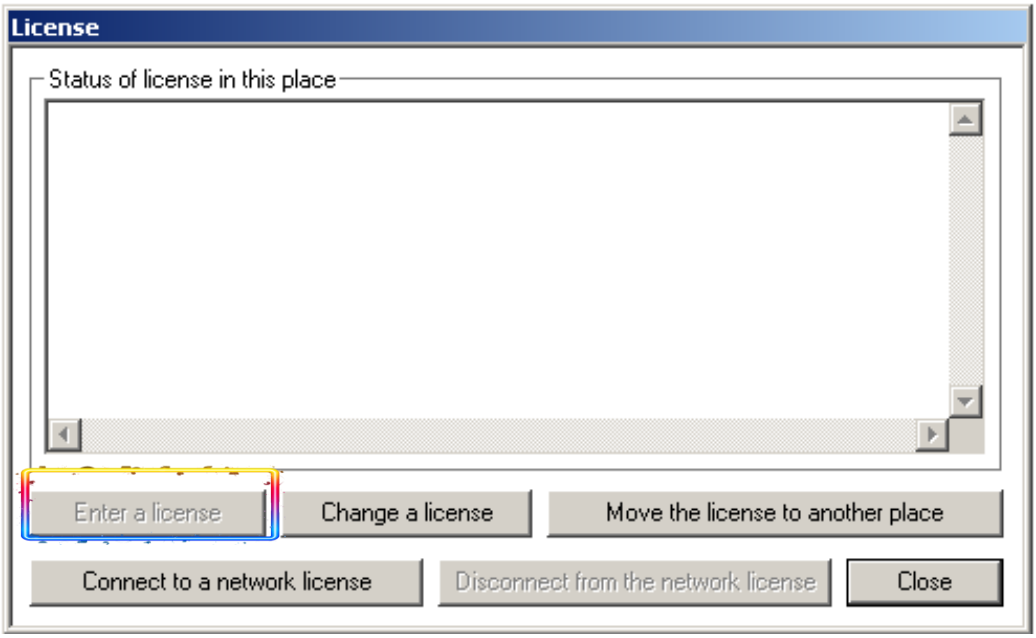
Register a license

VIRTUAL UNIVERSE PRO works in a trial version (limited to 15 days trial) as long as you have not registered the license.

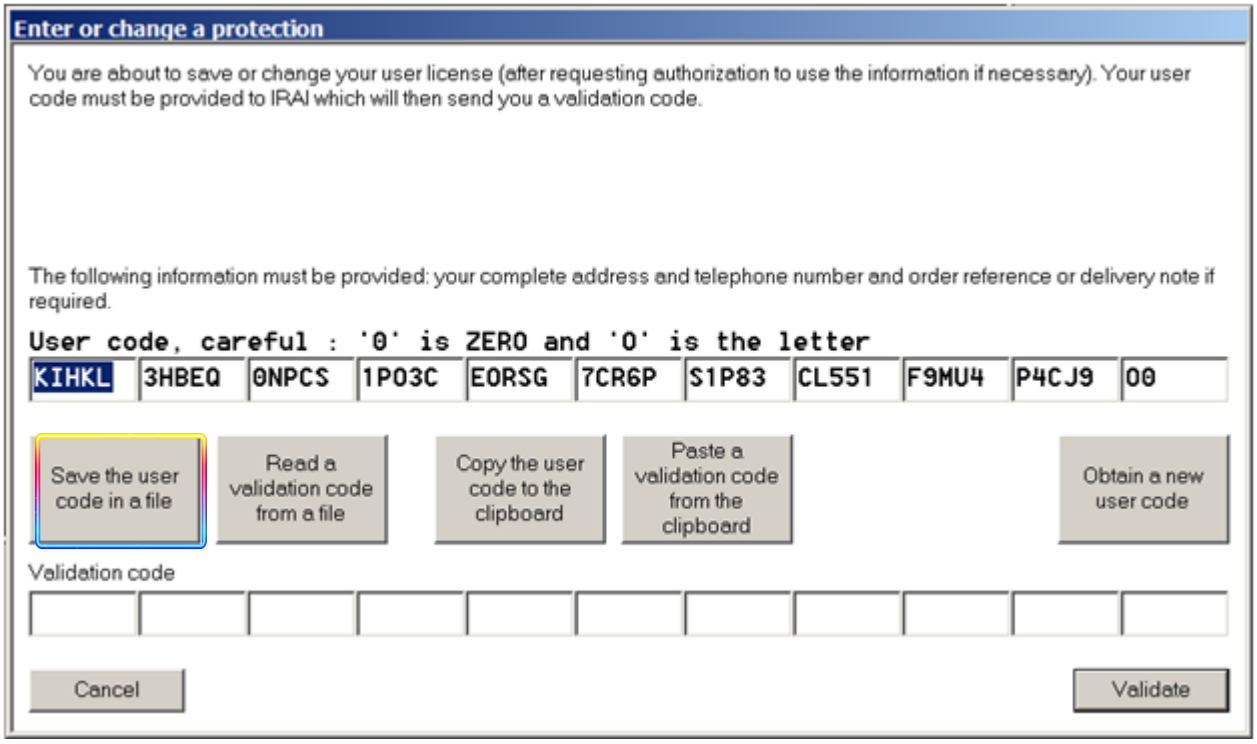
To register, select "License" in the "File" menu:



A license window is opened. Click on « Enter a license »:



Click on "Save the user code in a file" and save this file on your computer :



Then, send this file by email to the following address: contact@irai.com

You will receive by email a new file containing a validation code to register by clicking the button "Read a validation code from a file". You can also directly enter the validation code in the "Validation Code" field. Then you will click "Validate" to validate the license.

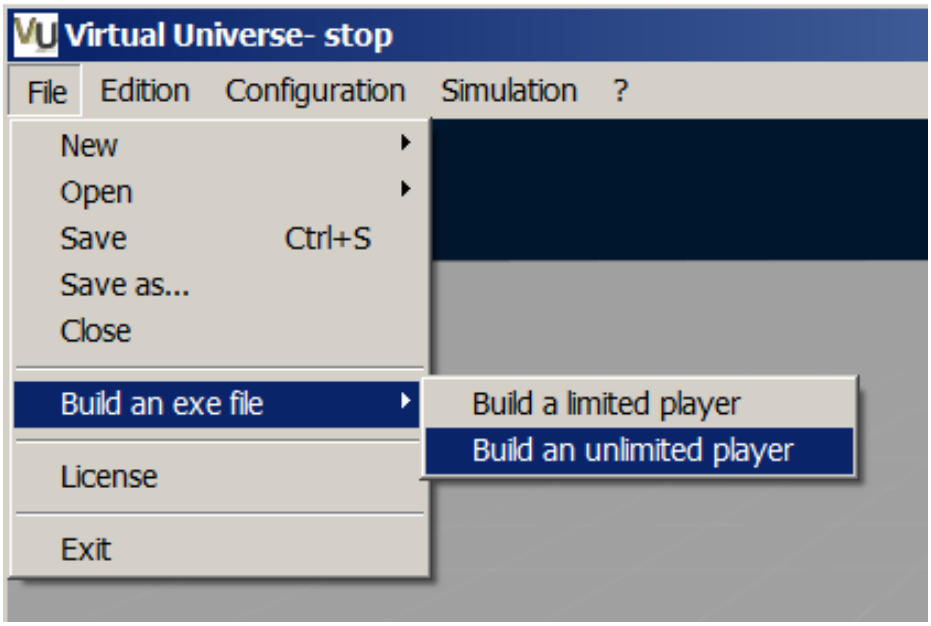
You have 20 days between the generation of a user code and the input of the validation code.

Register a security code (unlimited player)

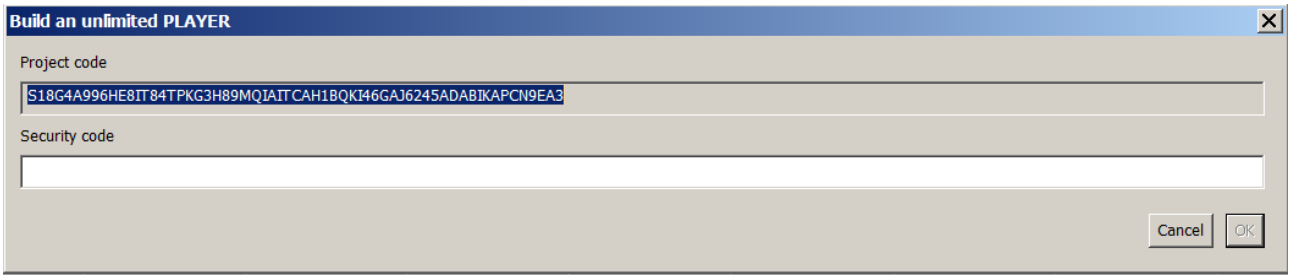
VIRTUAL UNIVERSE PRO enables the generation of standalone 3D emulators (executable files named « players ») having an unlimited period of use.

The player generation functionality is protected by a security code. Each player generated has its own security code, which is unique.

For registering a security code, select Build an exe file/Build an unlimited player in the File menu:



Once filled the player name, a window is opened, displaying a project code:



Copy and paste this project code in an email and send your email to the following address: contact@irai.com

You will receive an unique security code, to paste in the « Security code » field, allowing you to generate an unlimited player. When receiving the security code by email, you have 30 days for using this code to generate the player. Once generated the player; the generation functionality will be blocked again and a new security code will be needed.

It is highly recommended to use the limited players to make your tests (test the communication with an external controller...), before generating the unlimited player, considered as a definitive player.

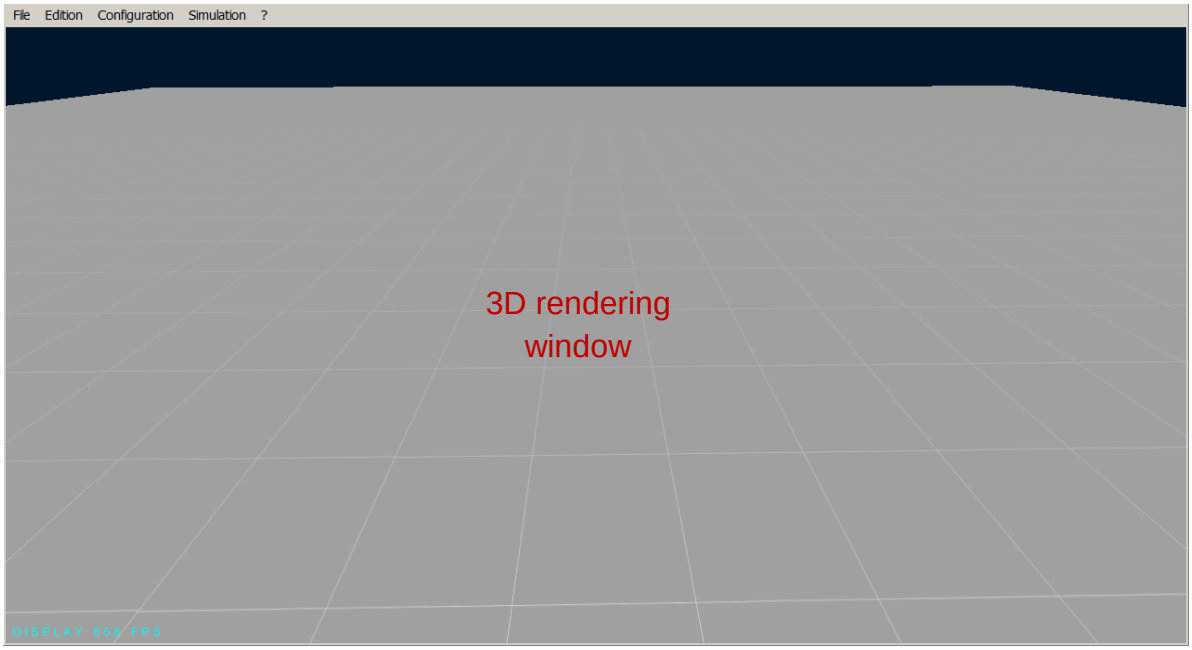
Network installation

VIRTUAL UNIVERSE PRO can be installed on a file server, and licenses can be managed by a network license manager. Please contact our technical support to complete this installation: contact@irai.com

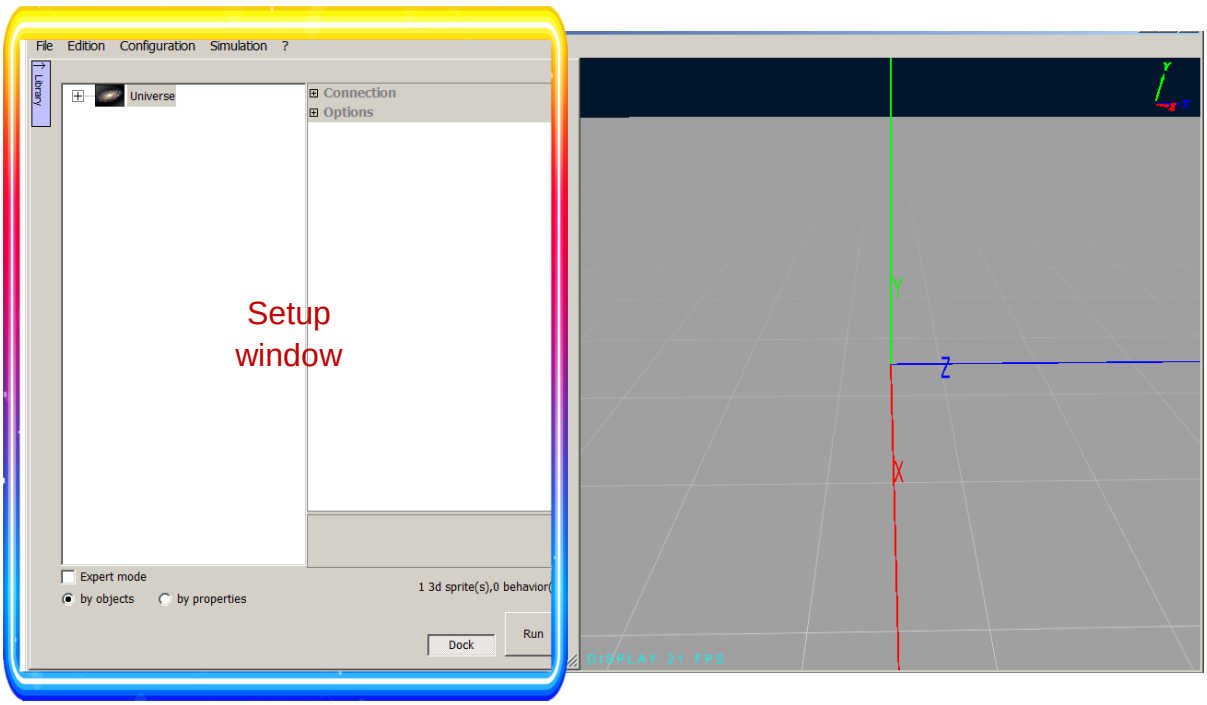
Overview

Menus and Windows

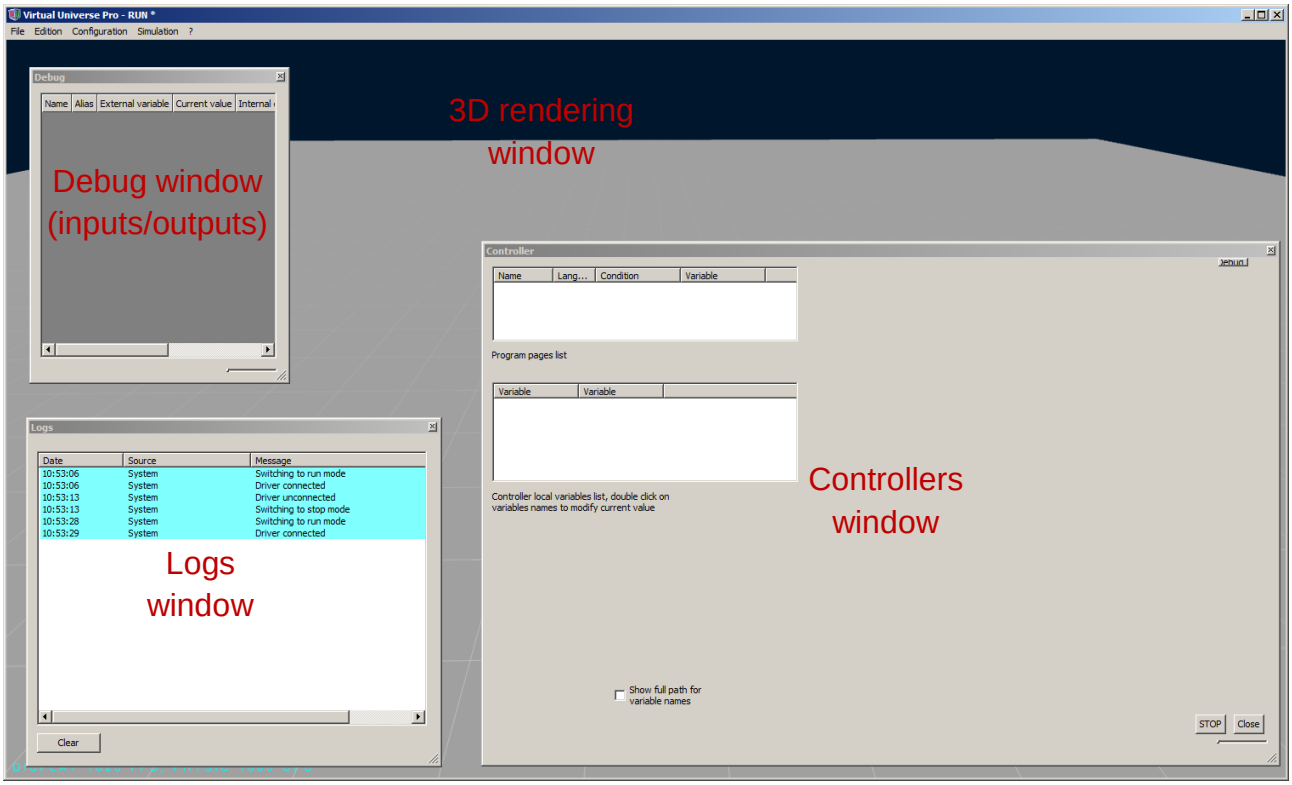
When launching VIRTUAL UNIVERSE PRO, an empty project is opened in the main 3D rendering window.



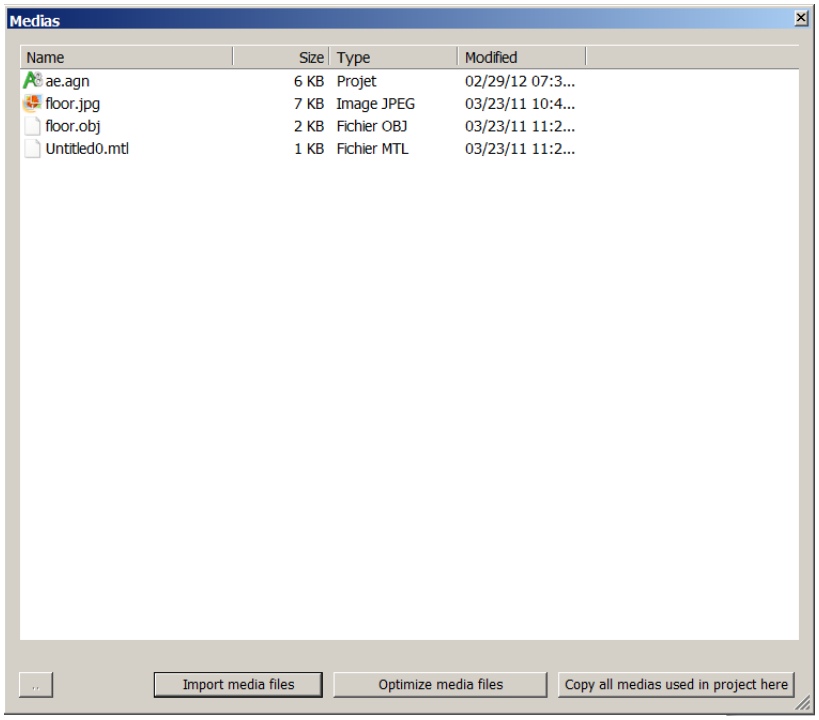
- The **Configuration** menu provides access to the Setup window and to the project construction tools.



- The **Simulation** menu provides access to the launch of the simulation and to the simulation debug tools.



- The **Medias** menu provides access to Media Manager containing the list of all the media files (3D files, bitmaps) used in the 3D emulator project. All these files are automatically saved in the 3D emulator project file.



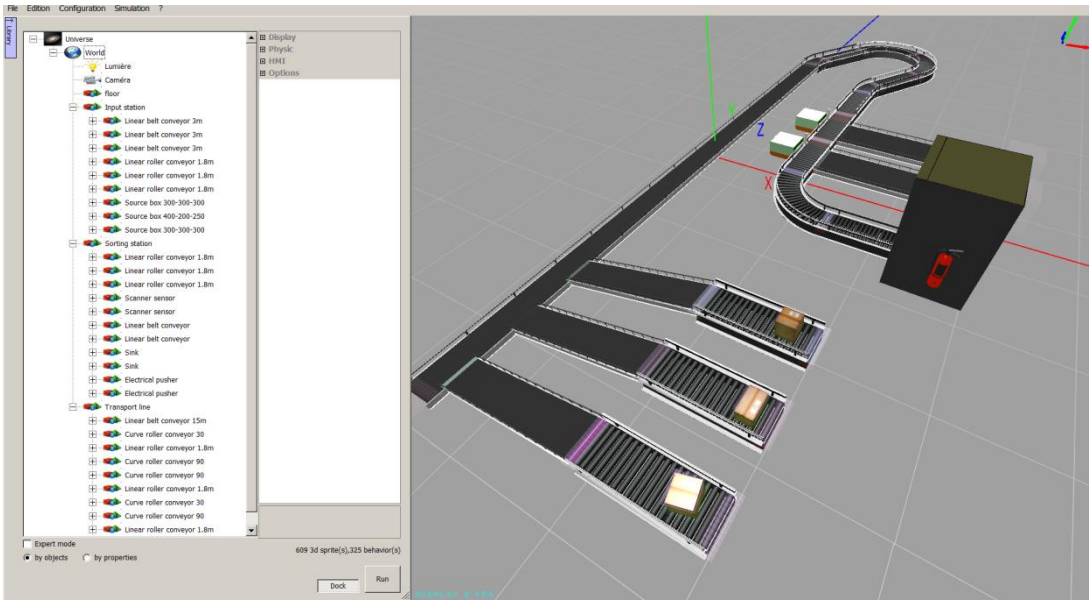
- The **?** menu provides access to the user documentation and software information.

- The **File** menu lets you create a new project, open an existing project, save and close a project. This menu also provides access to the generation of standalone 3D emulators (executable files called “players”) and to the licenses registration.

- The **Edition** menu allows you to undo or redo changes made in the current project.

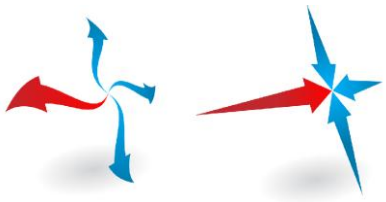
Example

For getting familiar with the various menus and windows of VIRTUAL UNIVERSE PRO, a simulation project example is available in the menu File/Open/Open a sample:



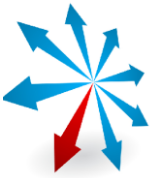
Navigation and interactions

By default, navigation in the 3D world and interactions with 3D objects during simulation are done with the mouse.



Mouse wheel = zoom

Mouse wheel pressed + moving mouse or rotating wheel = translation

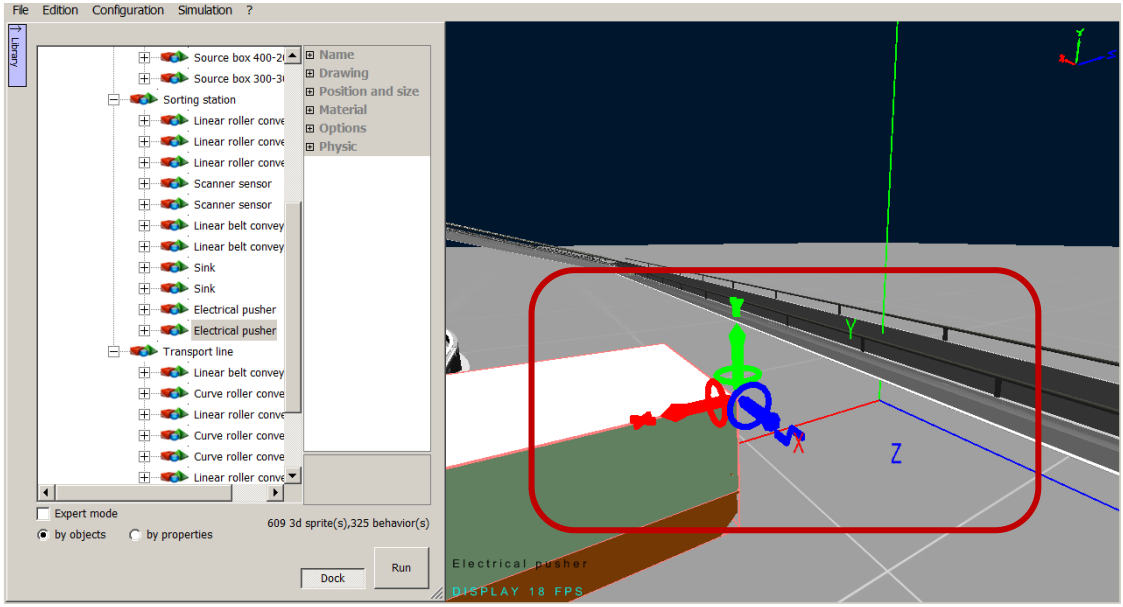


Left mouse button click = actions

Right mouse button pressed + moving mouse = rotation

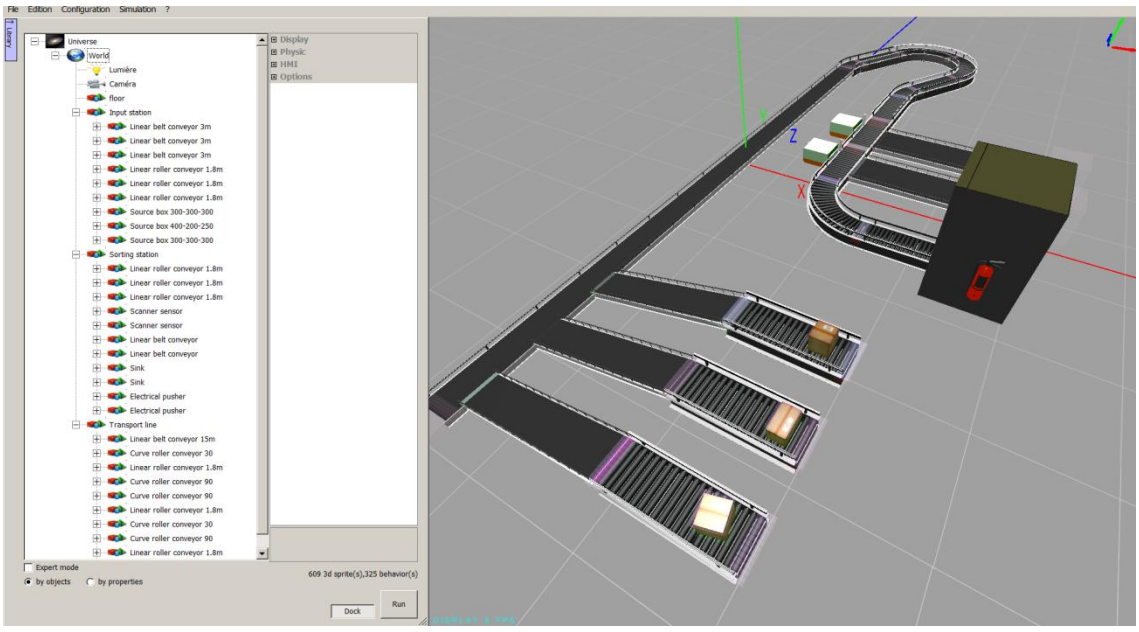
Left mouse button click and drag = grab object

When simulation is off and the setup window is opened, an (X, Y, Z) axis system is displayed at the center of the world in the 3D rendering window (global axis system), as well as on the 3D sprite selected with the mouse or in the tree (local axis system).



Example

For getting familiar with navigation and interactions in VIRTUAL UNIVERSE PRO, a simulation project example is available in the menu “File/Open/Open a sample”:



Main steps for building a 3D emulator

VIRTUAL UNIVERSE PRO enables to build interactive automated system 3D emulators (or “virtual machines”), composed of a 3D virtual system controlled by one or more controllers (internal or external to the 3D emulator).

- The 3D models are imported from CAD software into VIRTUAL UNIVERSE PRO to create the 3D resources (composed of 3D objects called "sprites") and assemblies of 3D resources which will constitute the elements of the 3D virtual machine.

- Adding behaviors to the sprites brings a real intelligence to 3D resources and enables to model the behavior of resources, such as actuators and sensors of the virtual system. Behaviors are either predefined behaviors available in VIRTUAL UNIVERSE PRO or custom behaviors (scripts) created with a script editor integrated in VIRTUAL UNIVERSE PRO.

- All the resources created in VIRTUAL UNIVERSE PRO (smart 3D resources, or simple behaviors without 3D representation) can be saved and stored inside a VIRTUAL UNIVERSE PRO resource library, and can be reused for quickly building future 3D emulator projects.

- It is possible to model one or more virtual controllers inside the 3D emulator (motion controller, control sequence) and to define a 2D control panel for modeling an human-machine interface.

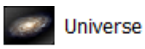
- The virtual system in VIRTUAL UNIVERSE PRO can be connected and simulated with an external controller (such as a Programmable Logic Controller).

- At any time during its construction, the performance of the 3D emulator (3D rendering quality, physics engine performance) can be measured and, if necessary, improved.

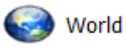
- Finally, it is possible to generate standalone 3D emulators (executable files called "players") limited to simulation and requiring no software installation.

Composition of a 3D emulator

At any time, the **Setup** window provides access to the setup and construction tools of a 3D emulator project. The 3D emulator project elements are structured and presented in a tree:

- The **Universe**  level provides access to the general use properties of the 3D emulator :
  - Using the 3D emulator with or without connection to an external software/controller
  - Automatic start of the simulation when opening the 3D emulator
  - Navigation mode in the 3D rendering
  - Setting rights to access the 3D emulator properties
  - …etc.

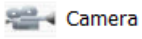
  A default setting of these properties is proposed. These properties can be modified by the user. For more details on the properties of the Universe, refers to Properties of the Universe.

- The **World**  level provides access to the general display properties of the 3D emulator :
  - Setting of the background color
  - Setting of the ambient light
  - Adding an image to sky
  - Setting of units
  - ... etc..

  A default setting of these properties is proposed. These properties can be modified by the user. For more details on the properties of the World, see Properties of the World.

The **Light**  level provides access to the lighting properties of the simulator.

By default, the 3D emulator already includes a preset light. You can define multiple lights in the 3D emulator. For more details on the properties of lights, see Properties of Lights.

- The **Camera**  level provides access the visualization options of the 3D emulator.

  By default, the 3D emulator already includes one camera. It is possible to define multiple cameras in the 3D emulator to create different views. For more details on the properties of cameras, see Properties of Cameras.

- The **Sprite**  level provides access to the properties of a Sprite.

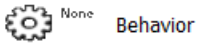
  Sprites are the 3D objects composing a virtual machine project. A Sprite is most frequently associated with an image file representing a 3D shape with its own size and location in the 3D world. A sprite can also have no associated image and it may be only used to structure the 3D data (assemblies). Sprites are structured and presented in a parent / child tree.

It is possible to manually add, copy, paste, move, delete sprites to build assemblies and 3D resources.

Importing 3D models from CAD software into VIRTUAL UNIVERSE PRO enables to create automatically 3D resources and assemblies of 3D resources, all consisting of a structured set of sprites.

These assemblies of sprites and 3D resources created in VIRTUAL UNIVERSE PRO can be saved and stored inside the VIRTUAL UNIVERSE PRO resource library and can be reused for quickly building future 3D emulator projects.

For more details on the properties of sprites, see Properties of Sprites.

- The **Behavior** 

level provides access to properties of a behavior.

Behaviors are at the heart of simulation in VIRTUAL UNIVERSE PRO. They represent the intelligence provided to sprites during simulation. Behaviors can turn inert 3D Sprites into smart resources, with the capability to move, to interact and to communicate with the other 3D resources during simulation.

There are different types of predefined behaviors available in VIRTUAL UNIVERSE PRO depending on the resource type (actuator, sensor, and controller) to model. Behavior can also represent inputs/outputs for sprites, allowing them to interact with other sprites and with any external controller.

In addition, a script editor can be used to describe (in Basic language) more developed and custom behaviors based on predefined behaviors to model a real behavioral logic for the resource.

Behaviors, like sprites, can be saved and capitalized inside the VIRTUAL UNIVERSE PRO integrated resource library to be reused for quickly building future 3D emulators projects.

It is possible to manually add, copy, paste, move, and delete behaviors.

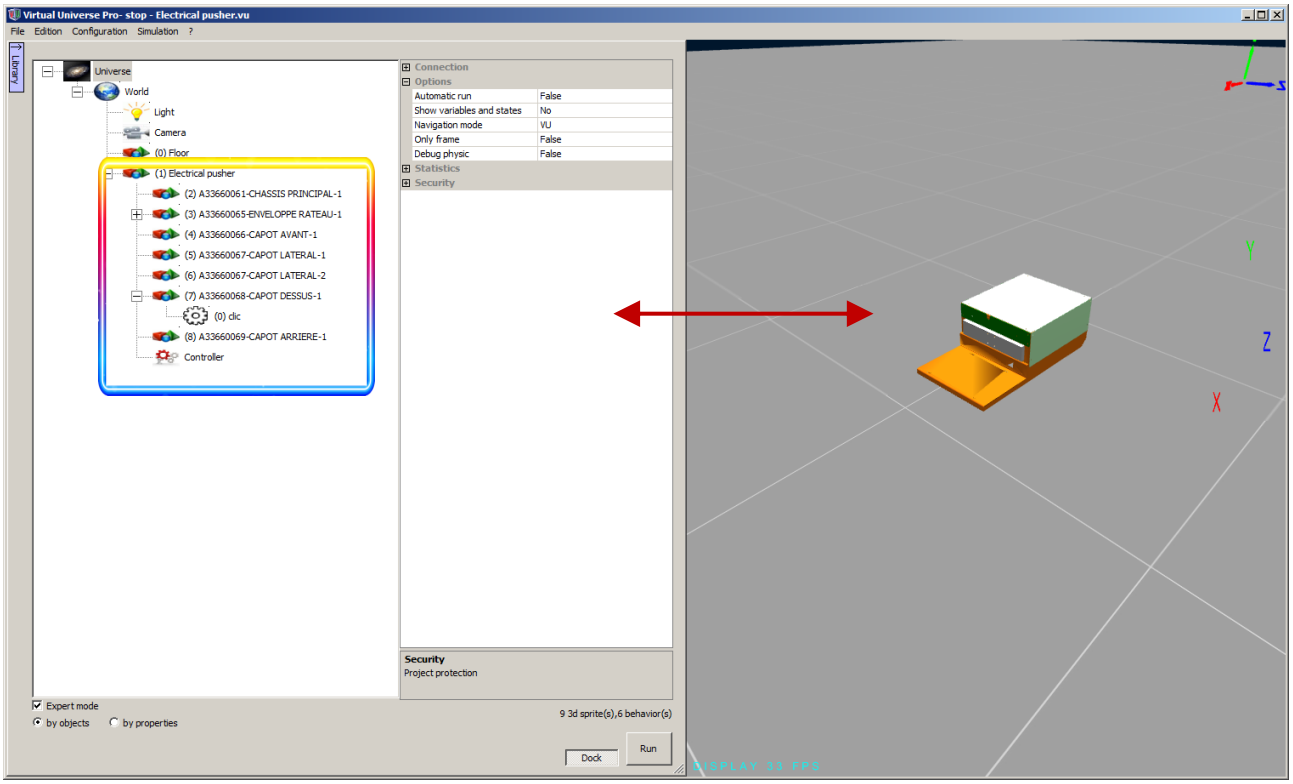
For more details on the properties of behaviors, see Properties of Behaviors.

The **HMI** level  HMI provides access to the properties of a HMI. HMIs allow to create consoles which will be displayed into the render window and will use items like push-buttons, lights, sliders, etc. For more details on the properties of HMIs, see Properties of HMIs.

The **Controller** level  controller provides access to the properties of a controller. Controllers may be children of the world or of a sprite. They allow to create programs to control a full system or a part of a system. For more details on the properties of controllers, see Properties of Controllers.

Example

For understanding the concept of sprites and behaviors in VIRTUAL UNIVERSE PRO, we suggest you to look at the Electrical Pusher resource used in the conveyor project example available in the menu File/Open/Open a sample:

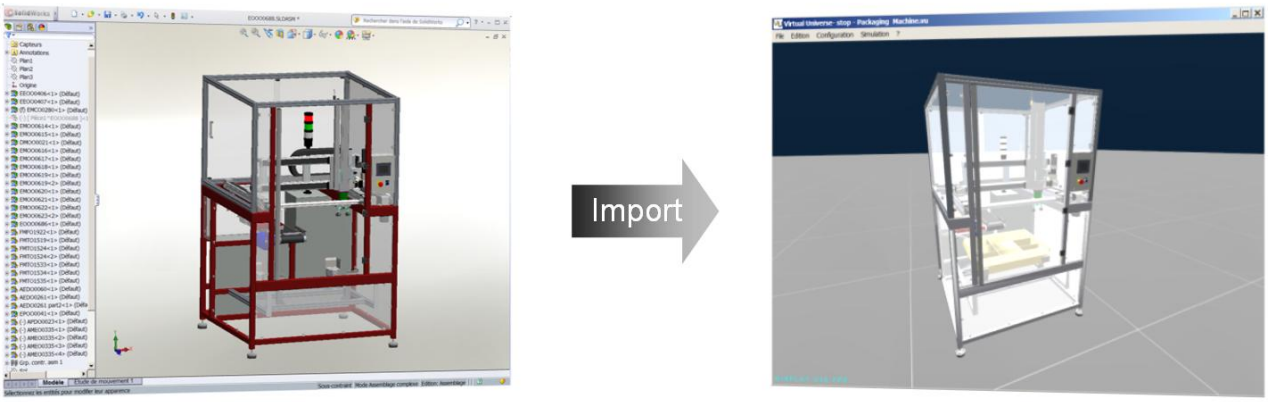


Types of imported 3D CAD data

VIRTUAL UNIVERSE PRO allows the reuse of 3D CAD (Computer Aided Design) models by a direct import of these data from their native CAD software, or indirectly through an exchange file.

CAD Software

VIRTUAL UNIVERSE



Here are the types of 3D CAD data that can be imported and reused today in VIRTUAL UNIVERSE PRO:

CAD Software	Exchange file (indirect import)
DS SolidWorks	3DXml
Autodesk Inventor	
DS Catia	3DXml
DS Delmia	3DXml

List of available PLC connectors

The virtual systems created in VIRTUAL UNIVERSE PRO can be connected to PLCs (Programmable Logic Controller), in order to create realistic virtual automated systems.

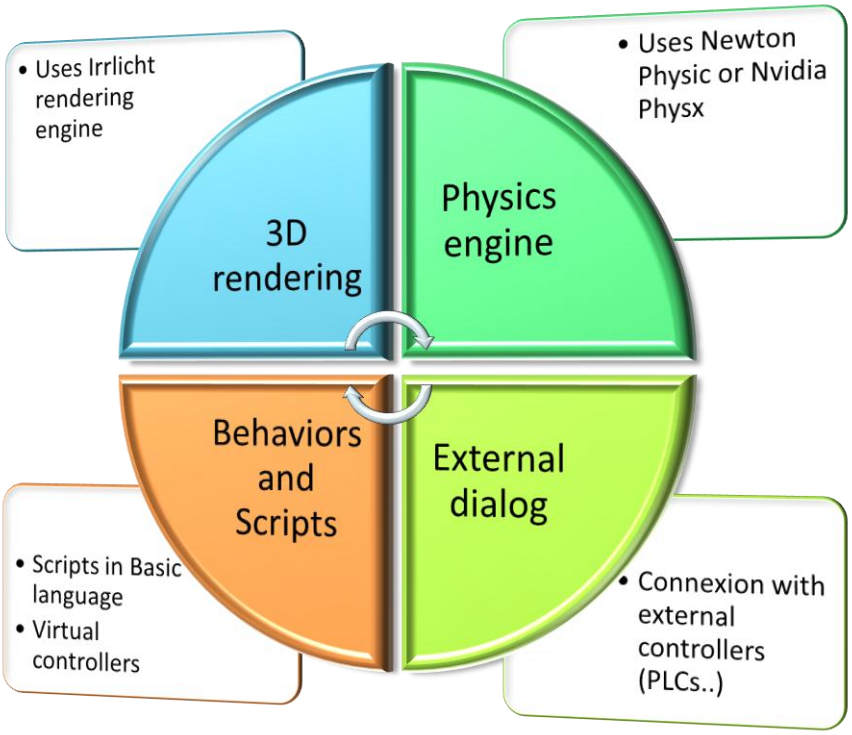
During simulation, the virtual system in VIRTUAL UNIVERSE PRO is controlled in real time by the PLC thanks to a permanent dialog established between the Inputs/Outputs of the PLC program and those of the virtual system.

VIRTUAL UNIVERSE PRO is compliant with some PLC brands, and it therefore offers several types of PLC connectors (communication protocols). These are listed below:

Connector	PLC brand	PLC types
OPC Client	Any PLC or control system compliant with OPC	Any PLC or control system compliant with OPC
Ethernet S7	Siemens	S7-300, S7-400, S7-1200
PLC-SIM	Siemens	PLC-SIM Emulator
Ethernet-IP	Rockwell, Allen Bradley	Compact Logix, Control Logix or Softlogix Emulator
Modicon 340	Schneider Electric	Modicon M340 or Unity software emulator
Automgen	Automgen PLC compatibles (see on www.irai.com)	All PLCs or other targets compayible with Automgen
Advantech I/O	Any system	Any system connected to one or more Advantech card
Omron Sim	Omron	CX-Simulator Emulator

Internal working of VIRTUAL UNIVERSE PRO

It may be helpful to understand which are the internal mechanisms and technologies used in VIRTUAL UNIVERSE PRO to enable simulation.



- For managing the display of the 2D and 3D sprites in the 3D emulator (during and off simulation), VIRTUAL UNIVERSE PRO operates the 3D rendering engine called "Irrlicht". This engine is run in parallel with the other engines, and does not affect the smooth operation of the simulation. By default, this engine is running at full capacity, in order to maximize graphics performance of the 3D emulator (refresh rate of 3D rendering). However, it is possible to set a maximum value for the refresh rate, in order to avoid consuming "unnecessarily" the computer resources. At any time, it is possible to know the current refresh rate of 3D rendering of the 3D emulator, by reading the FPS (Frame Per Second) metric, visible at the lower left of the 3D rendering window.

- For the modeling and simulation of physical phenomena applied to sprites (gravity, friction, collisions, forces...), two physics simulation engines can be used in VIRTUAL UNIVERSE PRO: Newton Physic (default engine) and NVIDIA Physx. The use of the physics engine is not systematic neither mandatory. Only the sprites with the property "Use physics" checked are exposed to the physics engine. By default, the physics engine is running at full capacity in VIRTUAL UNIVERSE PRO (in a separate thread with a variable sampling rate) to get the best realism of the simulated physical phenomena. At any time, it is possible to know the current computation speed of the physics engine, by reading the CPS (Computation Per Second) metric, visible at the lower left of the 3D rendering window.

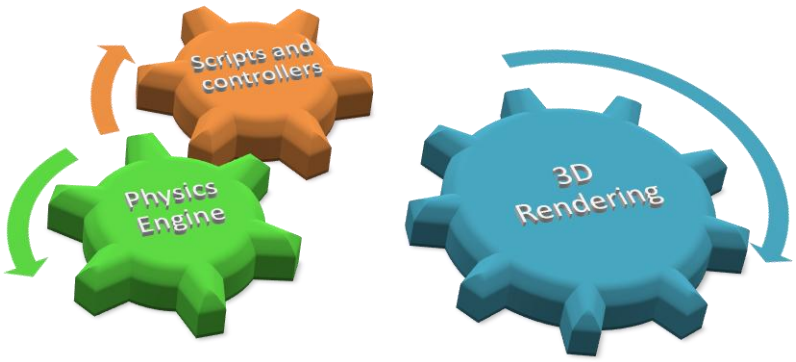
- For enabling the VIRTUAL UNIVERSE PRO 3D emulators to communicate in real time with an external controller (such as a Programmable Logic Controller), a permanent communication is also set up during simulation with the external controller (exchange of input/output variables). Several industrial communication protocols are available in VIRTUAL UNIVERSE PRO to connect and to communicate with different brands and types of industrial controllers (Siemens, Rockwell, Schneider...). By default, the communication speed between VIRTUAL UNIVERSE PRO and the external controller is maximum. However, it is possible to set a minimum value for the variables exchange period, in order to avoid consuming "unnecessarily" the computer resources.

- The simulation intelligence given to the 3D objects (sprites) is defined by a set of predefined behaviors and custom behaviors (scripts written in Basic or virtual controllers using ladder or SFC/FDB). All these behaviors, scripts and virtual controllers are executed in parallel (in a separate thread).

RUN/STOP simulation modes

In VIRTUAL UNIVERSE PRO, simulation can be in STOP mode (simulation stopped and initialized) or in RUN mode (simulation running).

In RUN mode, the 3D rendering engine, the physics engine and the dialog with the external software are processed. Behaviors and scripts are active.



The 3D sprites and cameras have a duplicate for some parameters (eg positions). The first set of parameters corresponds to initial values, the second set to current values. In STOP mode, the initial values are recopied into the current values.

Construction of a 3D emulator

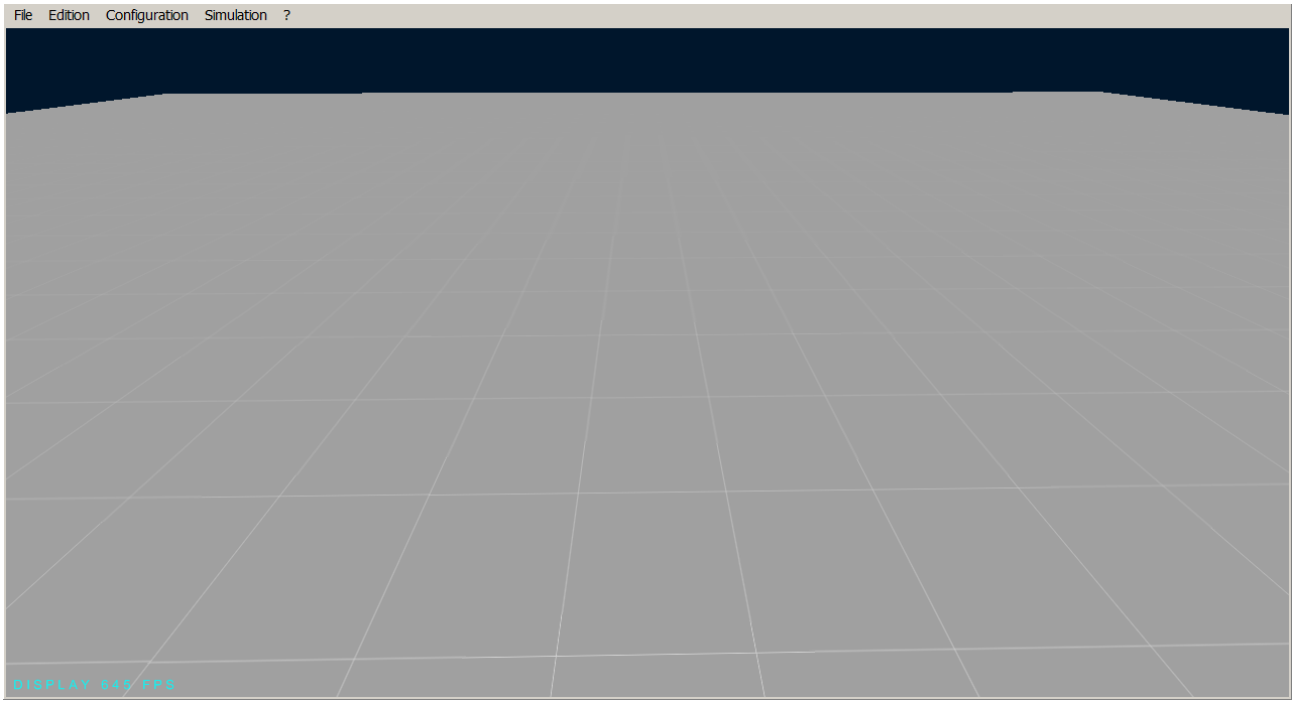
Set the general properties of a 3D emulator

Display properties

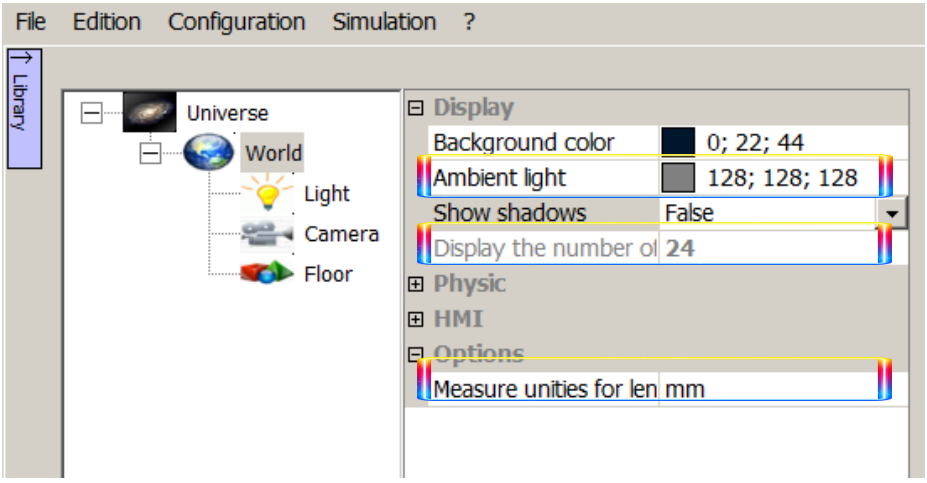
By default, VIRTUAL UNIVERSE PRO opens automatically to a preset pattern of world (in terms of display).

The units used by default in this world are:

- **millimeter** (length)
- **degree** (angle)

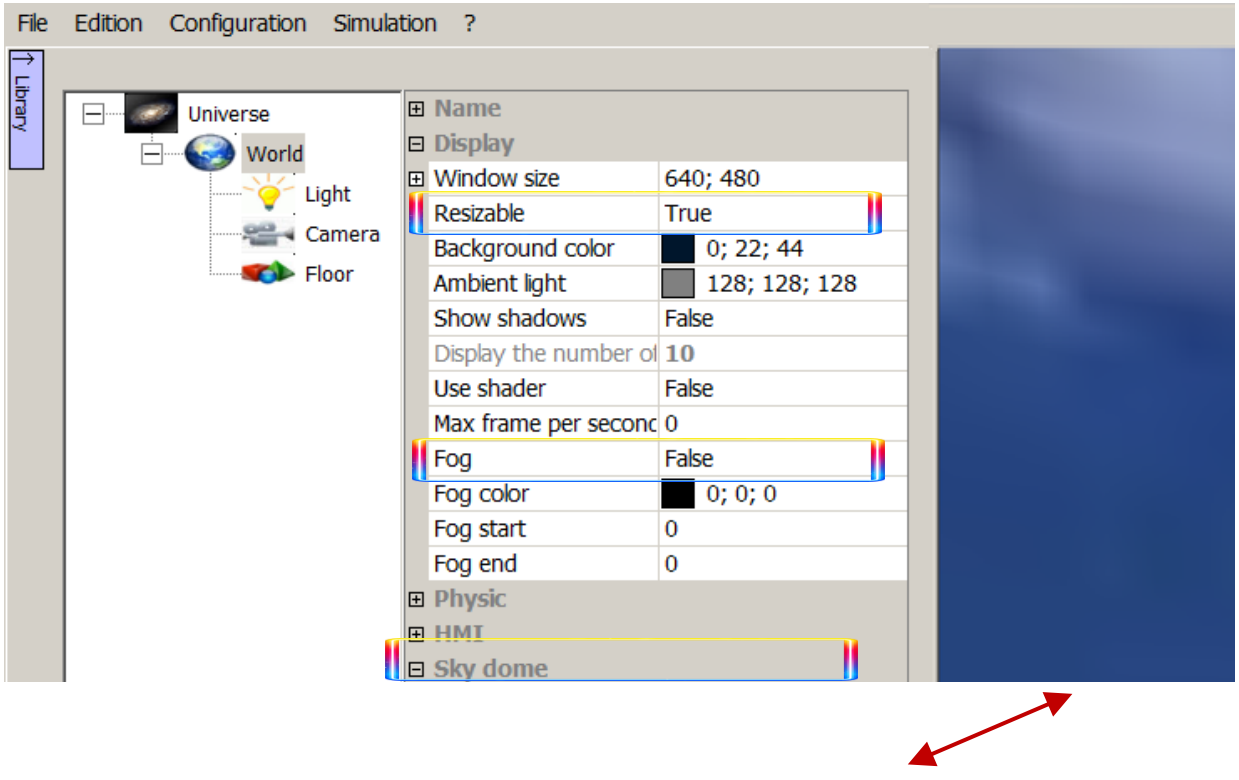


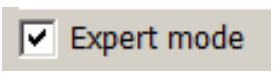
It is possible to change the units of length (meters or millimeters), the background color, or to add shadows, by accessing the properties of the world.



In Expert mode, it is possible to set many other display settings, always in the properties of the world:

- Resize the 3D emulator window when it opens
- Add an image to represent a sky above the world
- Impose to the 3D rendering engine a maximum graphics refresh rate

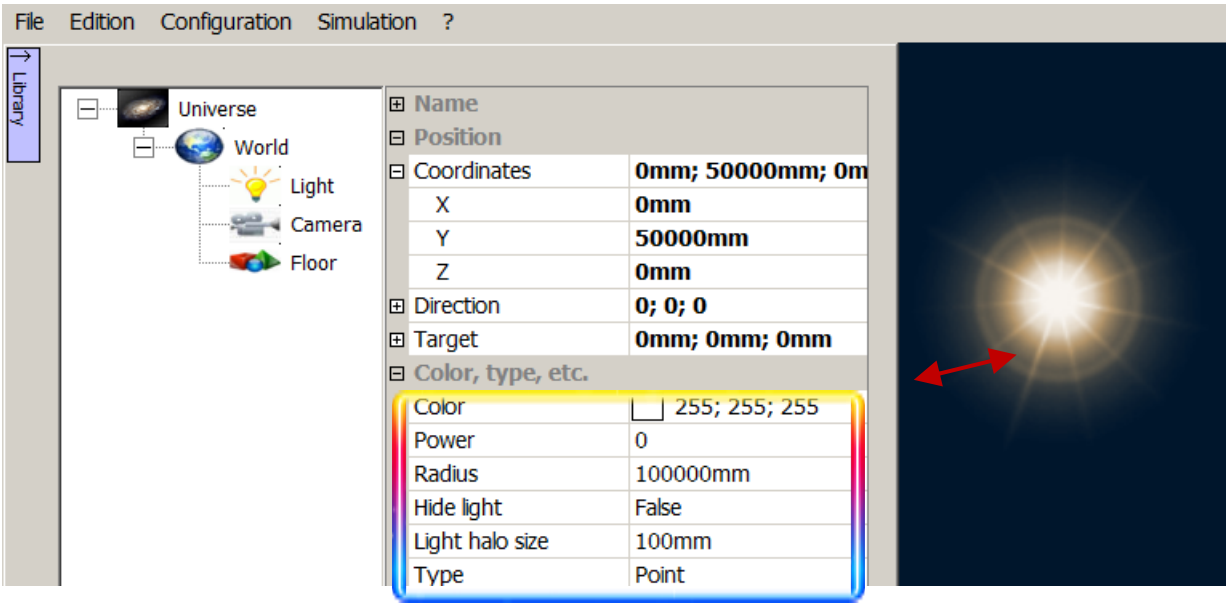




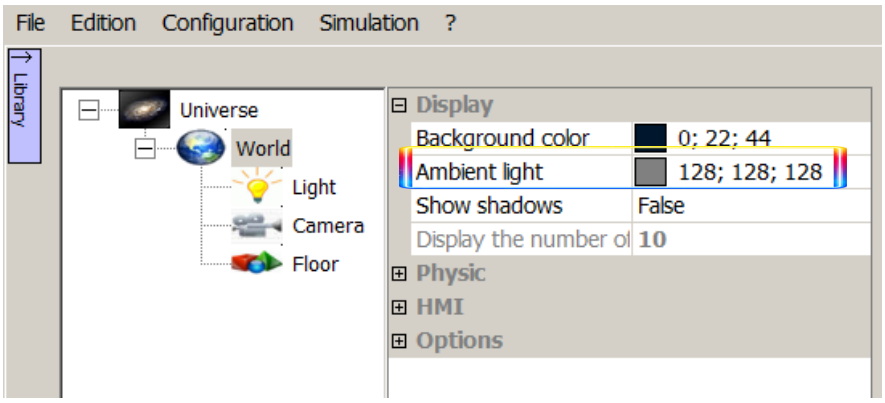
Lighting properties

The lighting of a 3D emulator is obtained by adding and adjusting lights. The 3D world already has a default preset light.

This light, white, is located 50 meters above ground and has a radius of 100 meters. It is possible to change the properties of this light, as it is possible to add more lights in the 3D emulator.



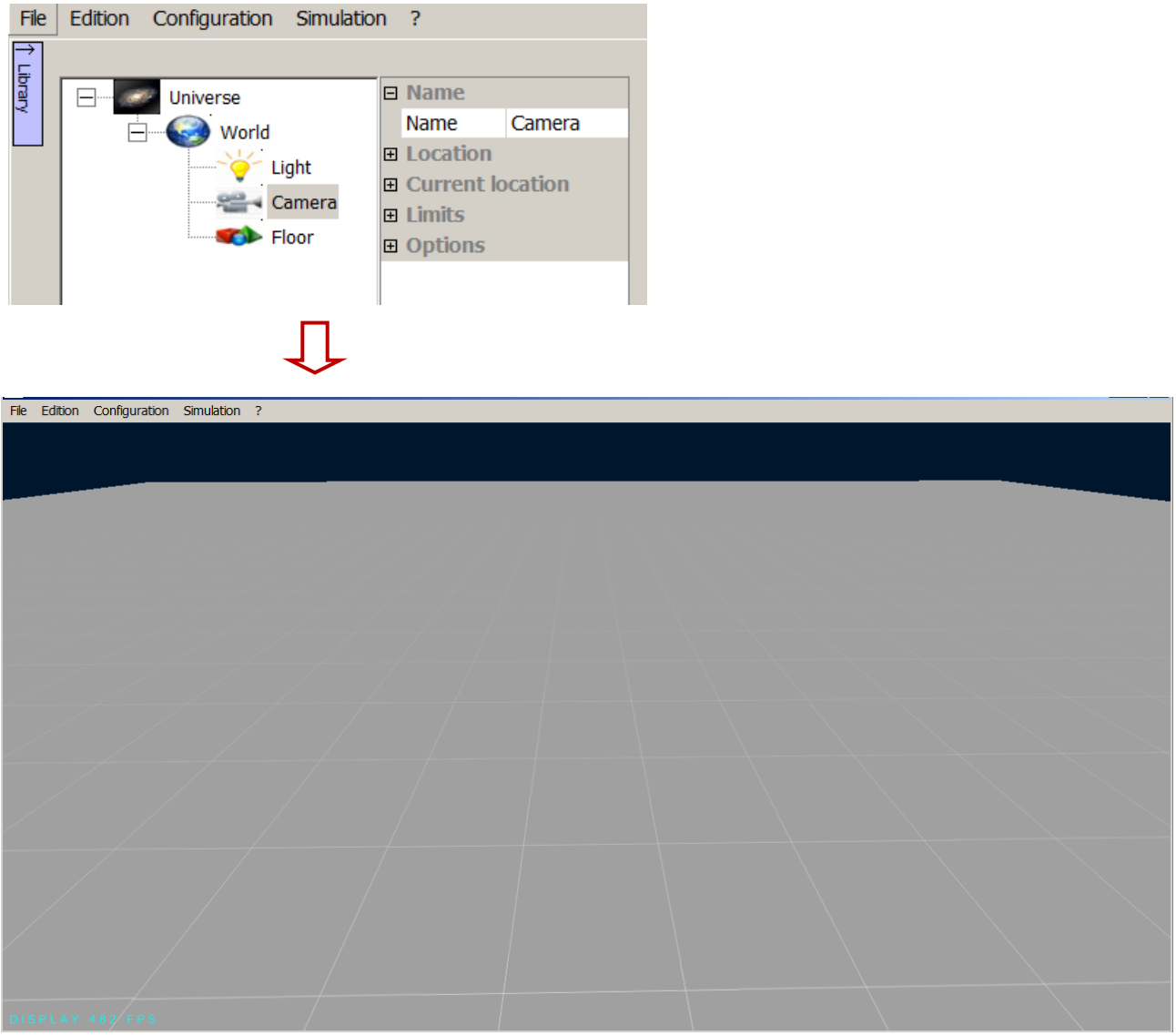
Another parameter enables to set the default ambient light in the world (the light and its intensity), regardless of the lights added to the 3D emulator. This parameter is located in the properties of the world.



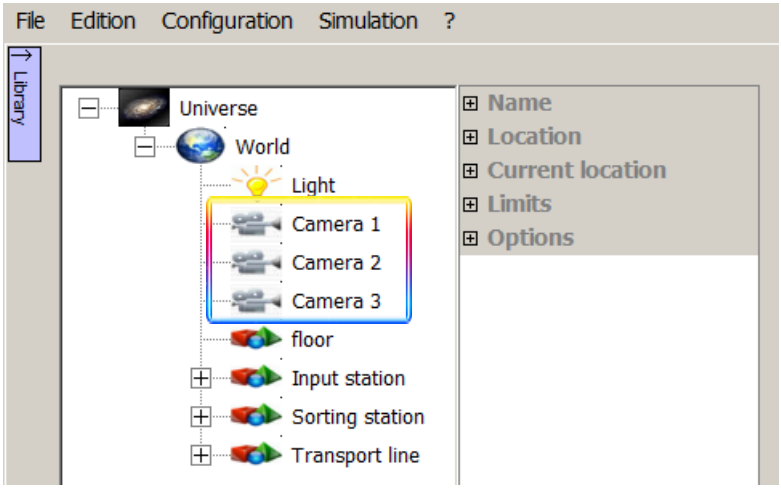
Visualization properties

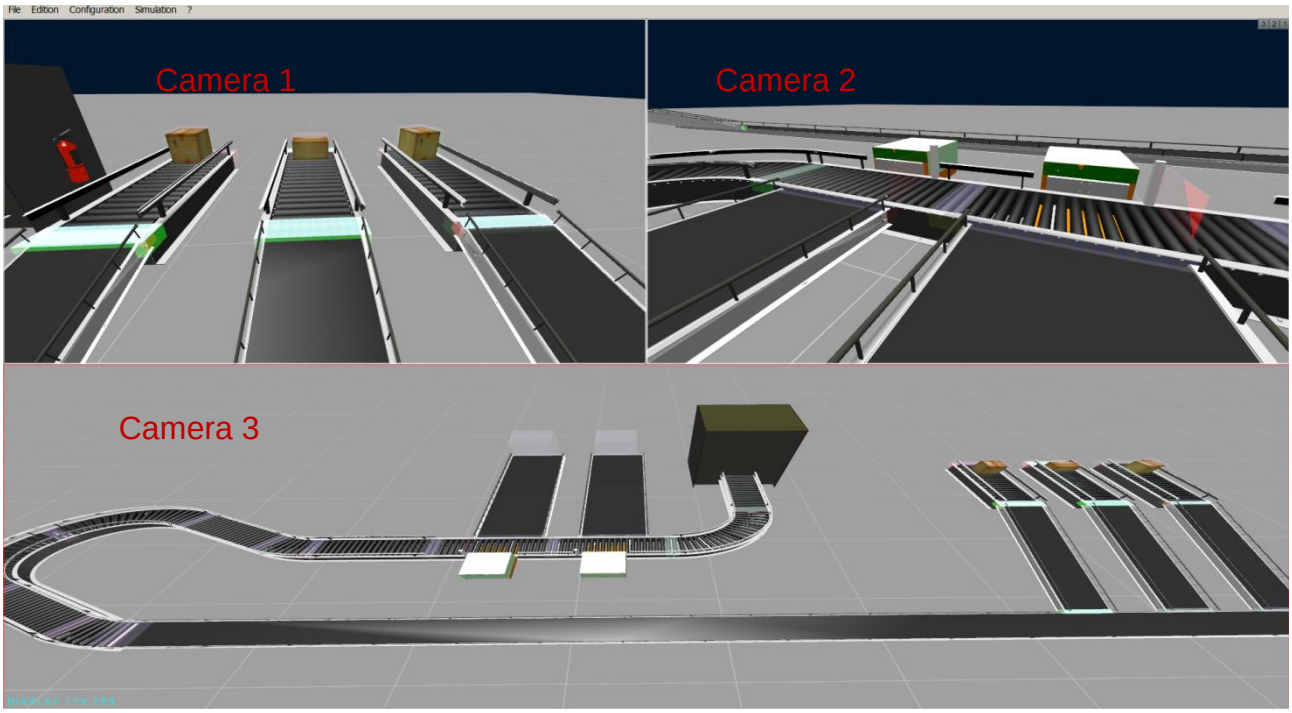
Visualization in a 3D emulator is managed by the use of cameras (viewpoints).

By default, the 3D emulator has a single fixed camera (single viewpoint), whose objective is directed to the center of the world.



It is possible to add more cameras (fixed or mobile) and so to create additional views within the 3D emulator.

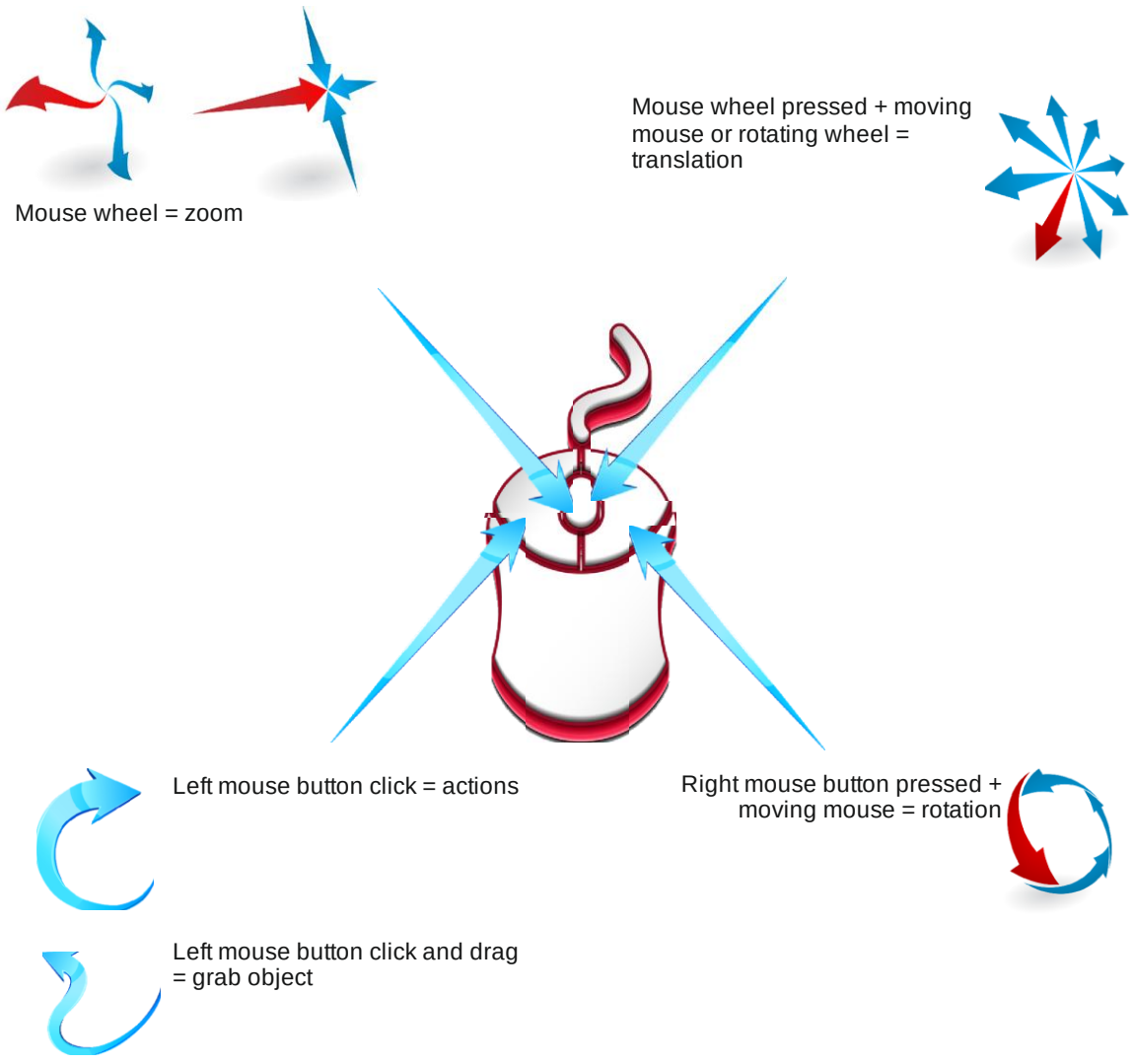




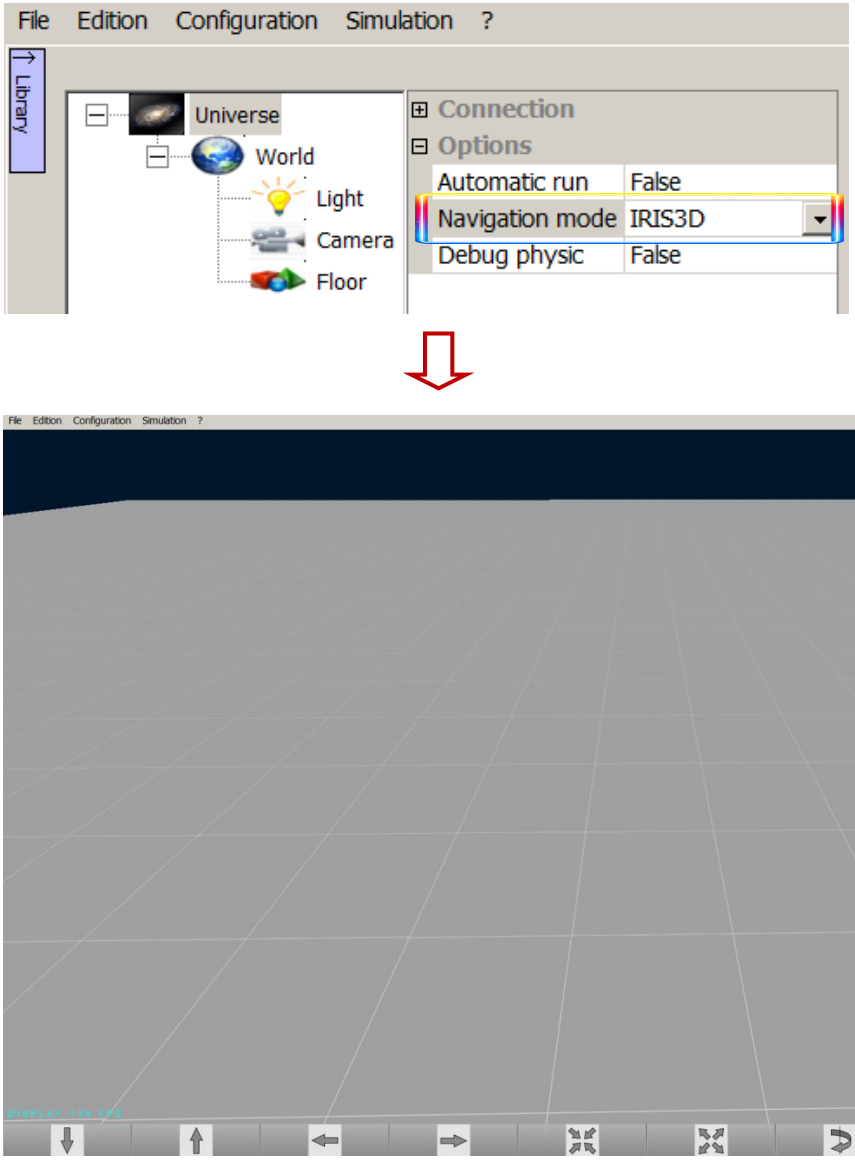
Caution! This method of multi-camera view can strongly reduce the performance of the graphics rendering (refresh rate)

Navigation properties

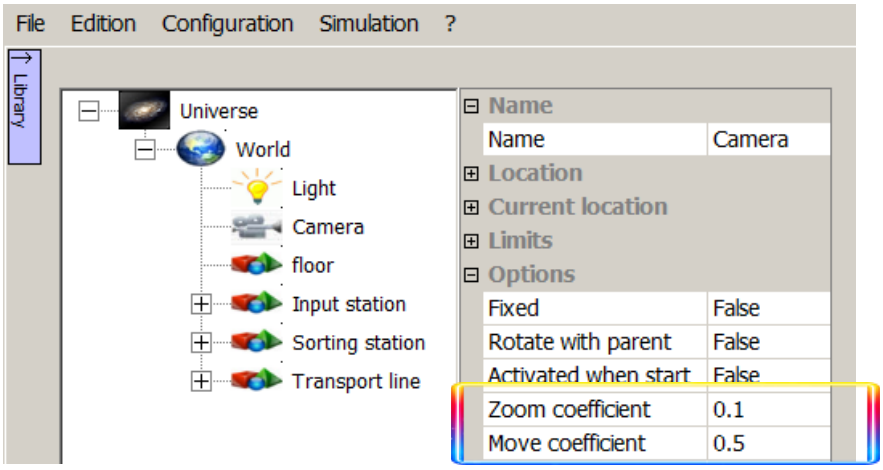
By default, the navigation within a 3D emulator is done using the mouse and its buttons (VU navigation mode).



It is possible to use another method of navigation (called IRIS3D) available in the properties of the universe. This method of navigation uses arrow buttons to move around and zoom in the 3D world. In this mode, the mouse navigation is still possible.

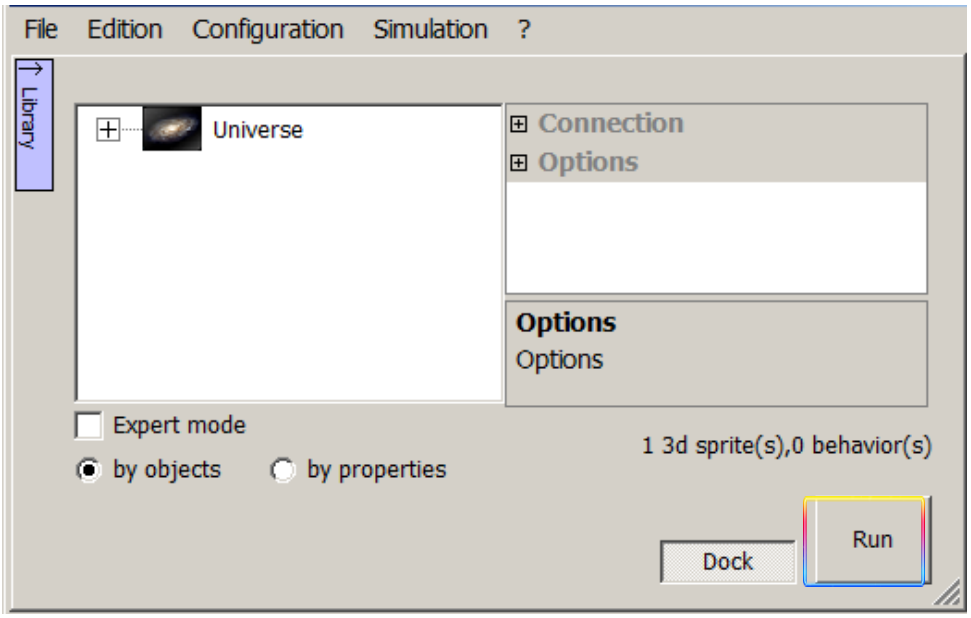


Finally, it is possible to adjust the move and zoom coefficients, in order to move or zoom faster or slower inside the 3D rendering (whatever the method used for navigation, VU or IRIS3D). These coefficients are specific to the camera and are accessible in its properties.

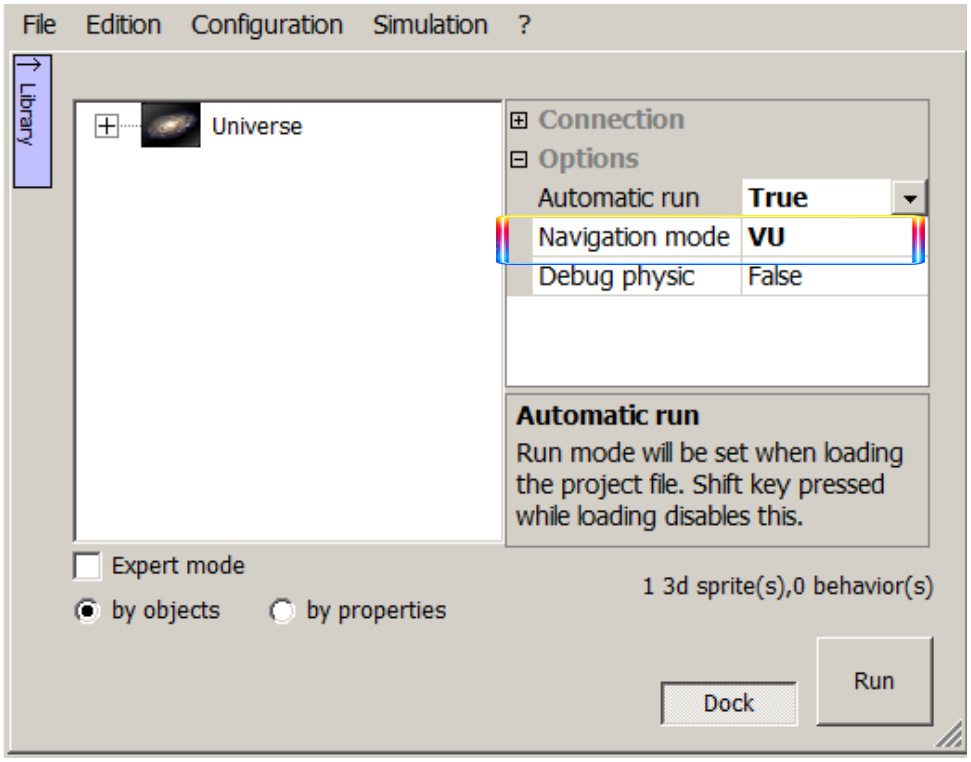


Simulation options

By default, at the opening, the simulation in VIRTUAL UNIVERSE PRO is in STOP mode. The launch of the simulation (switch to RUN mode) is obtained at any time by pressing the "RUN" button.



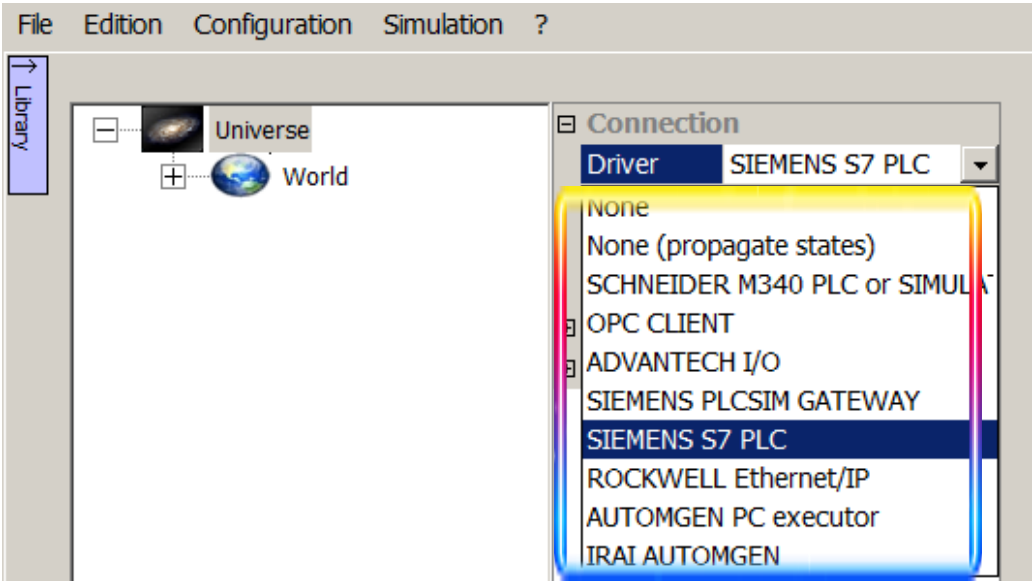
It is possible to automatically run the simulation at the opening of the VIRTUAL UNIVERSE PRO project, by selecting "Automatic Run" in the properties of the Universe.



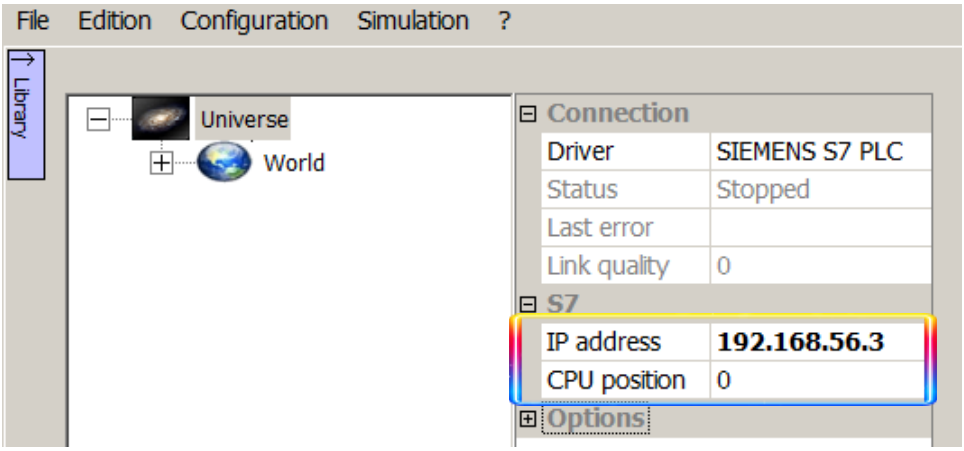
In the default project opened by default, VIRTUAL UNIVERSE PRO is not connected to any external software/controller.

To connect the VIRTUAL UNIVERSE PRO 3D emulator to an external software/controller, it is necessary to select a "driver" in the properties of the universe. The launch of the simulation (switch to RUN mode), a dialog will be established between VIRTUAL UNIVERSE PRO and the external controller.

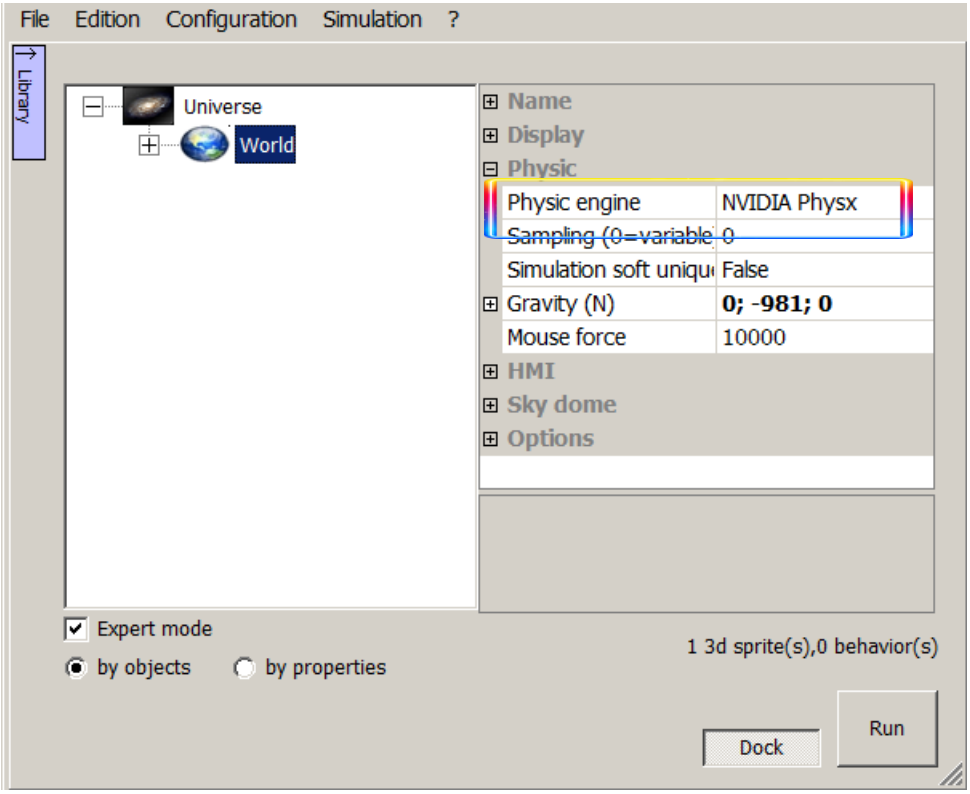
VIRTUAL UNIVERSE PRO is compatible with most major brands of Programmable Logic Controller and offers several types of drivers (for different PLC communication protocols).



Once the "Driver" is selected, the settings of the connection (IP address of the PLC, OPC server name, CPU position on the rack ...) are available in the Driver tab.



In Expert mode, and for advanced users, it is possible to change the physics engine for simulation. In the Properties in the World, it is possible to choose the Nvidia Physix engine in place of Newton Physics (engine used by default).


File
Edition
Configuration
Simulation
?
Library
Universe
World
Name
Display
Physic
Physic engine
NVIDIA Physx
Sampling (0=variable
0
Simulation soft uniqu
False
Gravity (N)
0; -981; 0
Mouse force
10000
HMI
Sky dome
Options
Expert mode
by objects
by properties
1 3d sprite(s),0 behavior(s)
Dock
Run

Import and simplify 3D CAD models

Import 3D CAD models

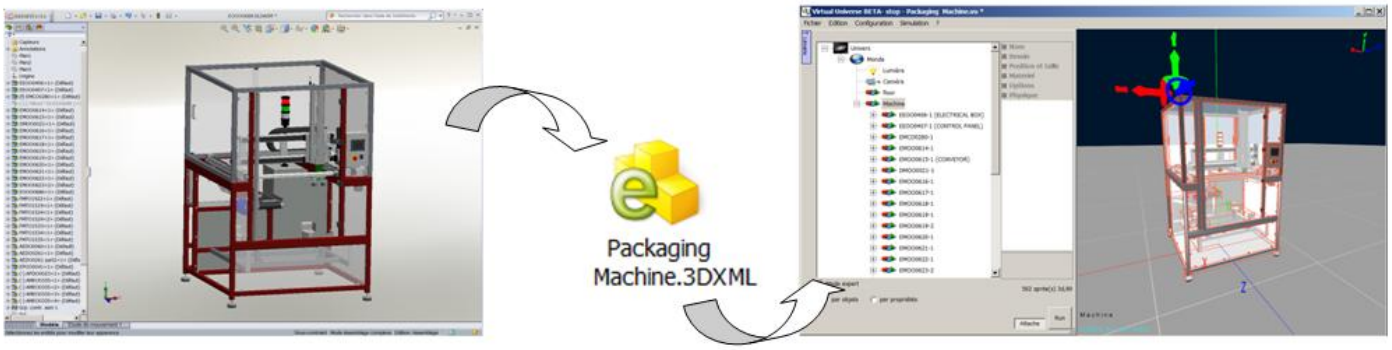
VIRTUAL UNIVERSE PRO allows the reuse of 3D CAD (Computer Aided Design) models by a direct import of these data from their native CAD software, or indirectly through an exchange file.

Here are the types of 3D CAD data that can be imported and reused today in VIRTUAL UNIVERSE PRO:

CAD Software	Exchange file (indirect import)
DS SolidWorks	3DXml
Autodesk Inventor	
DS Delmia	3DXml
DS Catia	3DXml

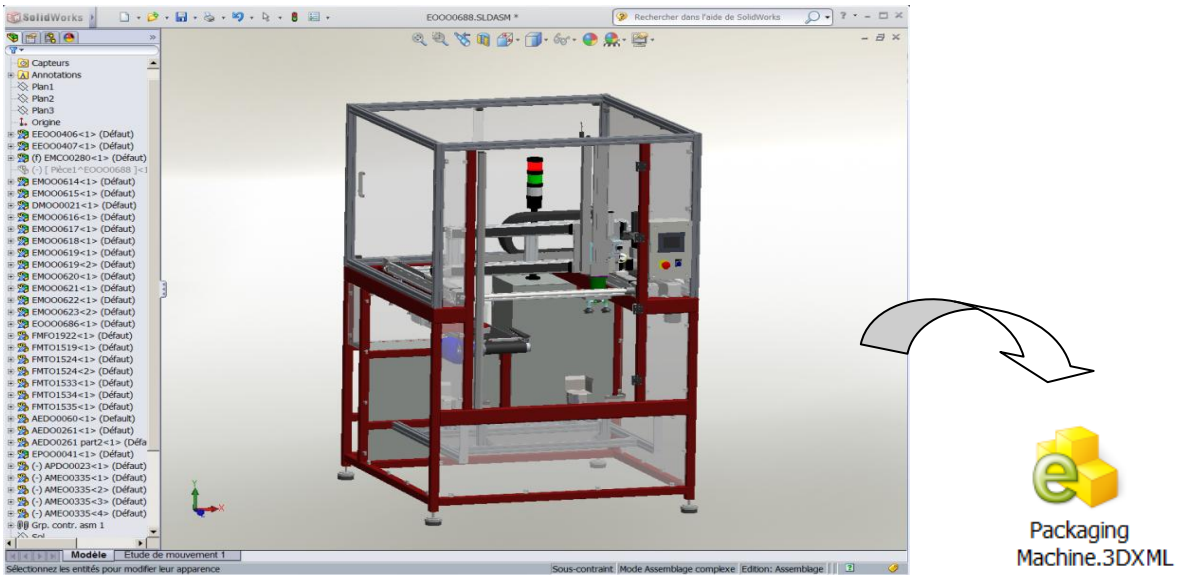
Import SolidWorks models

With this importation method, the SolidWorks data is first exported and saved as a 3DXML file. This file is next imported into VIRTUAL UNIVERSE PRO.

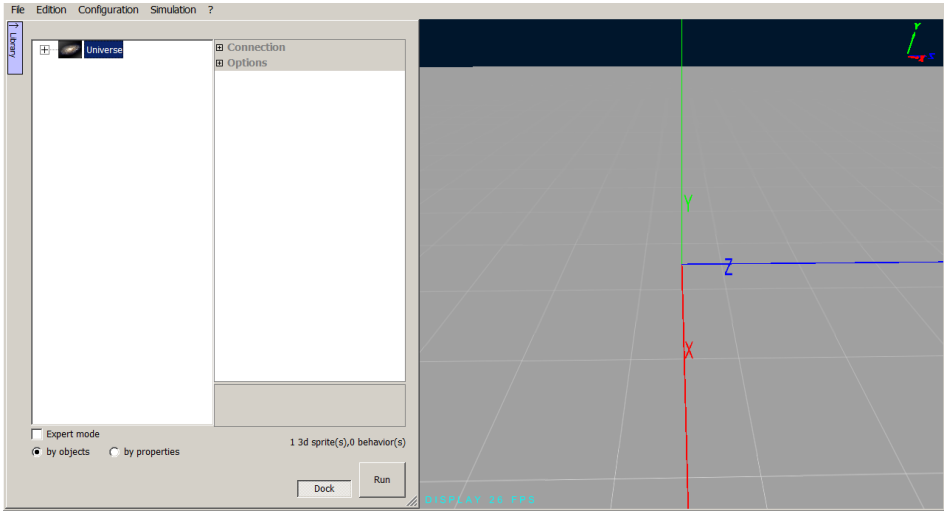


The advantage of this method is to not constraint the VIRTUAL UNIVERSE PRO user to have SolidWorks installed on his computer.

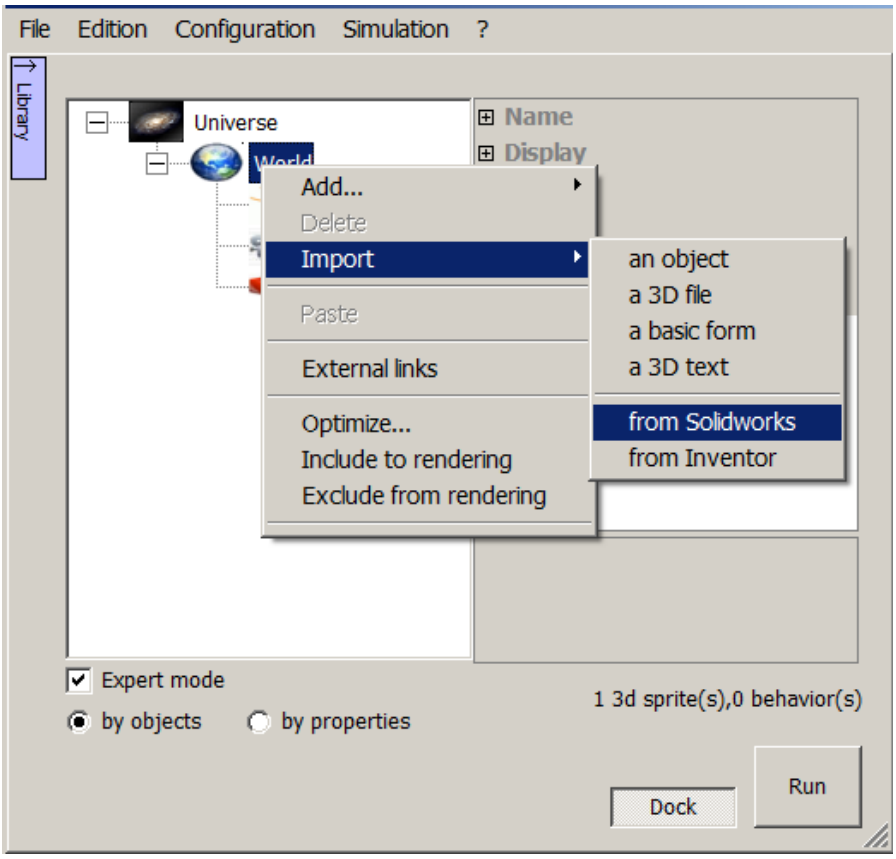
- Previously, the full 3D data has to be opened in SolidWorks and saved as a 3DXML file.



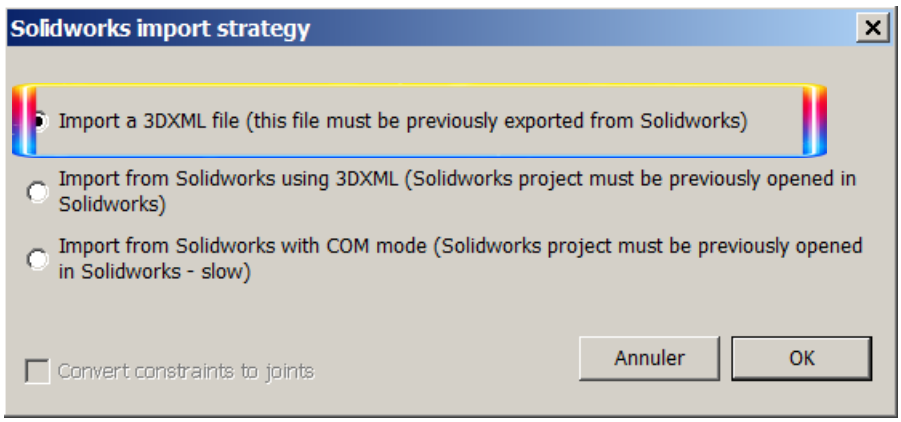
- In VIRTUAL UNIVERSE PRO, open the set-up window.



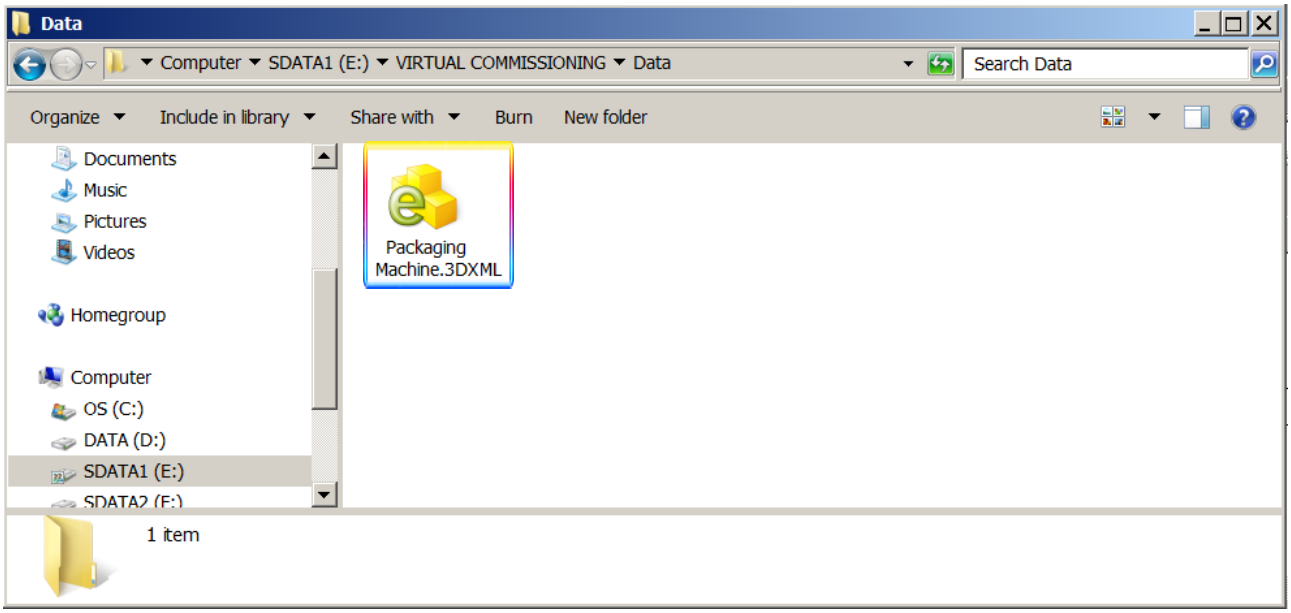
- Right-click at the World level, to access the Import/from SolidWorks menu. Importing data is also available at a sprite level, whatever its position in the project tree.



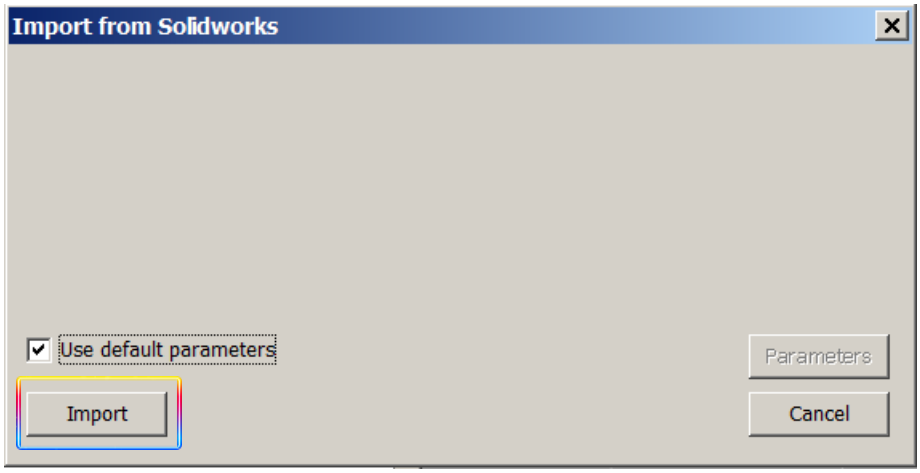
- A dialog box opens with several import methods. Choose the first method "Import a 3DXML file".



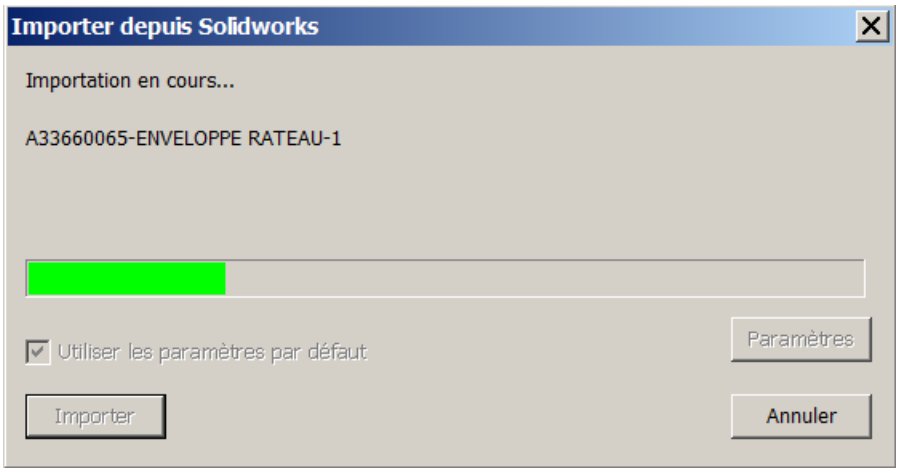
- An explorer window opens, you can select the 3DXML file to import.



- Then click on the “Import” button.

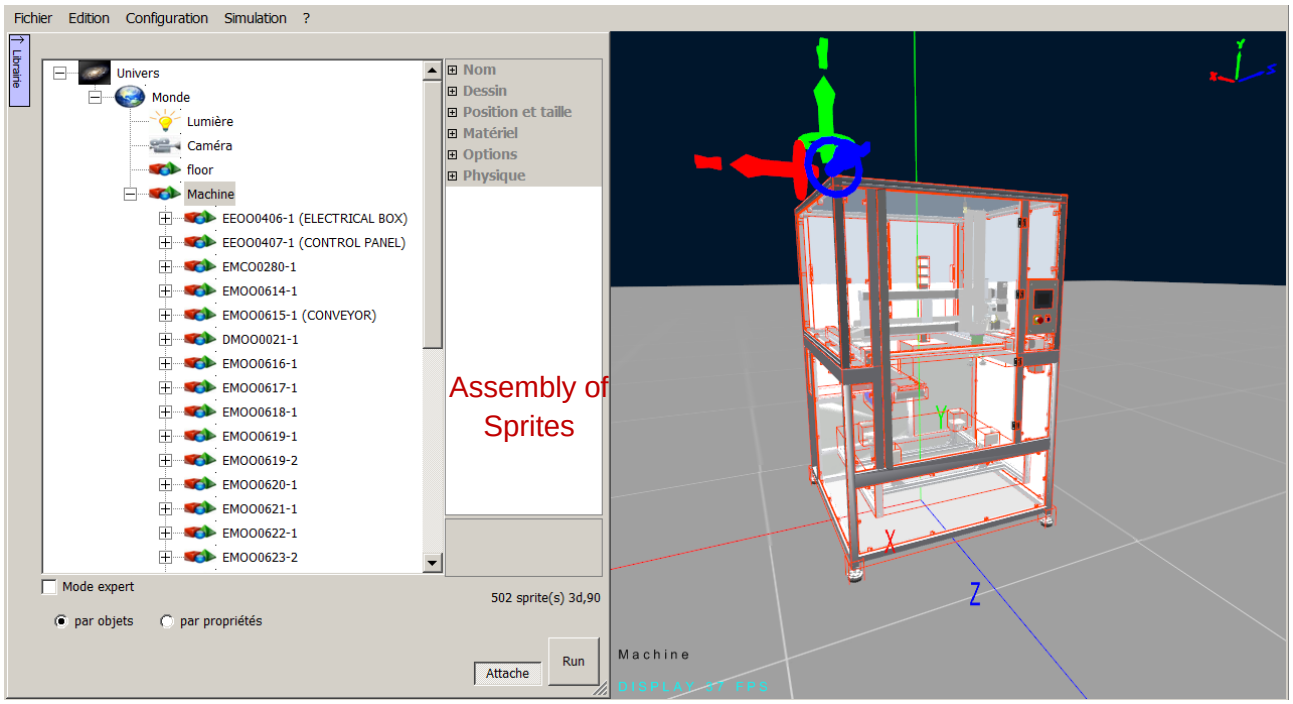


- The import process is then started. A bar indicates the import progress. The waiting time ranges from several seconds to several minutes, depending on the imported project size.



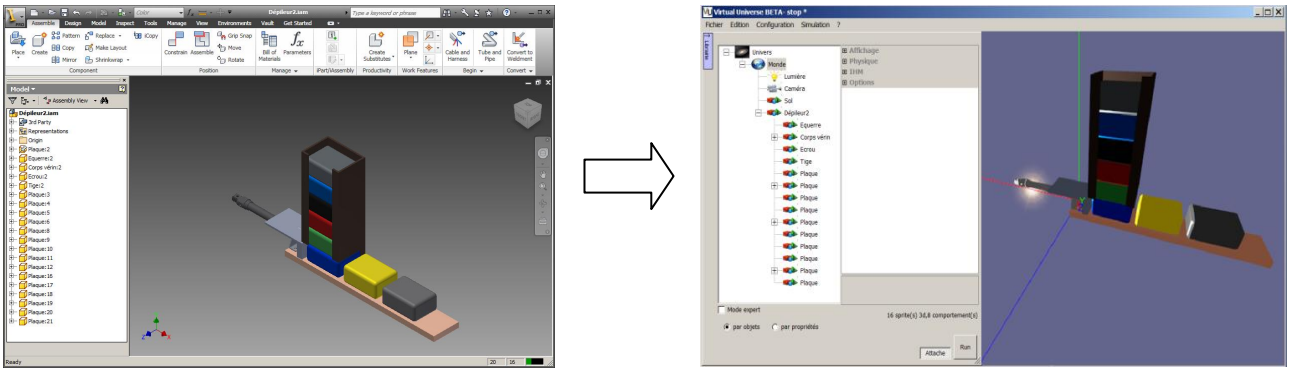
The 3D data is then imported into VIRTUAL UNIVERSE PRO as a tree of 3D objects (sprites), available in the set-up window.

This tree follows the original structure of the SolidWorks data (composed of parts and assemblies). With an assembly, each part is imported as an independent 3D object (sprite) in VIRTUAL UNIVERSE PRO. The initial colors and textures are also recovered in VIRTUAL UNIVERSE PRO.

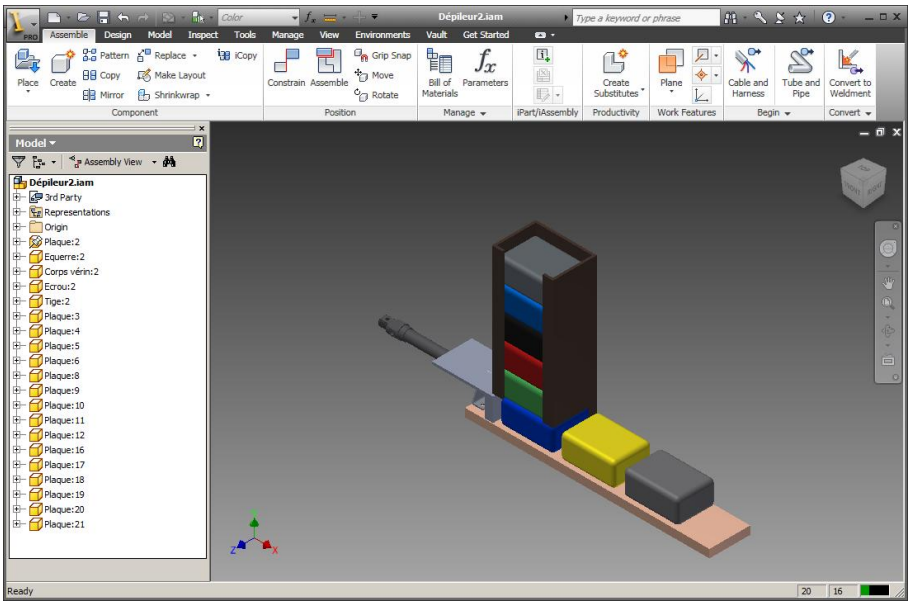


Import Inventor models

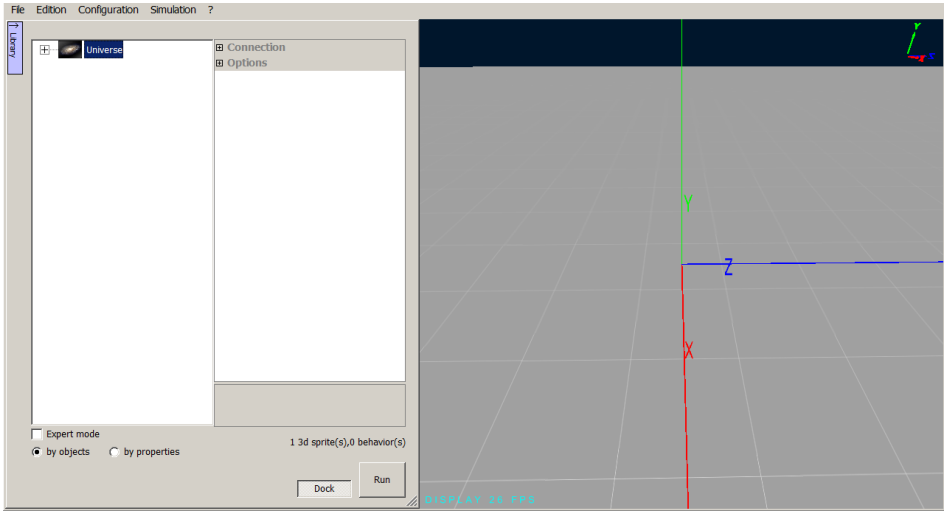
This method requires Autodesk Inventor opened simultaneously with VIRTUAL UNIVERSE PRO on the same computer.



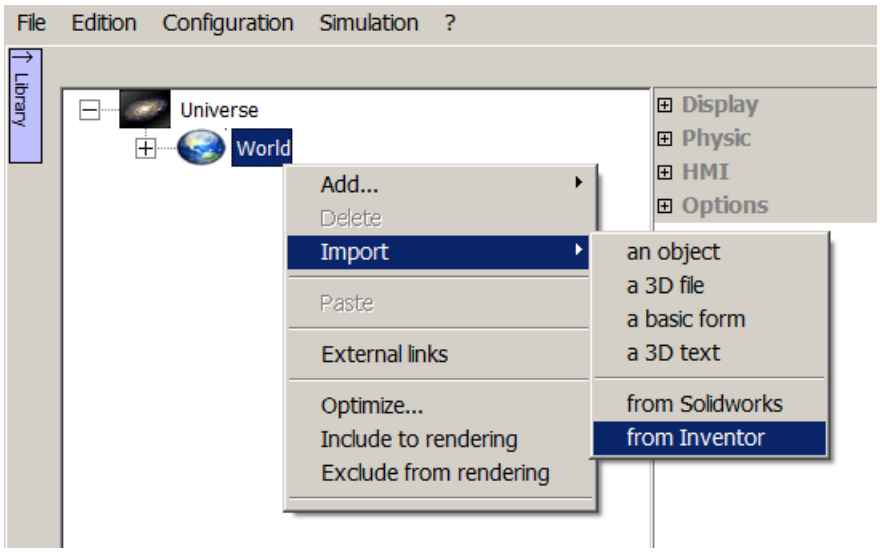
- First, open the full 3D data in Autodesk Inventor.



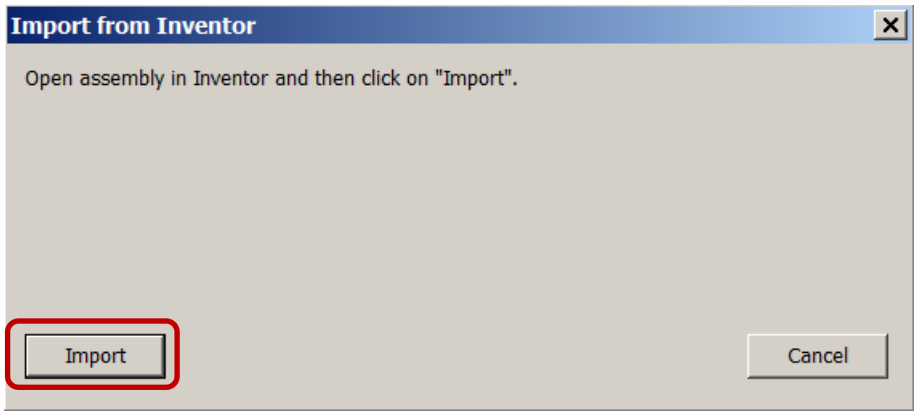
- In VIRTUAL UNIVERSE PRO, open the set-up window.



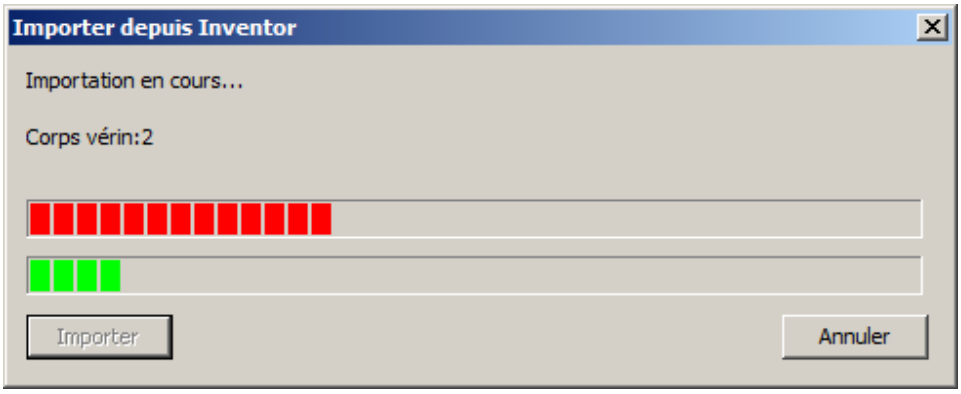
- Right-click at the World level, to access the **Import/from Inventor** menu. Importing data is also available at a sprite level, whatever its position in the project tree.



- Then, click on the “Import” button.

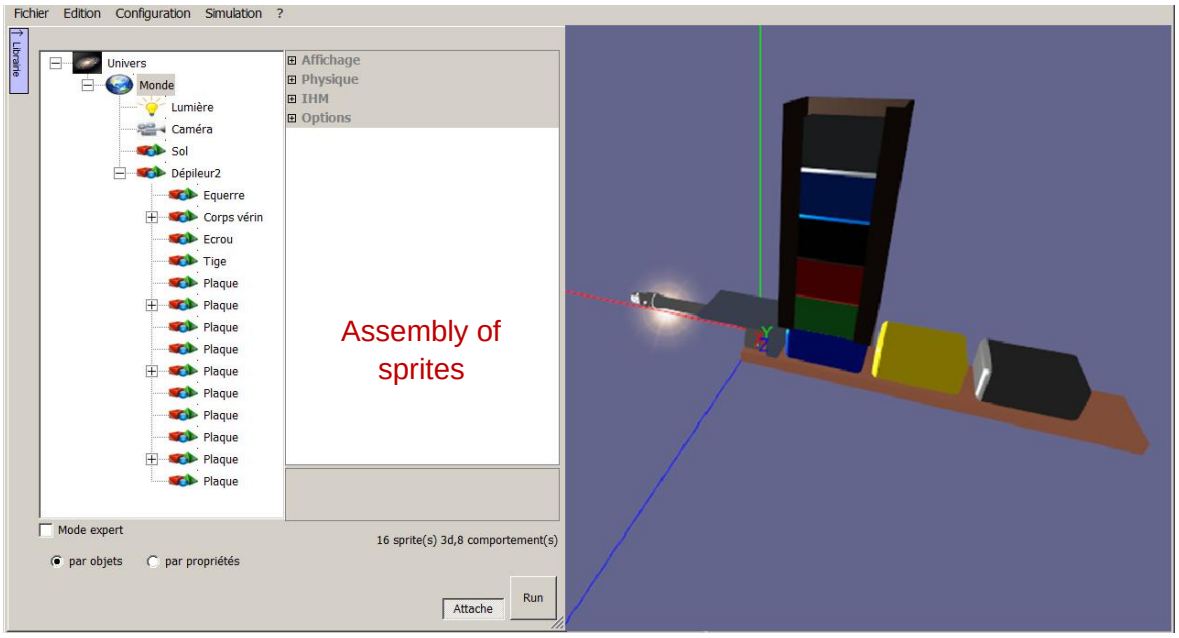


- The import process is then started. A bar indicates the import progress. The waiting time ranges from several seconds to several minutes, depending on the imported project size.



The 3D data is then imported into VIRTUAL UNIVERSE PRO as a tree of 3D objects (sprites), available in the set-up window.

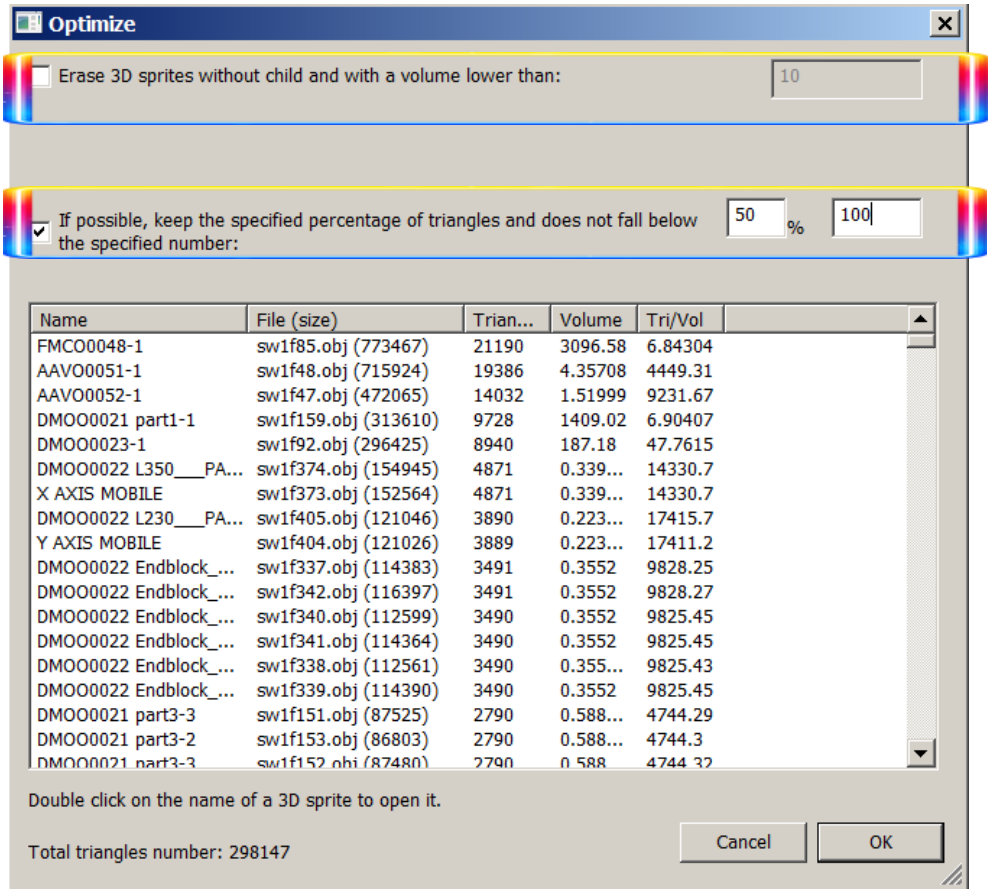
This tree follows the original structure of the Inventor data (composed of parts and assemblies). With an assembly, each part is imported as a 3D object (sprite) independent in VIRTUAL UNIVERSE PRO. The initial colors and textures are also recovered in VIRTUAL UNIVERSE PRO.



Simplify 3D CAD models

The 3D models created with CAD software often have very complex 3D geometries (number of triangles) and small parts that are not always necessary to VIRTUAL UNIVERSE PRO 3D emulators and may instead greatly reduce graphics performance.

For the simplification of 3D CAD models, VIRTUAL UNIVERSE PRO offers a 3D geometries optimization tool.

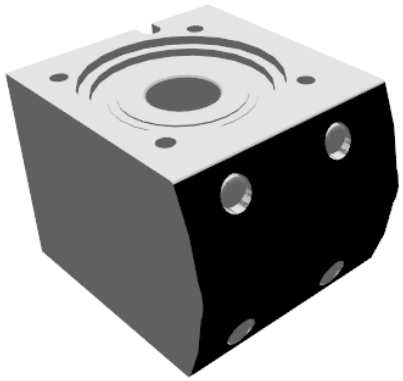


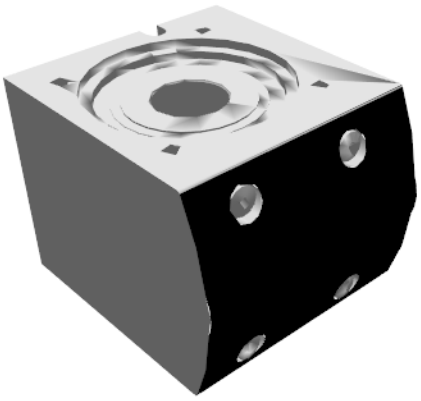
Removal of small 3D parts

Reduction of the number of triangles

This 3D geometry optimization tool enables to

- Reduce the number of triangles composing a 3D sprite: on the selected 3D sprites, the tool applies a percentage reduction in the number of triangles, respecting a minimum threshold (minimum number of triangles) not to be exceeded.





Original data :

6 982 triangles

Optimized data :

2 093 triangles (- 70 %)

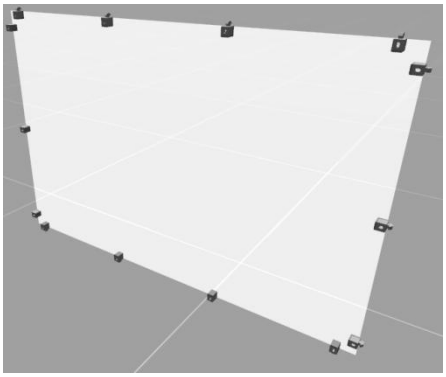
Original data :

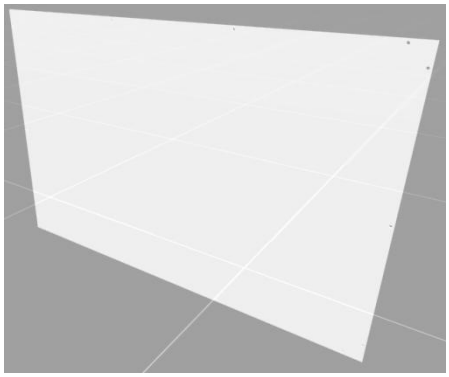
19 456 triangles

Optimized data:

2 234 triangles (- 89 %)

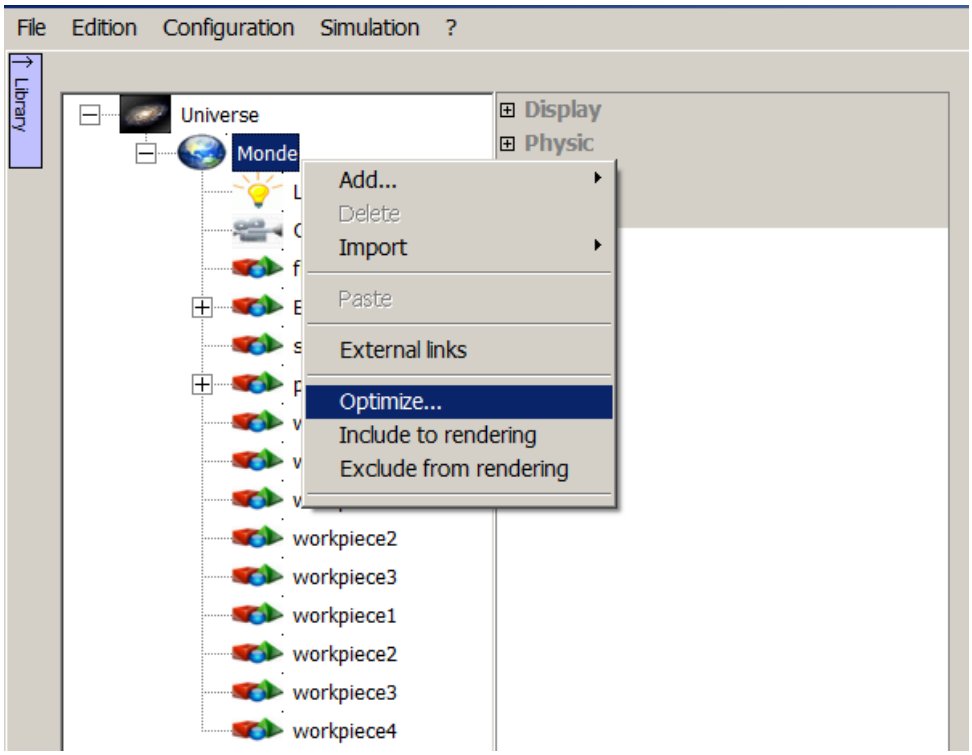
- Remove unnecessary small 3D parts: on the selected 3D sprites, the tool deletes the 3D parts whose size (volume) is below a threshold.



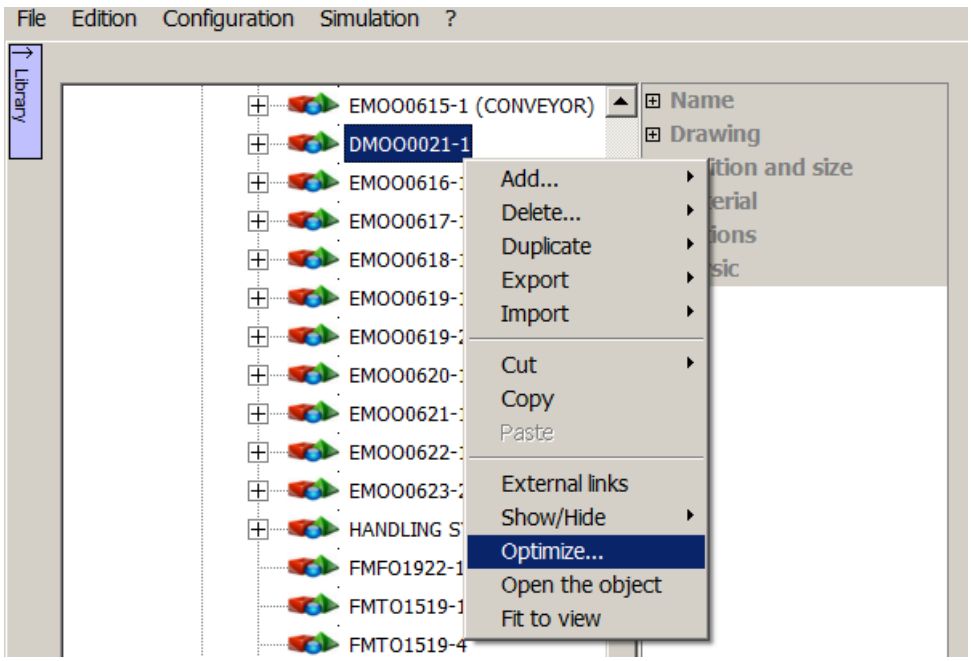


This optimization tool is available at two levels:

- At the World level: optimization is applied to all 3D sprites in the 3D emulator project



- At a 3D sprite level: optimization is only applied to the selected sprite and all its 3D sprites children



For more information on how to measure and optimize graphics performance of a 3D emulator, see Measure graphics performances

Design smart 3D resources and systems

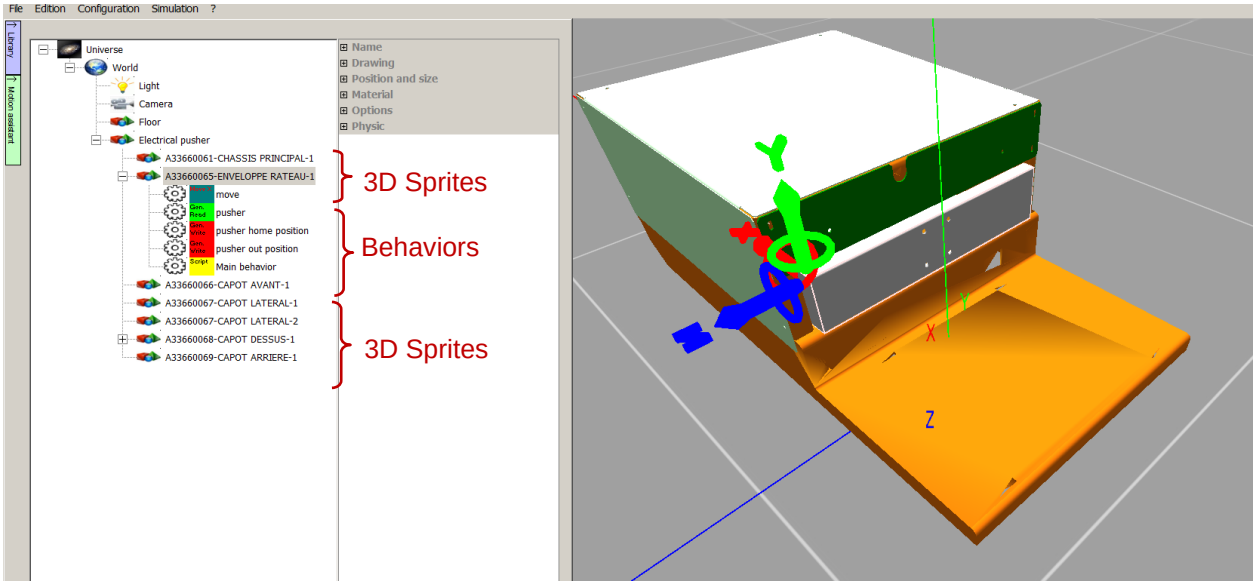
VIRTUAL UNIVERSE PRO enables the design of new smart 3D resources and systems, by reusing 3D CAD models (Computer Aided Design).

These smart 3D resources and systems can be saved independently and be added to the VIRTUAL UNIVERSE PRO resource library, to be reused for future 3D emulator projects.

In VIRTUAL UNIVERSE PRO, a smart 3D resource is often composed of several 3D sprites (structured assembly of sprites) to which behaviors are attached. During simulation, these behaviors constitute the intelligence of the resource.

Example

The Electrical Pusher, available in the Demo Library, is a smart 3D resource. It features a structured set of sprites and behaviors that controls the drawer release/return (cylinder).

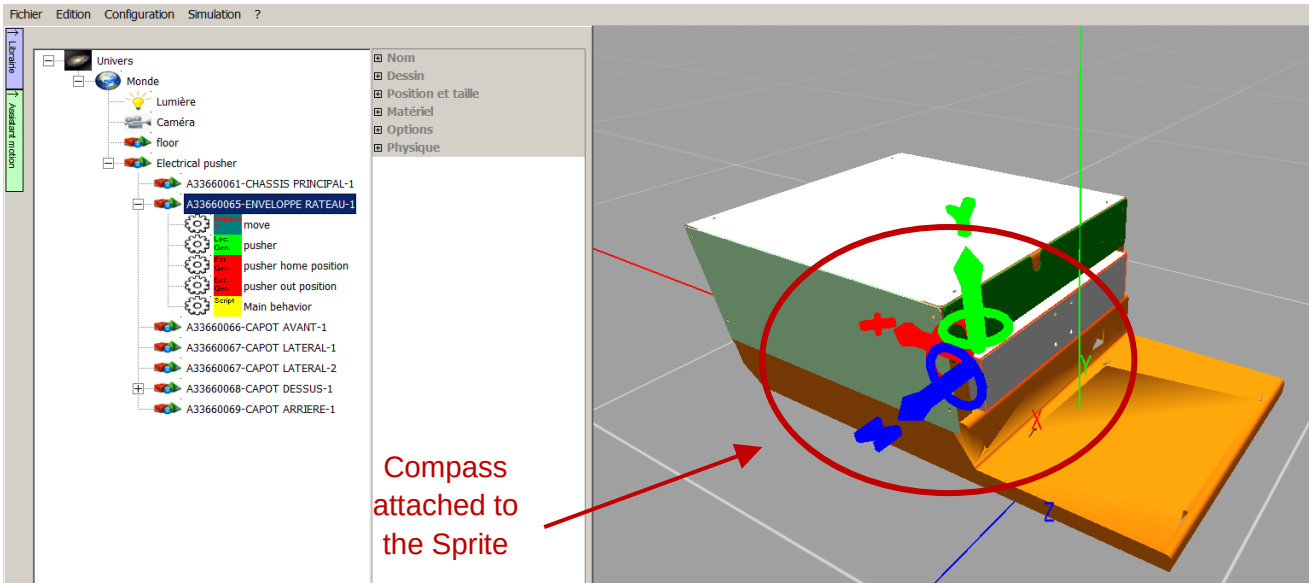


Modify position and dimensions of a 3D sprite

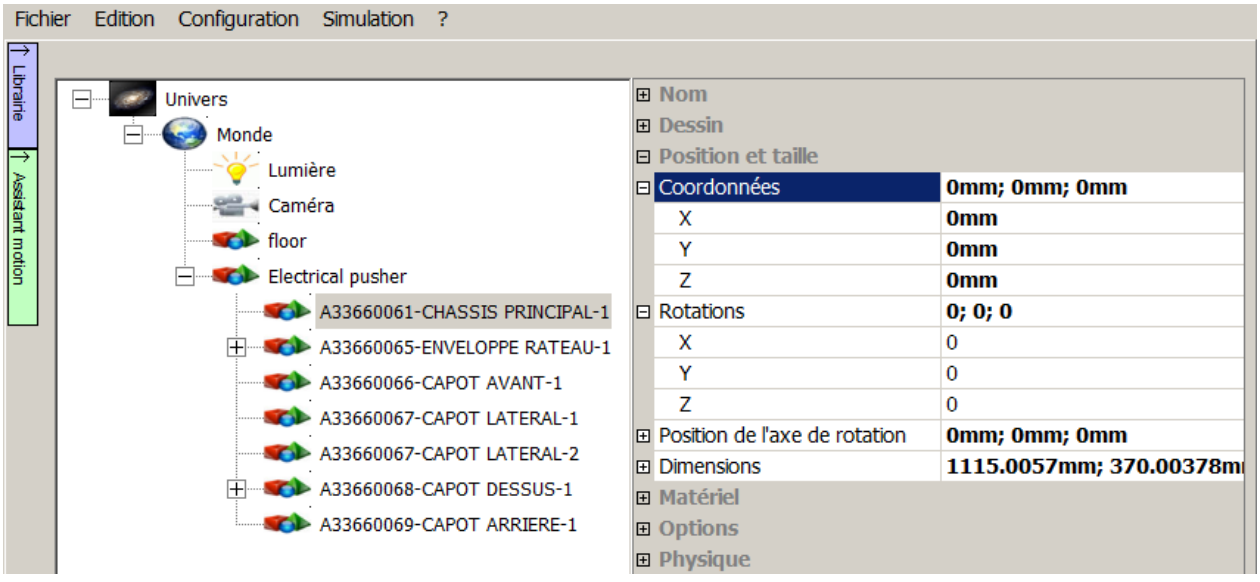
In VIRTUAL UNIVERSE PRO, the position of a sprite in the 3D world is defined in relative coordinates, relative to the parent sprite. Thus, when data is imported from 3D CAD software, each sprite has a default set of null coordinates (position and rotation) in the local frame (positions relative to the parent sprite).

For moving a sprite, there are two methods:

- It can be moved directly by the arrows (translation) or circles (rotation) of the (X,Y,Z) compass attached to the sprite.

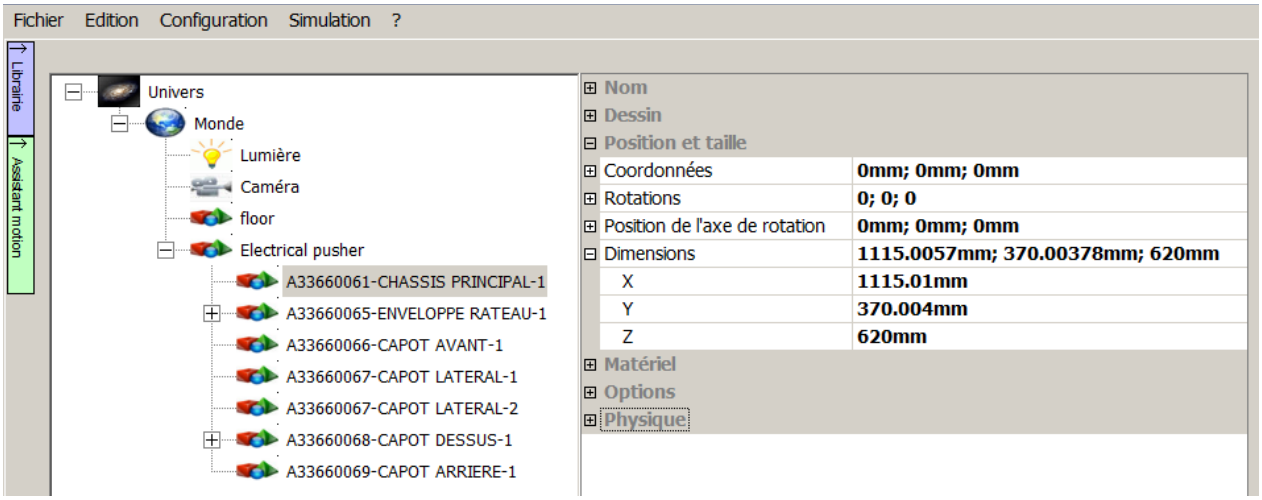


- It can also be moved precisely through the local coordinate settings "Position and Size": The displayed units (meters or millimeters) are those selected in the properties of the World.



- For changing the size of a sprite, just modify the dimensions parameters of this sprite in the "Position and Size" tab in the sprite options.

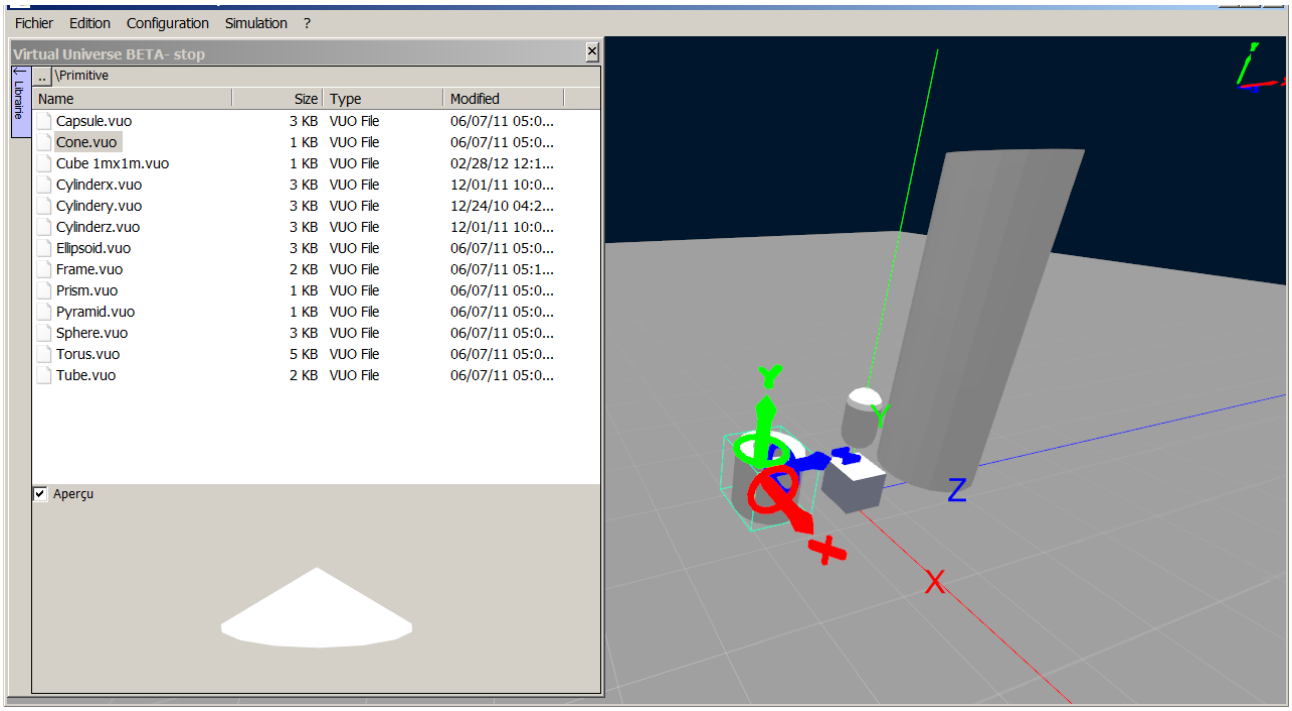
  The displayed units (meters or millimeters) are those selected in the properties of the World.



Add basic 3D shapes

VIRTUAL UNIVERSE PRO provides a set of basic 3D shapes (cubes, cylinders,..) available in the resource library.

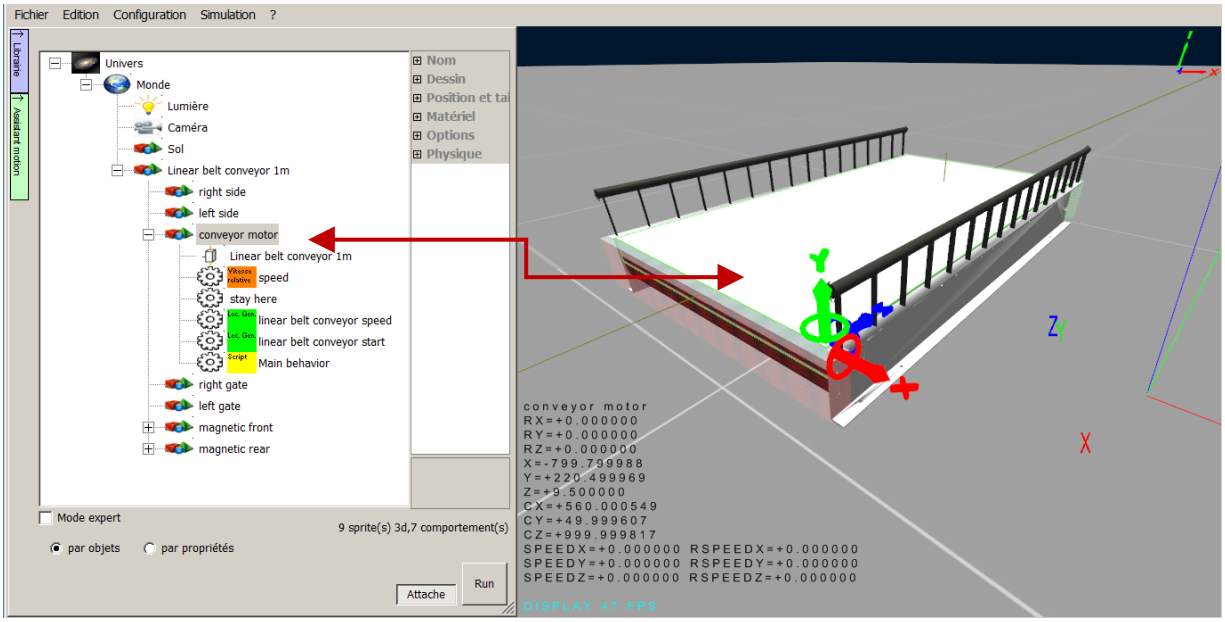
These basic shapes are used to represent the 3D elements missing in the original CAD 3D models (like a sensor beam, or the moving part of a conveyor belt...).



Example

For building the conveyors available in the demonstration library (Demo library), a basic 3D shape (parallelepiped) was used to model the conveyor moving part ("conveyor motor" sprite).

This primitive shape was first inserted into the conveyor resource, resized and positioned in the right place, and it was finally hidden.

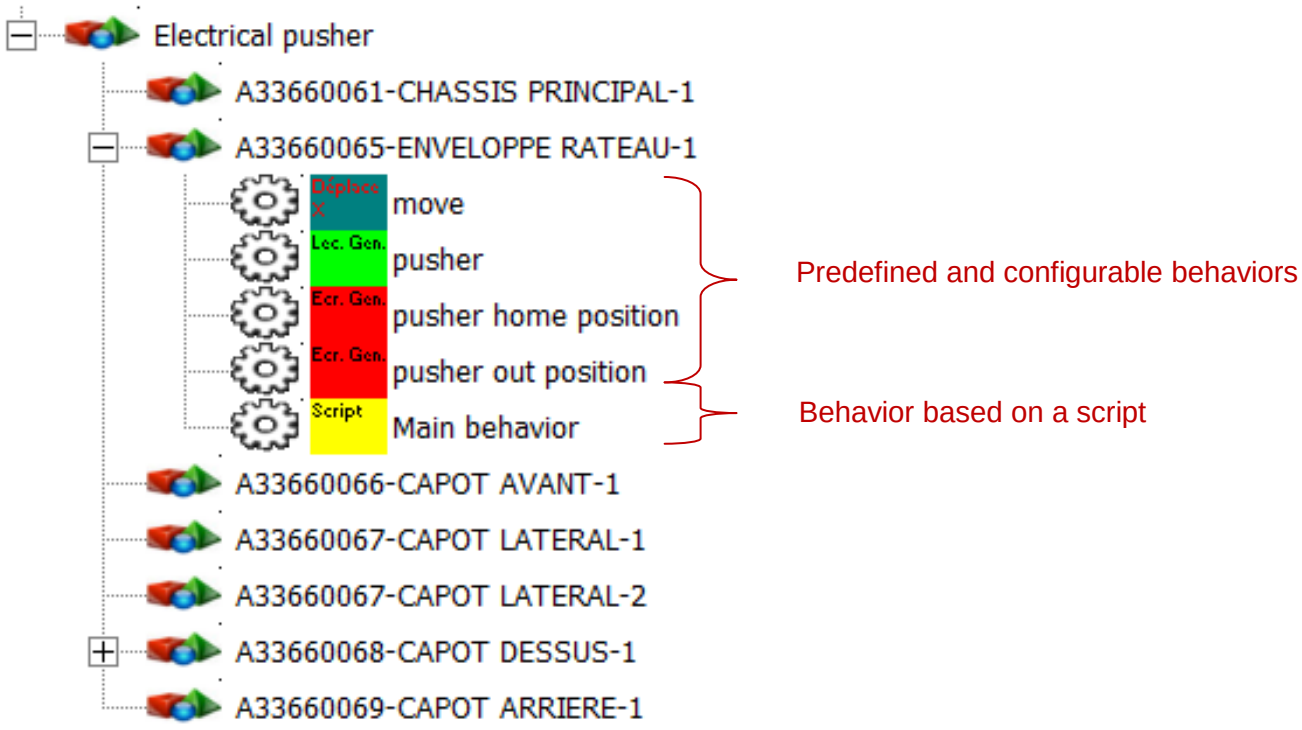

Fichier
Edition
Configuration
Simulation
?
Univers
Monde
Lumière
Caméra
Sol
Linear belt conveyor 1m
right side
left side
conveyor motor
Linear belt conveyor 1m
speed
stay here
linear belt conveyor speed
linear belt conveyor start
Main behavior
right gate
left gate
magnetic front
magnetic rear
Nom
Dessin
Position et tai
Matériel
Options
Physique
Mode expert
9 sprite(s) 3d,7 comportement(s)
par objets
par propriétés
Attache
Run
conveyor motor
RX=+0.000000
RY=+0.000000
RZ=+0.000000
X=-799.799988
Y=+220.499969
Z=+9.500000
CX=+560.000549
CY=+49.999607
CZ=+999.999817
SPEEDX=+0.000000 RSPEEDX=+0.000000
SPEEDY=+0.000000 RSPEEDY=+0.000000
SPEEDZ=+0.000000 RSPEEDZ=+0.000000
DISPLAY 47 FPS

Add behaviors to 3D sprites

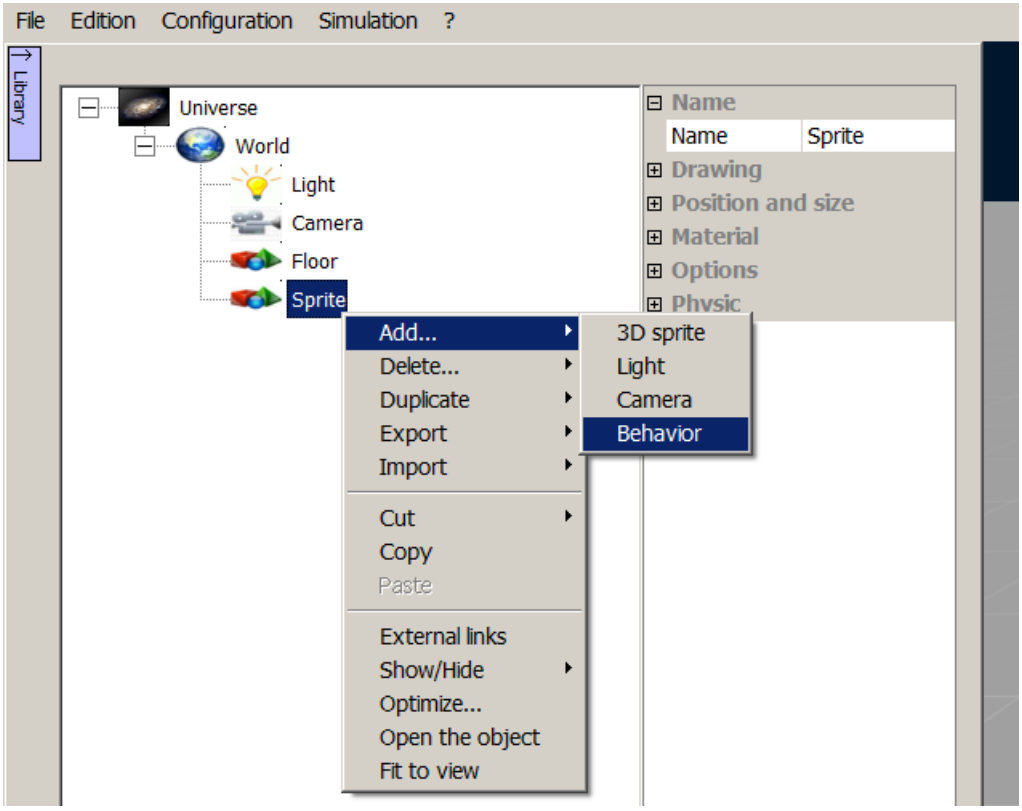
Behaviors are the intelligence provided to the sprites during simulation.

Adding behaviors to sprites enables you to build smart resources, capable to move in the 3D world, interact with other 3D resources or communicate with each other or with external software. For example, behaviors can be used to model the actuators and sensors within the operative part of an automated system.

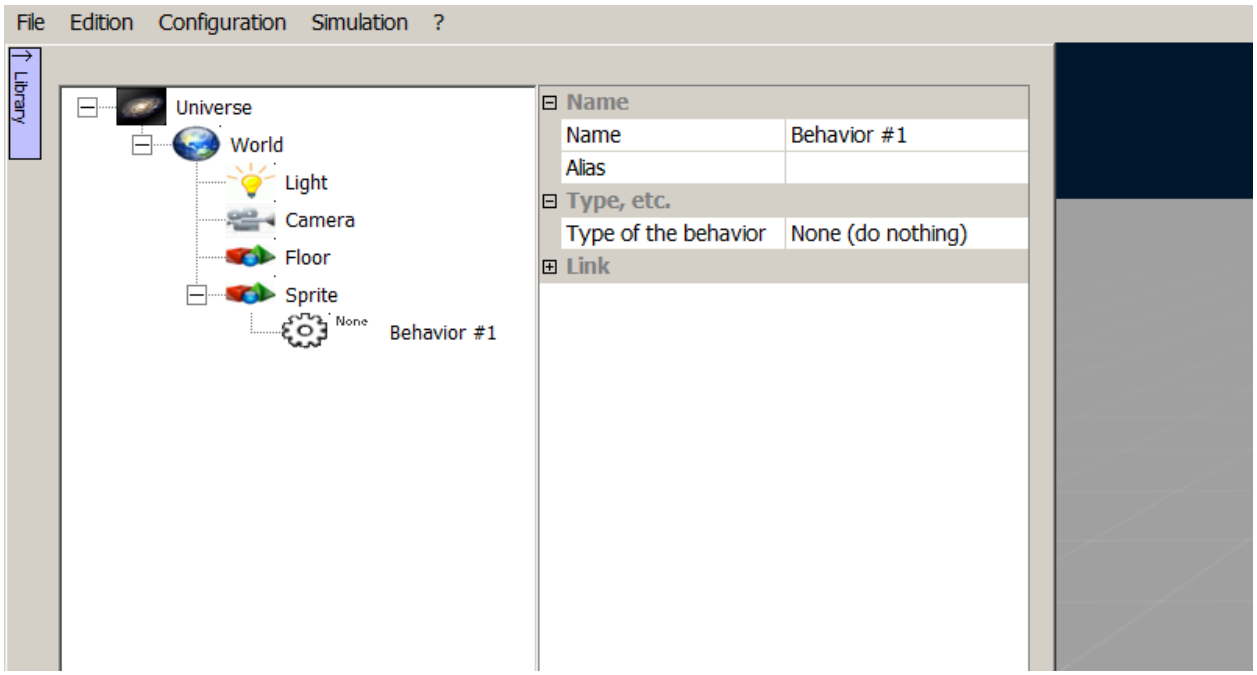
Behaviors are, either predefined and configurable behaviors, available in VIRTUAL UNIVERSE PRO, or custom and more complex behaviors based on scripts, created by user with the VIRTUAL UNIVERSE PRO integrated script editor. A script is frequently used to model the main controller of the resource (internal logic).



- Adding a behavior to a sprite is made by right-clicking on the selected sprite :

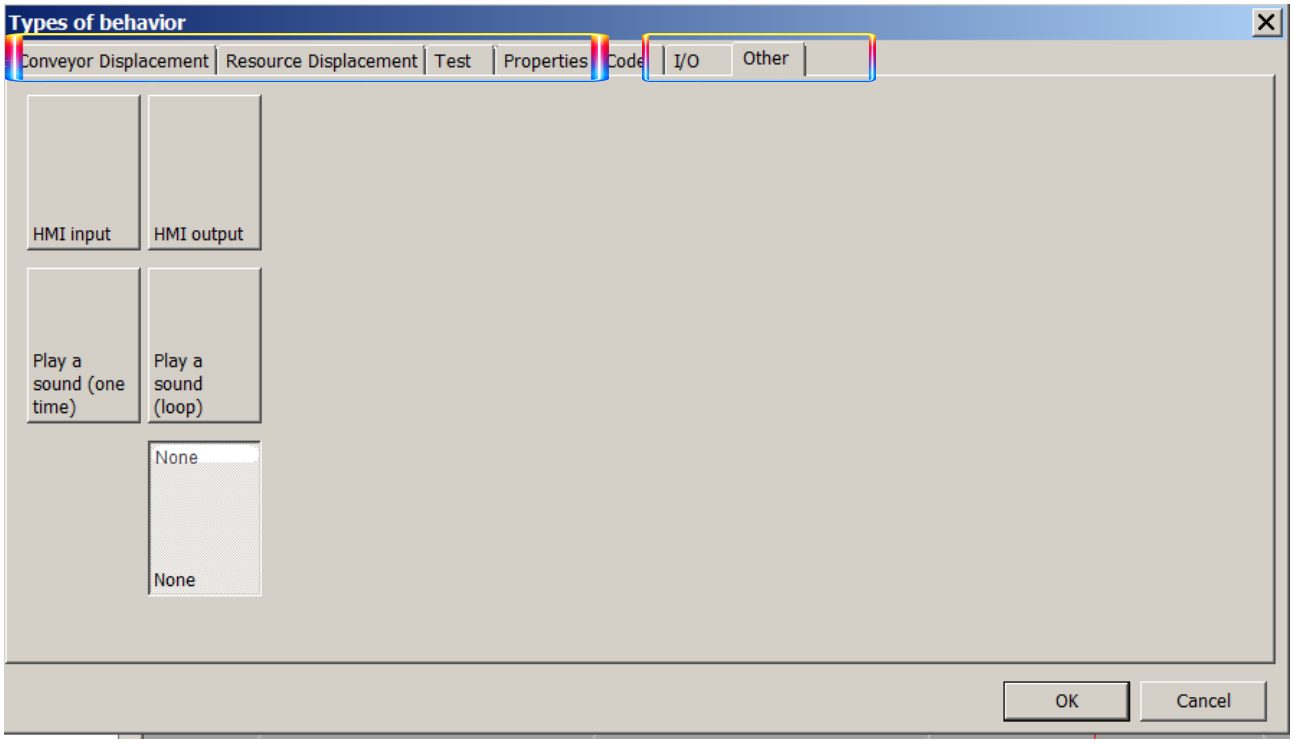


- In the behavior properties, it is possible to choose the behavior type:

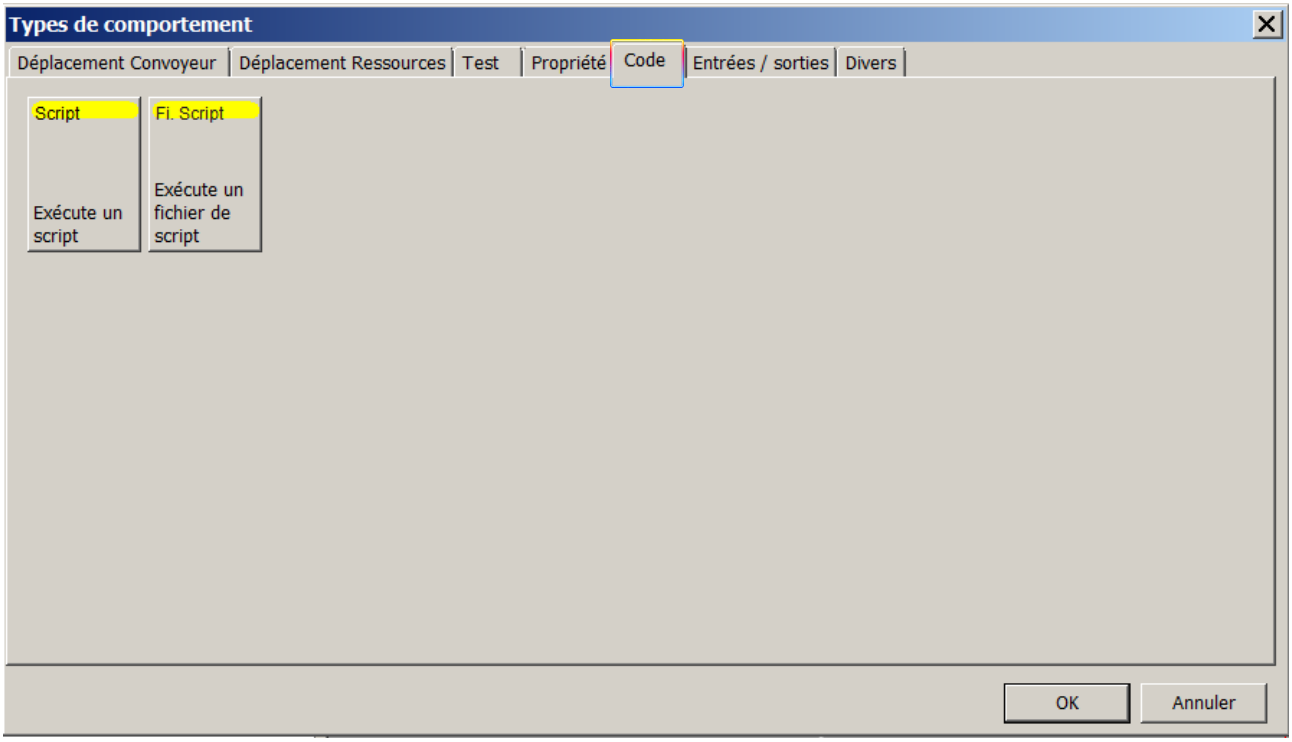


- Behaviors are organized into seven sub categories presented in the “Types of Behaviors” window:

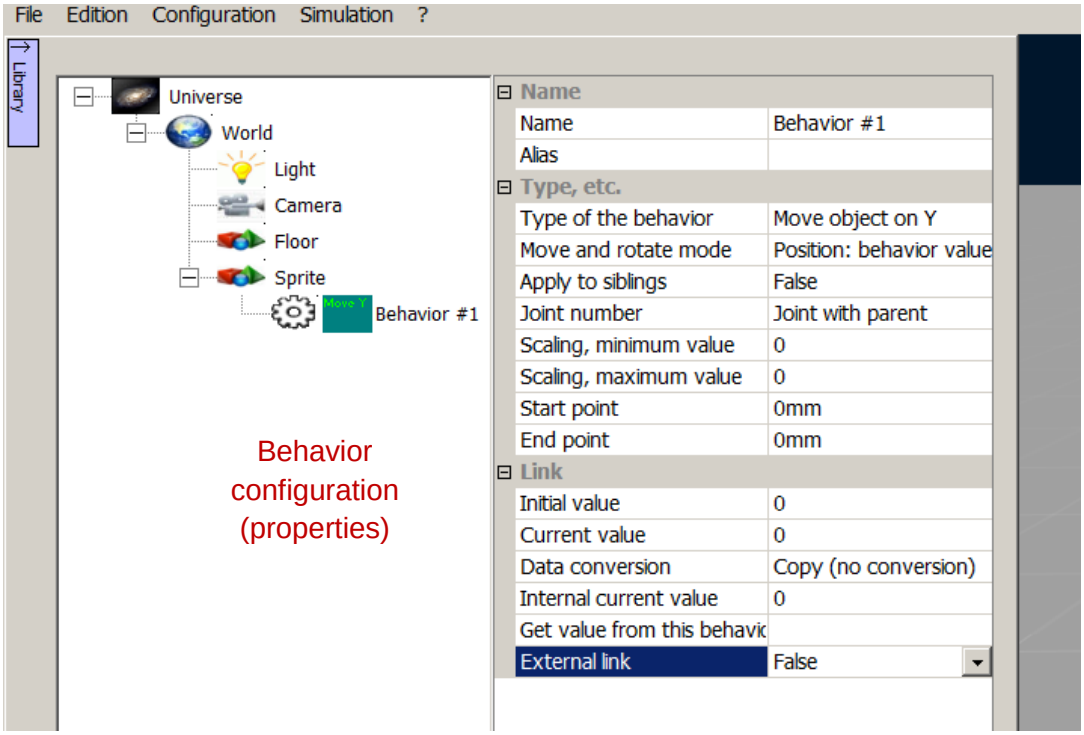
- The 6 categories outlined include the predefined behaviors in VIRTUAL UNIVERSE PRO



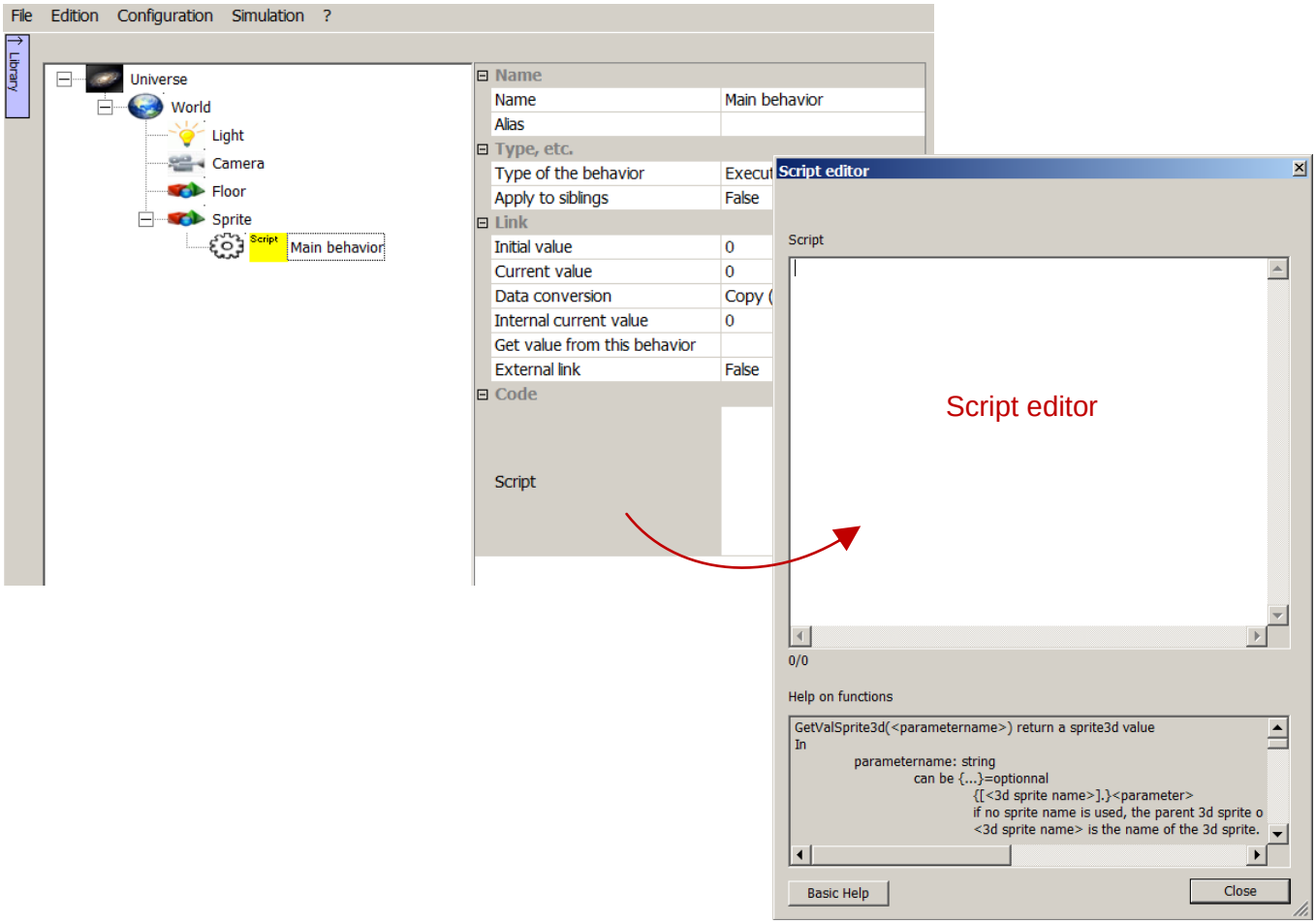
- The **Code** category providing access to Script behaviors.



- Once the behavior type selected, it is possible to configure the behavior by setting its properties



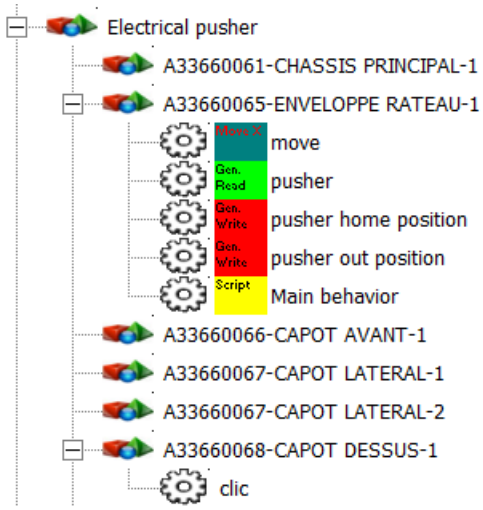
- With a Script behavior, the Scipt editor is opened via the code tab.

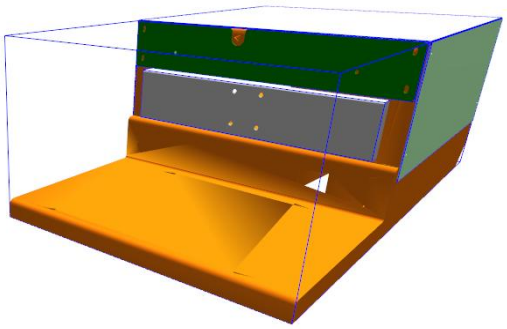


For more information on behavior types and properties, see Properties of Behaviors.

Example 1

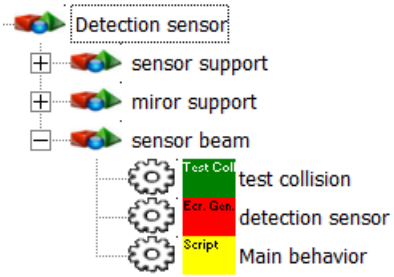
Take a look at the « Electrical Pusher » resource available in the VIRTUAL UNIVERSE PRO demo library

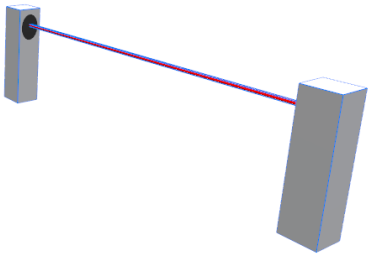




Example 2

Take a look at the « Detection sensor » resource available in the VIRTUAL UNIVERSE PRO demo library

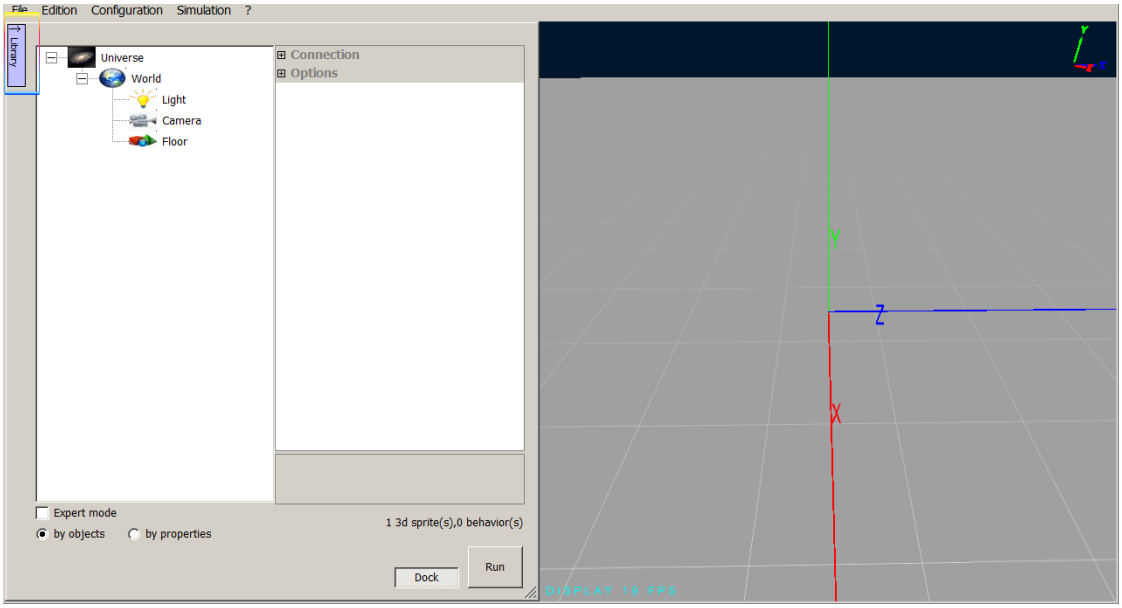




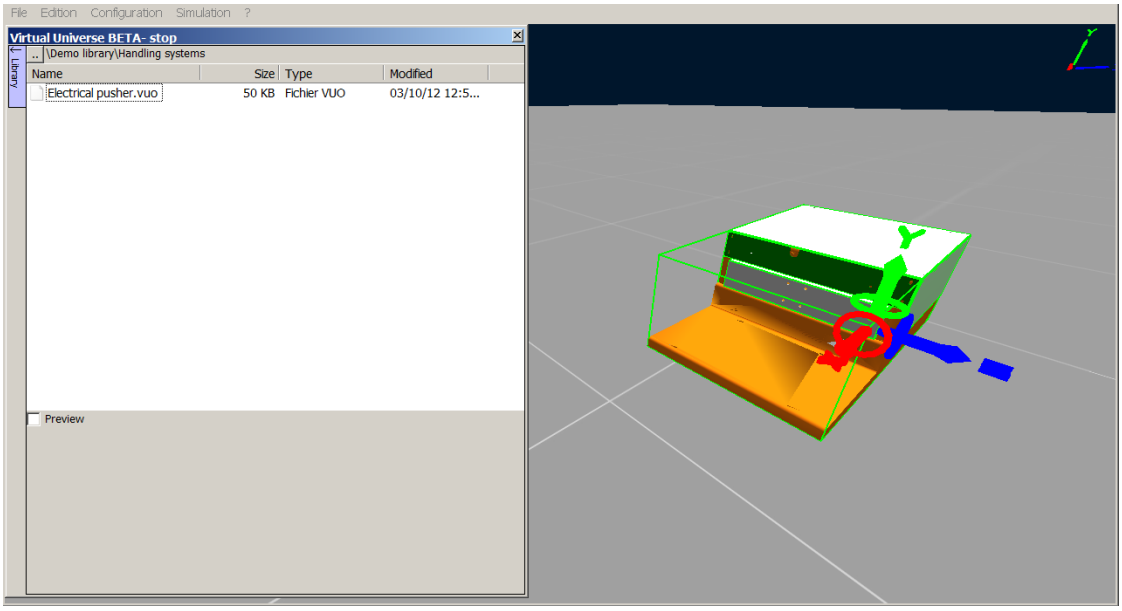
Exercise

The purpose of this short exercise is to add a new behavior to a 3D resource of the library (Electrical Pusher) and to modify its main script to include this new behavior.

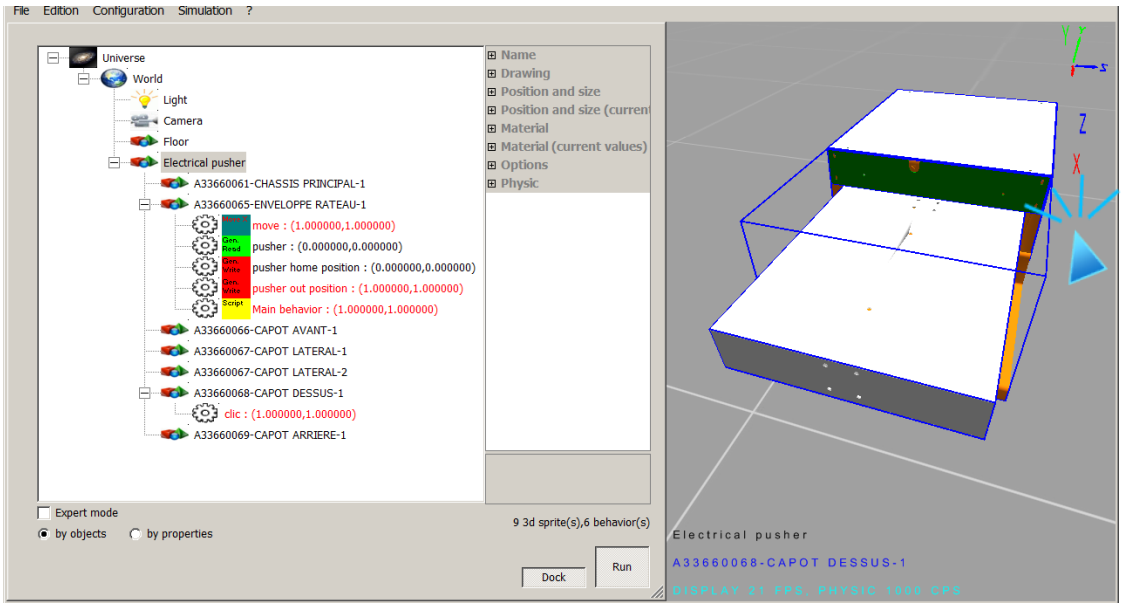
1. Open a void project and switch to the resources library window.



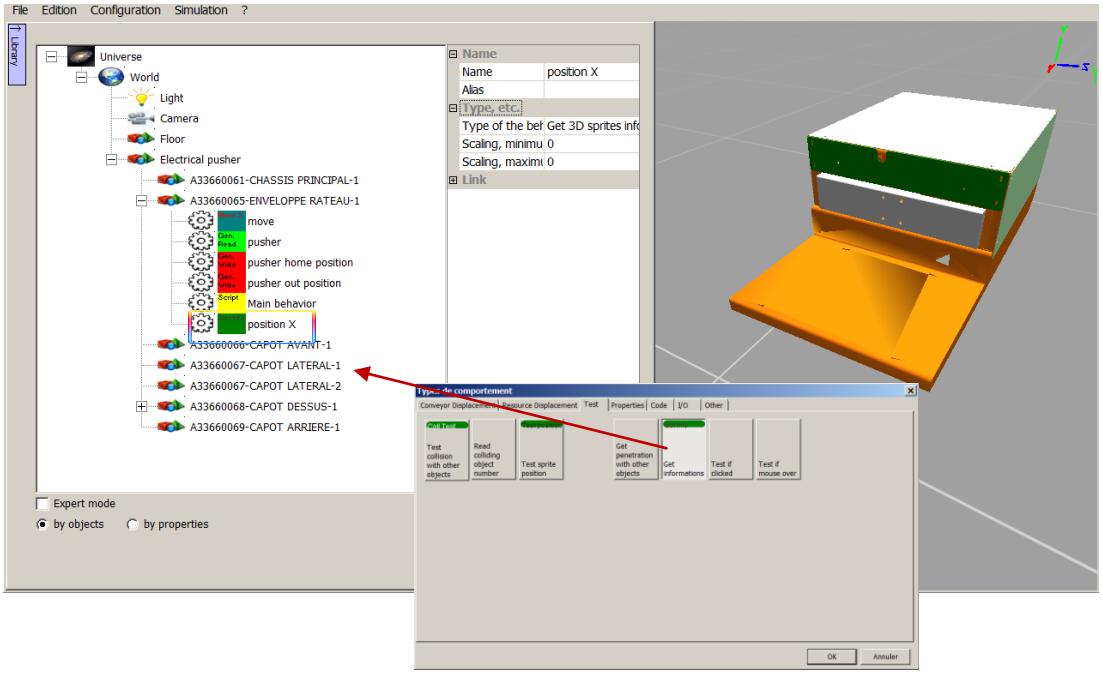
2. From the library, insert the “Electrical Pusher” resource inside the project, by slipping it with the mouse.



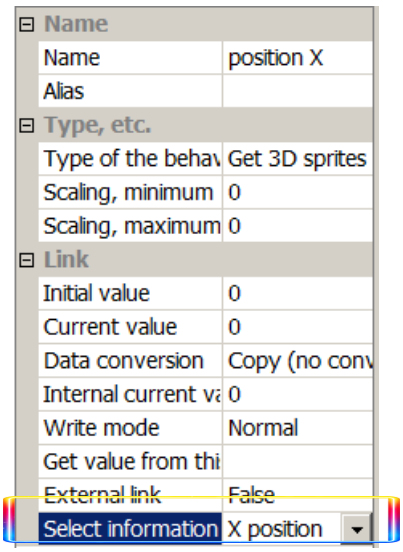
3. Get back to the set-up window and launch the simulation. During simulation, click on the top cover with the mouse to exit/enter the drawer. Watch the evolution of behaviors.



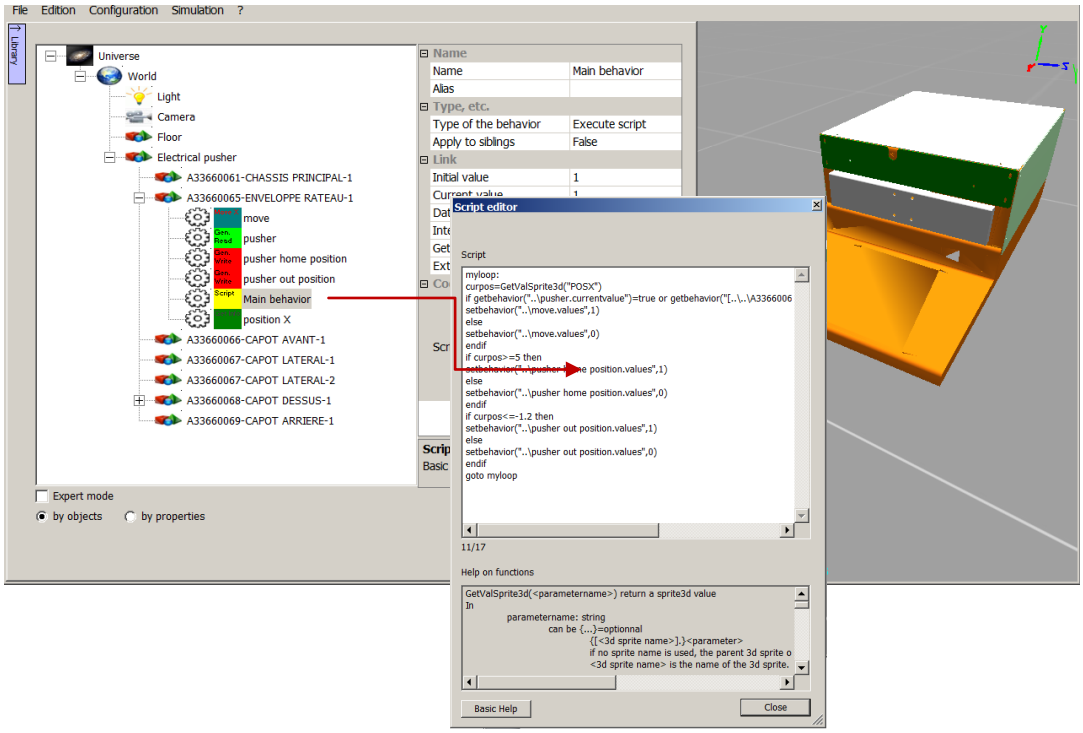
4. Stop the simulation. At the A33660065-ENVELOPPE sprite level, add a new “Get informations” behavior and rename it “position X”.



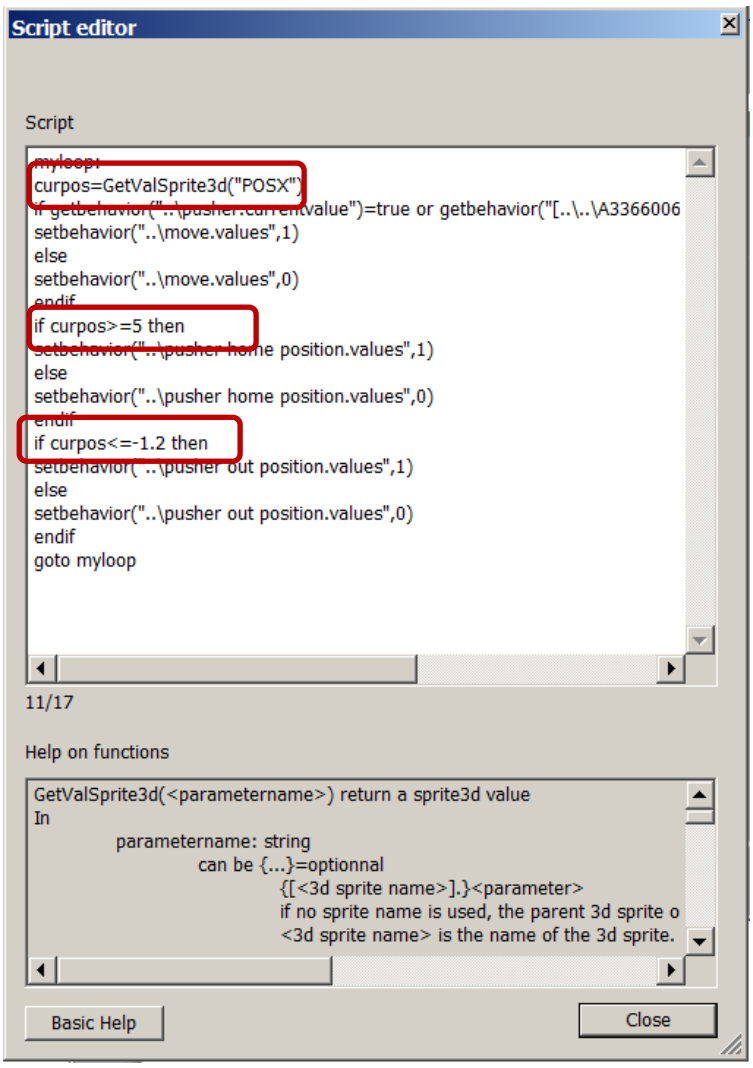
5. In the behavior properties, select "X position" in the "Select information to be read from the 3D sprite" field. This behavior then returns the position along the A33660065-ENVELOPPE sprite X axis (in the local coordinate system relative to the parent sprite “Electrical pusher”).



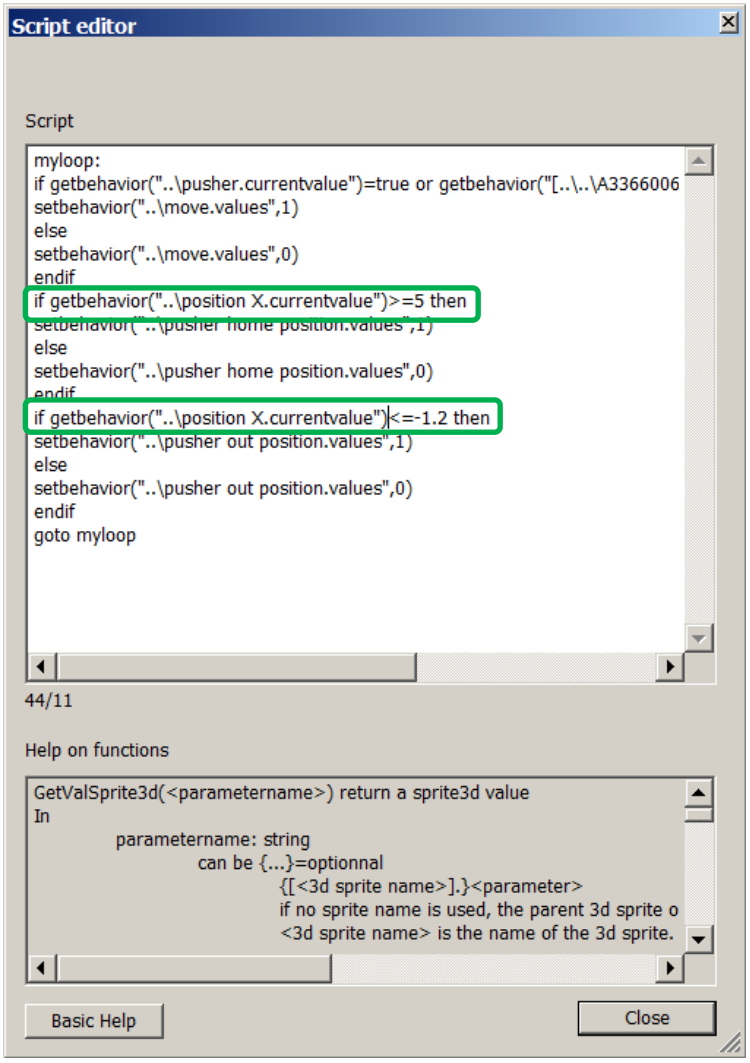
6. Click on the « Main behavior » behavior and open the script editor.



7. In this script, the modification is to use the new behavior “Position X” instead of the *curpos=GetValSprite3d("POSX")* function.

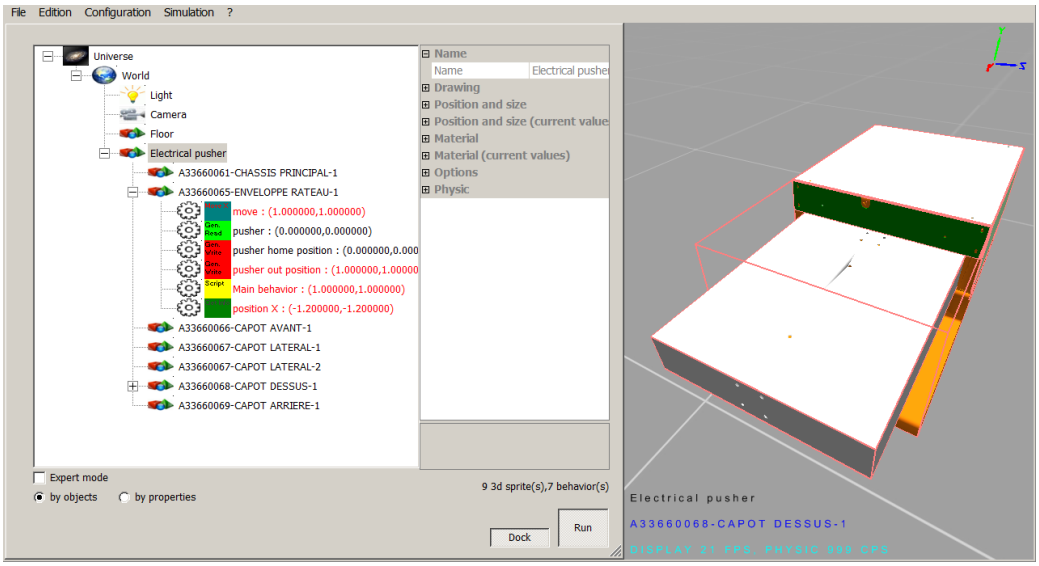

Script editor
Script
myloop:
curpos=GetValSprite3d("POSX")
if getbehavior("..\pusher.currentvalue")=true or getbehavior("[..\..\A3366006
setbehavior("..\move.values",1)
else
setbehavior("..\move.values",0)
endif
if curpos>=5 then
setbehavior("..\pusher home position.values",1)
else
setbehavior("..\pusher home position.values",0)
endif
if curpos<=-1.2 then
setbehavior("..\pusher out position.values",1)
else
setbehavior("..\pusher out position.values",0)
endif
goto myloop
11/17
Help on functions
GetValSprite3d(<parametername>) return a sprite3d value
In
parametername: string
can be {...}=optionnal
{[<3d sprite name>].}<parameter>
if no sprite name is used, the parent 3d sprite o
<3d sprite name> is the name of the 3d sprite.
Basic Help
Close

Old script



New script

8. Restart the simulation with this new script and note that the pusher behavior remains unchanged.



Define motion profiles with the Motion Assistant

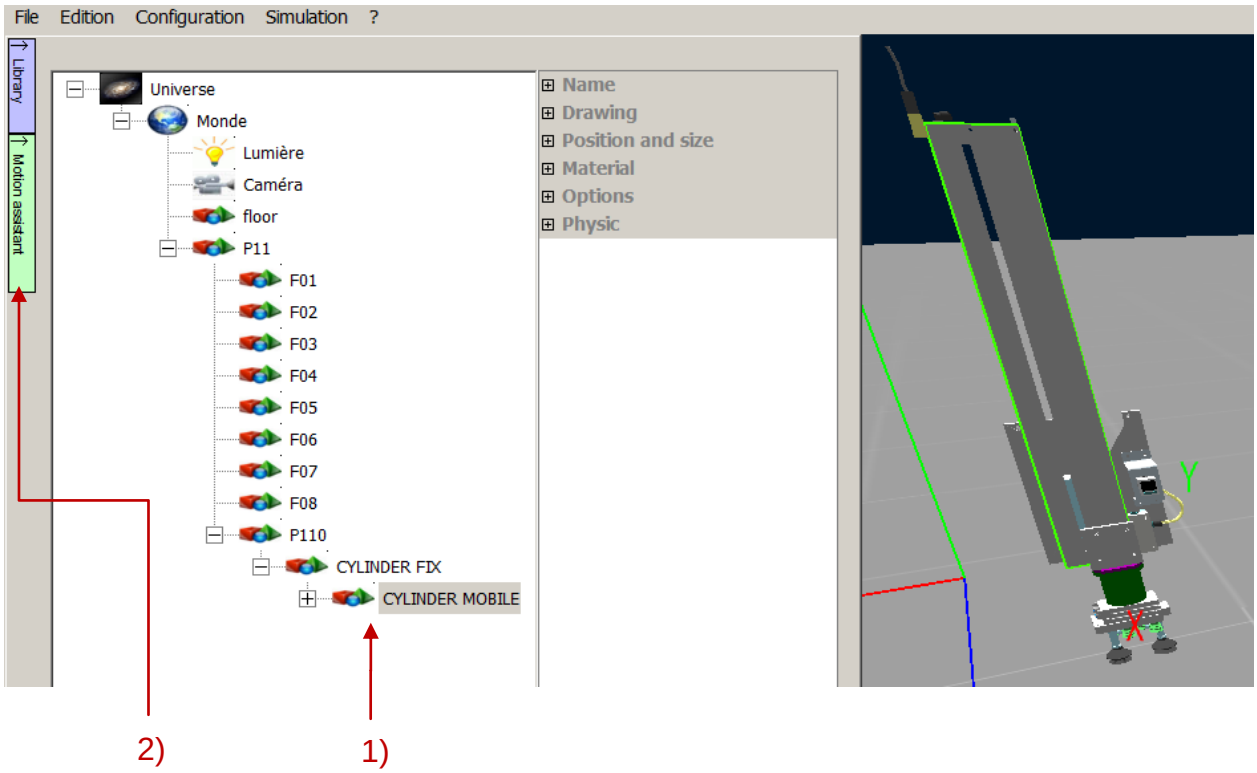
The Motion Assistant enables to easily define the motion profiles of a 3D mechanical resource or system, by creating and adjusting automatically the behaviors needed for simulation of these motion profiles.

A mechanical resource can be, for instance, a cylinder or an axis driven by a set {motor + variable-frequency drive}. The Motion Assistant enables, as appropriate, to control a resource in different ways:

- In time
- In speed
- In position

Moving as a function of time

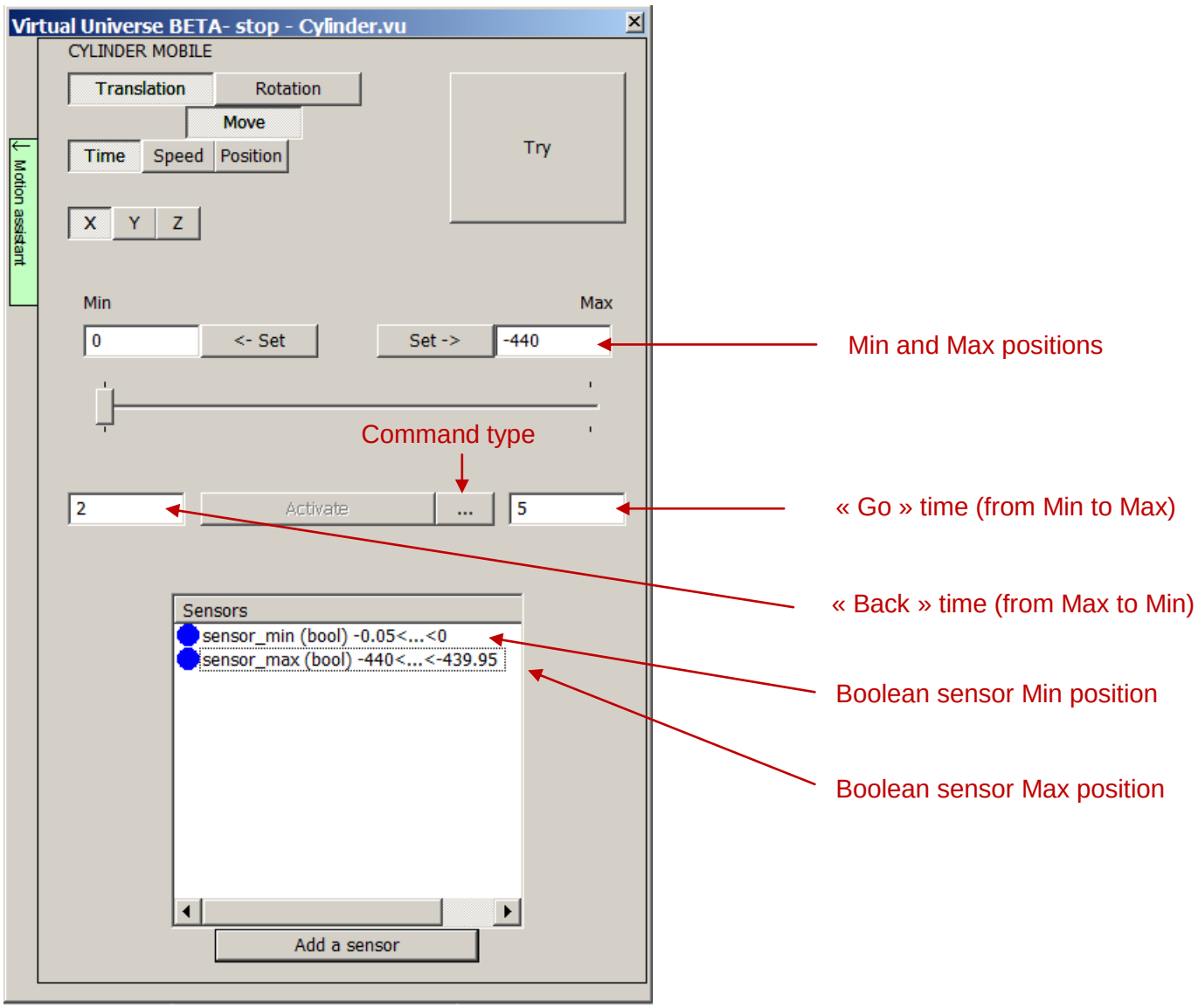
This is typically the motion profile used in the case of a cylinder driven in time by an external controller.



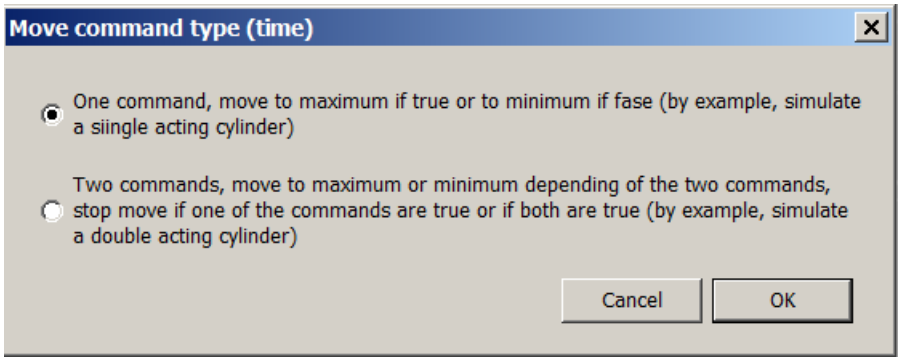
- First, select (in the project tree) the mobile part of the mechanical resource that you wish to set in motion (this is the cylinder rod in the example above).

- Then click the green tab "Motion assistant" to the left of the tree.

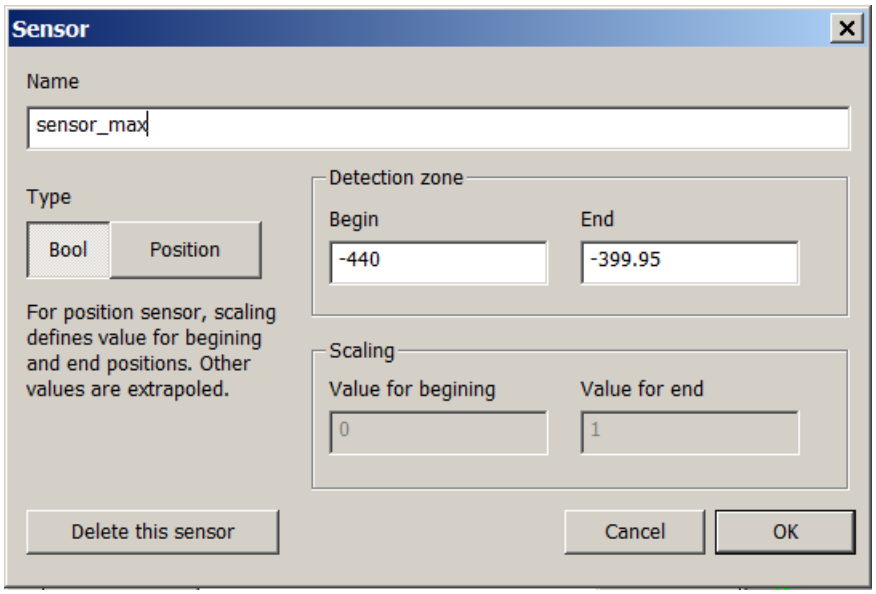
- The Motion assistant window appears on screen.



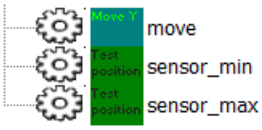
- Set the Min and Max stops, either manually or by moving the resource directly in the rendering window (these stops will be reached by the resource).
- Set the time to "go" and to "return" of the mechanical resource (cylinder rod).
- Select the Move command type.



- It is also possible to add Boolean sensors (AON sensors, all or nothing), returning 1 if the mechanical resource is within the detection range of the sensor and 0 otherwise. The following example illustrates the creation of a Boolean sensor (sensor_max) located around the maximum position.



- The generated behaviors are:

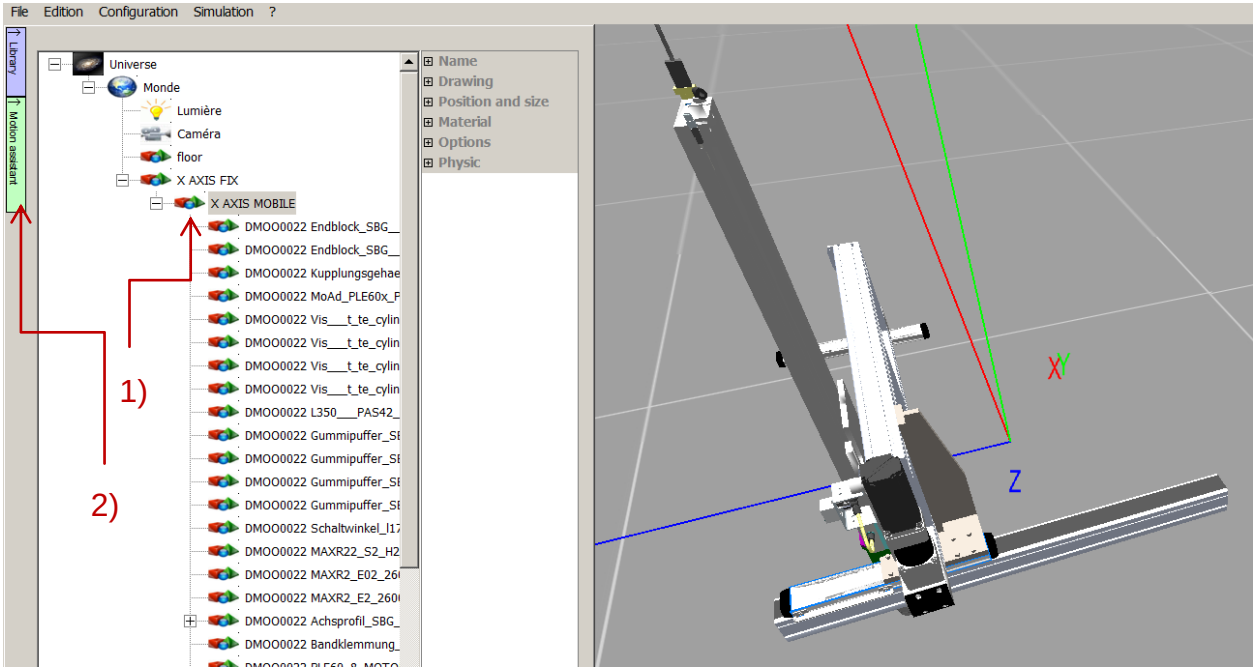


Behavior “move”, cylinder actuator, **3D emulator input**
Boolean sensor “min”, **3D emulator output**
Boolean sensor “max”, **3D emulator output**

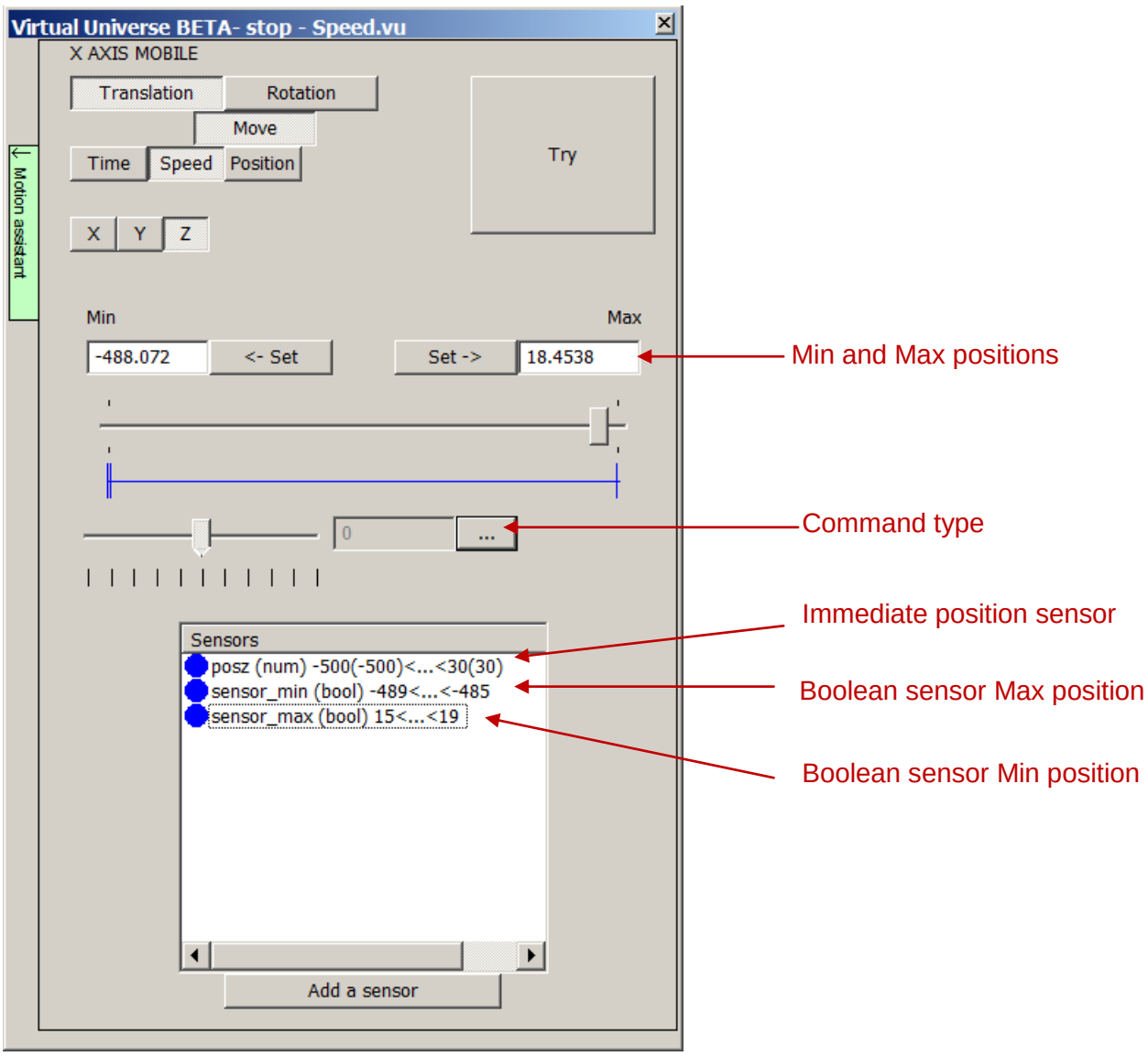
  - A value of 0 for the "move" behavior will result in a "return" displacement and a value of 1 to a "go" displacement.
  - Min and Max have a value of 1 when the resource is located within the sensor range, 0 otherwise.

Moving as a function of speed

This is typically the motion profile used in the case of an axis, driven by a set {motor + variable-frequency drive} receiving speed setpoints from a programmable logic controller, external to the 3D emulator.

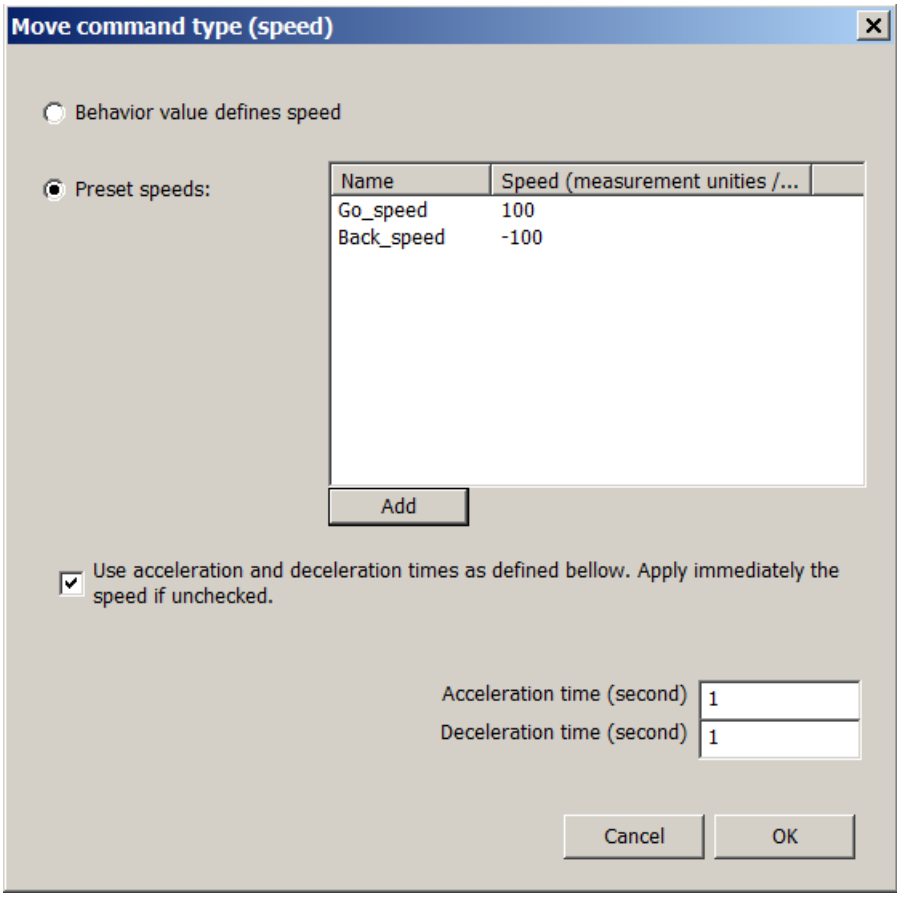


- First, select (in the project tree) the mobile part of the mechanical resource that you wish to set in motion (this is the cylinder rod in the example above).
- Then click the green tab "Motion assistant" to the left of the tree.
- The Motion assistant window appears on screen.

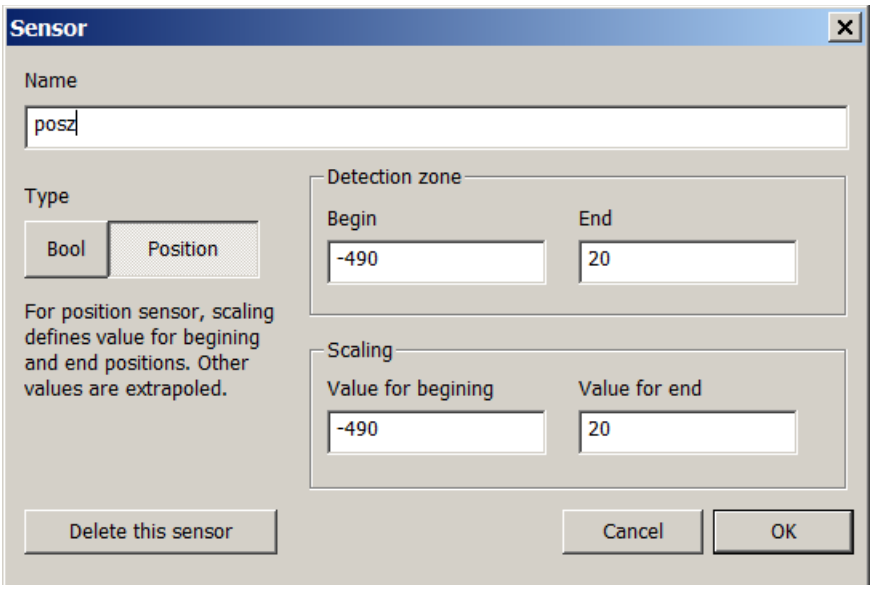


- Set the Min and Max stops, either manually or by moving the resource directly in the rendering window (these stops will not be reached by the resource).

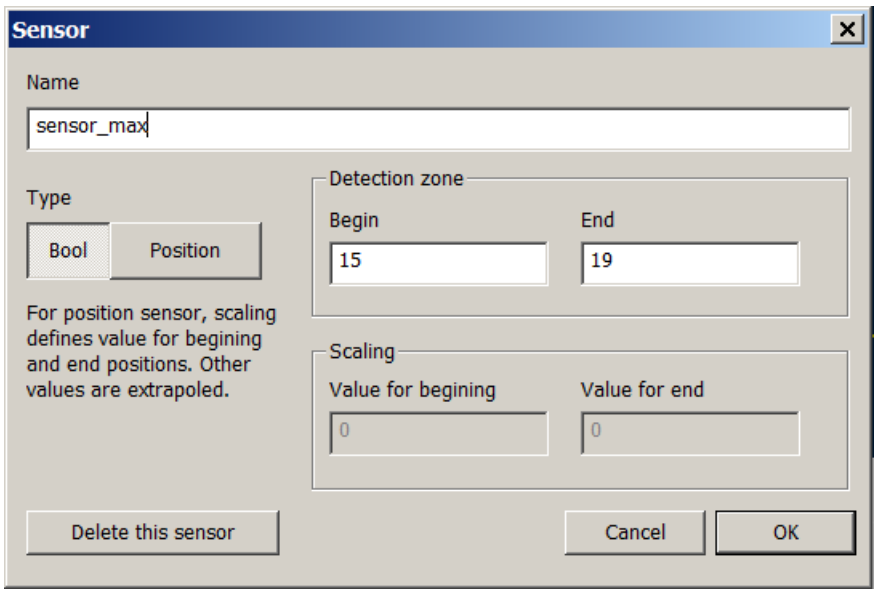
- Select the Move command type according to the command you wish to apply (preset speed, preset acceleration/deceleration time). The following image illustrates how to create a movement with two preset speeds (100 and -100 mm/s) and a profile of acceleration/deceleration (one second to reach the applied speed).



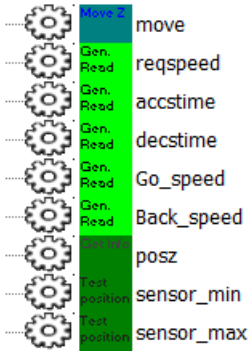
- The addition of an "immediate" position sensor (returning directly the resource position along the motion axis) constitutes a position feedback. The following image illustrates the creation of such a sensor “posz”.



- It is also possible to add Boolean sensors (AON sensors, all or nothing), returning 1 if the mechanical resource is within the detection range of the sensor and 0 otherwise. The following example illustrates the creation of a Boolean sensor (sensor_max) located around the maximum position which can, for example, model a security sensor.



- The generated behaviors are :

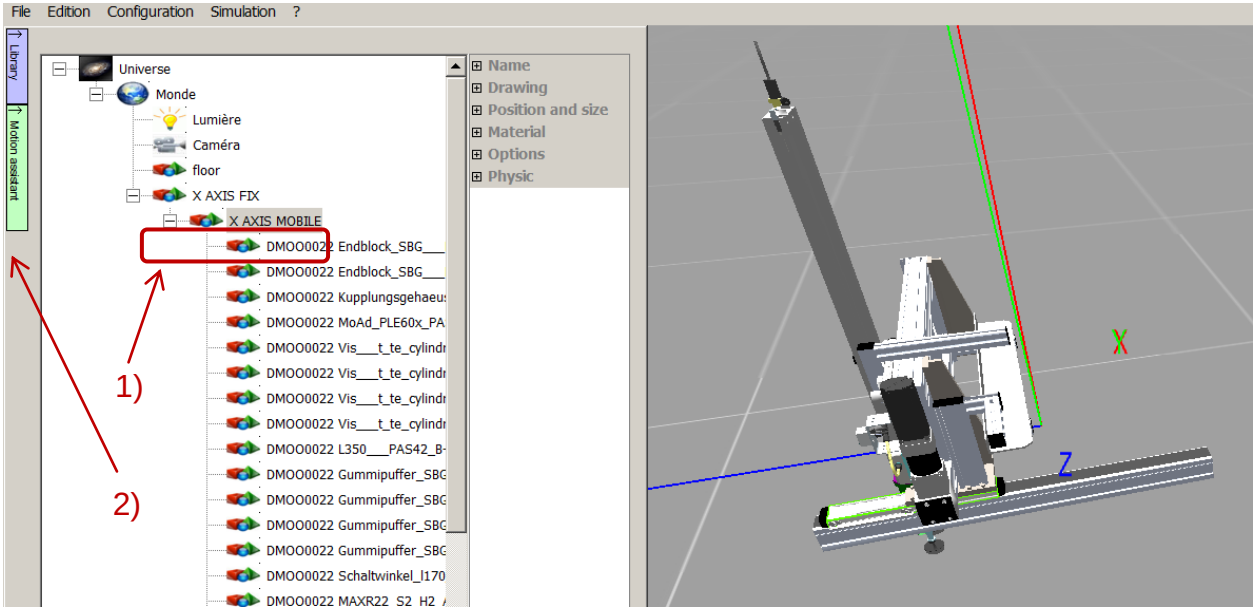


The behavior value sets the speed
Speed setpoint, **3D emulator input**
Acceleration time, **3D emulator input**
Deceleration time, **3D emulator input**
Preset speed 1, **3D emulator input**
Preset speed 2, **3D emulator input**
“Immediate” position sensor, **3D emulator output**
Boolean sensor min, **3D emulator output**
Boolean sensor max, **3D emulator output**

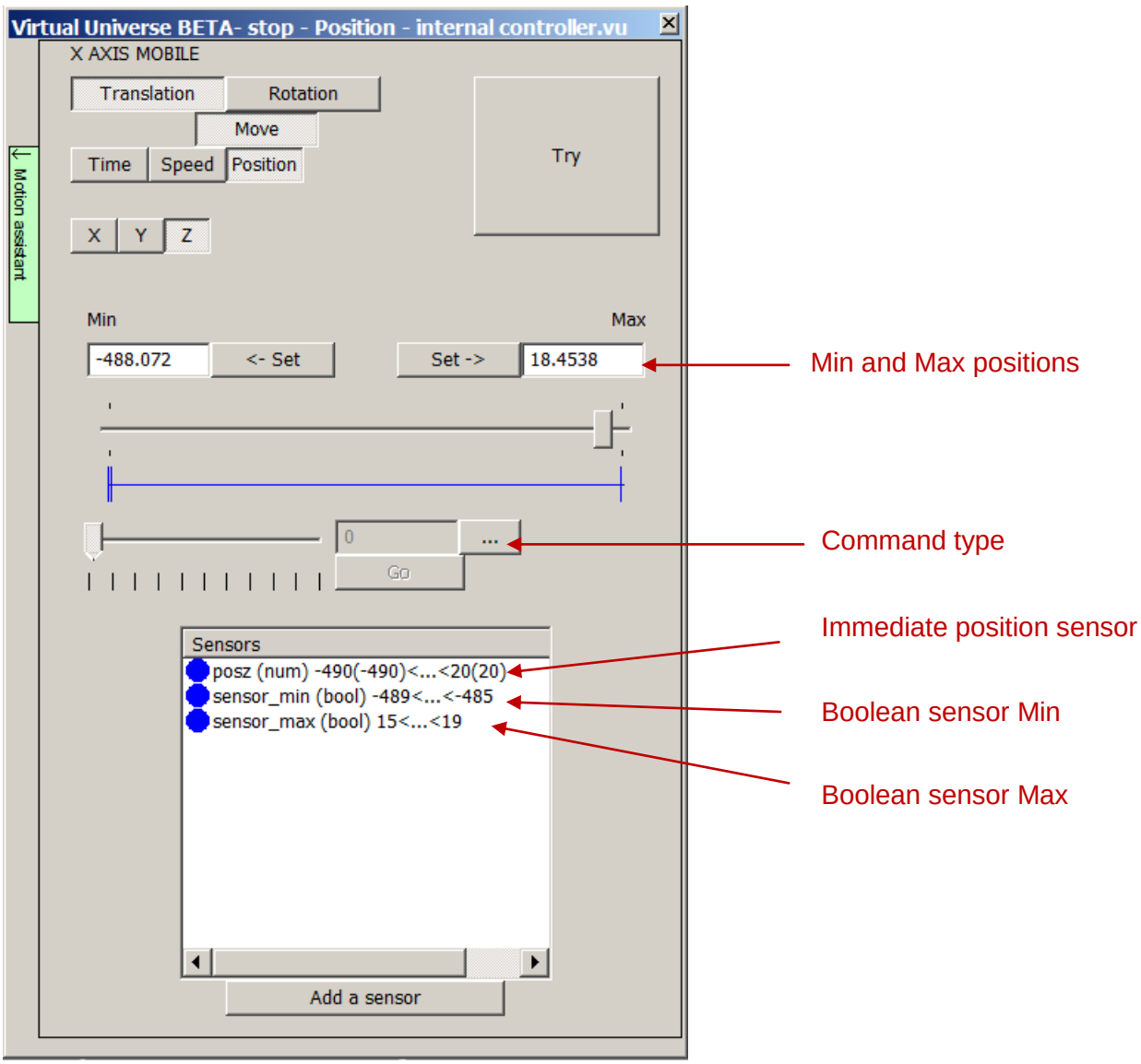
  - A value of 1 for a "Preset speed" behavior activates the preset speed, forcing the speed setpoint to its value.
  - Reqspeed, accstime, decstime are only created if the user defines a profile of accel/decel profile. If no profile is defined, the command part will directly write "move" that will be an input for the 3D emulator

Moving as a function of an “immediate” position

This is typically the motion profile used in the case of an axis, driven by a set {motor + variable-frequency drive} receiving position setpoints from an axis controller, external to the 3D emulator.

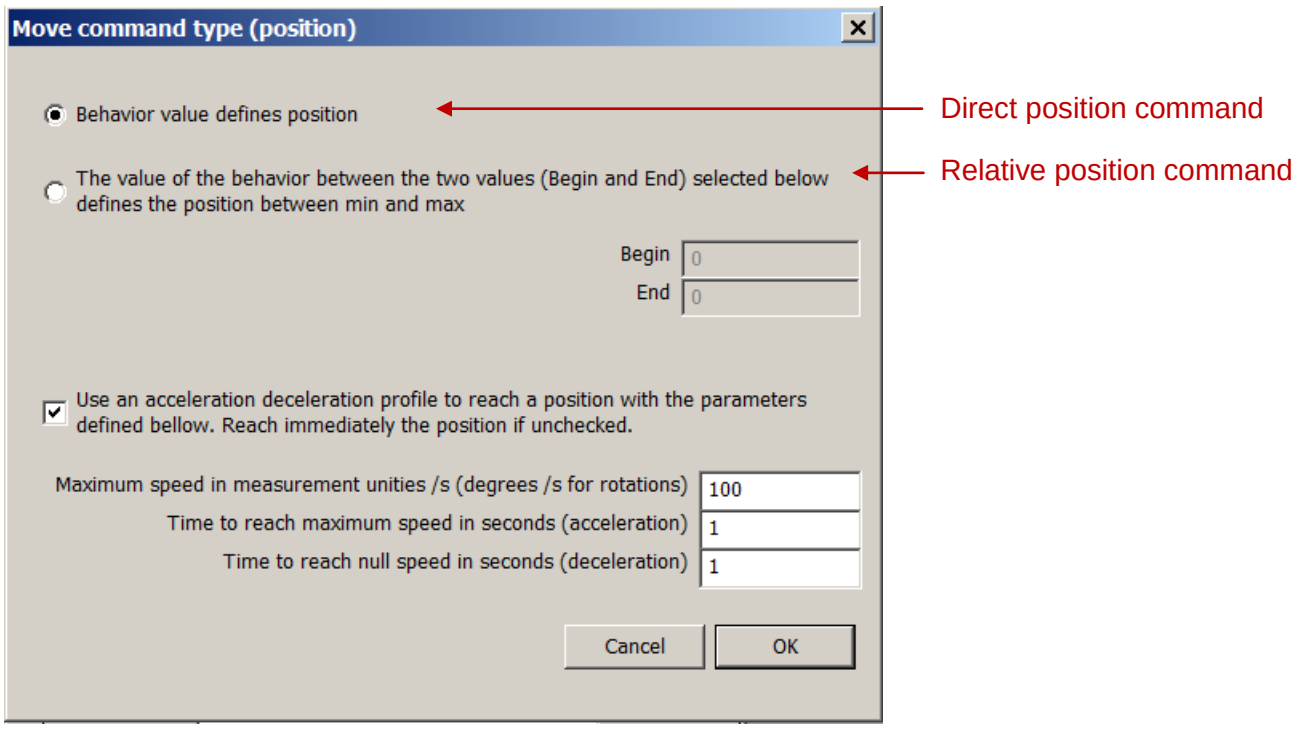


- First, select (in the project tree) the mobile part of the mechanical resource that you wish to set in motion.
- Then click the green tab "Motion assistant" to the left of the tree.
- The Motion assistant window appears on screen.

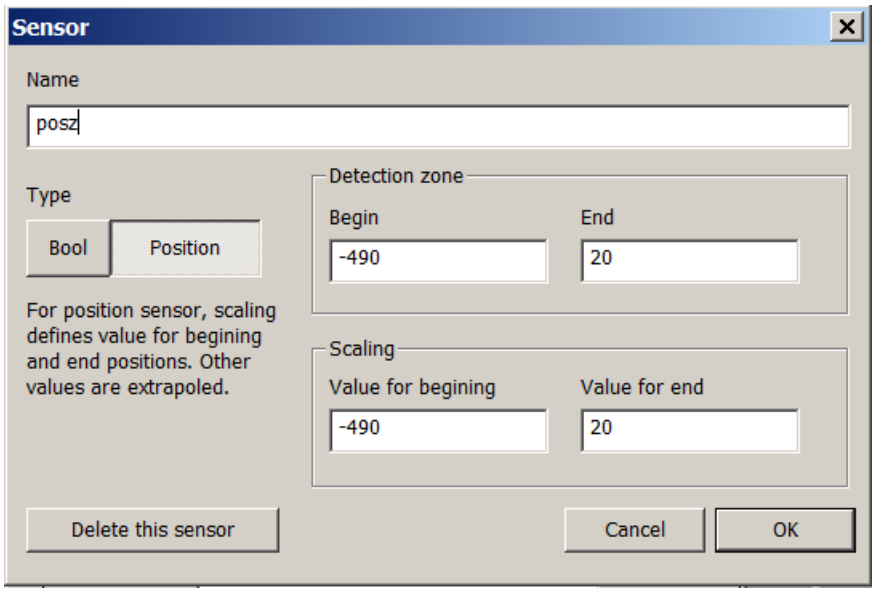


- Set the Min and Max stops, either manually or by moving the resource directly in the rendering window (these stops will not be reached by the resource).

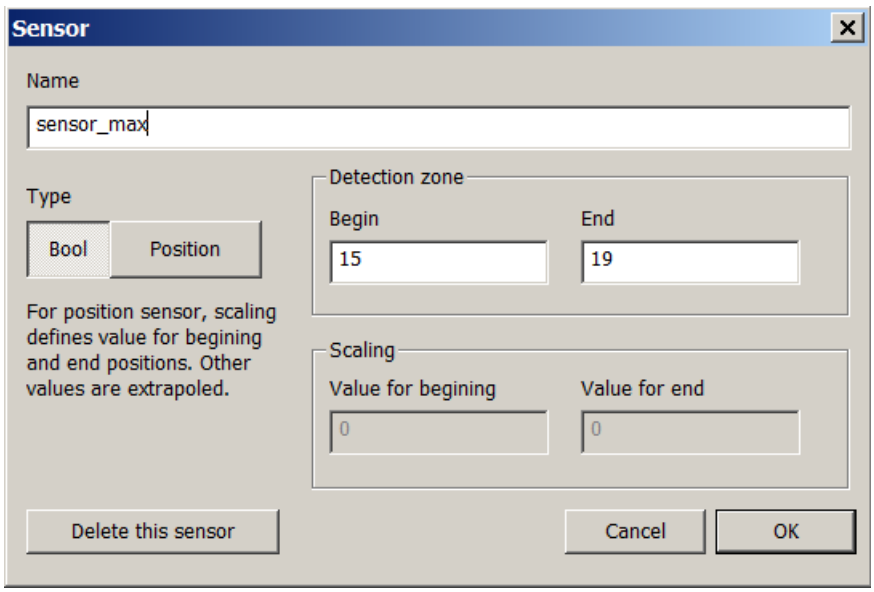
- Select the command type (direct or relative).



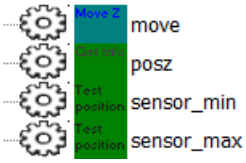
- The addition of an "immediate" position sensor (returning directly the resource position along the motion axis) constitutes a position feedback. The following image illustrates the creation of such a sensor “posz”.



- It is also possible to add Boolean sensors (AON sensors, all or nothing), returning 1 if the mechanical resource is within the detection range of the sensor and 0 otherwise. The following example illustrates the creation of a Boolean sensor (sensor_max) located around the maximum position which can, for example, model a security sensor.



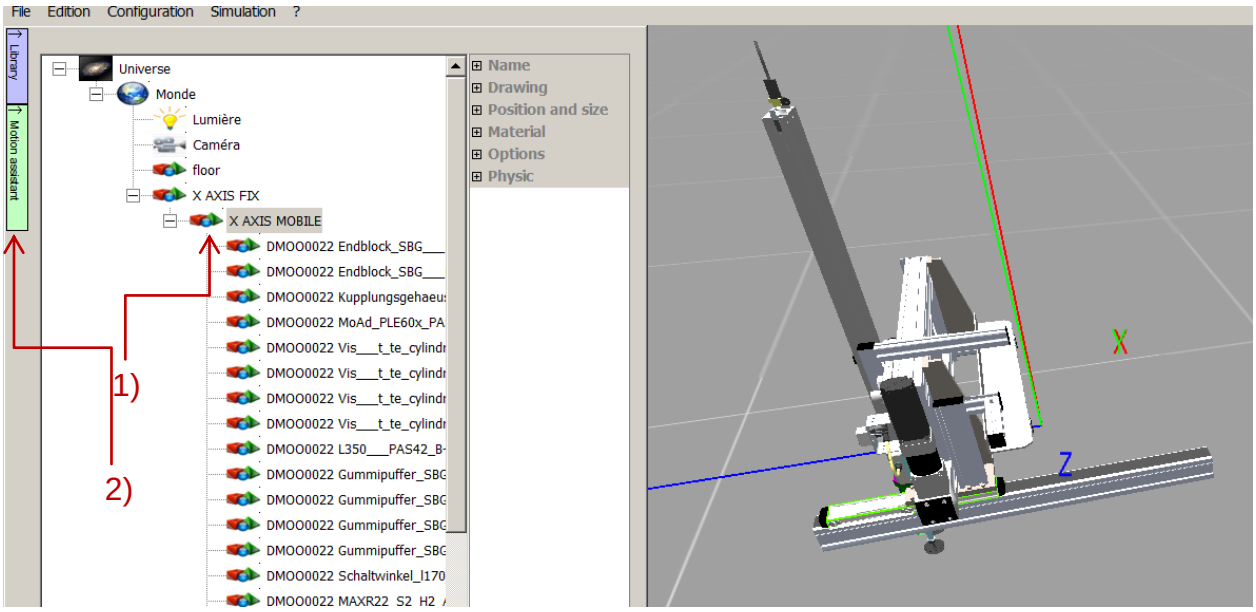
- The generated behaviors are :



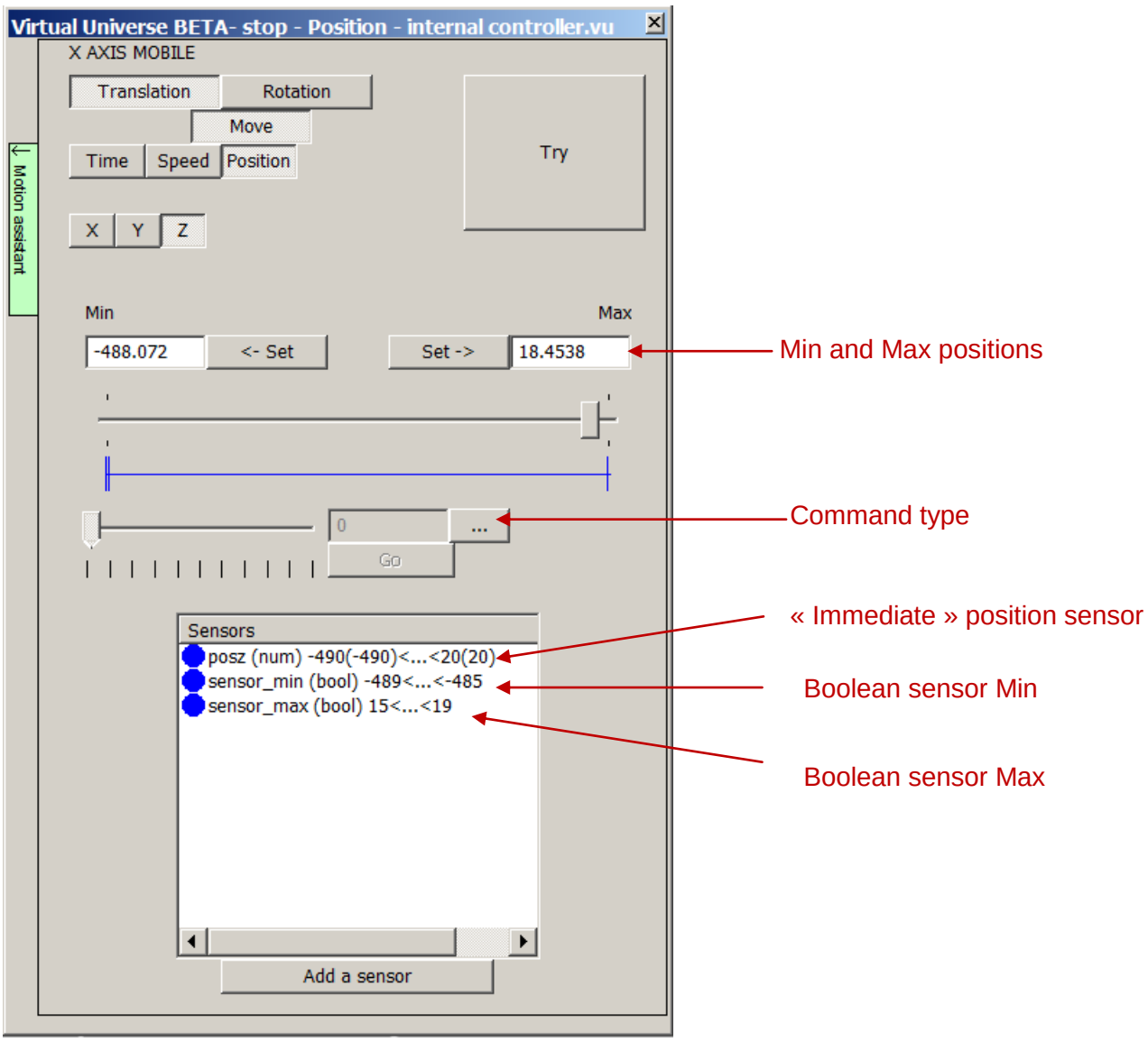
Direct writing of the resource position, **3D emulator input**
“immediate” position sensor **3D emulator output**
Boolean sensor “min”, **3D emulator output**
Boolean sensor “max”, **3D emulator output**

Moving as a function of a position with acceleration and deceleration

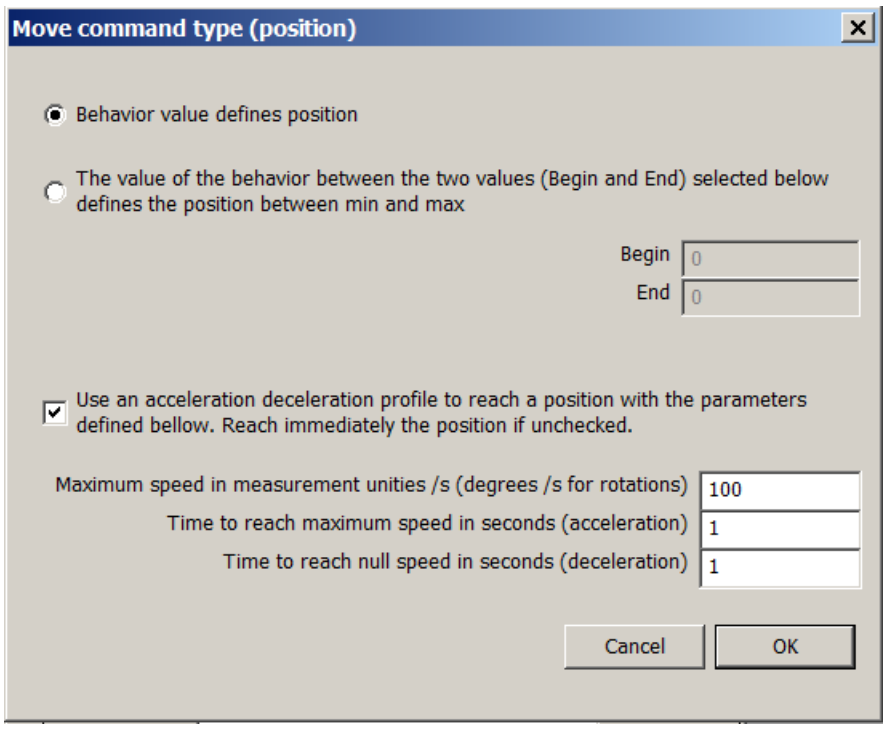
This is typically the motion profile used in the case of an axis, driven by a set {motor + variable-frequency drive} receiving position setpoints from an axis controller, internal to the 3D emulator.



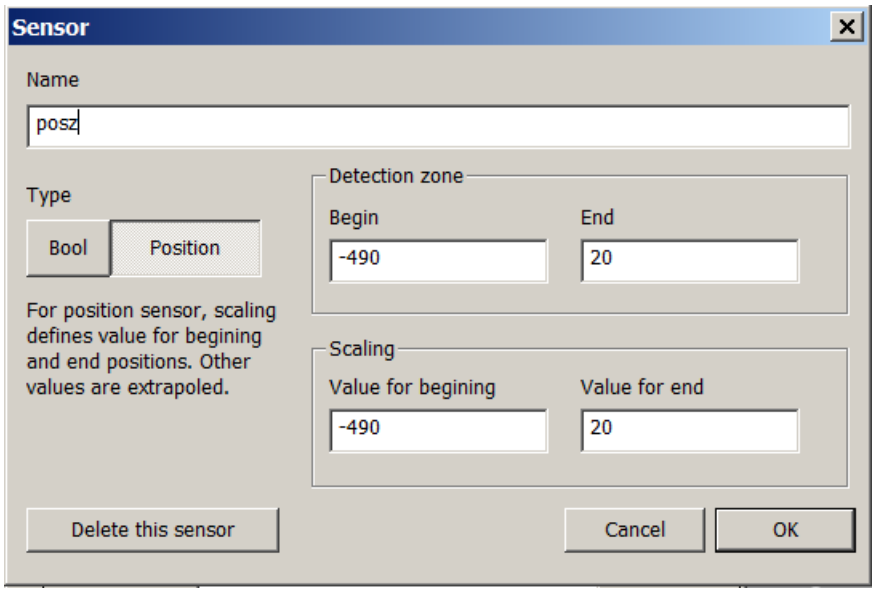
- First, select (in the project tree) the mobile part of the mechanical resource that you wish to set in motion.
- Then click the green tab "Motion assistant" to the left of the tree.
- The Motion assistant window appears on screen.



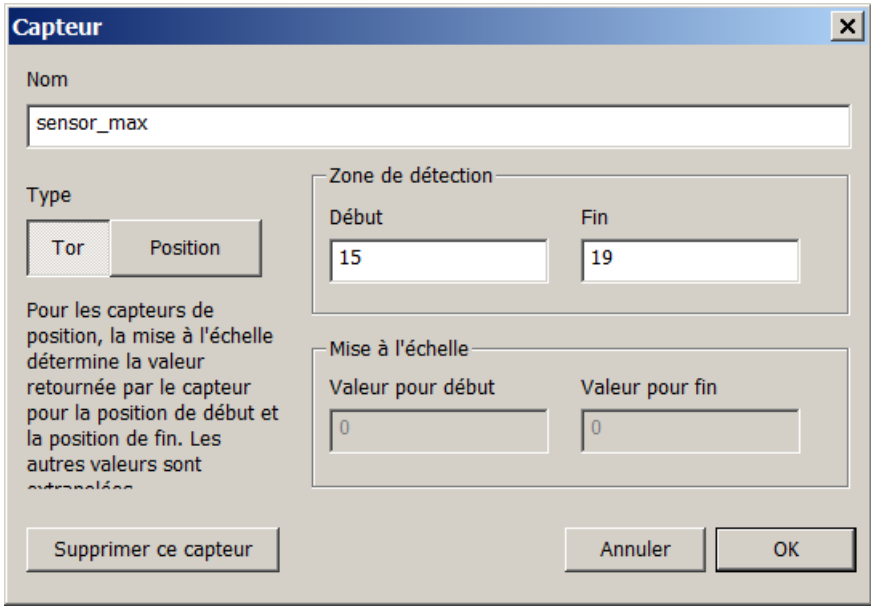
- Set the Min and Max stops, either manually or by moving the resource directly in the rendering window (these stops will not be reached by the resource).
- Select the command type (direct or relative).
- Select the option « Use an acceleration/deceleration profile to reach a position… » and define the profile parameters.



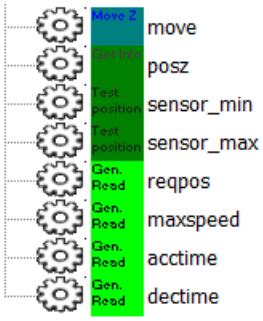
- The addition of an "immediate" position sensor (returning directly the resource position along the motion axis) constitutes a position feedback. The following image illustrates the creation of such a sensor “posz”.



- It is also possible to add Boolean sensors (AON sensors, all or nothing), returning 1 if the mechanical resource is within the detection range of the sensor and 0 otherwise. The following example illustrates the creation of a Boolean sensor (sensor_max) located around the maximum position which can, for example, model a security sensor.



- The generated behaviors are:

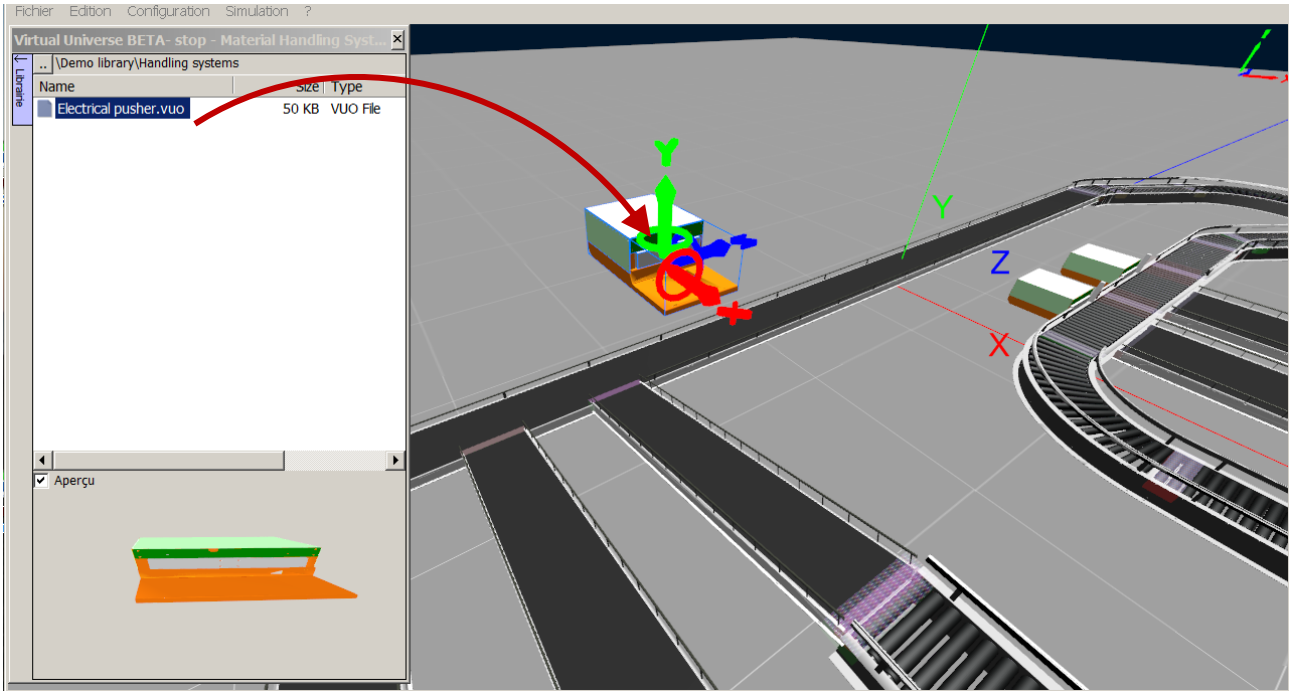


Direct writing of the resource position, **internal value**
“immediate” position sensor, **3D emulator output**
Boolean min sensor, **3D emulator output**
Boolean max sensor, **3D emulator output**
Position setpoint, **3D emulator input**
Maximum speed in units/s (mm or degrees), **3D emulator input**
Time to reach maximum speed, **3D emulator input**
Time to reach 0 speed, **3D emulator input**

Use a library of smart resources

VIRTUAL UNIVERSE PRO provides access to a library of 3D resources enabling to capitalize on your own smart resources in order to later use them for building new 3D emulator projects.

As example, VIRTUAL UNIVERSE PRO provides a first demonstration library (called Demo library) including some smart 3D resources (conveyors, electrical pusher, sensors, electrical enclosure, source, sink...) that should help to train you and design your own 3D smart resources.

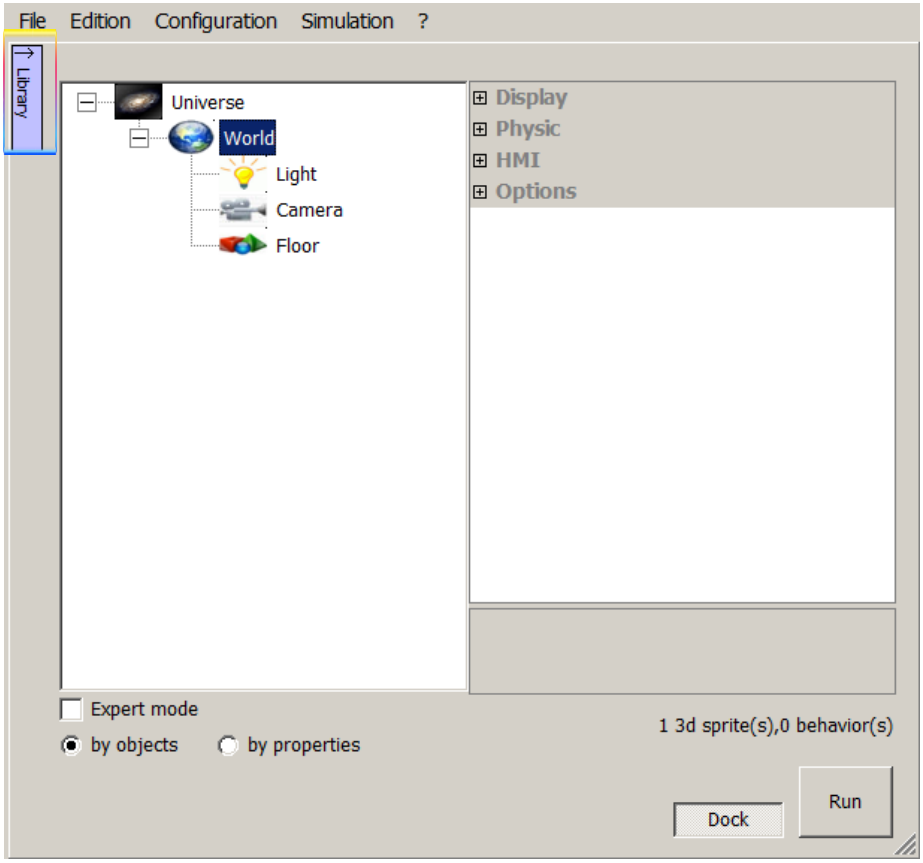


The VIRTUAL UNIVERSE PRO library is stored in the "library" folder, itself located in the VIRTUAL UNIVERSE PRO Industry installation directory:

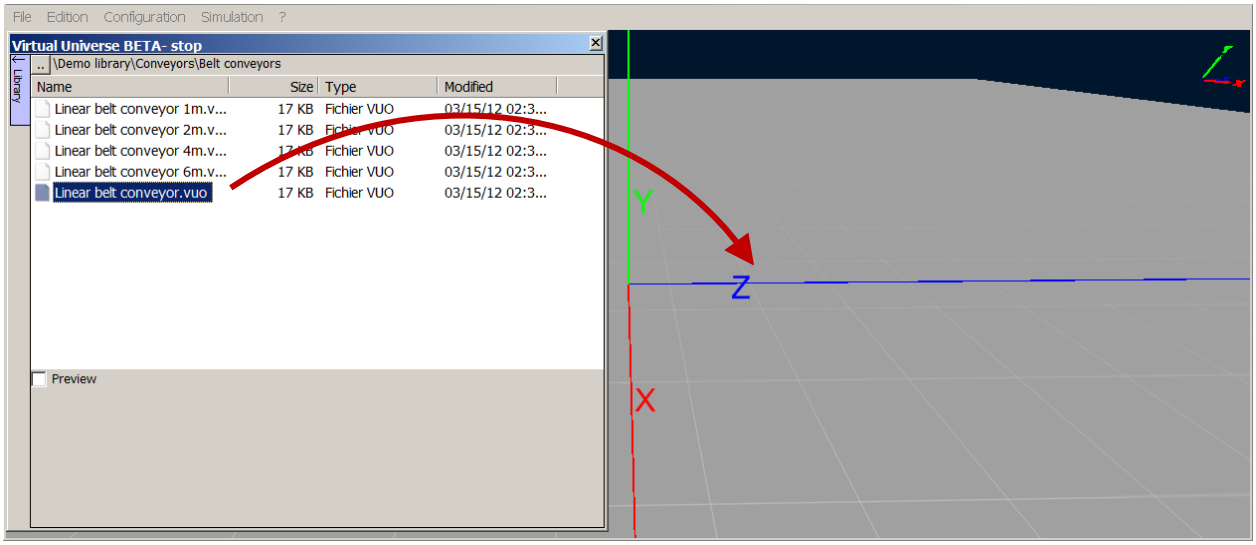
VIRTUAL UNIVERSE PRO Industry\VUI 1.113\library

Import a smart resource from library

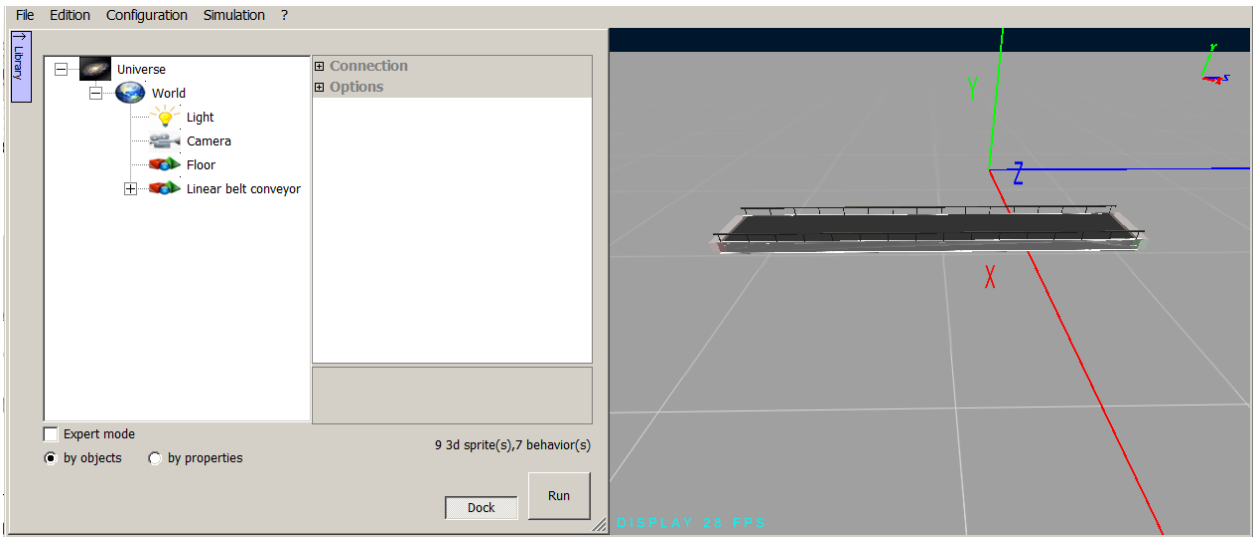
- For accessing to a 3D resource already available in the library, click on the blue tab "Library" on the left of the set-up window. A browser will open, allowing you to access all 3D resources available in the library.



- In the library, select the resource to add to the project and drag and drop it from the library onto the 3D rendering window.



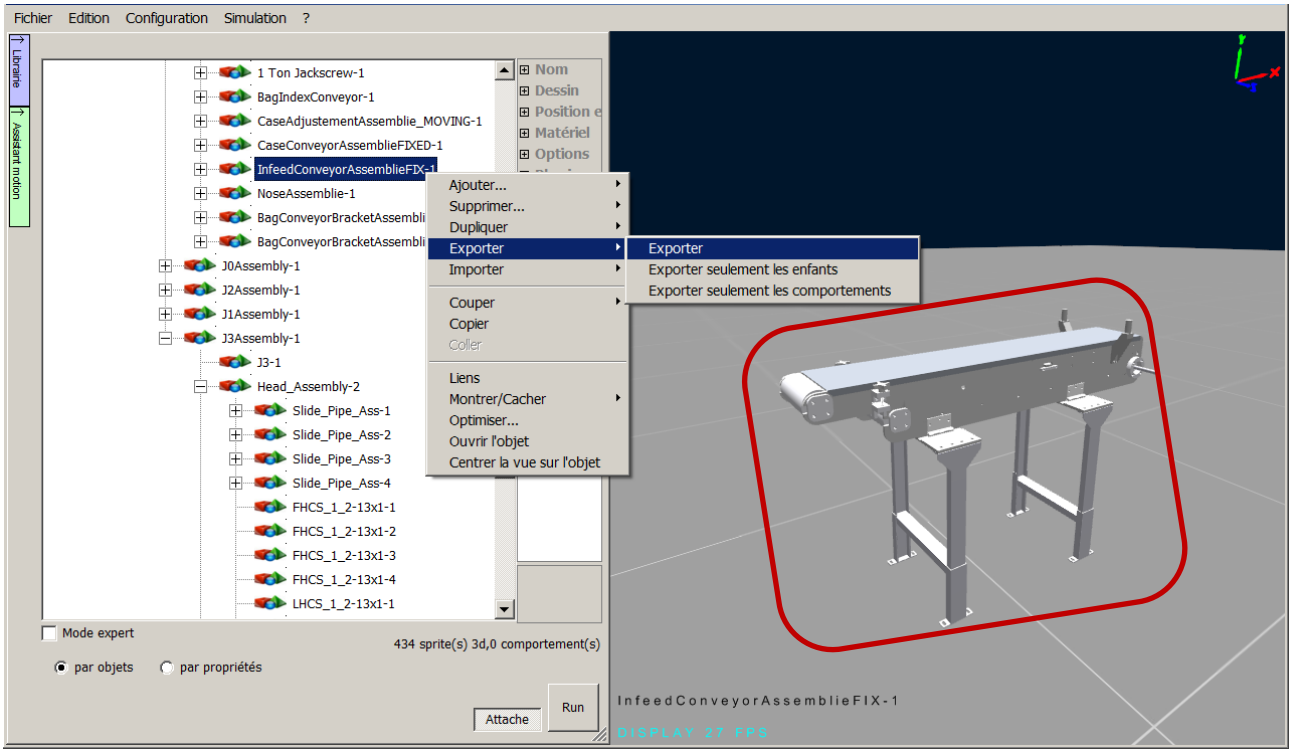
- The 3D resource is immediately added to the project, and visible in the project tree.



Export a smart resource into the library

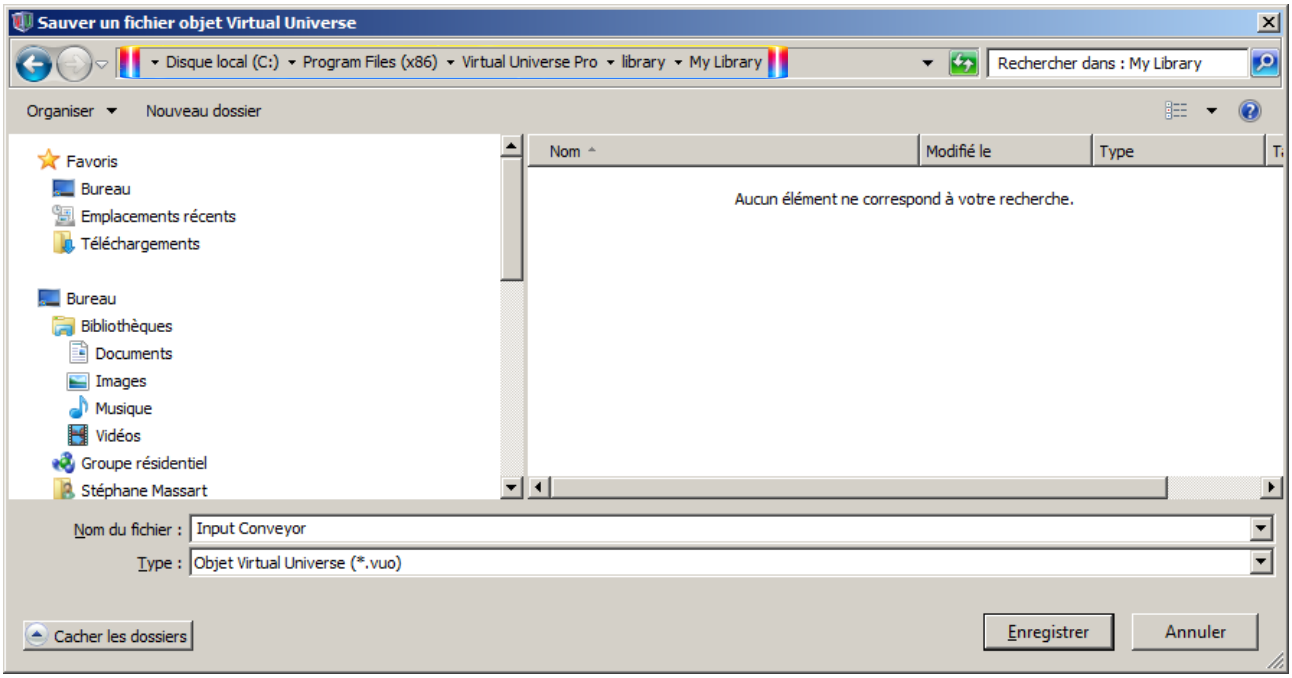
VIRTUAL UNIVERSE PRO enables you to capitalize on your own smart resources in order to later use them for building new 3D emulator projects.

- In the project tree, select the 3D resource to add to the library and right click. Select Export/Export.

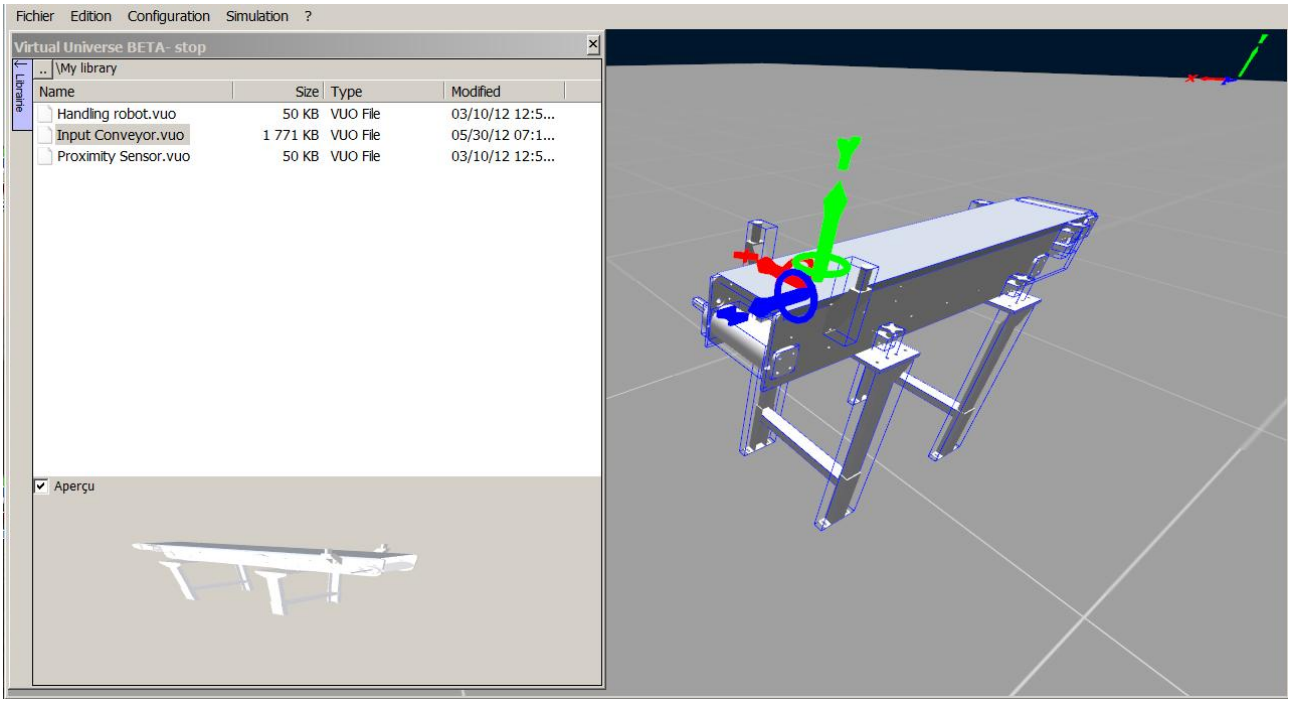


A Windows Explorer opens. In the VIRTUAL UNIVERSE PRO installation directory, select the "library" folder. This folder is not write-protected, you can organize it the way you want (adding new directories).

- After having selected the backup path and named your resource, click "Save".



- The new 3D resource is then available in the VIRTUAL UNIVERSE PRO library, ready for reuse.



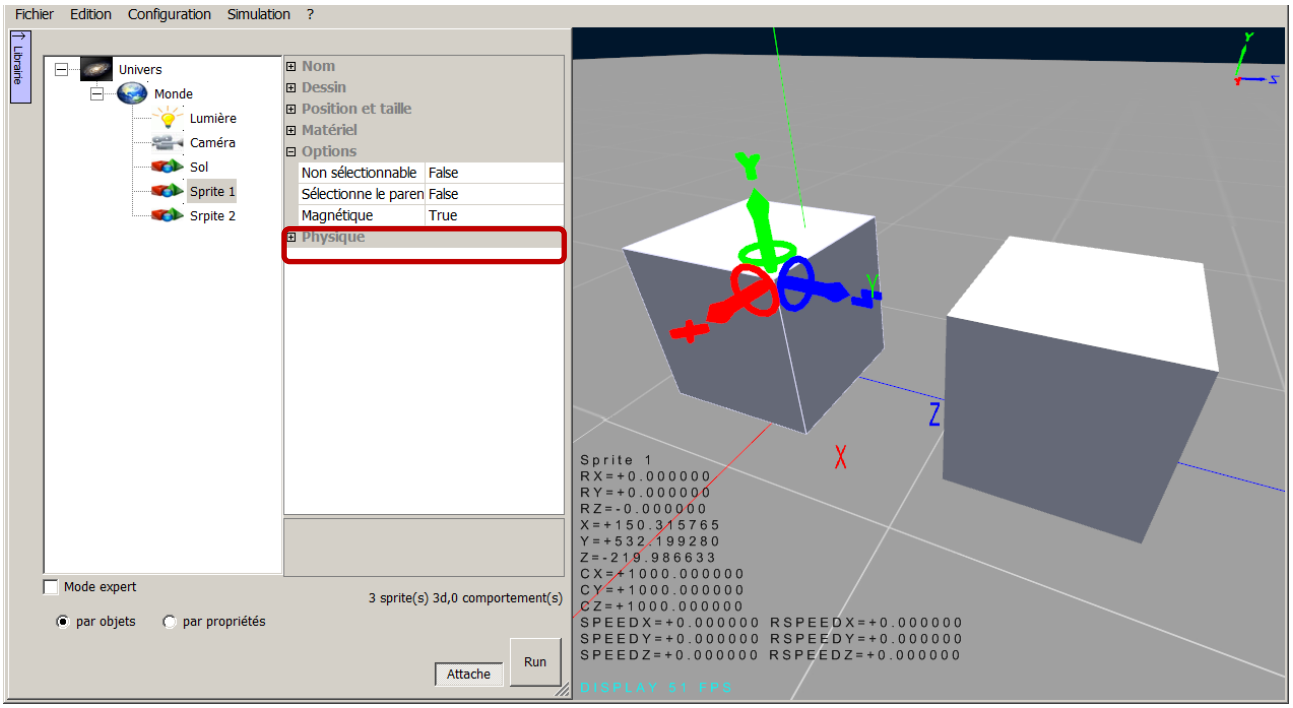
Quickly connect 3D resources with the « Magnetic » option

VIRTUAL UNIVERSE PRO features a "Magnetic" function which enables to easily connect 3D resources by using the mouse.

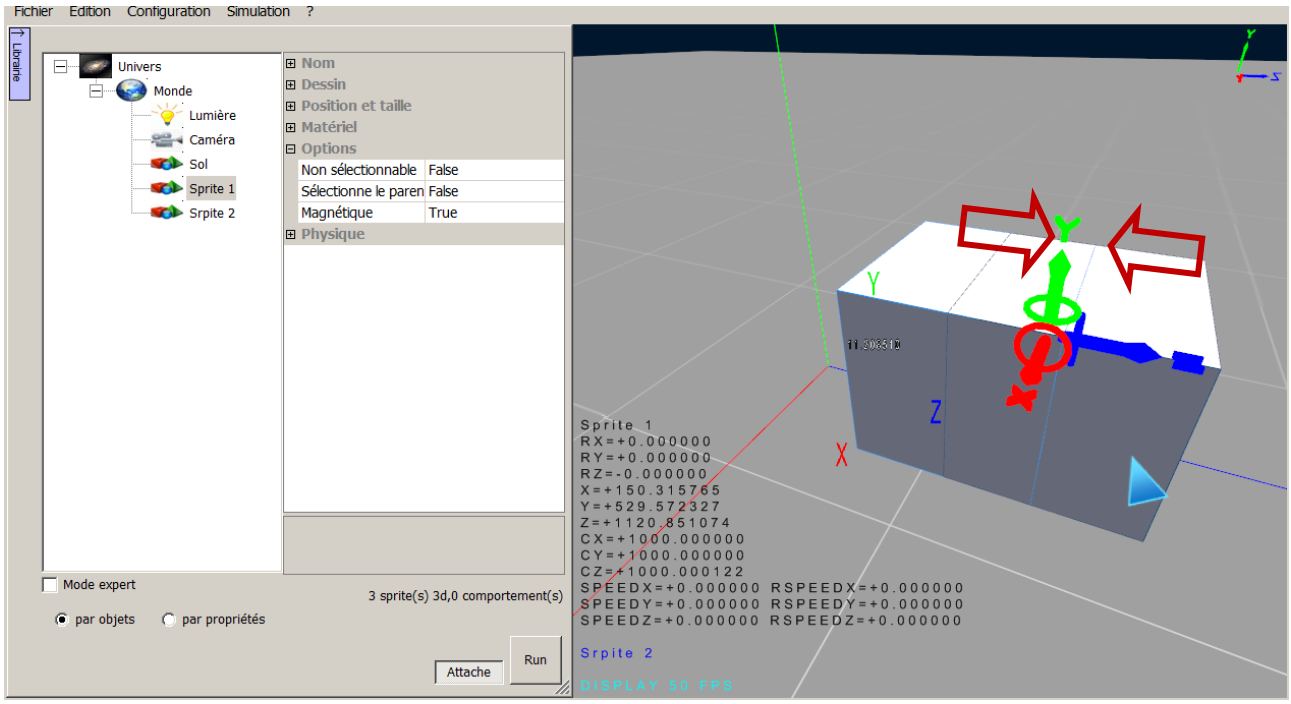
This is particularly useful for quickly positioning the resources of a production line in the 3D world.

In the 3D resource, a sprite is designated as the connector and has the particularity of being "magnetized" with other 3D sprites that also use the "Magnetic" function.

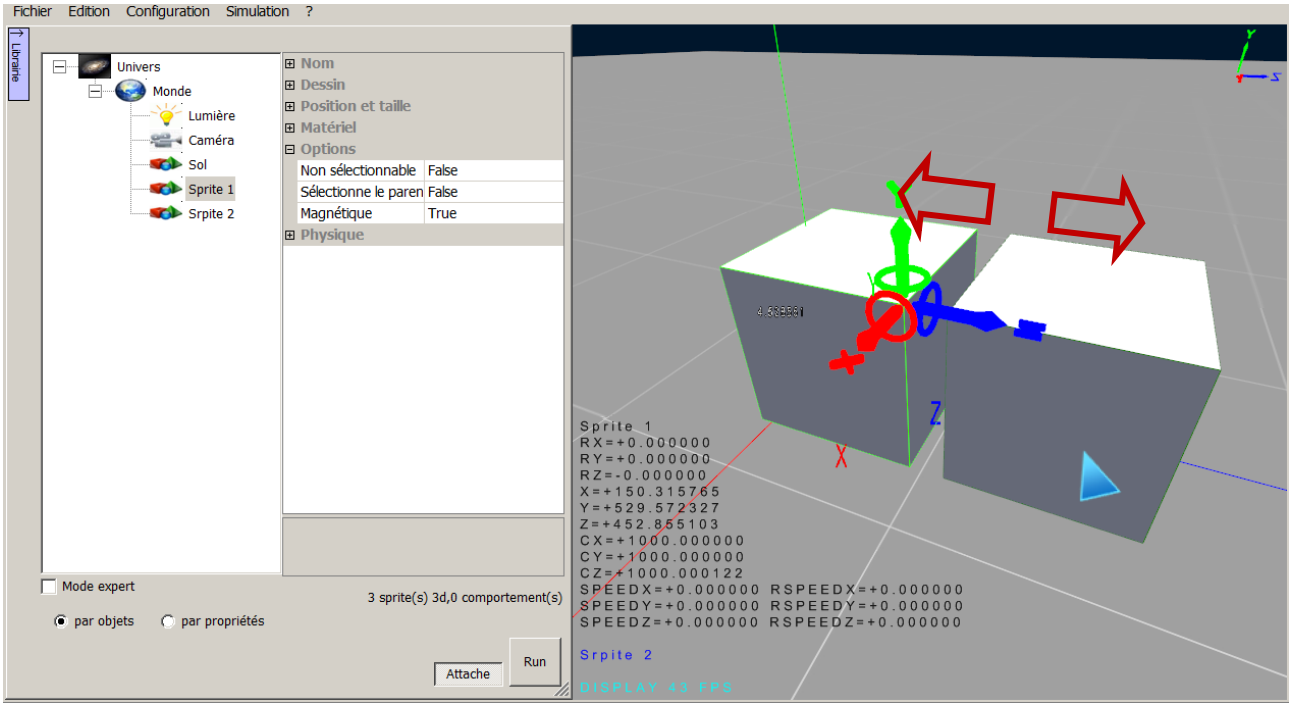
- The "Magnetic" function is available in the sprite properties



- When the 3D sprites using the "Magnetic" function are moved in the 3D world and approached to one another with the mouse (off simulation), they are "magnetized" towards each other. The connection is complete when the overlap between the two sprites is maximum.



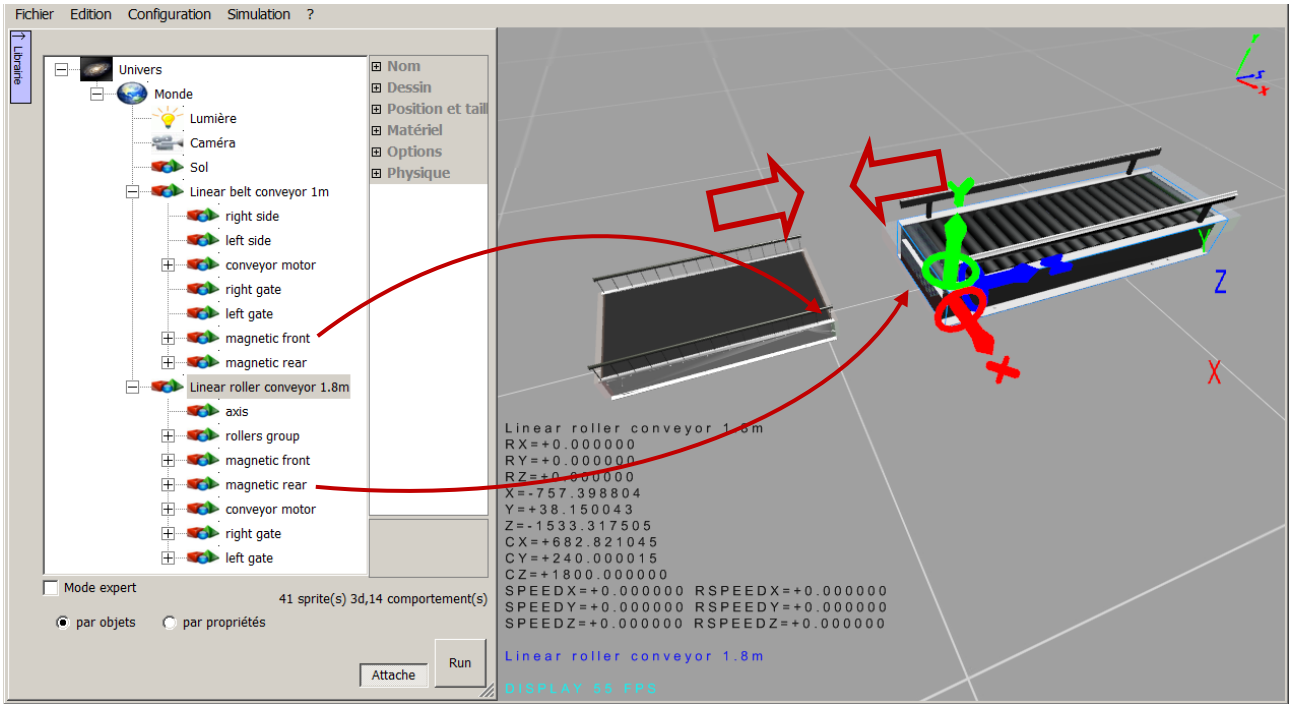
- To disable the "Magnetic" function and disconnect the sprites, simply hold the "Alt" key while moving the sprite with the mouse.



Example

All the conveyors available in the demo library of VIRTUAL UNIVERSE PRO use the "Magnetic" function.

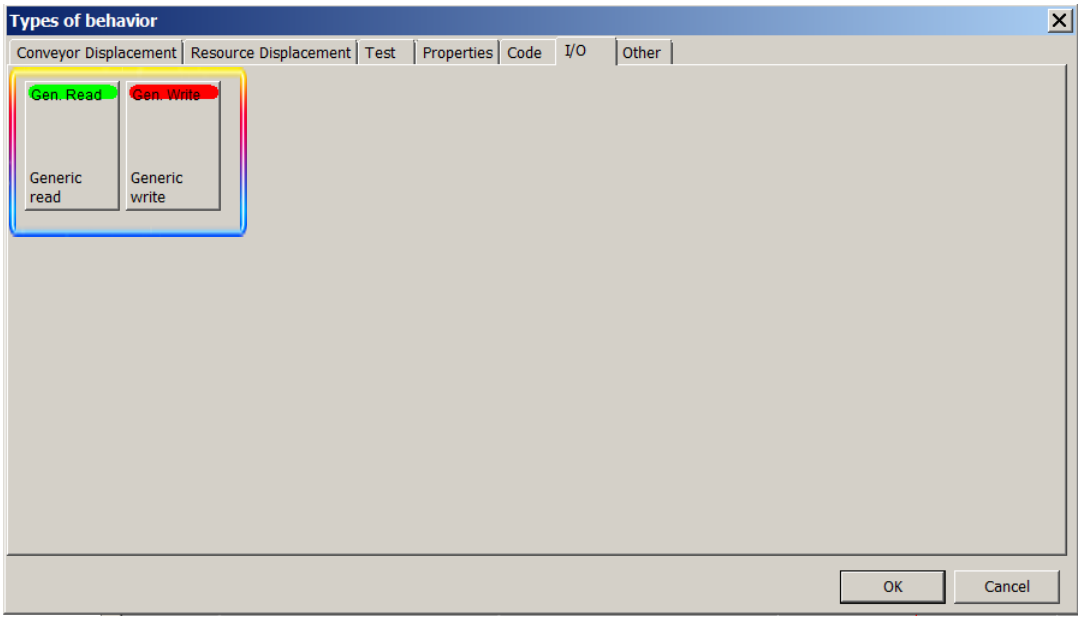
When a conveyor is approached to another conveyor using the mouse, it positions and connects itself perfectly.



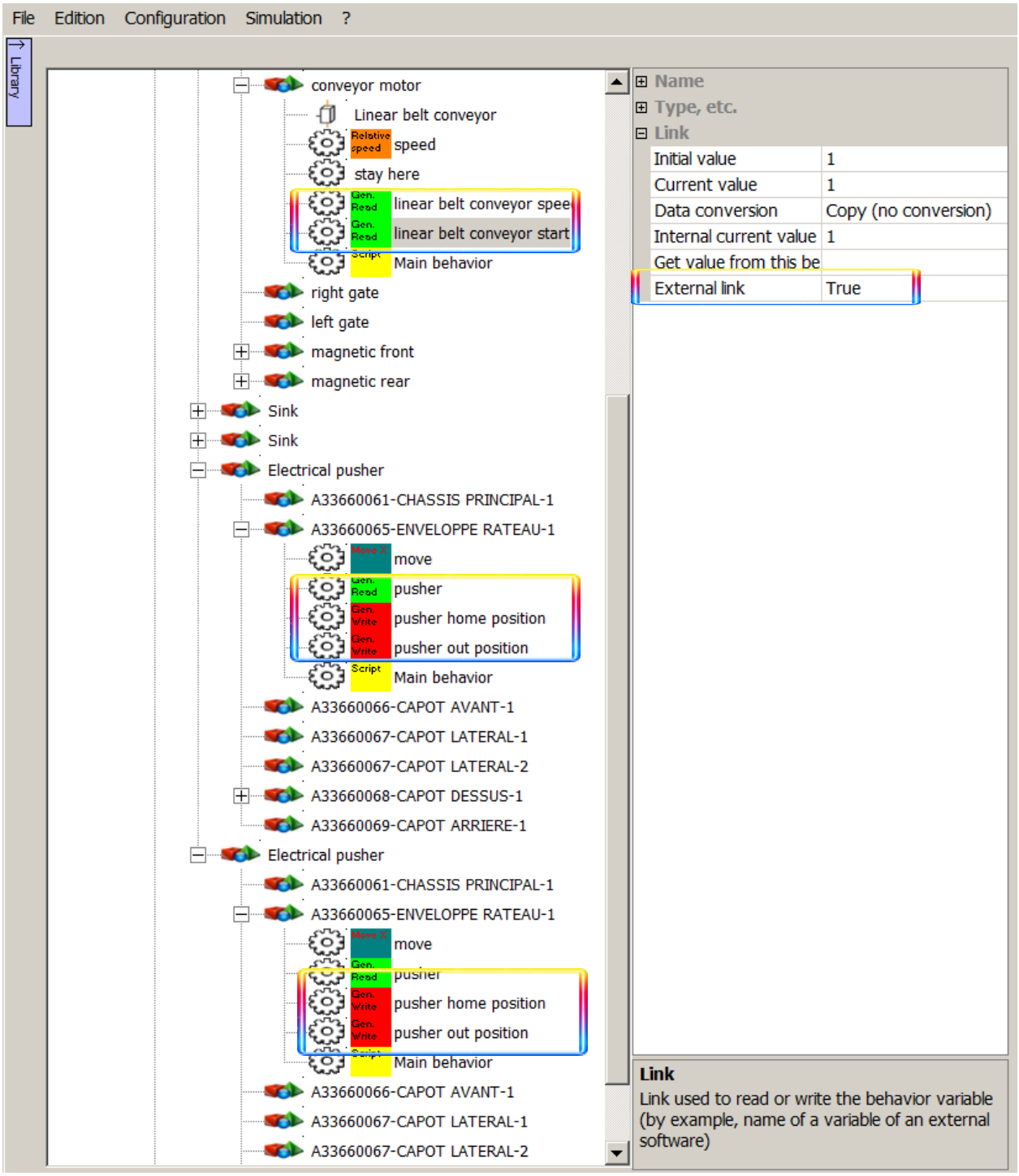
Connect a 3D emulator to an external software/controller

Define the list of 3D emulator inputs/outputs

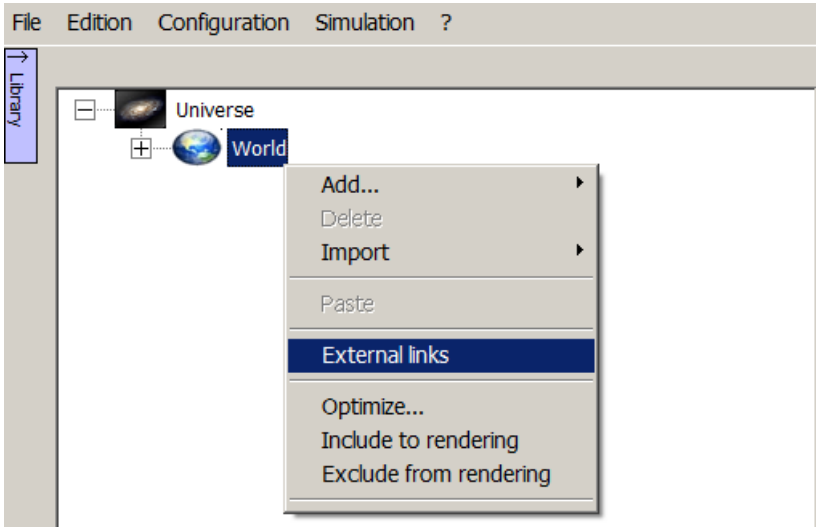
The list of 3D emulator inputs/outputs corresponds to all "Generic read" and "Generic write" behaviors added to the project and reported as "external links".

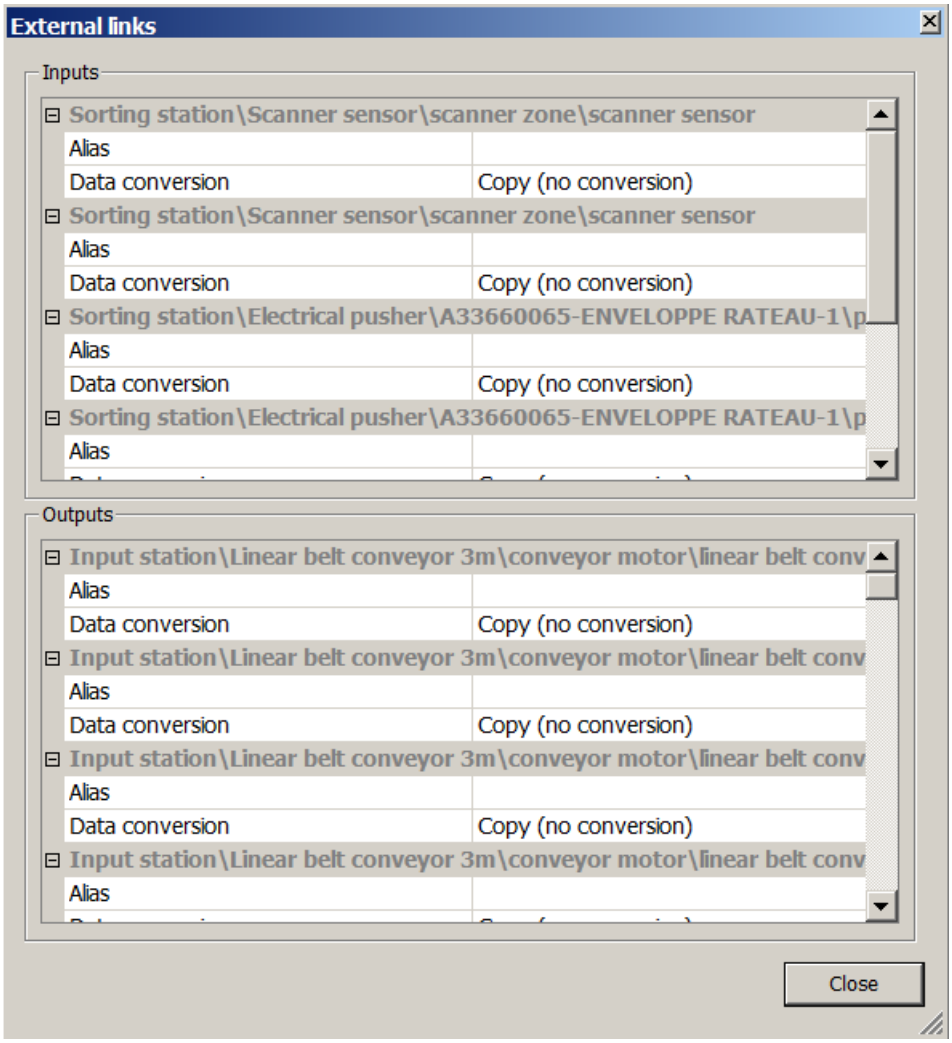


The "External Link" property is available in the behavior properties.



The list of 3D emulator inputs/outputs is available by a right click at the world level, in the "External Links" window.



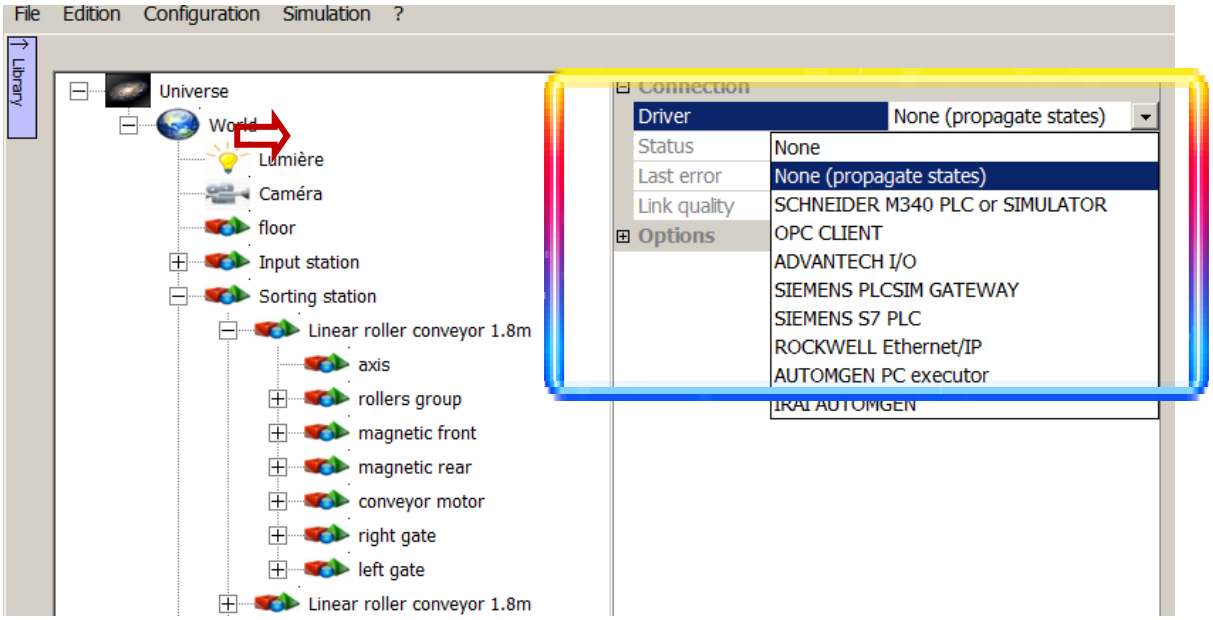


List of 3D emulator inputs/outputs

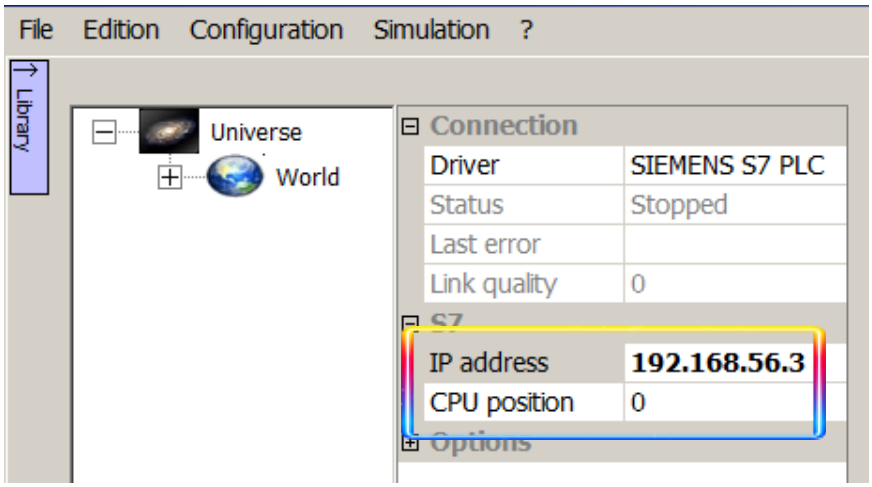
Connect a 3D emulator with an external software/controller

By default, the 3D emulator project is not connected to an external software/controller.

Configuring a connection with an external software/controller (Programmable Logic Controller, PLC emulator, OPC server...) is made in the Universe, in the "Connection" tab, by choosing a type of connector (driver).



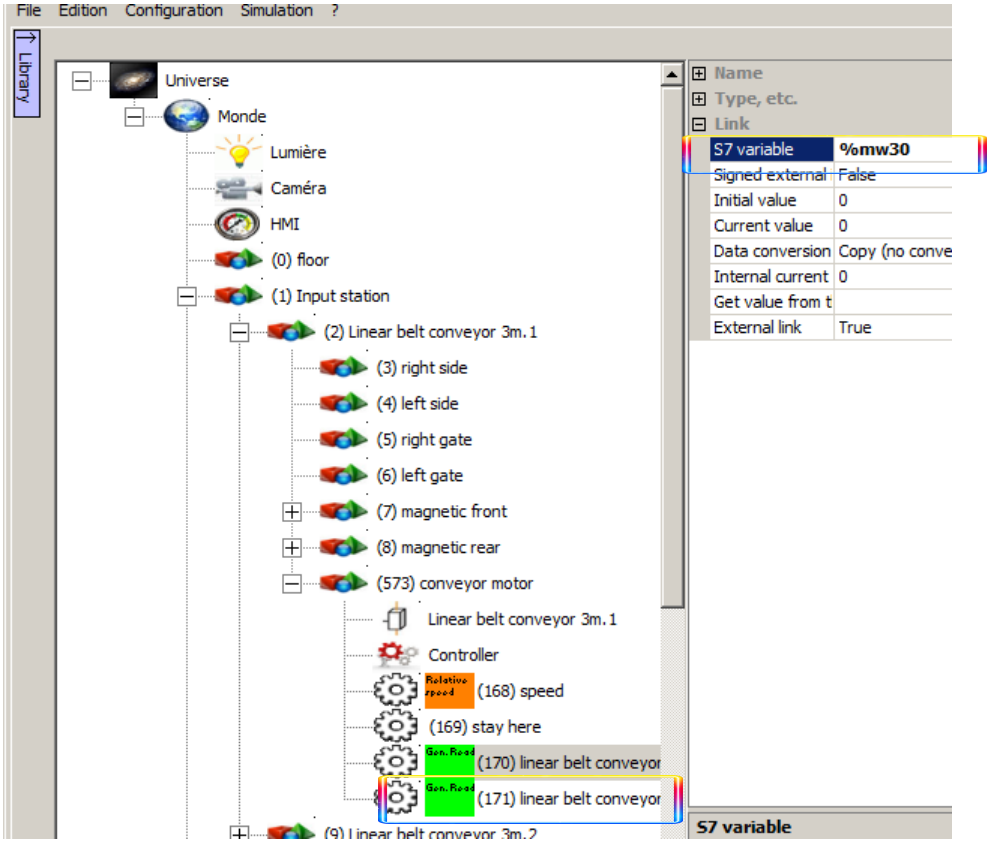
Once the "Driver" is selected, the connection settings (PLC IP address, CPU position on the rack, OPC server name...) are made in the tab-specific driver.



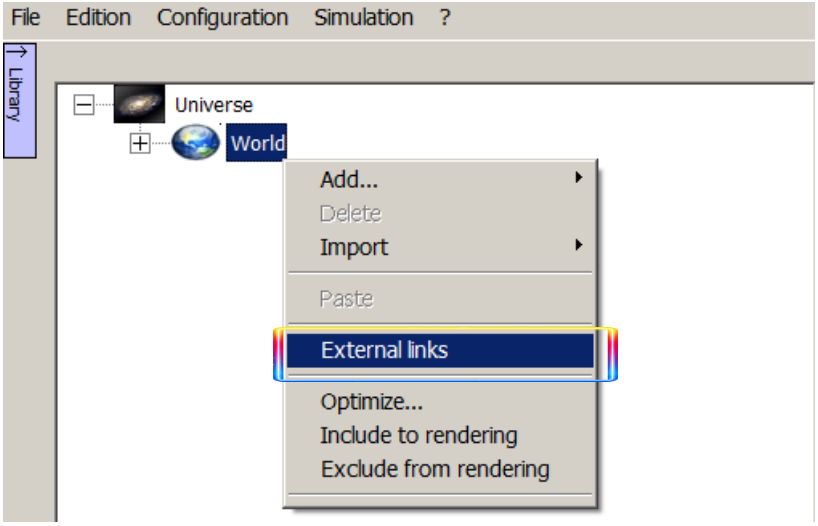
Mapping between the 3D emulator inputs/outputs and the external controller variables

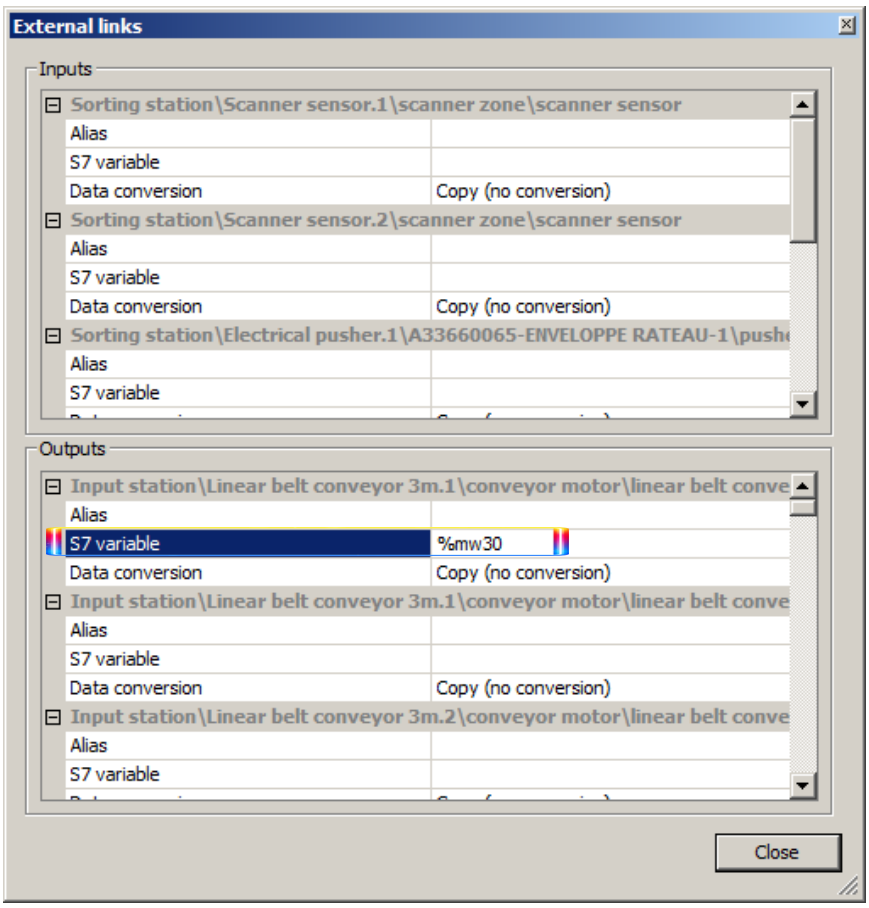
The 3D emulator inputs/outputs can be linked to the external software/controller variables at two levels

- at the input/output behavior level (declared as external link).

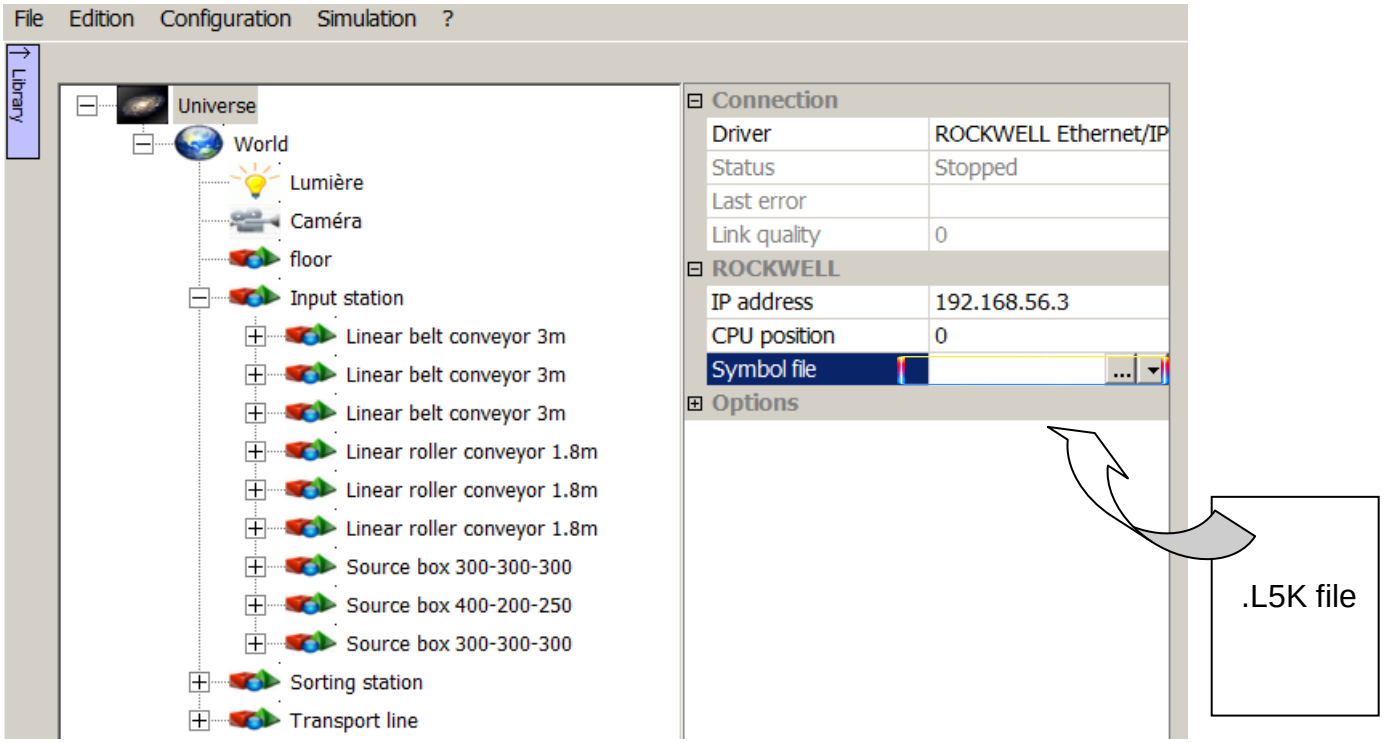


- at the World level, in the External Links window (listing all the 3D emulator inputs/outputs), accessible by a right click.





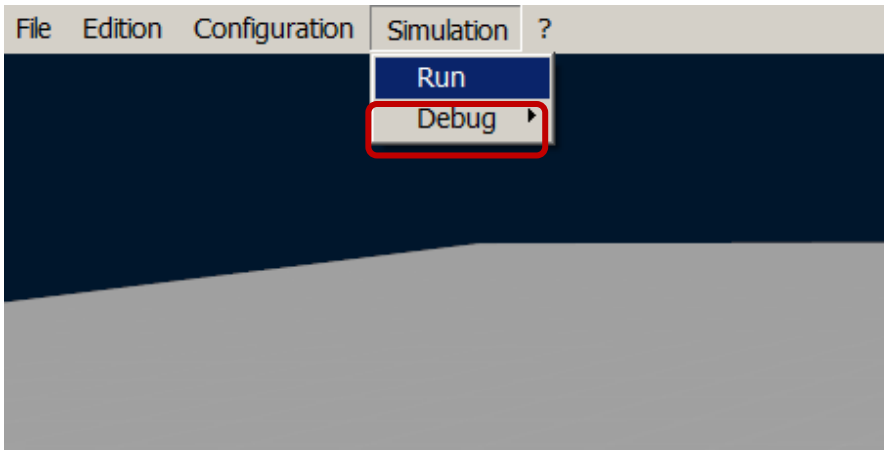
In the special case of ROCKWELL/Allen Bradley PLCs, it is possible to import a list of tags used in the PLC program in VIRTUAL UNIVERSE PRO, as a .L5K file, previously exported from the ROCKWELL/Allen Bradley programming software.



Test and debug a 3D emulator

Launch simulation

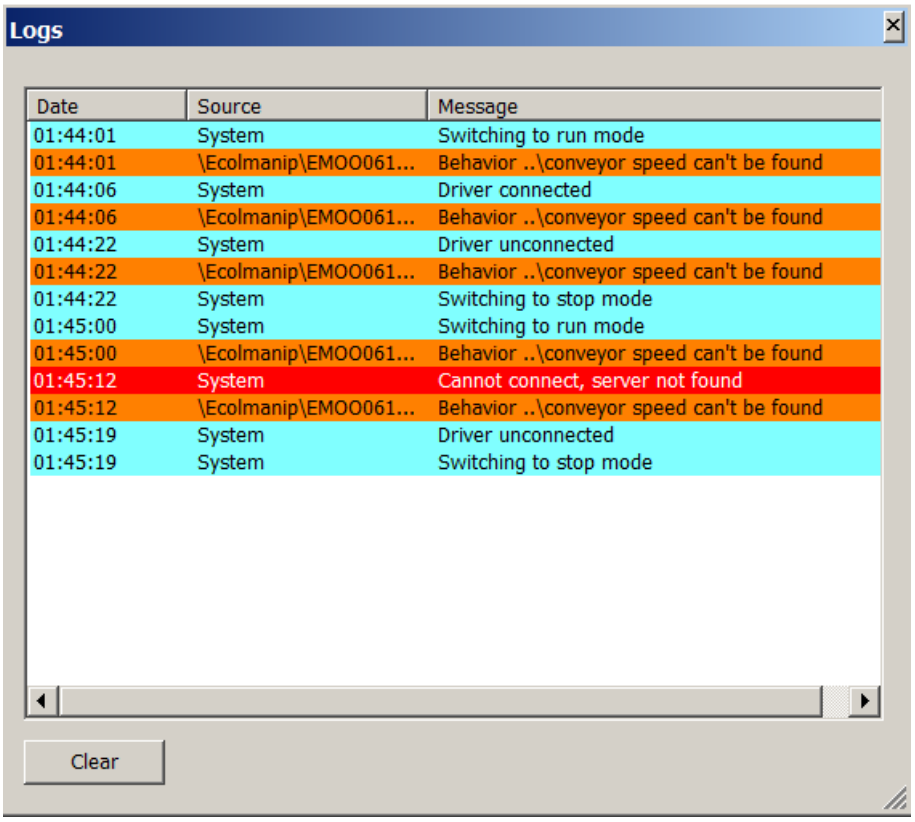
- To start the simulation, select Simulation in the menu bar and then "Run".



Simulation messages

During simulation, a simulation diagnosis is carried out continuously.

- If a problem (warning or fatal error) occurs at startup or during simulation (design error in the 3D emulator, communication problem with the external software/controller...), a message window opens with the list of identified problems :



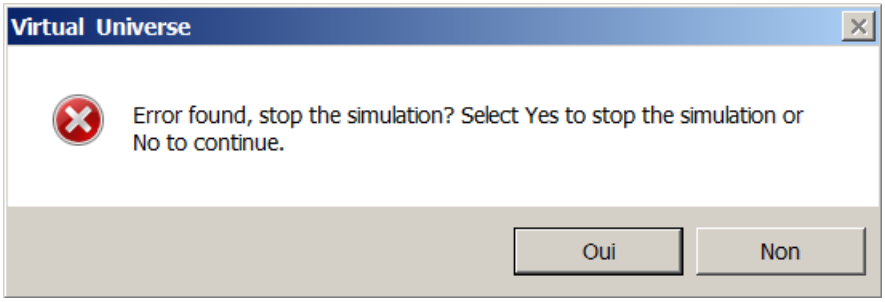
Here is the meaning of colors:

Light blue: normal operation of the simulation

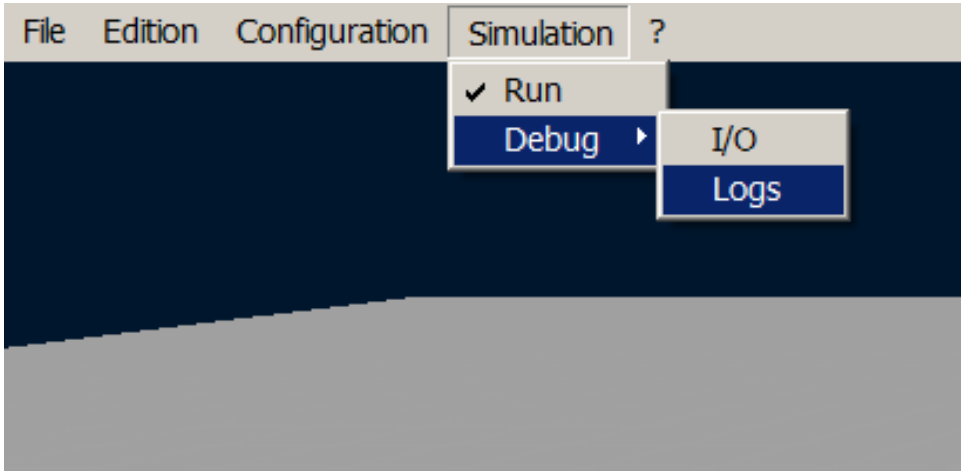
Orange: warning (design error in the 3D emulator)

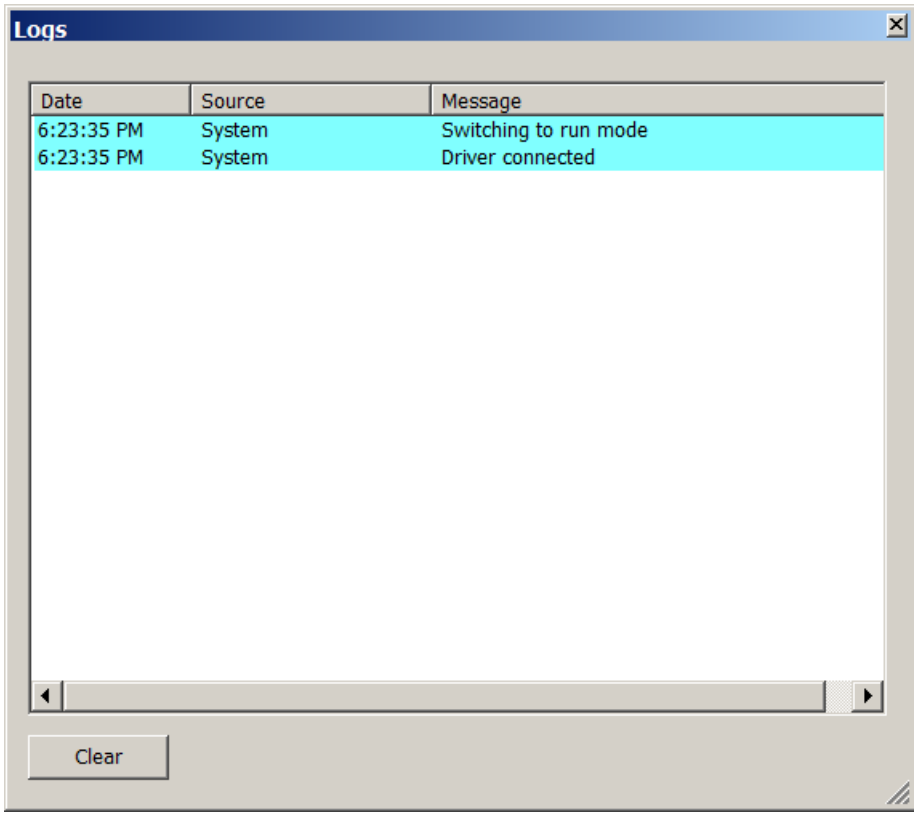
Red : fatal error (script failure, connection problem,..)

A second window also opens in parallel, prompting the user to stop the simulation (to correct the problem) or to continue with the simulation.



- At any time during simulation, the user can access the logs window, in the "Simulation/Debug/Logs" menu.



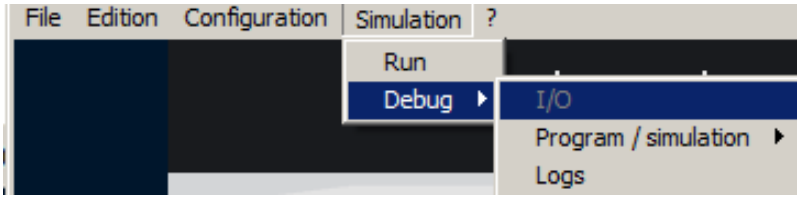


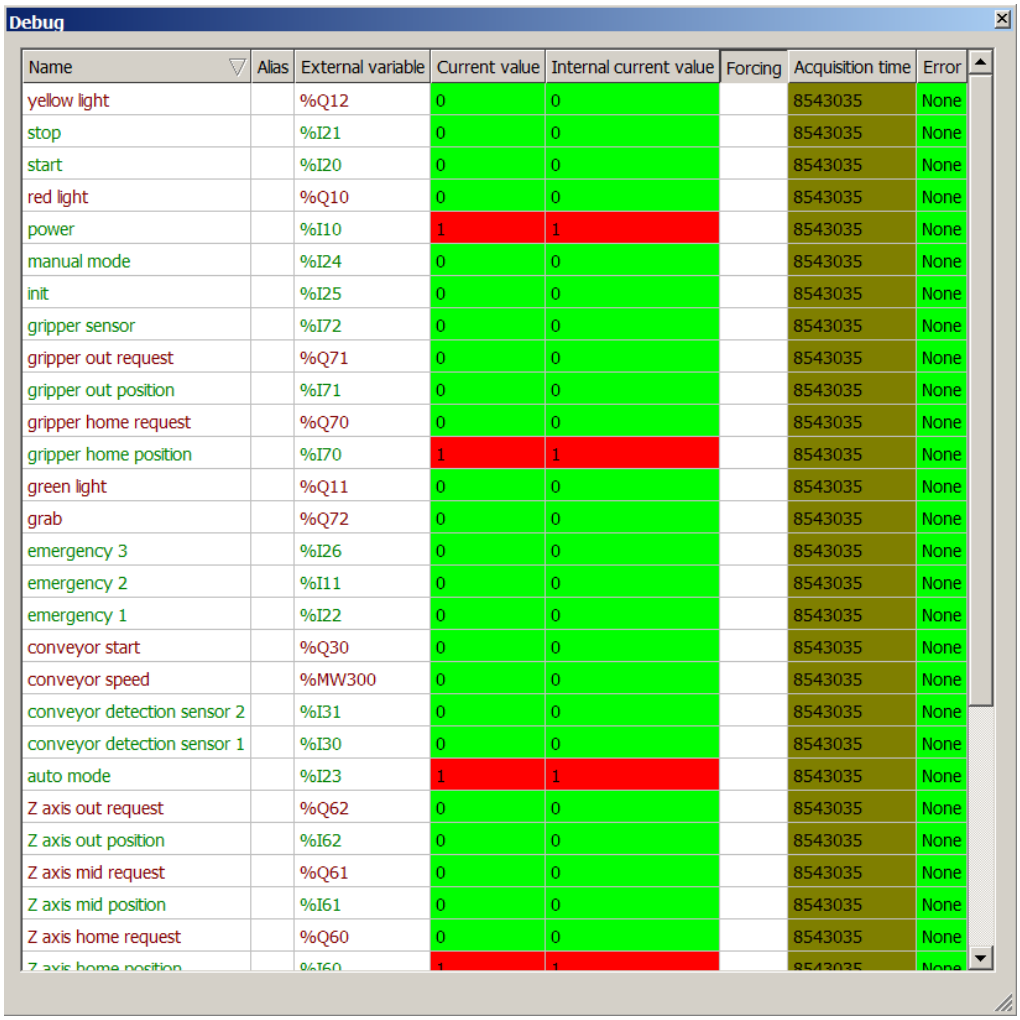
Test the 3D emulator

During simulation, it is possible to test the 3D emulator by forcing and displaying its input/output variables, connected or not to an external software/controller.

This allows, for example, to test the 3D emulator before the PLC program is fully complete or available, or to simulate unexpected scenarios (PLC connection interruption...).

In the Simulation/Debug/I/O menu, the user can access to the 3D emulator I/O monitoring and debugging window.





Name	Alias	External variable	Current value	Internal current value	Forcing	Acquisition time	Error
yellow light		%Q12	0	0		8543035	None
stop		%I21	0	0		8543035	None
start		%I20	0	0		8543035	None
red light		%Q10	0	0		8543035	None
power		%I10	1	1		8543035	None
manual mode		%I24	0	0		8543035	None
init		%I25	0	0		8543035	None
gripper sensor		%I72	0	0		8543035	None
gripper out request		%Q71	0	0		8543035	None
gripper out position		%I71	0	0		8543035	None
gripper home request		%Q70	0	0		8543035	None
gripper home position		%I70	1	1		8543035	None
green light		%Q11	0	0		8543035	None
grab		%Q72	0	0		8543035	None
emergency 3		%I26	0	0		8543035	None
emergency 2		%I11	0	0		8543035	None
emergency 1		%I22	0	0		8543035	None
conveyor start		%Q30	0	0		8543035	None
conveyor speed		%MW300	0	0		8543035	None
conveyor detection sensor 2		%I31	0	0		8543035	None
conveyor detection sensor 1		%I30	0	0		8543035	None
auto mode		%I23	1	1		8543035	None
Z axis out request		%Q62	0	0		8543035	None
Z axis out position		%I62	0	0		8543035	None
Z axis mid request		%Q61	0	0		8543035	None
Z axis mid position		%I61	0	0		8543035	None
Z axis home request		%Q60	0	0		8543035	None
Z axis home position		%I60	1	1		8543035	None

- The « **Name** » column indicates the 3D emulator input/output variable name
  - red = 3D emulator input variable
  - green = 3D emulator output variable
- The « **External variable** » column the connected external software/controller variable name.
- The « **Current value** » column the variable current value.
- The « **Internal Current value** » column indicates the variable internal (before any conversion) current value.
- The « **Forcing** » column is used to force the variable.
- The « **Acquisition time** » column gives the instantaneous time required for the exchange of the variable between the 3D emulator and the external software/controller.
- The « **Error** » column indicates a possible connection error with the external software.

Measure and optimize the performances of a 3D emulator

There are three criteria to measure and optimize the performances of a 3D emulator :

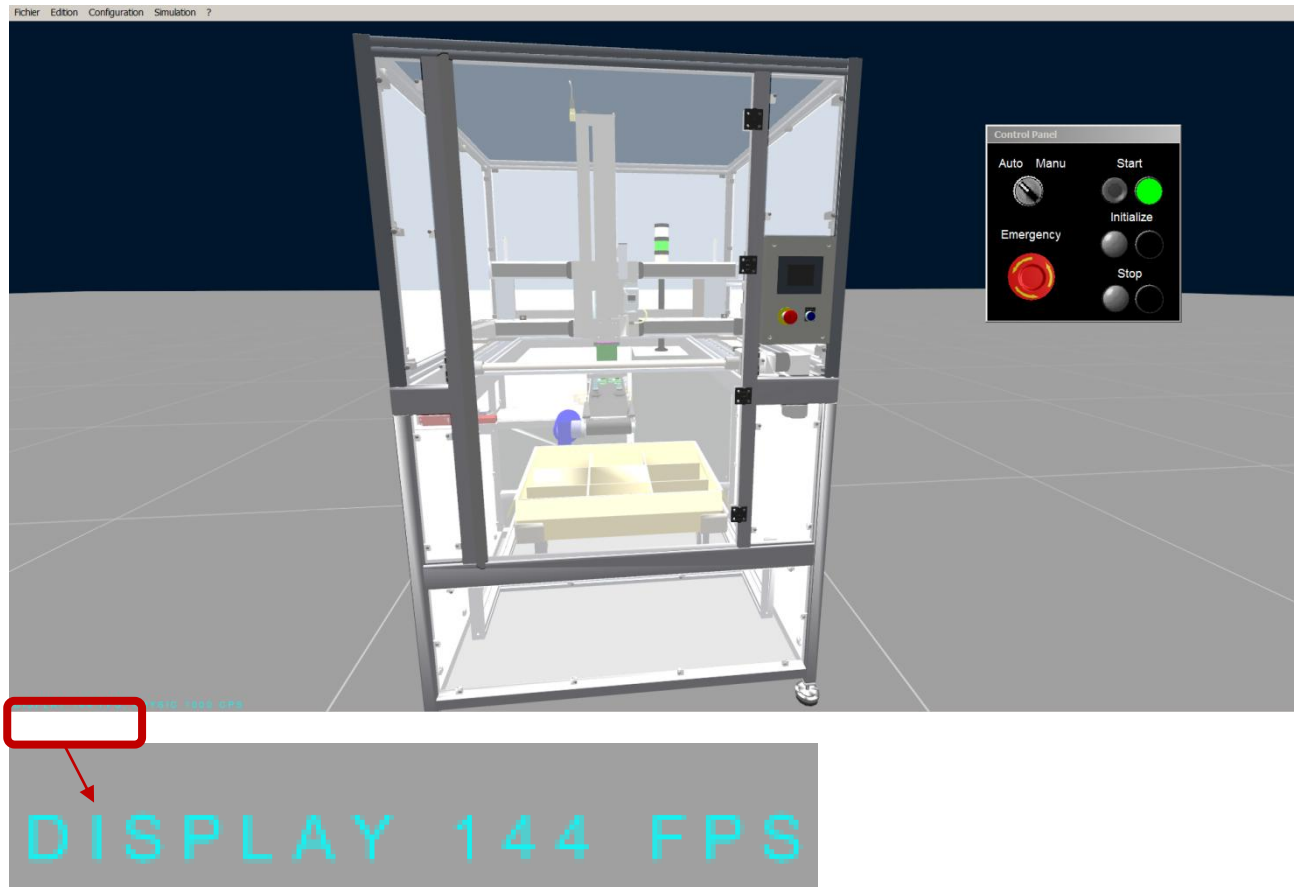
- The graphics refresh rate (3D rendering speed and fluidity)
- The physics engine performances (realism of the simulated physical phenomena)
- The speed of dialog with the external software/controller

Measure graphics performances

It is important to measure and optimize graphics performances of a 3D emulator. Indeed, these graphics performances directly influence the fluidity and visual quality of 3D rendering.

The graphics performances of a 3D emulator are represented by the number of frames displayed per second (FPS) in the 3D rendering window (during simulation and offline simulation). It also corresponds to the graphic refresh rate.

In VIRTUAL UNIVERSE PRO, this information is constantly displayed in the lower left of the 3D rendering window.



The graphic performances of a 3D emulator mainly depend on the following factors:

- The graphics card quality and performance (installed on the computer running the 3D emulator)
- The 3D geometry complexity (displayed in the emulator 3D rendering)
- The number of cameras (views) used in the 3D emulator
- The set-up window opening or closing during simulation

We consider that the graphics performances of a 3D emulator are good when the graphics refresh rate always remains above **15 FPS (15 frames per second)**.

Optimize graphics performances

Graphic card

For getting good graphics performance with the VIRTUAL UNIVERSE PRO 3D emulators, it is strongly recommended to use a modern computer equipped with a good graphics card.

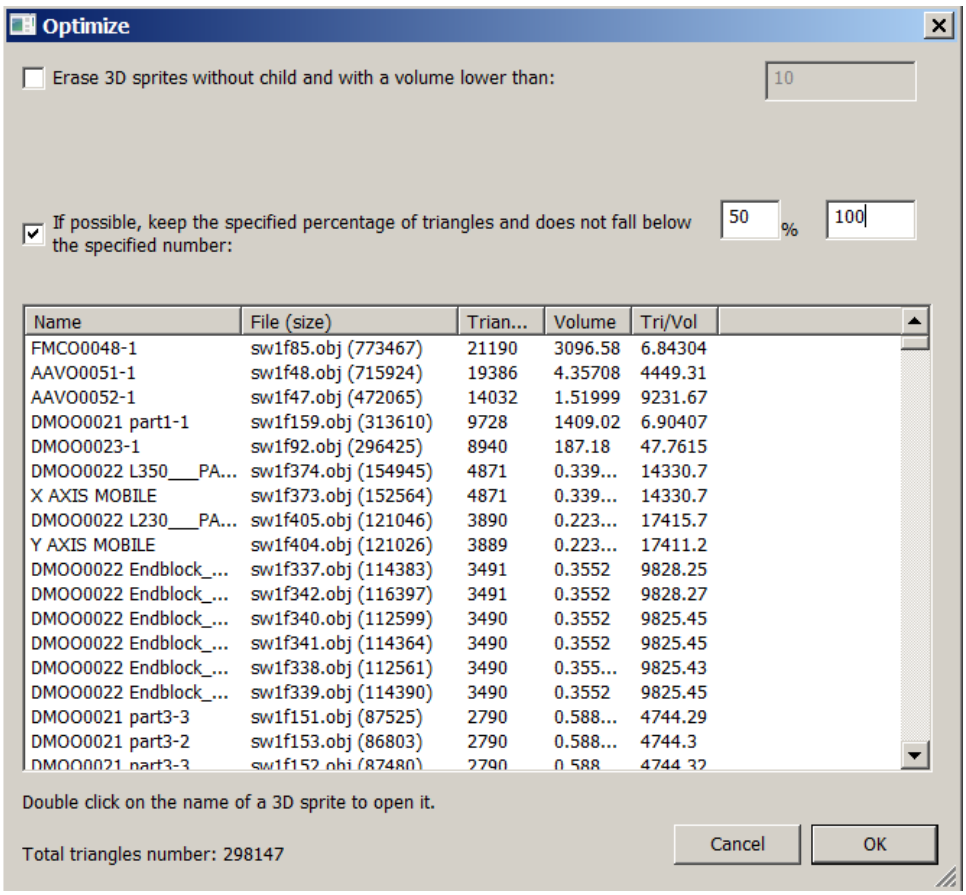
Graphics cards dedicated to the world of video games are particularly powerful and adapted for VIRTUAL UNIVERSE PRO 3D emulators.

3D geometry simplification

The 3D models created with CAD software often have very complex 3D geometries (number of triangles) and small parts that are not always necessary to VIRTUAL UNIVERSE PRO 3D emulators and may instead greatly reduce graphics performance.

For the simplification of 3D CAD models, VIRTUAL UNIVERSE PRO offers a 3D geometries optimization tool.

For more information about usage of the 3D geometry optimization tool, see Simplify 3D CAD models.


Optimize
Erase 3D sprites without child and with a volume lower than:
10
If possible, keep the specified percentage of triangles and does not fall below the specified number:
50
%
100
Name
File (size)
Trian...
Volume
Tri/Vol
FMCO0048-1 sw1f85.obj (773467) 21190 3096.58 6.84304
AAVO0051-1 sw1f48.obj (715924) 19386 4.35708 4449.31
AAVO0052-1 sw1f47.obj (472065) 14032 1.51999 9231.67
DMOO0021 part1-1 sw1f159.obj (313610) 9728 1409.02 6.90407
DMOO0023-1 sw1f92.obj (296425) 8940 187.18 47.7615
DMOO0022 L350___PA... sw1f374.obj (154945) 4871 0.339... 14330.7
X AXIS MOBILE sw1f373.obj (152564) 4871 0.339... 14330.7
DMOO0022 L230___PA... sw1f405.obj (121046) 3890 0.223... 17415.7
Y AXIS MOBILE sw1f404.obj (121026) 3889 0.223... 17411.2
DMOO0022 Endblock_... sw1f337.obj (114383) 3491 0.3552 9828.25
DMOO0022 Endblock_... sw1f342.obj (116397) 3491 0.3552 9828.27
DMOO0022 Endblock_... sw1f340.obj (112599) 3490 0.3552 9825.45
DMOO0022 Endblock_... sw1f341.obj (114364) 3490 0.3552 9825.45
DMOO0022 Endblock_... sw1f338.obj (112561) 3490 0.355... 9825.43
DMOO0022 Endblock_... sw1f339.obj (114390) 3490 0.3552 9825.45
DMOO0021 part3-3 sw1f151.obj (87525) 2790 0.588... 4744.29
DMOO0021 part3-2 sw1f153.obj (86803) 2790 0.588... 4744.3
Double click on the name of a 3D sprite to open it.
Total triangles number: 298147
Cancel
OK

Opening the set-up window

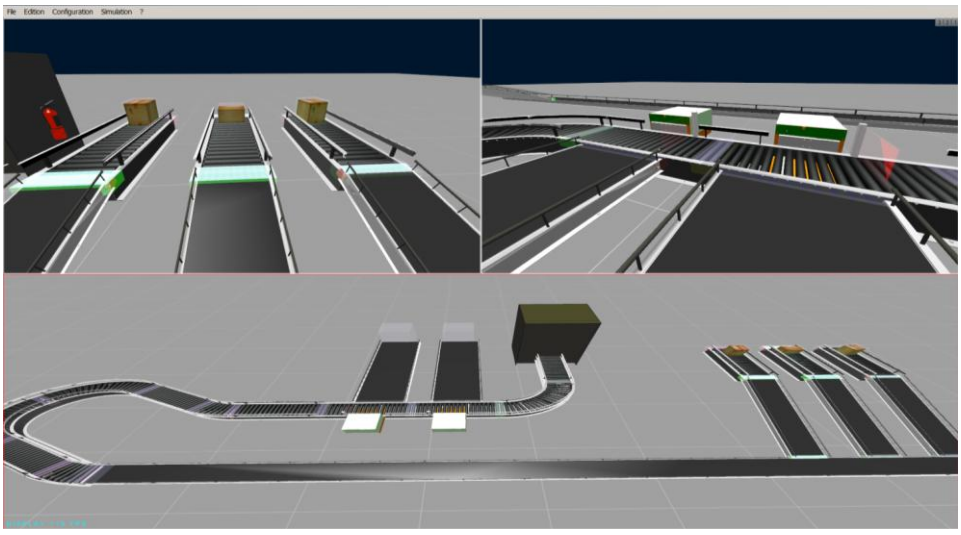
When opened, the set-up window can significantly reduce the graphics refresh rate during simulation. During simulation, when possible, it is recommended to keep this set-up window closed.

Another possibility is to use a second screen to open the set-up window (and Debug window), in order to keep only the 3D rendering window opened on the first screen. Thus, the graphics performances of the 3D emulator (during simulation) are not reduced.

Set-up window 3D rendering window

Use of multiple cameras

The use of multiple cameras in a 3D emulator may reduce the graphics performances.



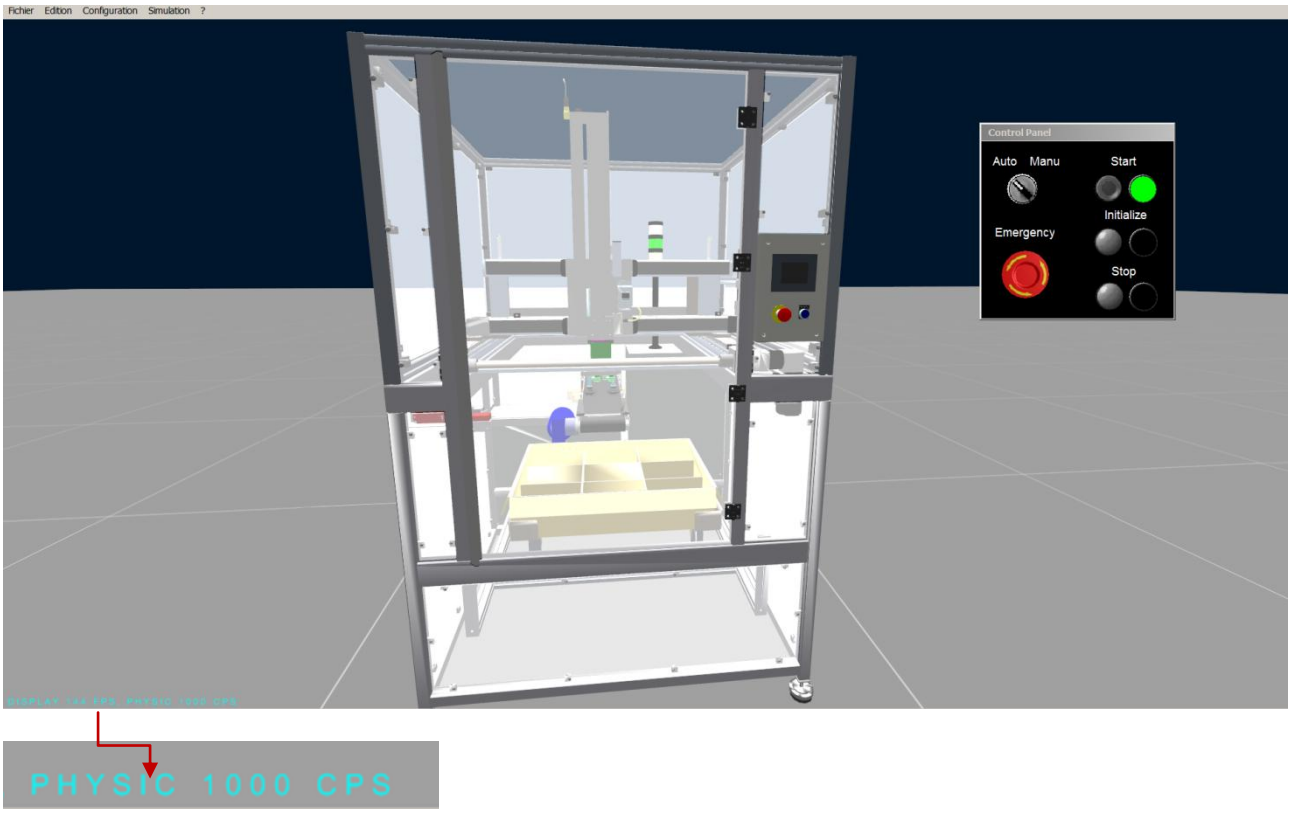
Measure the physics engine performances

It may be useful to know how to measure and optimize performances of the VIRTUAL UNIVERSE PRO physics engine, particularly for 3D emulators which require the simulation of many physical phenomena (gravity of bodies, friction, collisions).

The VIRTUAL UNIVERSE PRO physics engine performances directly influence the realism of the simulated physical phenomena.

The physics engine performances are expressed in number of physics engine calculation per second (CPS).

In VIRTUAL UNIVERSE PRO, this information is constantly displayed in the lower left of the 3D rendering window.



The physics performances of a 3D emulator strongly depend on the following factors:

- The processor (CPU) performance of the computer used to run the 3D emulator
- The number and shape of 3D objects used by the physics engine during simulation

It is considered that the physics performances of a 3D emulator are good when the number of physics calculations, during simulation, is still greater than **100 CPS (100 calculations per second)**.

Optimize the physics engine performances

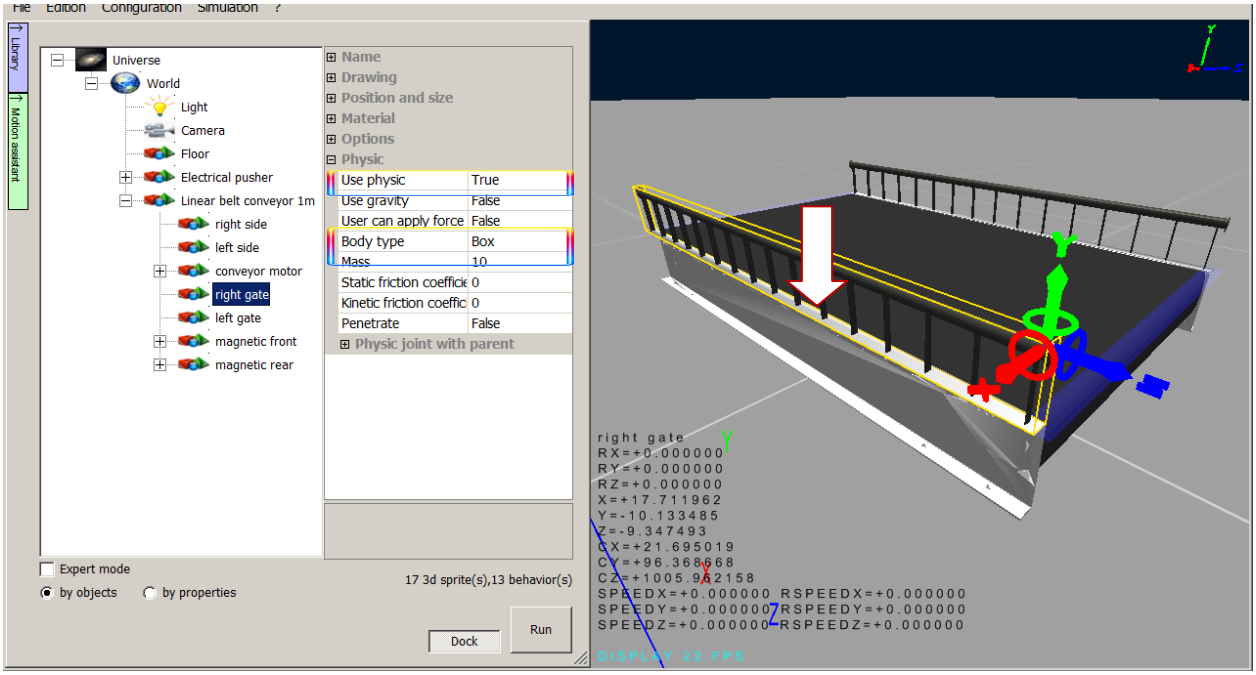
Computer CPU

To achieve good physics performances in the VIRTUAL UNIVERSE PRO 3D emulators, it is strongly recommended to use a modern computer equipped with a powerful processor (multi-core).

Shape of 3D sprites used by the physics engine

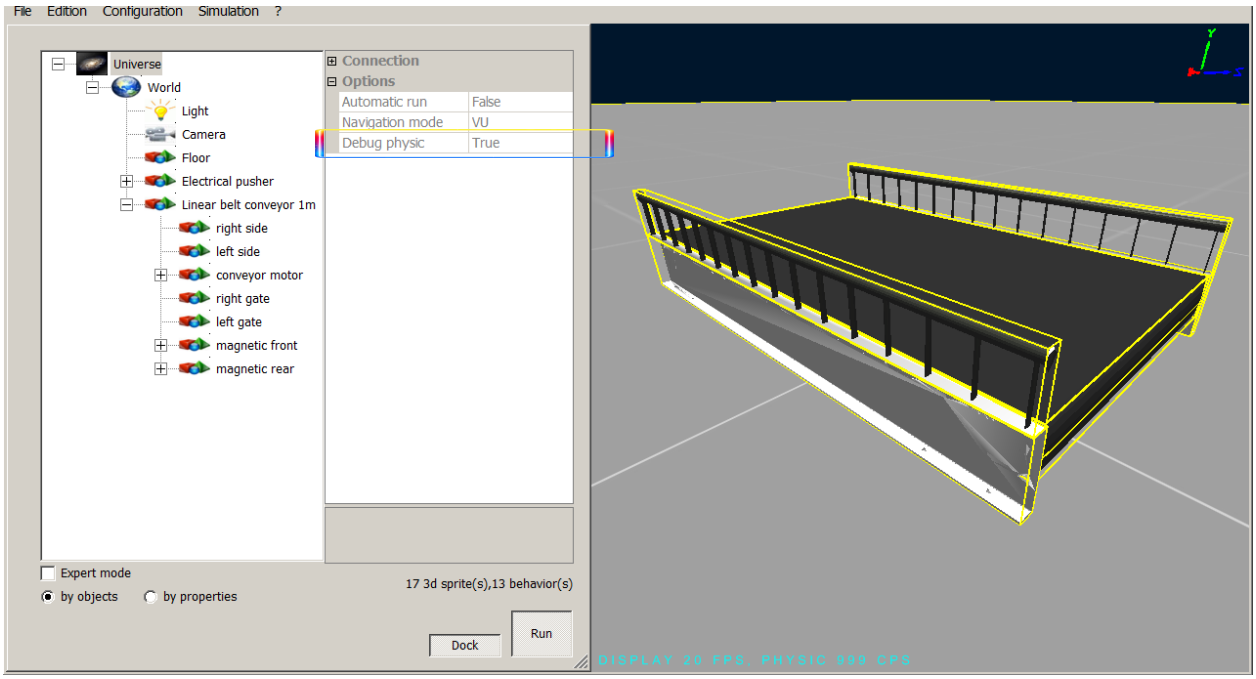
By default, when added to the project, 3D sprites are not exposed to the physics engine, their option "Use physics" is disabled.

For optimizing the physics engine performance, it is recommended to not declare too many 3D Sprites used by the physics engine and to choose the body type "box" when possible.



The "Any" or "Convex" body types are those which provide the greater realism in the simulated physical phenomena, but also strongly impact the physics engine.

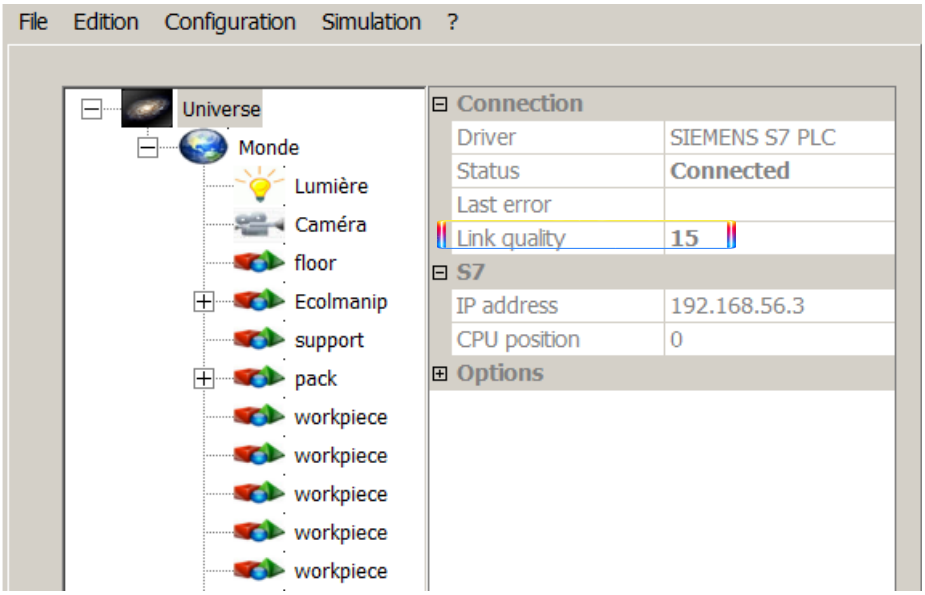
In development phases, it can be useful to visualize the geometries handled by the physics engine by activating the "debug physics" mode in the properties of the Universe.



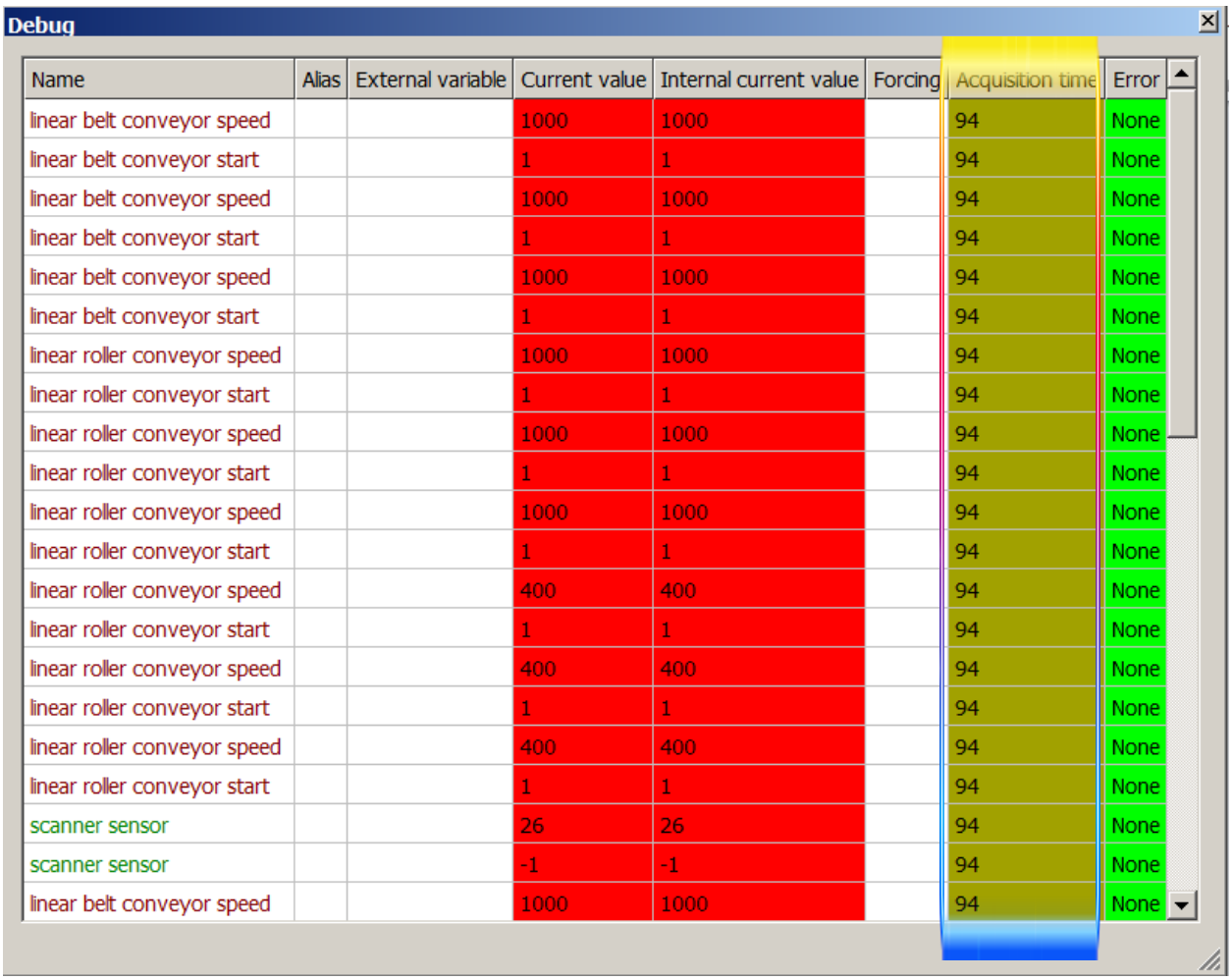
Performances of dialog with the external software/controller

The dialog performances with the external software/controller are expressed by the time (in milliseconds) needed to exchange the set of variables shared between the 3D emulator and the external software/controller.

This information is available, during simulation, in the Universe/Connection/Link Quality tab.



The Debug I/O window also provides the time needed for exchanging each variable with the external software.


Debug
Name
Alias
External variable
Current value
Internal current value
Forcing
Acquisition time
Error
linear belt conveyor speed 1000 1000 94 None
linear belt conveyor start 1 1 94 None
linear belt conveyor speed 1000 1000 94 None
linear belt conveyor start 1 1 94 None
linear belt conveyor speed 1000 1000 94 None
linear belt conveyor start 1 1 94 None
linear roller conveyor speed 1000 1000 94 None
linear roller conveyor start 1 1 94 None
linear roller conveyor speed 1000 1000 94 None
linear roller conveyor start 1 1 94 None
linear roller conveyor speed 1000 1000 94 None
linear roller conveyor start 1 1 94 None
linear roller conveyor speed 400 400 94 None
linear roller conveyor start 1 1 94 None
linear roller conveyor speed 400 400 94 None
linear roller conveyor start 1 1 94 None
linear roller conveyor speed 400 400 94 None
linear roller conveyor start 1 1 94 None
scanner sensor 26 26 94 None
scanner sensor -1 -1 94 None
linear belt conveyor speed 1000 1000 94 None

Generate standalone 3D emulators (players)

VIRTUAL UNIVERSE PRO can generate independent 3D emulators, as executable files (.exe) called "players".

A player embeds all the simulation intelligence from its original 3D emulator project (from which it is created). It can be connected to an external software/controller (depending on the type of connection defined in the original 3D emulator project) and offers, during simulation, the same functionalities as the original 3D emulator project.

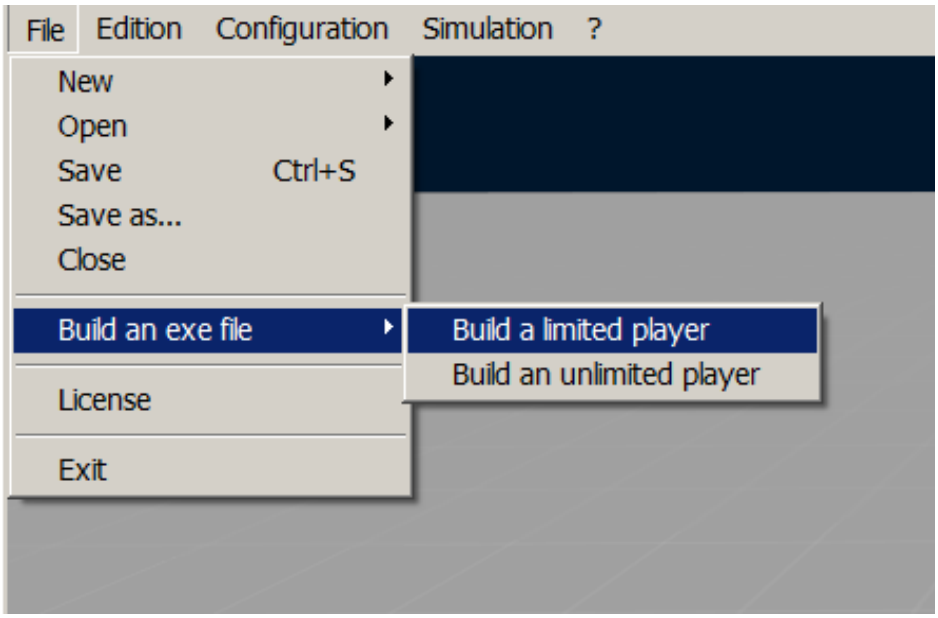
There are two types of players:

- Limited player (2 minutes of use)
- Unlimited player (unlimited use)

Limited players

VIRTUAL UNIVERSE PRO allows the generation of players, limited to a life of 2 minutes (the player closes 2 minutes after it opened).

The generation of a limited duration player is available in the File/Generate an executable/Generate a limited player menu.

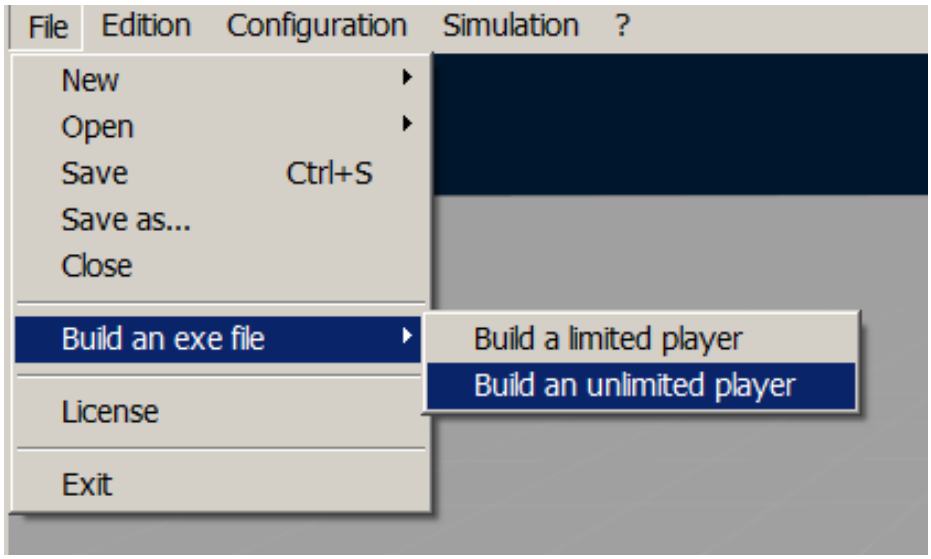


Unlimited players

VIRTUAL UNIVERSE PRO allows the generation of unlimited players (in term of duration), but protected by a unique security code.

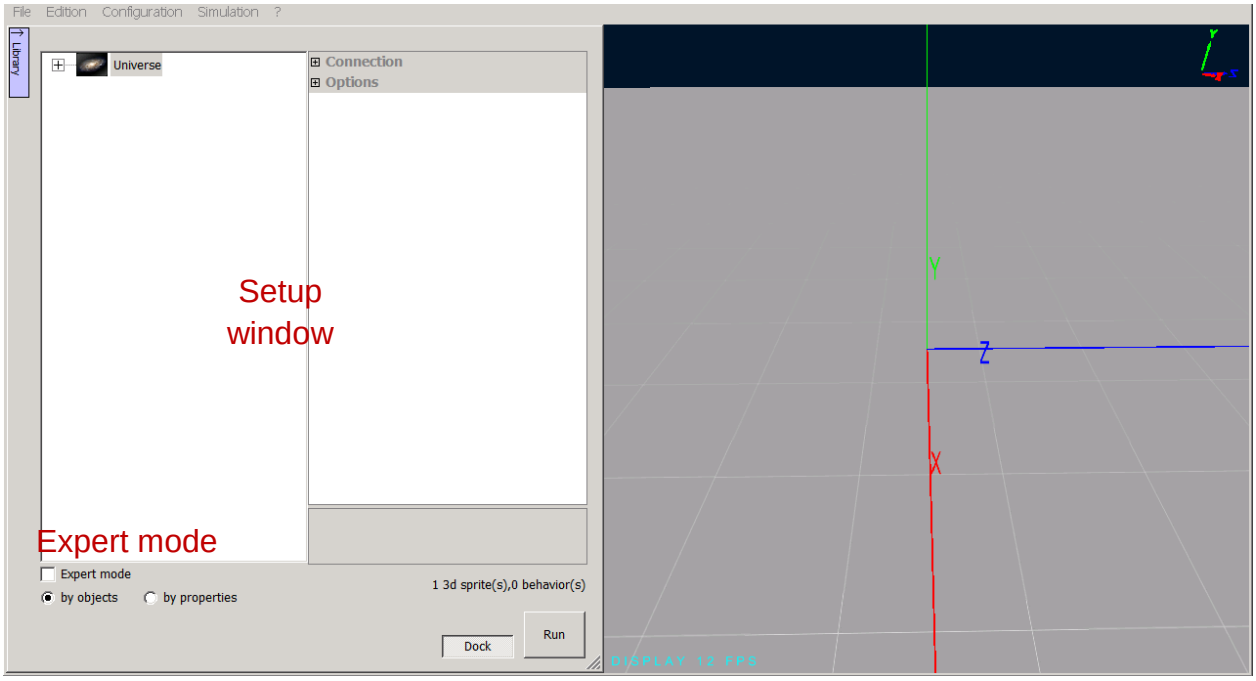
The generation of an unlimited duration player is available in the "Build an exe file/Build an unlimited player" in the “File” menu.

This generation requires a security code. For more information, see Register a security code (unlimited player)



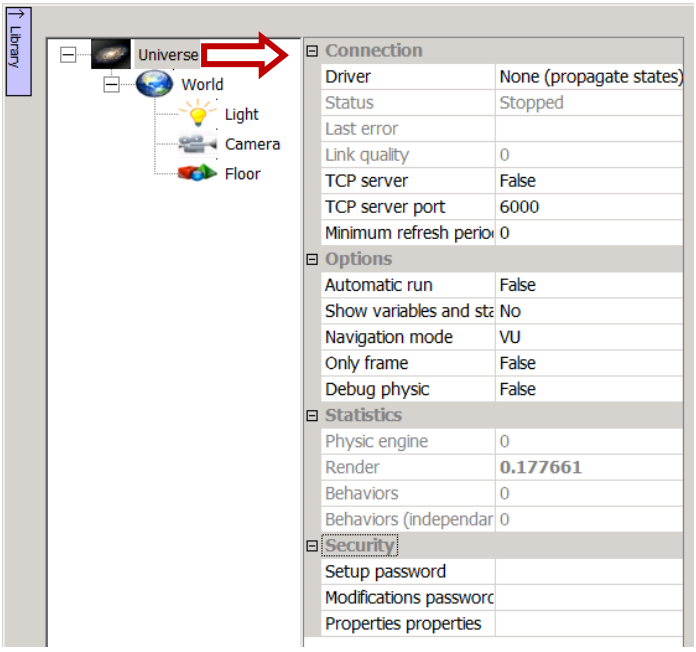
Detailed properties of a 3D emulator

The setup window provides access to the detailed properties of a 3D emulator project. Some of these properties are for advanced users only and are accessible by activating the “Expert mode”.



Properties of the Universe

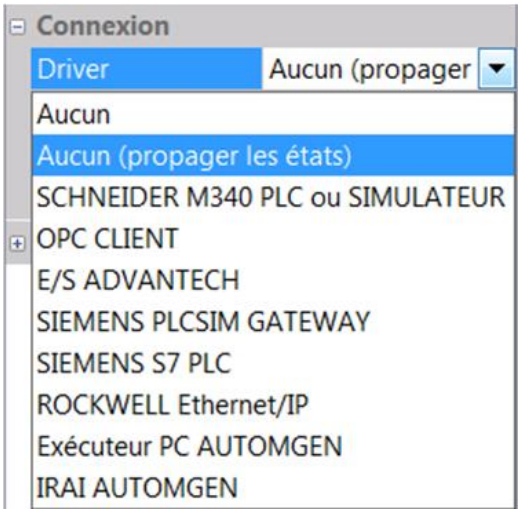
Detailed properties of the Universe



Universe properties

Connection

Driver**:** Define the connection with external software (software which communicates with VIRTUAL UNIVERSE PRO). The available drivers are:



Status**:** Current status of the connection (stopped, external software not ready, connected).

Last error: Last communication error.

Link quality: Time needed to exchange all data with the external software (in milliseconds).

TCP server (Expert mode): If true, activates a TCP server allowing one or more external applications to dialog with VIRTUAL.

TCP server port (Expert mode): TCP port number.

Minimum refresh period for external variables (Expert mode): Minimum limit (in milliseconds) for the exchange of all variables. 100: exchange of all variables every 100 milliseconds as a minimum.

Options

Automatic run: If true, simulation is launched automatically when opening the project. Press “Shift” key at the opening of the project to deactivate the automatic run.

Show variables and states (Expert mode): If true, displays (in the rendering window) variables names and states on 3D Sprites containing such a behavior (using external variables or states).

Navigation mode: VU (navigation with mouse) or IRIS3D (in the absence of mouse, this mode enables navigation with buttons and arrows).

Only frame (Expert mode): Draw only frames for 3D Sprites.

Debug physic: If true, the volumes used by the physics engine are displayed in the 3D rendering window (yellow color lines). This can be very useful in the development phase of a project to see the different volumes used by the physics engine.

Statistics (Expert mode)

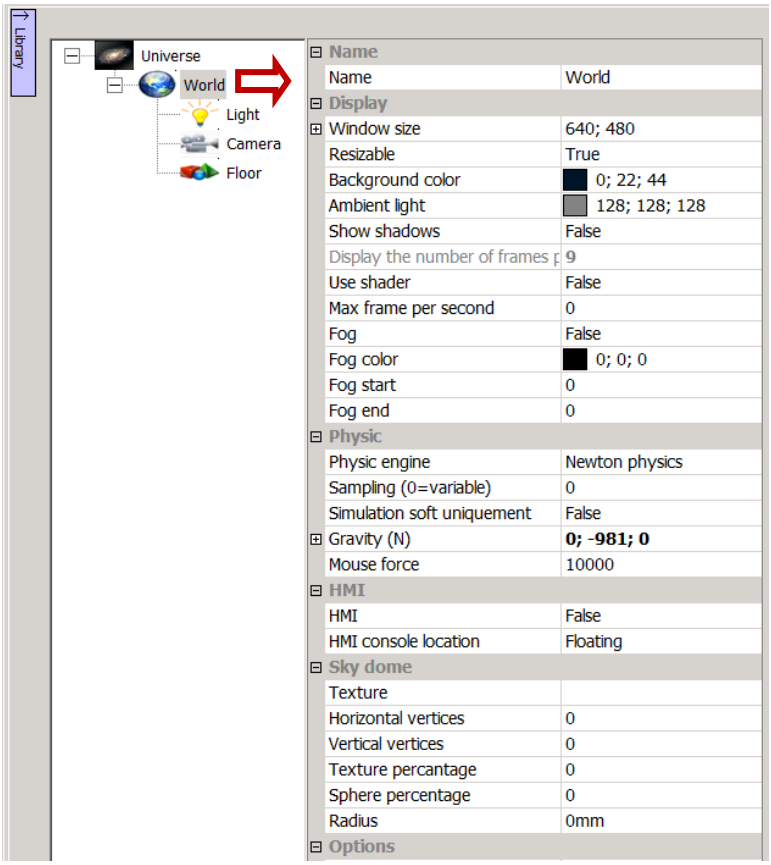
This section provides information on the 3D rendering time, on the physics engine performance and on processing of behaviors.

Security (Expert mode)

Set a password to control access to the setup window and to the project modification.

Properties of the World

Detailed properties of the World



World properties

Name (Expert mode)

Name of the World.

Display

Window size (Expert mode): Rendering window size (in pixels) in full screen mode. Full screen mode is activated by running VIRTUAL UNIVERSE PRO with /fullscreen on the command line. This is necessary for use of VIRTUAL UNIVERSE PRO with a 3D view screen and 3D glasses.

Resizable (Expert mode): Allow the user to resize the window.

Background color: Define the color of the 3D world background.

Ambient light: Determine the color and intensity of ambient light (light illuminating all objects regardless of their positions and orientations).

Show shadows: If true, handles the display of shadows, requires that the properties of objects relating to the shadows is also positioned. The display of shadows can significantly slow down the 3D rendering.

Display the number of frames per second: refresh rate.

Use shader (Expert mode): Use of a shader (advanced technology for rendering shadows and other effects. Only spot lights or directional lights must be used if true).

Max frames per second (Expert mode): If not 0, limit the number of frames displayed per second to the specified value. Also preserves CPU time.

Fog, Fog color, Fog start, Fog end (Expert mode): Display a fog effect.

Physic

Physic engine (Expert mode): Select the physic engine to be used: NEWTON Physics by default.

Sampling (0=variable) (Expert mode): Physic engine sampling period in seconds, 0 indicates a variable sampling.

Simulation soft only (Expert mode): Force a software physics simulation for the physics engine, does not use the hardware accelerator for physical simulation.

Gravity (N) (Expert mode): Gravity value in Newton.

Mouse force: Define the force used when the user picks an object in RUN mode (2000 is the default value).

HMI

HMI: Activation of a Human-Machine Interface..

HMI console location: Determine the HMI console location (Floating or Anchored below).

Sky dome (Expert mode)

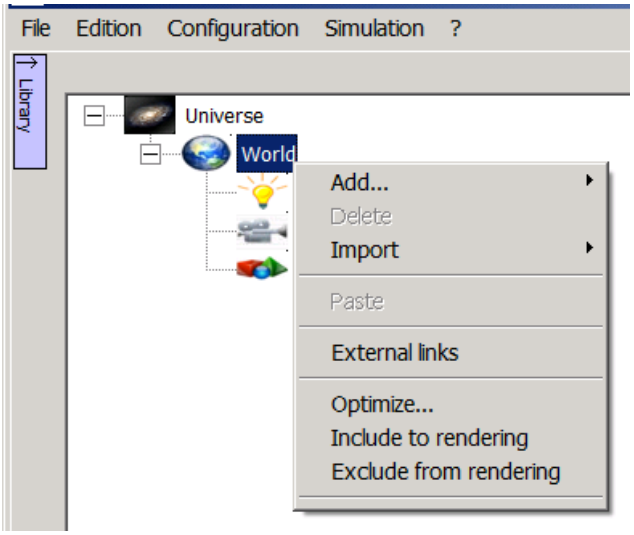
Set a texture used as sky.

Options

Global size coefficient (mode expert): Modify the size, position and speed values. By default, the global size coefficient is 100 for each axis, corresponding to a display of measurements units in millimeters.

Measure unities for length and location: Define the unities used for coordinates and size properties (by default, the millimeter).

Functionalities at the World level



Add

Add a 3D Sprite, Light or Camera as « Child » of the World.

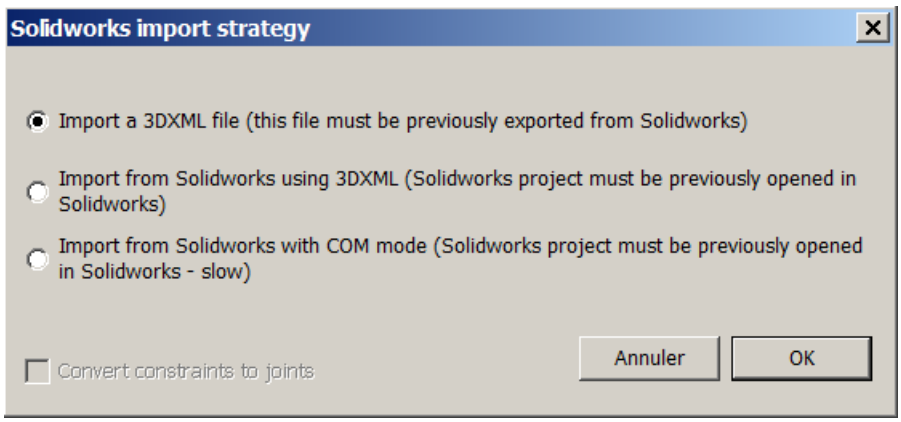
Import

An object: Import a VIRTUAL UNIVERSE PRO object (.VUO file) in the 3D emulator project. A VIRTUAL UNIVERSE PRO object is a simulation resource that is to be reused in a 3D emulator project. An object may be limited to a simple sprite or behavior, or may represent a more complex smart 3D resource (assemblies of sprites and behaviors). The smart 3D resources available in the VIRTUAL UNIVERSE PRO demonstration library are objects.

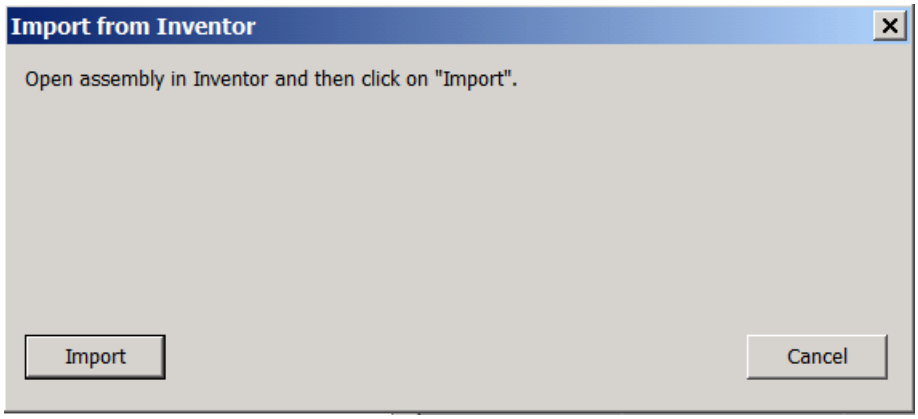
A 3D file: Import 3D files from standard formats (.3DS, VRML, STL, .OBJ files, etc...) in the VIRTUAL UNIVERSE PRO 3D emulator project.

A basic form: Provides access to the library of basic 3D shapes available in VIRTUAL UNIVERSE PRO. A basic is an object file (.VUO file) representing a basic 3D shape. VIRTUAL UNIVERSE PRO provides in its library a list of basic 3D shapes that can be reused in a 3D emulator project. The basic object files are located in the library/Primitive folder in the VIRTUAL UNIVERSE PRO Industry installation directory.

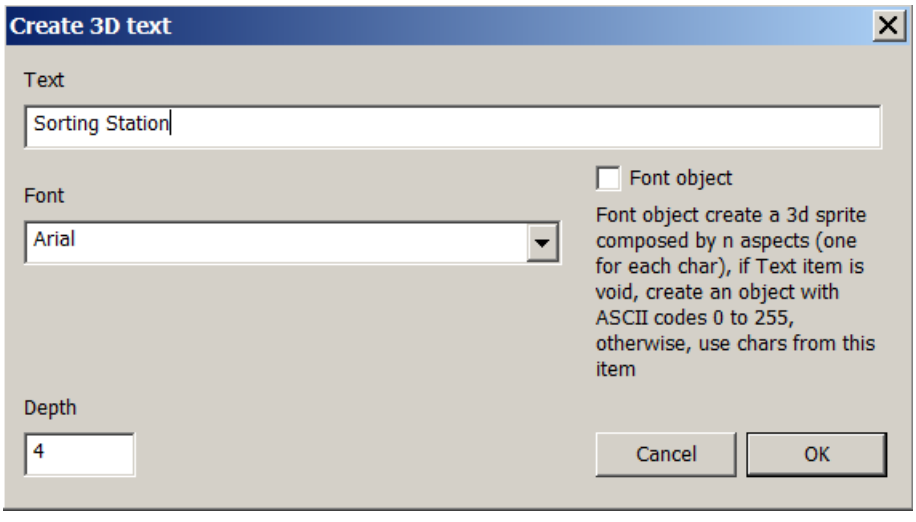
From SolidWorks: Provides access to the SolidWorks data import tool. For more information on how to import 3D models from SolidWorks, see Import SolidWorks models.



From Inventor: Provides access to the Inventor data import tool. For more information on how to import 3D models from Inventor, see Import Inventor models.

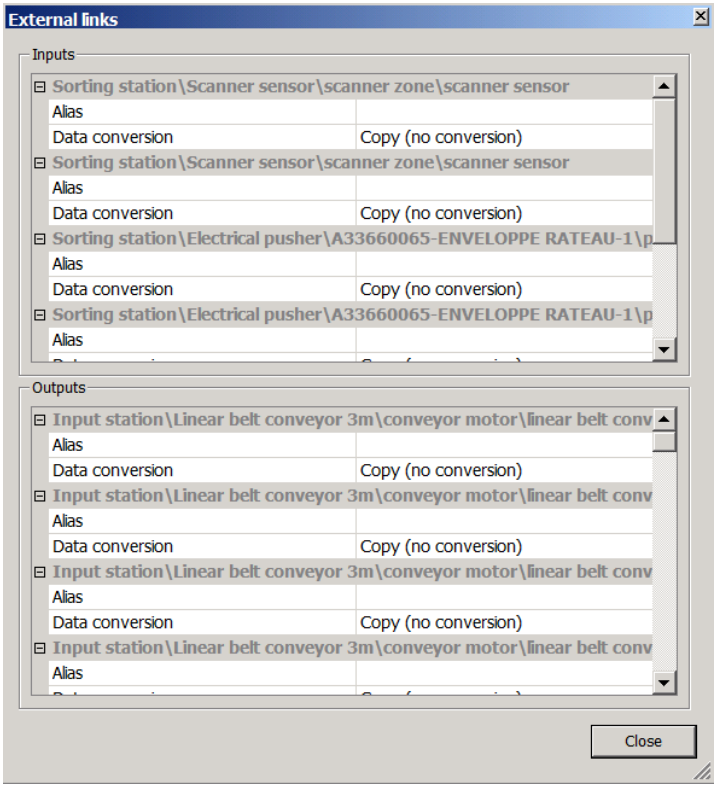


A 3D text: Add a 3D sprite in a text shape.



External links

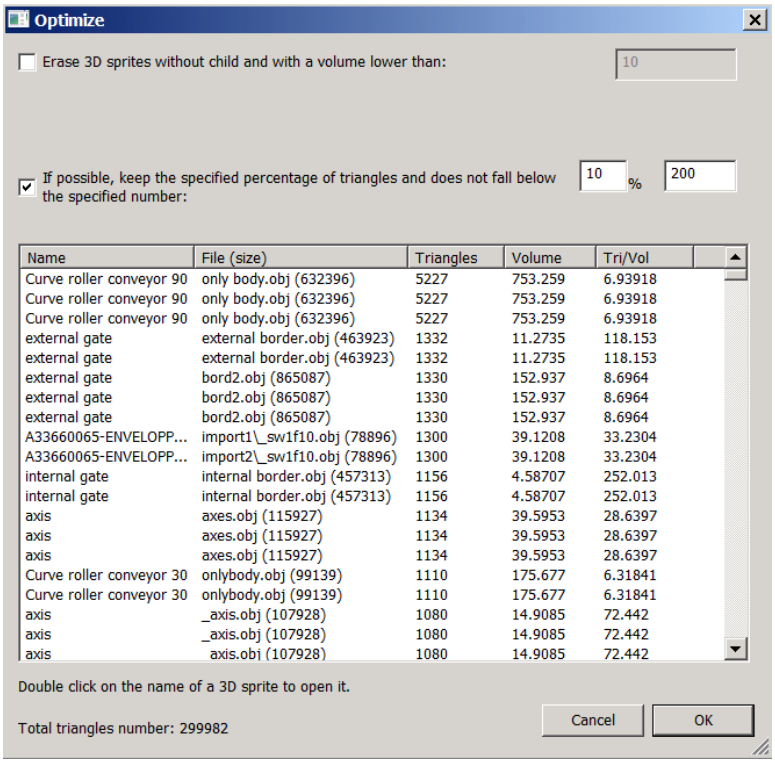
Open the External Links window, listing all behaviors defined as "External links" in the 3D emulator project (at the World level). For more information regarding the inputs/outputs of a 3D emulator, see Define the list of 3D emulator inputs/outputs.



Optimize

Open the geometries optimization tool of VIRTUAL UNIVERSE PRO, allowing user to simplify the geometry of all the 3D sprites used in the 3D emulator project (world level).

Simplifying the 3D geometries enables to optimize the performance of 3D rendering. For more information on how to use this optimization tool, see Simplify 3D CAD models.



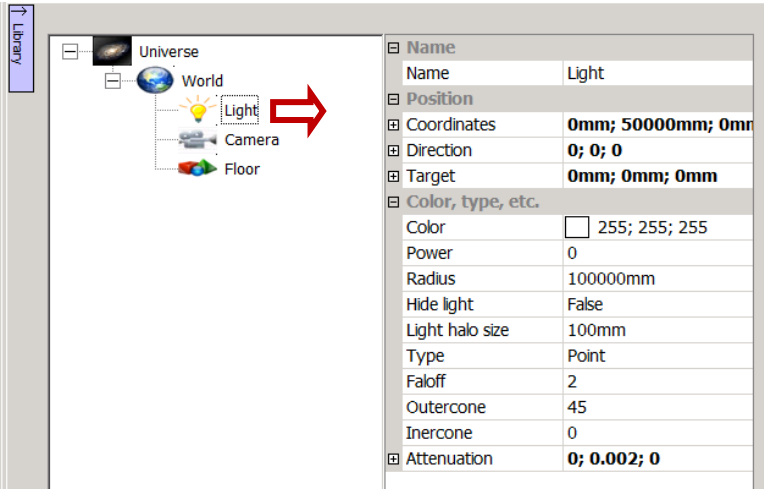
For more information on how to measure and optimize graphics performance of a 3D emulator, see Measure graphics performances.

Include to rendering / Exclude from rendering

Make all the 3D sprites visible/invisible in the 3D rendering window.

Properties of Lights

Detailed properties of Lights



Light properties

Name

Name of the Light.

Position

Coordinates: Define the Light's coordinates (unused for Directional Light).

Direction: Set the direction of the Light in degrees (only used for Spot or Directional Light).

Target (Expert mode): Only used for the shader shadows, this parameter sets the target coordinates (X, Y and Z).

Color, type, etc.

Color: Color of the Light.

Power: Multiplicative factor of the Light's power, 0 value = default (multiplication by 1).

Radius: Radius of the Light (distance of effect).

Hide light: Hide the Light spot and halo.

Light halo size: Modify only the halo size, not the power and radius of the light.

Type: Select the Light type: point (all directions lighting), spot or directional.

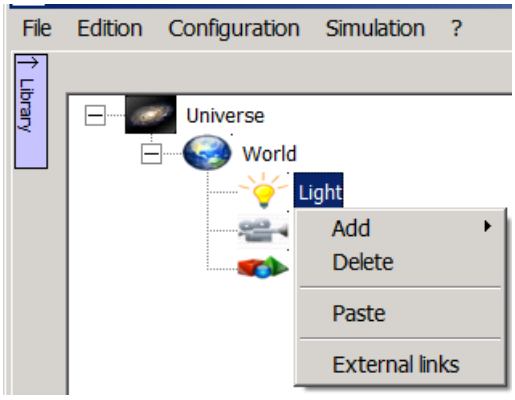
Falloff: The light strength's decrease between Outer and Inner cone.

Outercone: Angle of the outer cone for spots (ignored for other lights).

Inercone: Angle of the outer cone for spots (ignored for other lights).

Attenuation: Light strength fading over distance.

Functionalities at the Light level



Add

Add a new Behavior.

Delete

Remove the Light.

Paste

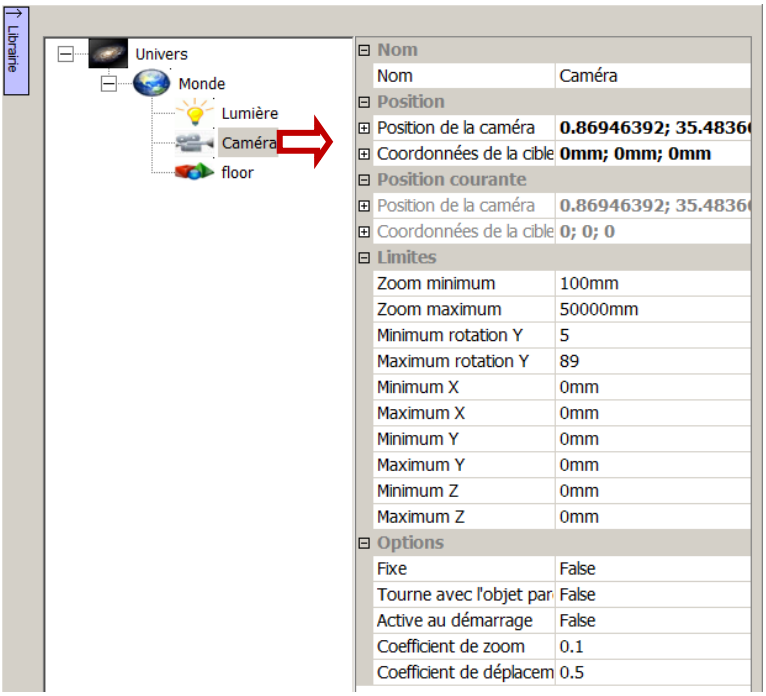
Paste the behaviors copied from another light.

External links

Open the Light External Links window, listing all behaviors reported as "External links". This can be useful to control the Light with an external variable.

Properties of Cameras

Detailed properties of Cameras



Cameras properties

Name

Name of the Camera.

Location

Determine the initial position of the Camera by the coordinates of the target (where the Camera is looking) as well as rotation about the axes X and Y and zoom.

Current location

Same as above, but for the current position. The current position can be copied into the original position by clicking the downward arrows appearing on the right elements of initial positions and selecting "Copy from current values".

Limits

Restrict the Camera's movements.

Options

Fixed: The user cannot move the Camera.

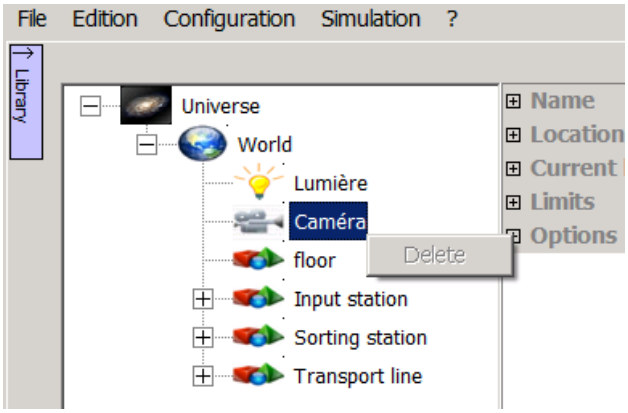
Rotate with parent: If true, the Camera moves along with its parent.

Activated when start: In multi-camera mode, sets the Camera as active.

Zoom coefficient: Increase or decrease the zoom speed (0 = default).

Move coefficient: Increase or decrease the displacement speed (0 = default).

Functionalities at the Camera level

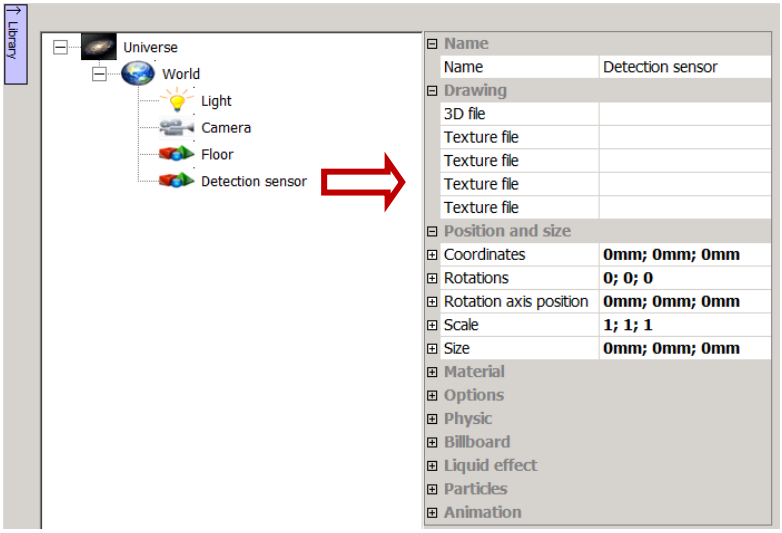


Delete

Delete the Camera.

Properties of Sprites

Detailed properties of Sprites



Properties of a Sprite

Name

Name of the Sprite.

Drawing

Determine the 3D file used to define the geometry of the 3D Sprite and possible texture files.

Position and size

Define the initial position, rotation (along the axis) and scale (expert mode) of the 3D Sprite. The rotations are in degrees (from -180 to +180 degrees).

Position and size (current values) – RUN mode

Same as above, but for the current values. Also displays the relative translation and rotation (relative to the 3D parent Sprite), the position of the object center and the absolute rotation (to the world).

Material (current values)

These properties include the characteristics of the equipment used to display the object. These characteristics are directly related to the Irrlicht rendering engine.

Transparency: Set the transparency of the object from 0 (no transparency) to 1 (full transparency).

Design both faces (Expert mode): Display the two faces of the object's surfaces.

Material (current values) – RUN mode

Same as above, but for the current values.

Options

Non selectable: If true, the 3D sprite is not selectable by mouse (useful for large 3D sprites to let smaller Sprites be selectable).

Select parent: If true, select the parent instead of the object itself when clicked.

Magnetic: Magnetic object for automatic positioning and connection of 3D Sprites.

Physic

Use physic: If true, the physics engine will process this 3D Sprite. If false, the 3D Sprite will be ignored by the physics engine and have no physical interaction with the other 3D objects.

Use gravity: If true, the 3D Sprite will be subject to gravity. Its mass must also be non-zero.

User can apply force: If true, the user can, in RUN mode, acting on the object by holding down the right mouse button while the cursor is on the Sprite 3D and moving the cursor.

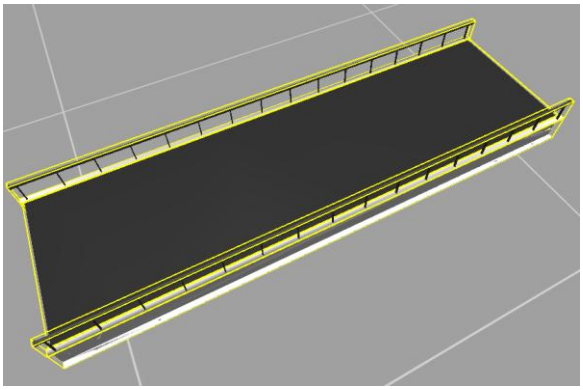
Body type: Determine the geometry type of the 3D Sprite that will be handled by the physics engine:

- Any (convex): a convex shape derived from the 3D geometry of the Sprite.
- Any (Nvidia Physx only)
- Box : rectangular parallelepiped
- Sphere
- Capsule
- Cloth (Nvidia Physx only)
- Soft body (Nvidia Physx only)
- Fluid (Nvidia Physx only).

Caution! The type "Any", when is used with a complex 3D Sprite (with many sides), may consume a lot of resources during physics simulation. Prefer, when possible one of the other types (as “Box” type).

In the development phases, it is possible and often useful to visualize the geometries handled by the physics engine by activating the "debug physic" mode in the properties of the universe.

Example:



Mass: Mass of the object (0 = freeze).

Moment of inertia (Expert mode): Determine the object’s resistance to changes to its rotation.

Linear/Angular damping (Expert mode): Linear/angular viscous friction.

Auto adjusts center of mass (Expert mode): If true, computes the center of mass from the geometric center of the object. If false, the center of mass is the point of coordinates (0,0,0).

Center of mass (Expert mode): Define the object’s center of mass.

Coefficients: Determine the static and kinetic friction, the elasticity and static softness coefficients. A zero value uses the default parameters of the physics engine. The coefficient used by the physics engine between an object A and object B is a combination (product) of the coefficients of object A and object B.

Penetrate: If true, the 3D Sprite will penetrate other 3D sprites. In the case of objects linked by joints (see below) collisions are automatically disabled.

Physic joint with parent

Joint: Define a joint with the parent of the 3D Sprite. The 3D Sprite must use physic and have a non-zero mass:

- Hinge
- Slider
- Ball and socket

- Hinge and slider
- Pulley (Nvidia Physx only)
- Fix

Pivot joint position or anchor position: Origin of the pivot or anchor solution.

Line of action: Line of action of the Joint: slide (translation axis) and pivot (rotation axis).

Limits: Minimum and Maximum positions for the joint. Limits are disabled if min limit = max limit.

Joint power: Power (rigidity) of the joint.

Joint force (N) (Expert mode): Force supported by the joint.

Break joint force (N) (Expert mode): If joint force is greater than this value, the joint is breaks.

Pivot joint position or anchor position (parent or other objects): Origin of the pivot or anchor position

Pulley (Expert mode): Options for the creation of a pulley.

Billboard (Expert mode)

2D graphic file displayed in the 3D world.

Liquid effect (Expert mode)

Simulate a liquid effect.

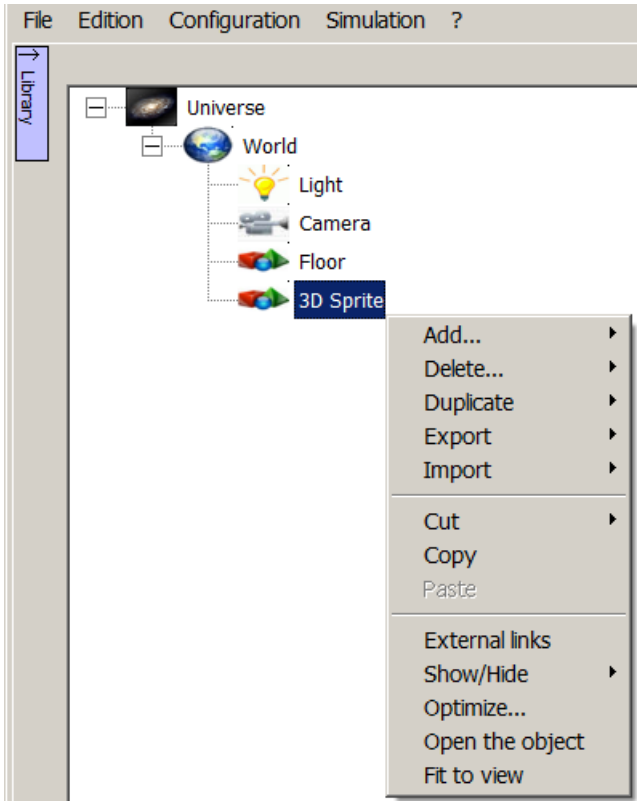
Particles (Expert mode)

Display particles (simulates small moving objects: water, fire…).

Animation (Expert mode)

Manage the animation defined in a 3D file (only available for .x files).

Functionalities at the Sprite level



Add

Add a new 3D Sprite, Light, Camera or behavior as a child of the 3D Sprite.

Delete

Delete the 3D Sprite (ability to delete a group of 3D Sprite with “This 3d sprite and the next…”).

Duplicate

Online or chained duplication of 3D resources. Associated with the "Magnetic" function, this option is particularly useful for the construction of a conveyor line.

Export

Export a Sprite (with eventual behavior and children) as a VIRTUAL UNIVERSE PRO reusable object (.vuo file), just like a smart 3D resource. By default, VIRTUAL UNIVERSE PRO proposes to save this object in the VIRTUAL UNIVERSE PRO library directory, but it is possible to select another location on the computer.

It is also possible to export only the children's behavior of the selected sprite.

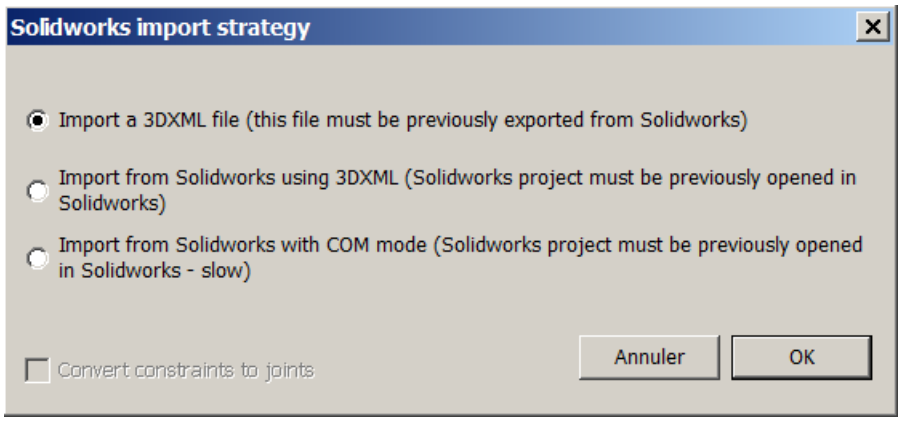
Import

An object: Import a VIRTUAL UNIVERSE PRO object (.VUO file) in the 3D emulator project. A VIRTUAL UNIVERSE PRO object is a simulation resource that is to be reused in a 3D emulator project. An object may be limited to a simple sprite or behavior, or may represent a more complex smart 3D resource (assemblies of sprites and behaviors). The smart 3D resources available in the VIRTUAL UNIVERSE PRO demonstration library are objects.

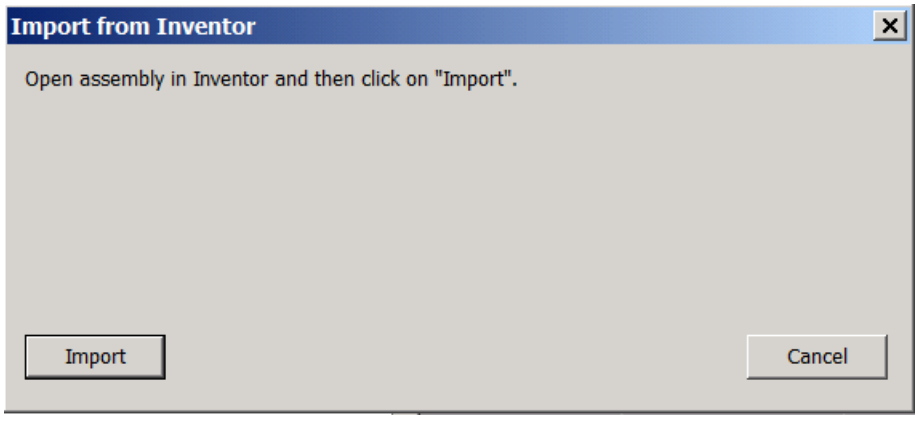
A 3D file: Import 3D files from standard formats (.3DS, VRML, STL, .OBJ files, etc...) in the VIRTUAL UNIVERSE PRO 3D emulator project.

A basic form: Provides access to the library of basic 3D shapes available in VIRTUAL UNIVERSE PRO. A basic is an object file (.VUO file) representing a basic 3D shape. VIRTUAL UNIVERSE PRO provides in its library a list of basic 3D shapes that can be reused in a 3D emulator project. The basic object files are located in the library/Primitive folder in the VIRTUAL UNIVERSE PRO Industry installation directory.

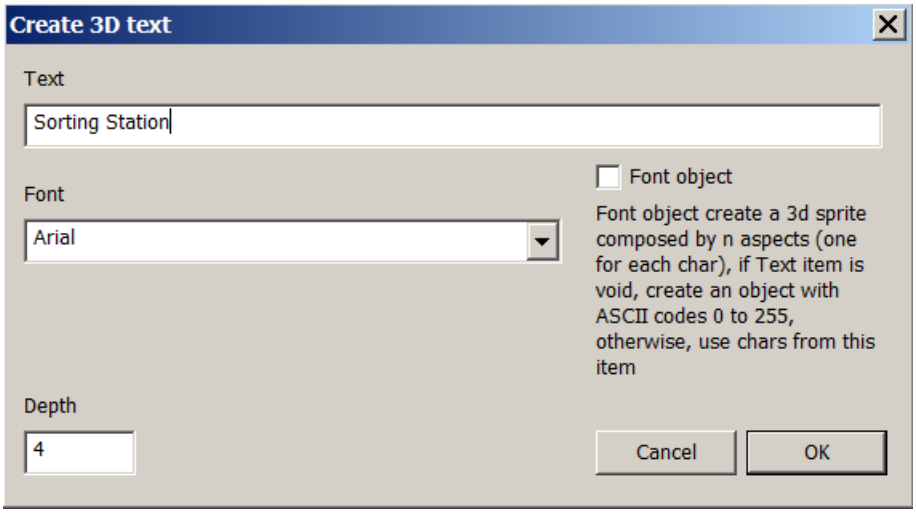
From SolidWorks: Provides access to the SolidWorks data import tool. For more information on how to import 3D models from SolidWorks, see Import SolidWorks models.



From Inventor: Provides access to the Inventor data import tool. For more information on how to import 3D models from Inventor, see Import Inventor models.

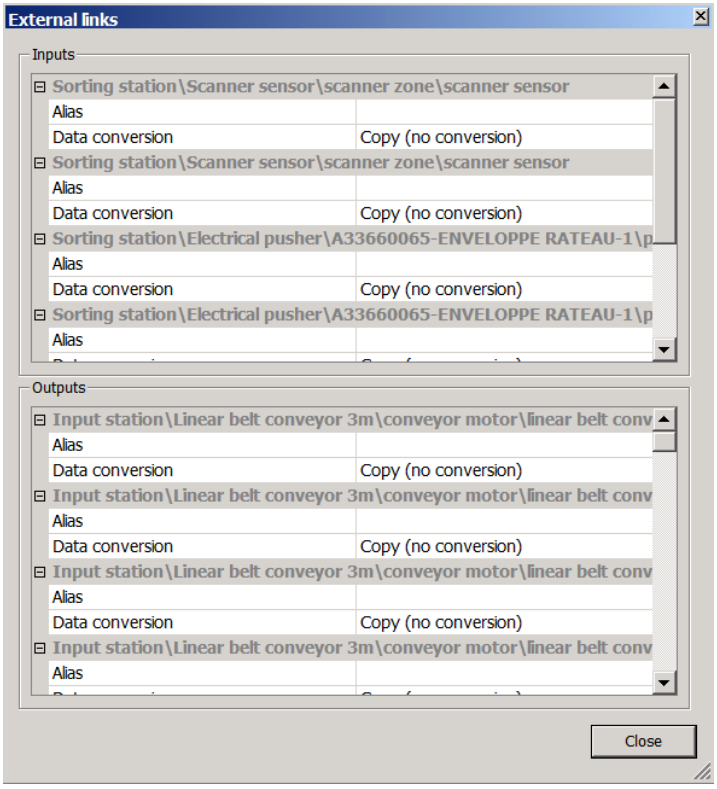


A 3D text: Add a 3D sprite in a text shape.



External links

Open the External Links window, listing all behaviors defined as "External links" at the selected sripte level.



Show/Hide

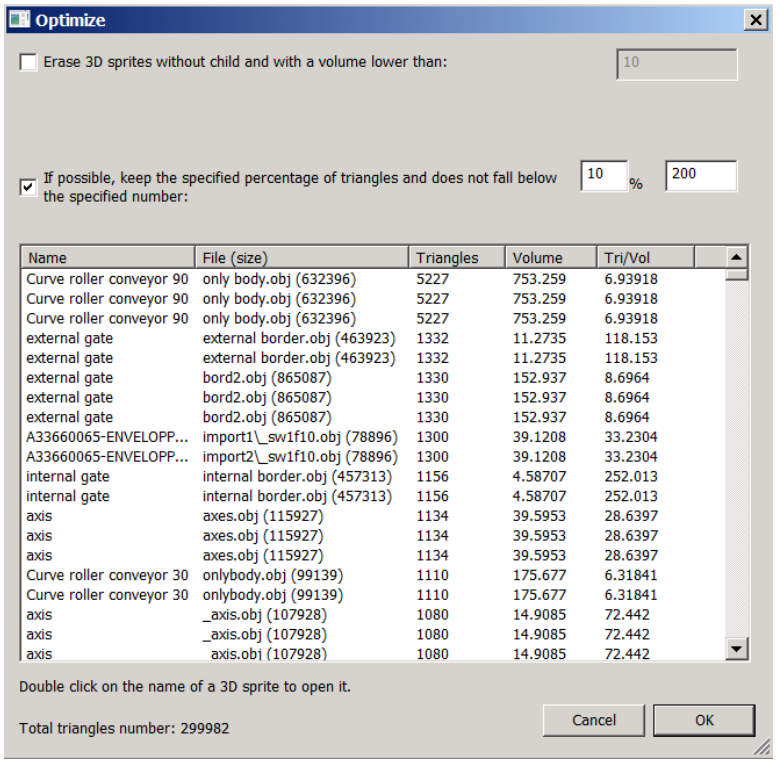
Show, hide or exclude a 3D resource from the 3D rendering windows.

Optimize

Provides access to the geometries optimization tool of VIRTUAL UNIVERSE PRO, enabling user to simplify the 3D geometry of the selected Sprites (including its children).

Simplifying the 3D geometries enables to optimize the performance of 3D rendering.

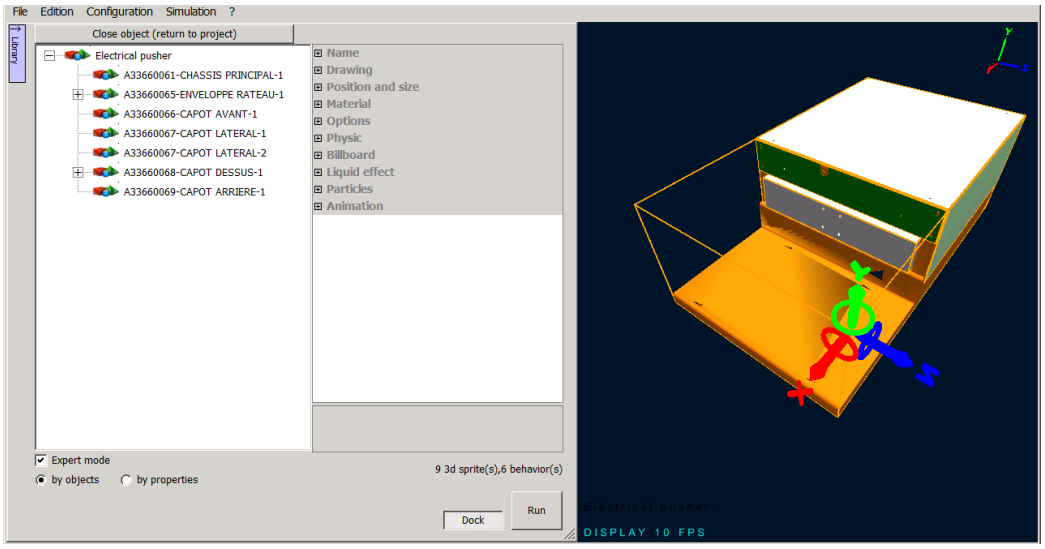
For more information on how to use this optimization tool, see Simplify 3D CAD models.



For more information on how to measure and optimize graphics performance of a 3D emulator, see Measure graphics performances.

Open the object

Show only the selected object in the 3D rendering window and in the project tree

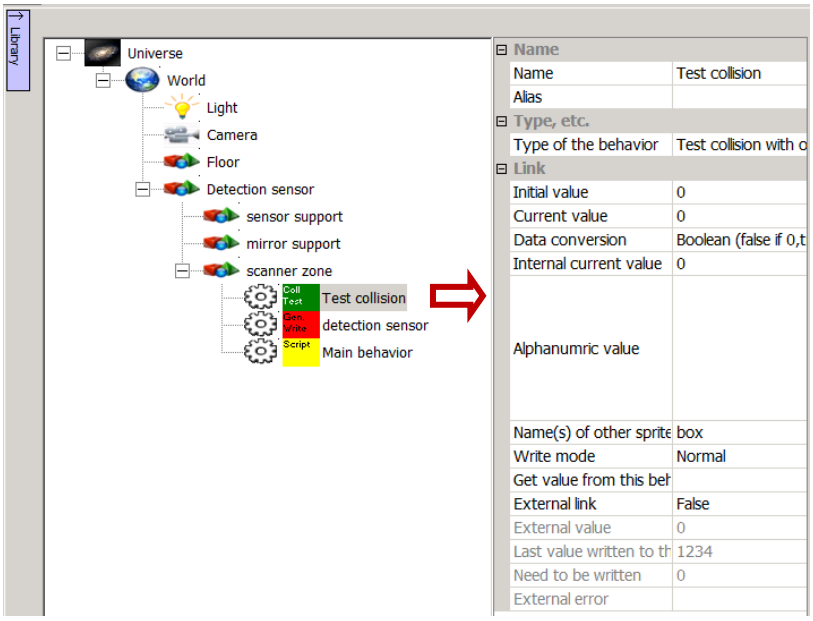


Fit to view

Center the view on the selected 3D Sprite.

Properties of Behaviors

Detailed properties of Behaviors



Behaviors properties

Name

Name of the Behavior

Type, etc.

Behavior type (see details below)

Link

Initial value: Will be copied into the current value when switching to simulation RUN mode. Can be used for a permanent activation of a behavior. For example, a script will be executed unconditionally from the start of the simulation by setting this property to 1.

Current value: Current value of the Behavior's variable, this value can be copied from or copied to the external software.

Data conversion: Determines how the data is converted between the current value and the internal current value:

- Copy (no conversion)
- Boolean (false if 0, true otherwise)

- Non Boolean (true if 0, false otherwise)
- From real world
- To real world

Internal current value: Internal current value of the Behavior's variable, this variable is used to drive the simulation or this value comes from the simulation.

Alphanumeric value (Expert mode): Used to store a string value.

Write mode: Define the method of writing data to the external software:

- Normal: data is written on each exchange cycle.
- Only when changed: data is only written on a value change.
- Safe: Data is read before writing and is not written if the value is the same. When writing data, verification is made to check that the value is correctly written.

Get value from this behavior: If nonempty, this area gives the name of a behavior whose value will be read and copied into the internal current value.

External link: If true, this Behavior will be displayed in the I/O list.

External value (Expert mode): Used by the "only when changed" mode.

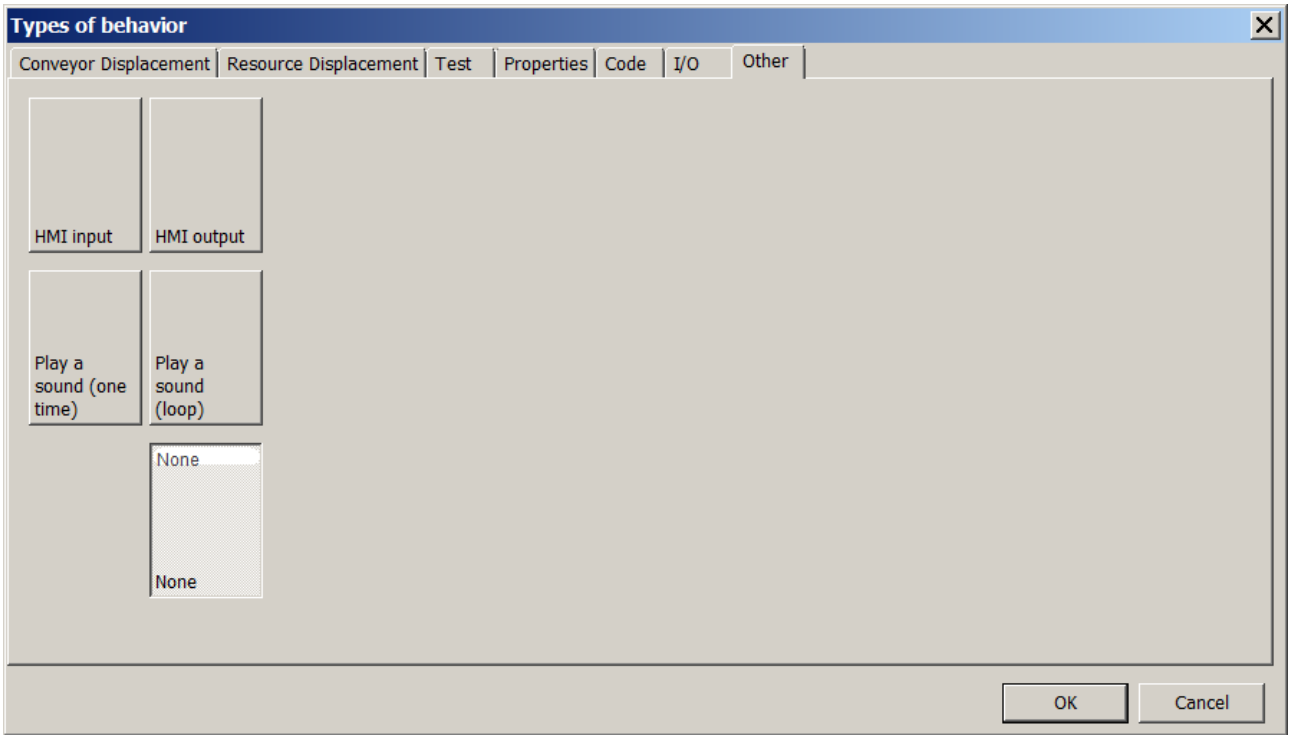
Last value written to the external software (Expert mode): Used by the "safe" mode.

Need to be written (Expert mode): Used by the "safe" mode.

External error (Expert mode): Error message if an error occurred while accessing an external variable. Empty if no error.

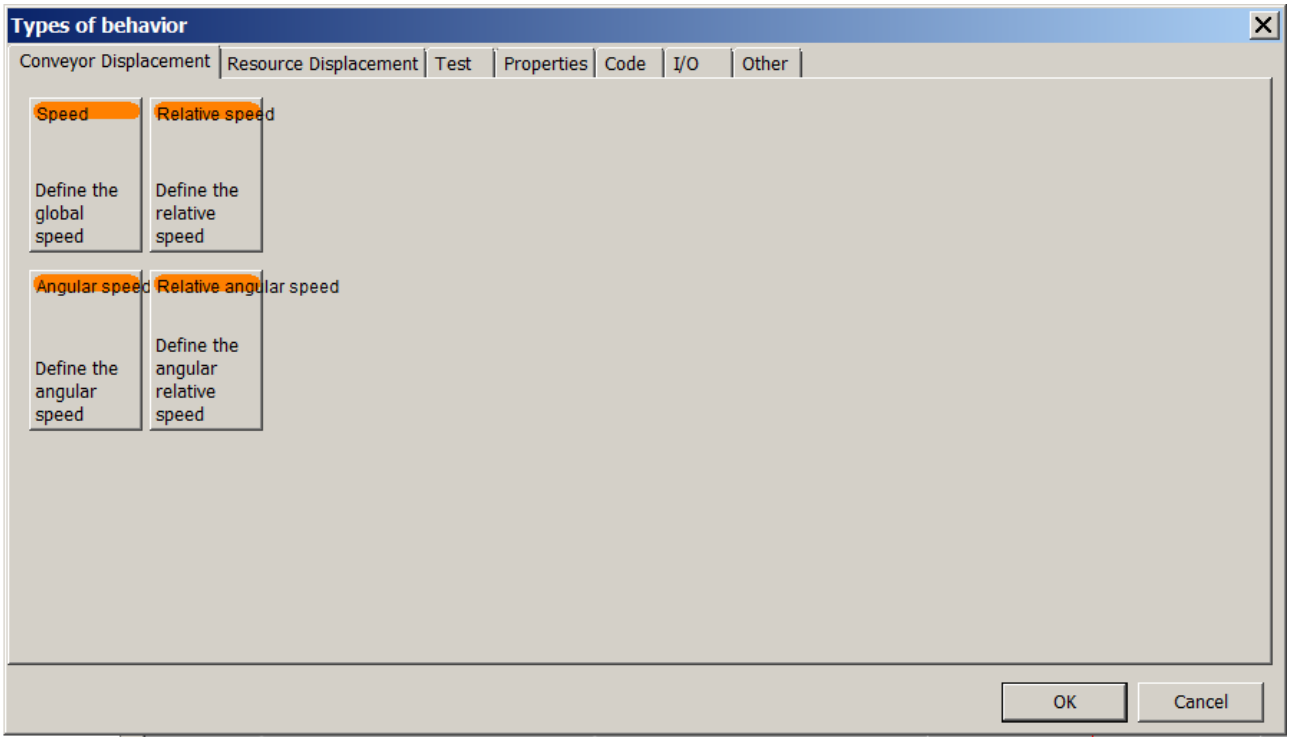
Types of Behavior

VIRTUAL UNIVERSE PRO offers a wide variety of predefined behaviors to be applied to 3D sprites. These behaviors are organized in 7 categories presented in the "Type of behavior" window.



Conveyor displacement

These behaviors are used to control the conveyor belt speed (in translation or rotation, depending on the conveyor shape).



Speed:

- Set the linear velocity of the sprite relatively to the reference axis of the World (absolute axis system). In meters/sec or mm/s according to the units selected in the properties of the world.

  Important! This behavior relies on the VIRTUAL UNIVERSE PRO physics engine. It requires the option "Physics" to be checked for the sprite and that a slide joint is defined between the sprite (sprite moved) and the parent sprite.

Relative speed:

- Sets the linear velocity of the sprite relative to the axes of the local coordinate system relative to the sprite (local frame). In meters/sec or mm/s according to the units selected in the properties of the world.

  Important! This behavior relies on the VIRTUAL UNIVERSE PRO physics engine. It requires the option "Physics" to be checked for the sprite and that a slide joint is defined between the sprite (sprite moved) and the parent sprite.

  Example

This behavior is used in the conveyor types "Linear belt conveyor" and "Linear Roller conveyor" available in VIRTUAL UNIVERSE PRO demo library, to manage the conveyor belt speed.

Angular speed:

- Defines the angular velocity of the Sprite relative to the reference axes of the World (absolute frame). In degrees/s.

  Important! This behavior relies on the VIRTUAL UNIVERSE PRO physics engine. It requires the option "Physics" to be checked for the sprite and that a revolute (hinge) joint is defined between the sprite (sprite moved) and the parent sprite.

Relative angular speed:

- Defines the angular velocity of the Sprite relative to the axes of the local coordinate system relative to the sprite (local frame). In degrees/s.

  Important! This behavior relies on the VIRTUAL UNIVERSE PRO physics engine. It requires the option "Physics" to be checked for the sprite and that a revolute (hinge) joint is defined between the sprite (sprite moved) and the parent sprite.

  Example

  This behavior is used in the conveyor types "Curve roller conveyor" available in VIRTUAL UNIVERSE PRO demo library, to manage the conveyor belt speed.

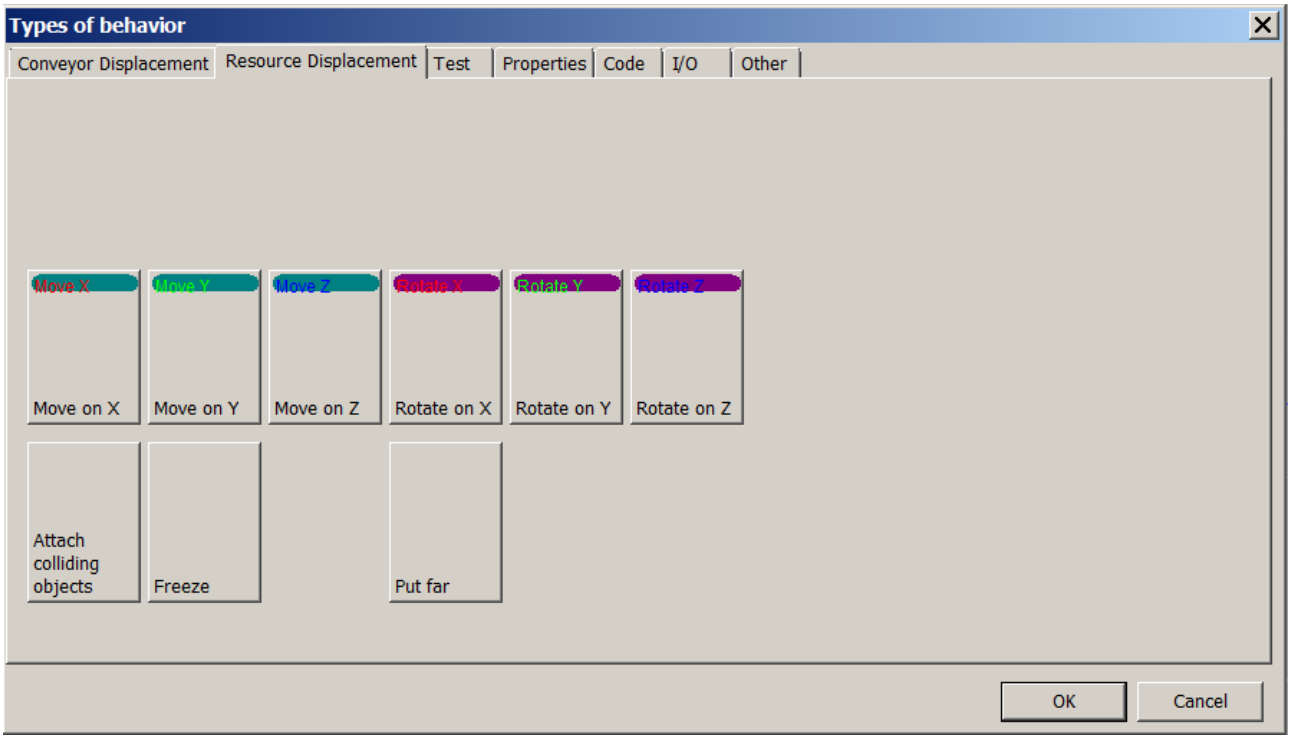
Resource displacement

These behaviors are intended to control the movement of 3D objects and 3D resources (translation, rotation, position).

These are the behaviors the most widely used to manage the resources movements in the VIRTUAL UNIVERSE PRO 3D emulators.

Most of these behaviors can be created and configured automatically using the Motion Assistant. For details, see Define motion profiles with the Motion Assistant.

These behaviors do not rely on the VIRTUAL UNIVERSE PRO physics engine and do not require a kinematic joint between the child and parent sprites.



Move on X:

- Move the sprite in translation relative to the axis X of the parent sprite local frame (local frame).

  The property “Move and rotate mode" lets you choose the drive mode for this displacement (Time, Speed, Position).

  Speed is in meters/sec or mm/s depending on the units selected in the properties of the world.

  Important! This behavior does not rely on the VIRTUAL UNIVERSE PRO physics engine. However, the Physics option can be enabled on the sprite, to take into account the potential collisions of this sprite with other sprites.

  ### Example

  The behavior "Move on X" in control mode "Time" is used in the electrical pusher "Electrical pusher" present library, to control the drawer return/release.

Move on Y:

- Move the sprite in translation relative to the axis X of the parent sprite local frame (local frame).

  The property “Move and rotate mode" lets you choose the drive mode for this displacement (Time, Speed, Position).

  Speed is in meters/sec or mm/s depending on the units selected in the properties of the world.

  Important! This behavior does not rely on the VIRTUAL UNIVERSE PRO physics engine. However, the Physics option can be enabled on the sprite, to take into account the potential collisions of this sprite with other sprites.

Move on Z:

- Move the sprite in translation relative to the axis Z of the parent sprite local frame (local frame).

  The property “Move and rotate mode" lets you choose the drive mode for this displacement (Time, Speed, Position).

  Speed is in meters/sec or mm/s depending on the units selected in the properties of the world.

  Important! This behavior does not rely on the VIRTUAL UNIVERSE PRO physics engine. However, the Physics option can be enabled on the sprite, to take into account the potential collisions of this sprite with other sprites.

Rotate X:

- Move the sprite in rotation relative to the axis X of the parent sprite local frame (local frame).

  The property “Move and rotate mode" lets you choose the drive mode for this displacement (Time, Speed, Position).

  Speed is in degrees/seconds.

  Important! This behavior does not rely on the VIRTUAL UNIVERSE PRO physics engine. However, the Physics option can be enabled on the sprite, to take into account the potential collisions of this sprite with other sprites.

Rotate Y:

- Move the sprite in rotation relative to the axis Y of the parent sprite local frame (local frame).

  The property “Move and rotate mode" lets you choose the drive mode for this displacement (Time, Speed, Position).

  Speed is in degrees/seconds.

  Important! This behavior does not rely on the VIRTUAL UNIVERSE PRO physics engine. However, the Physics option can be enabled on the sprite, to take into account the potential collisions of this sprite with other sprites.

Rotate Z:

- Move the sprite in rotation relative to the axis Z of the parent sprite local frame (local frame).

  The property “Move and rotate mode" lets you choose the drive mode for this displacement (Time, Speed, Position).

  Speed is in degrees/seconds.

  Important! This behavior does rely on the VIRTUAL UNIVERSE PRO physics engine. However, the Physics option can be enabled on the sprite, to take into account the potential collisions of this sprite with other sprites.

Attach colliding objects:

- Attach the selected 3D sprite with other sprites whose name is specified in the Link/Name(s) field.

  Important! This behavior has to be applied to the moving sprite, which comes in contact with other sprites. The name of sprites to be attached must be filled.

Freeze:

- Freezes the object.

  Important! This behavior relies on the VIRTUAL UNIVERSE PRO physics engine. It requires the option "Physics" to be checked for the sprite.

  Example

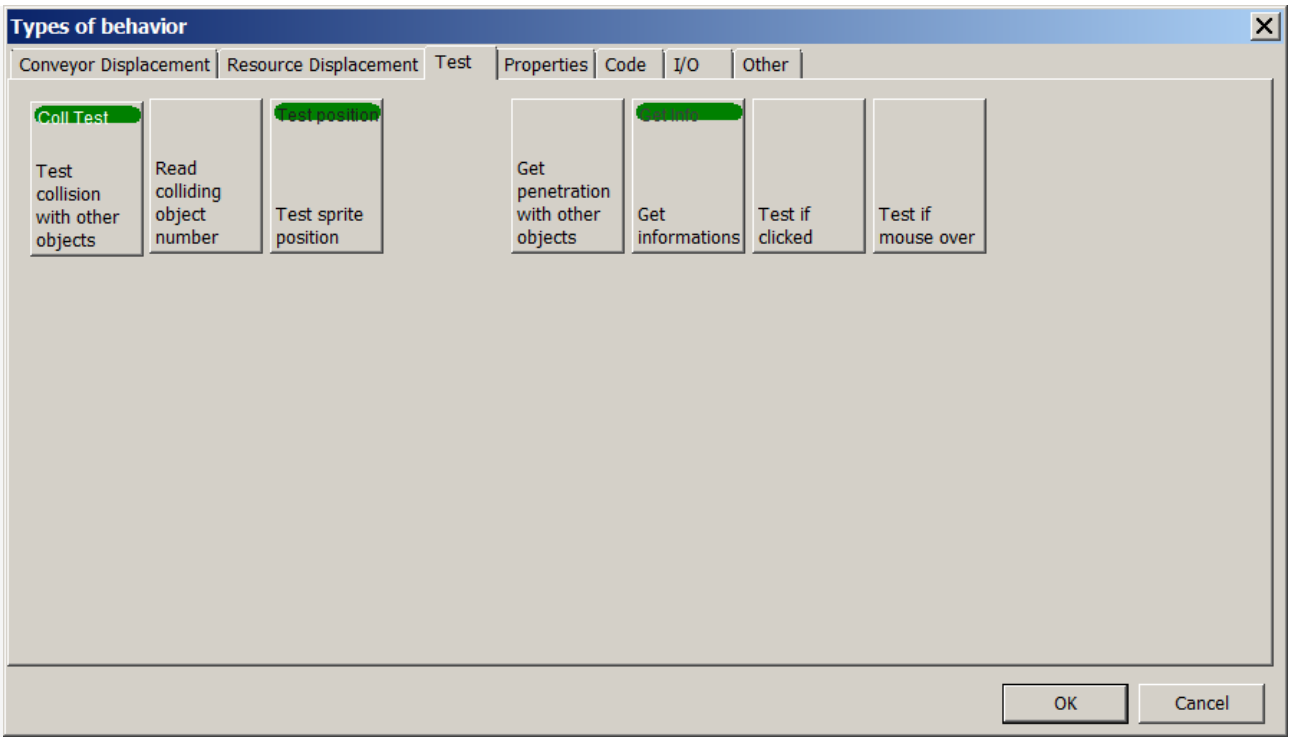
This behavior is used in the conveyors available in VIRTUAL UNIVERSE PRO demo library, to freeze the conveyor belt and bring it back to its initial position, when moved.

 Put far:

- Place the 3D Object "far" by moving it off to a distant position (not visible).

Test

These behaviors are designed to retrieve information about the 3D objects during simulation (position or identifier of an object, detection an object by a sensor, collision detection...).



 Test collision with other objects:

- Enables to detect the collision of this spite with other 3D sprites. The internal current value returns the number of triangles involved in the collision.

Important! The parameter "Name(s) other sprite" limits the action of this behavior to a group of 3D sprites. The name of monitored sprites has to be specified.

Read colliding object number:

- Provides the identifier of the 3D sprite colliding with the current sprite.

Example

The scanner sensor available in the VIRTUAL UNIVERSE PRO library operates this behavior.

Test sprite position:

- Enables to test whether the 3D Sprite position is between two bounds called “Min position” and "Max Position". This behavior can model a position sensor located on an actuator.

Important! This behavior relies on the VIRTUAL UNIVERSE PRO physics engine. It requires the option "Physics" to be checked for the sprite and a kinematic joint to be defined between the sprite and its parent.

Get penetration with other objects:

- Gives the penetration depth between the 3D Sprite associated with the behavior and other 3D sprites. The parameter "Name (s) of other sprites" limits the action of this behavior to a group of 3D sprites.

Important! This behavior relies on VIRTUAL UNIVERSE PRO physics engine. It requires the option "Physics" to be checked for the sprite. Frequent use of this behavior may decrease the physics engine performances.

Example

This behavior can model a proximity sensor.

Get information:

- Provides access to dynamic values belonging to a 3D Sprite. The parameter "Select information to be read from the 3D sprite" determines the information received.

Test if clicked:

- Takes the value 1 if the user clicks on the associated object (with the left mouse button).

  Important! This behavior only works if the sprite is selectable by the mouse (option Not selectable = false)

  Example

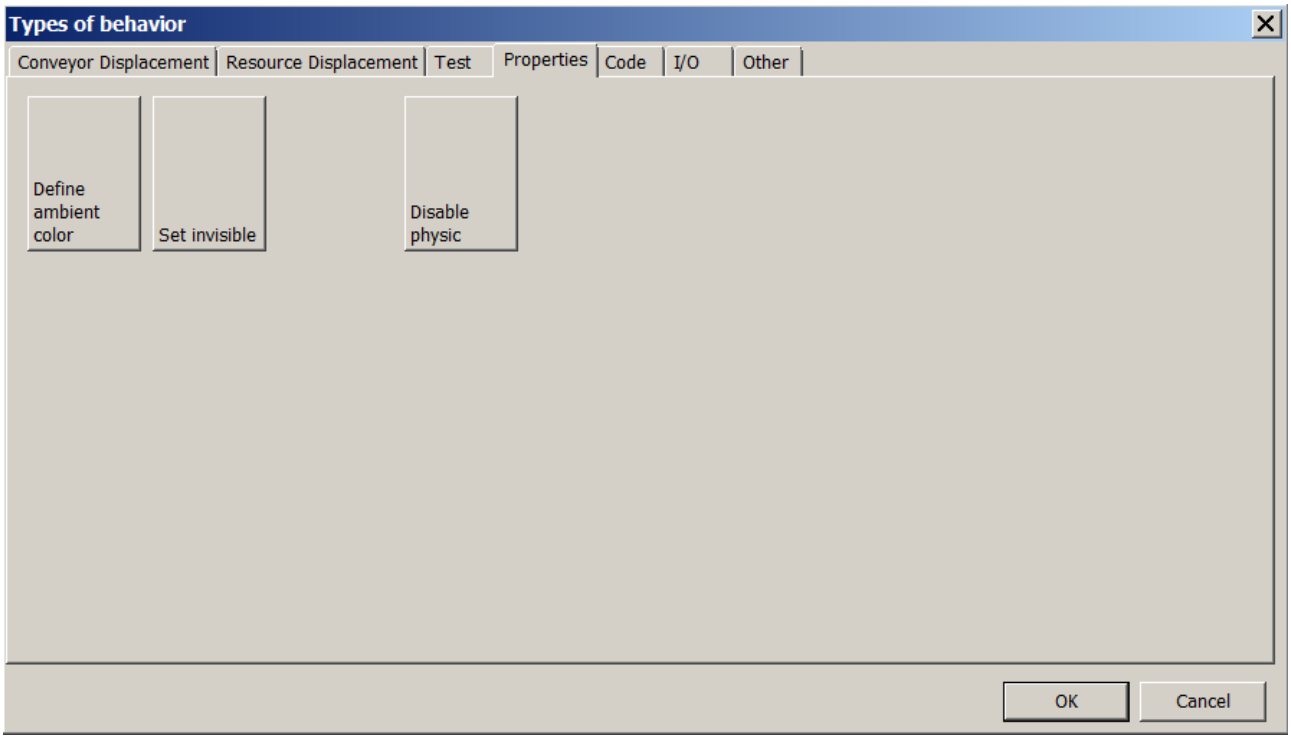
  The 3D resource « Electrical Pusher » available in the library uses this behavior.

Test if mouse over:

- Takes the value 1 if the user rolls over the object with the mouse.

Properties

These behaviors are used to modify some properties.



Define ambient color:

- Sets the ambient color of a 3D Sprite.

Important! This behavior colors the sprite materials. The coloring of these materials has to be enabled in the sprite options.

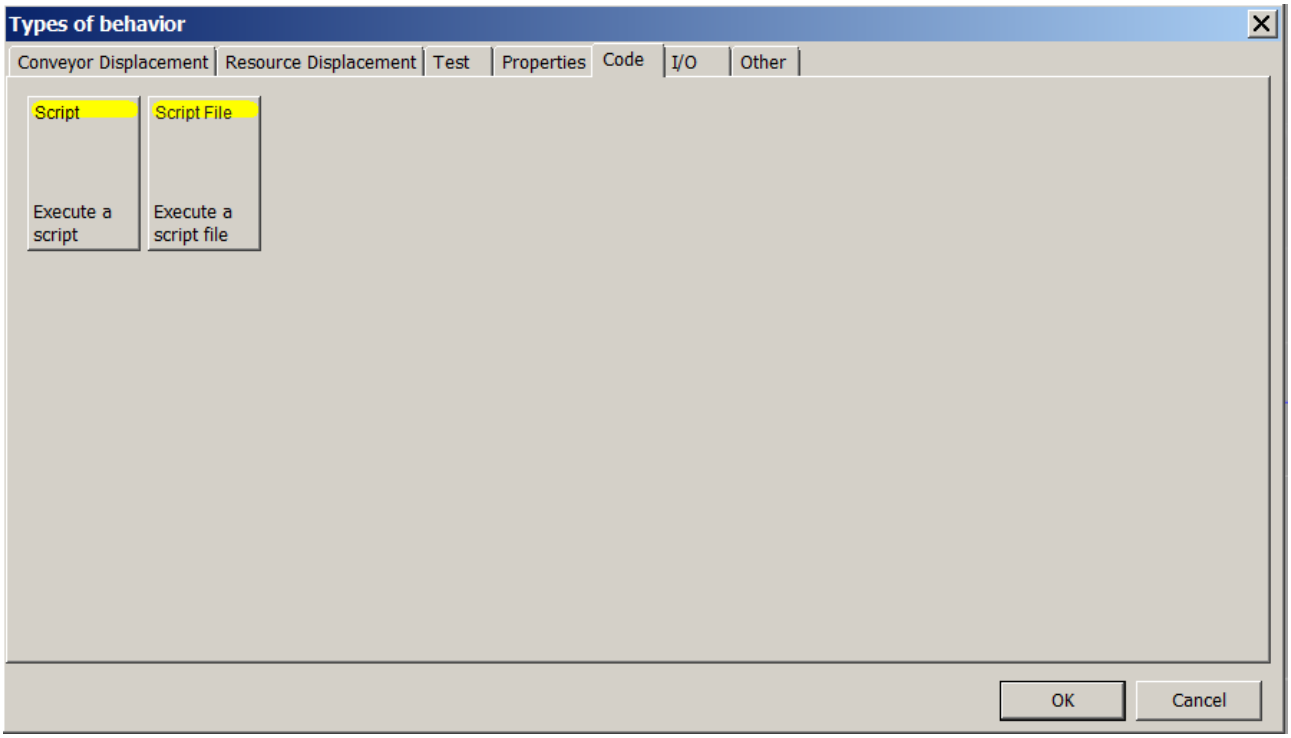
Set invisible:

- Makes the object invisible if enabled (a value of 1).

Disable physic:

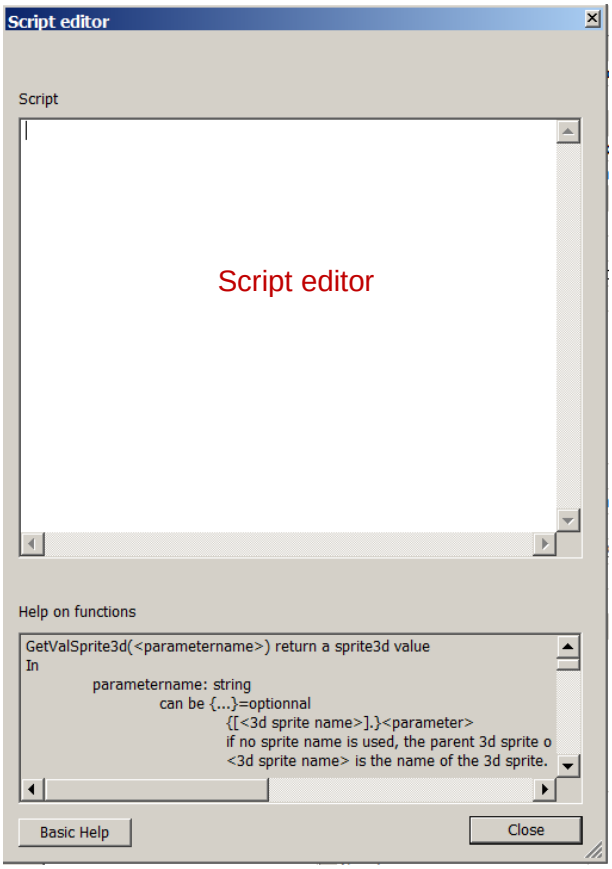
- Disables physics for the 3D Sprite if activated (a value of 1).

Code



Execute a script:

- Runs the script written in the tab "Code" if the behavior's current value is different from 0.



The language used in the script editor is the Beebasic. For more information, please refer to the basic_api.chm file available in VIRTUAL UNIVERSE PRO installation directory and to the “Help on functions” field under the script editor.

Here is a list of the main functions used by the script editor:

- **Getbehavior**(<parameter>) : returns a value associated to a Behavior.

  <parameter> refers to the parameter. It may designate a 3d Sprite by its name. If this is not the case, the 3d Sprite parent of the Behavior is used. The syntax is [<3d sprite name>].<parameter name>.

- **SetBehavior**(<parameter>,<value>) : writes a Behavior value

- **GetValSprite3d**(<parameter>): returns a value associated to a 3d Sprite.

- **SetValSprite3d**(<parameter>,<valuer>): modifies a value associated to a 3d Sprite.

- **Getuniverse**("time") : Returns the current 'time”.

The name for the reference to a behavior must respect the following syntax:

- A name without a path: will seek the Behavior whose name starts with that text in all Behaviors (in the World).
- ..\<behavior name>: A behavior brother of the Behavior.
- <3d sprite name>\<behavior name>: A behavior child of a 3d sprite.
- It is also possible to reference the value type as follows: [<behavior name>].<value type>. The value types are: « internalvalue », « currentvalue » and «values».

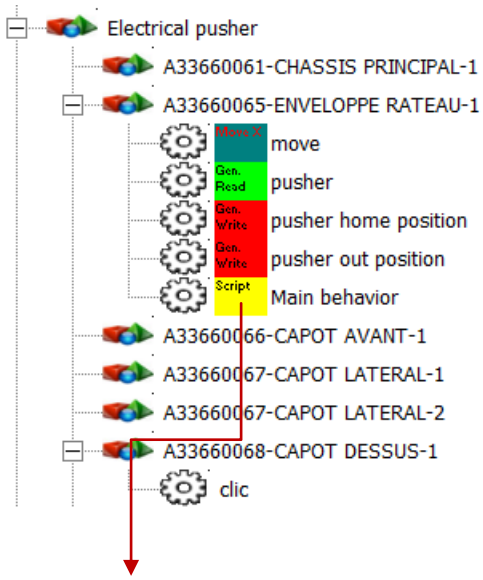
Execute a script file:

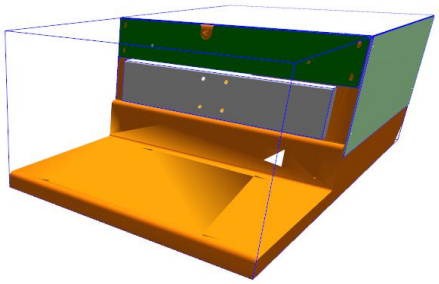
- Runs a script contained in a file if the current value of the behavior is different from 0.
  This solution is particularly useful if a script is too long for the integrated script editor.

Example 1

Case of the « Electrical Pusher » resource available in the VIRTUAL UNIVERSE PRO resource library.

The "Main behavior" script is used to control the drawer retraction/release by writing the "move" behavior based on an input order coming from the "pusher" behavior. It also returns the drawer current position ("pusher home position" and "pusher out position" external variables).





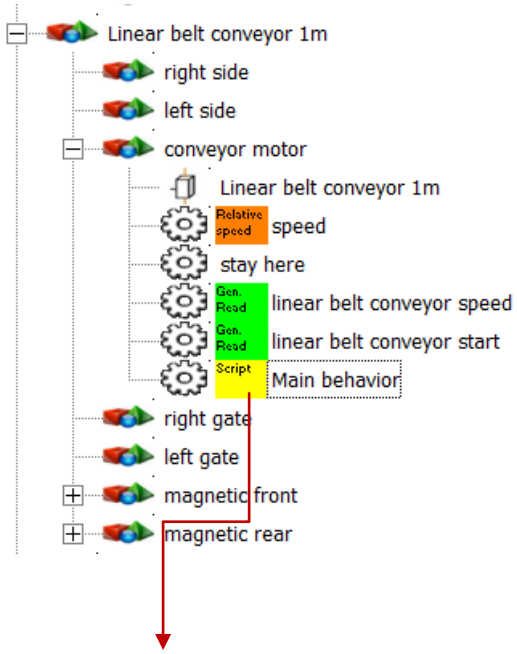
Main behavior

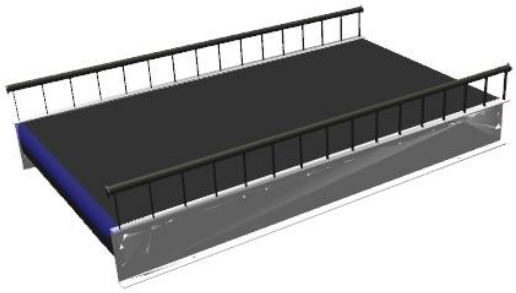
```
myloop:
curpos=GetValSprite3d("POSX")
if getbehavior("..\pusher.currentvalue")=true or getbehavior("[..\..\A33660068-CAPOT DESSUS-
1\clic].currentvalue")=true then
setbehavior("..\move.values",1)
else
setbehavior("..\move.values",0)
endif
if curpos>=5 then
setbehavior("..\pusher home position.values",1)
else
setbehavior("..\pusher home position.values",0)
endif
if curpos<=-1.2 then
setbehavior("..\pusher out position.values",1)
else
setbehavior("..\pusher out position.values",0)
endif
goto myloop
```

Example 2

Case of the « Linear belt conveyor » resource available in the VIRTUAL UNIVERSE PRO resource library.

The « Main behavior » script manages the conveyor belt speed by writing the « speed » behavior based on an order coming from the two inputs («linear belt conveyor speed » and « linear belt conveyor start »).





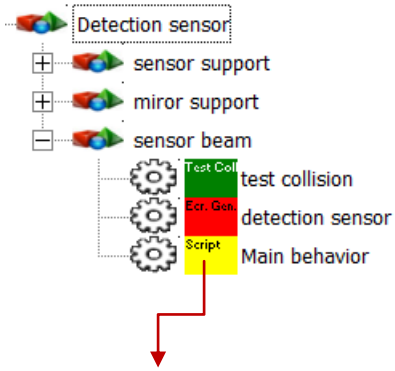
Main behavior

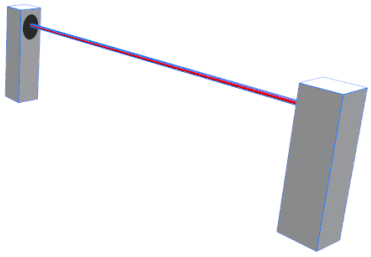
myloop:
conveyor_speed=getbehavior("[..\linear belt conveyor speed]")
conveyor_start=getbehavior("[..\linear belt conveyor start]")
if conveyor_start=true then
setbehavior("[..\speed].values",conveyor_speed)
else
setbehavior("[..\speed].values",0)
endif
goto myloop

Exemple 3

Case of the « Detection sensor » resource available in the VIRTUAL UNIVERSE PRO resource library.

The « Main behavior » is used to copy the collision test result from the « test collision » behavior to the « detection sensor » behavior.





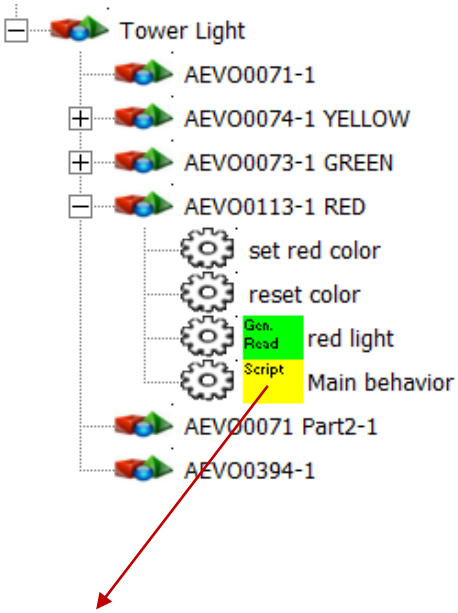
Main behavior

collision=getbehavior("..\test collision")
setbehavior("..\detection sensor.values", collision)

Exemple 4

Case of the « Tower light » resource available in the VIRTUAL UNIVERSE PRO resource library.

The RED lamp « Main behavior » handles the light flashing (0.5s period) based on the « red light » input signal.



Main behavior

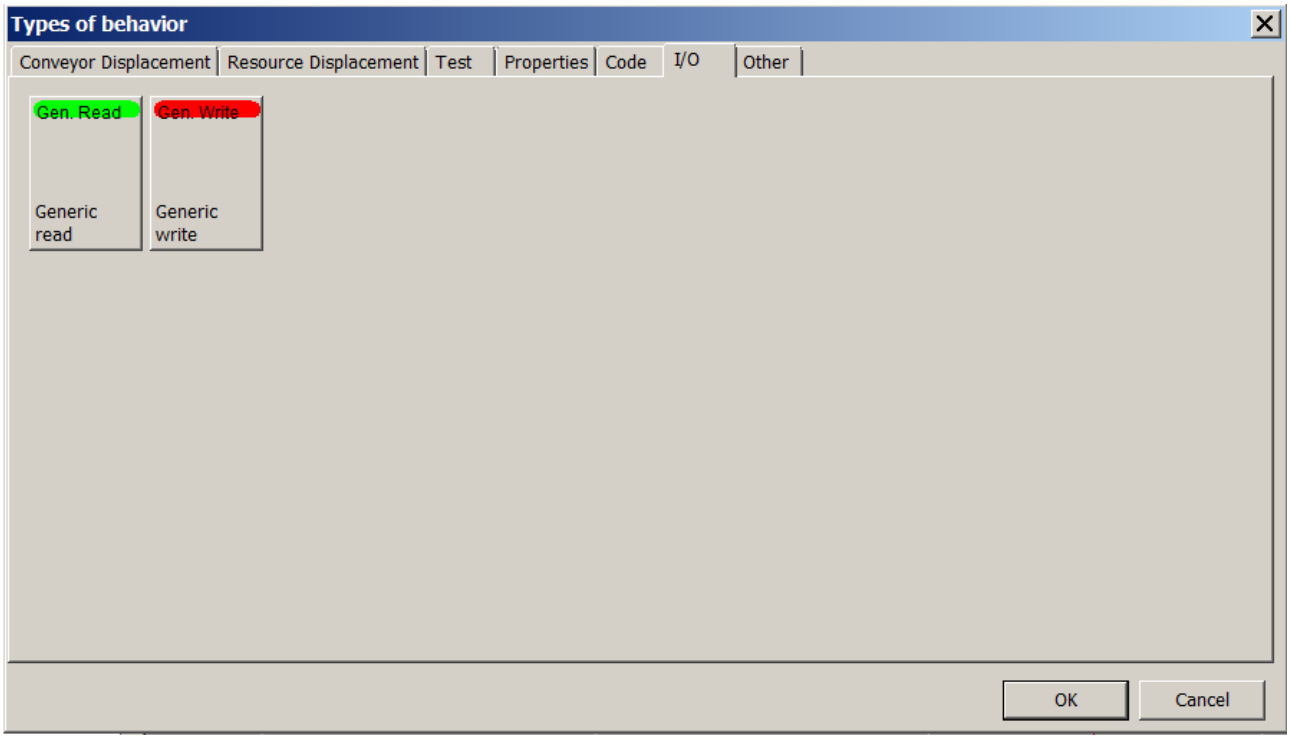
Sub wait(t)

```
    t1=getuniverse("time")
        miloop:
            t2=getuniverse("time")
            if(t2-t1<t*1000) then
                goto miloop
            endif
End Sub

Sub flash(t)
wait(t)
setbehavior("[..\set red color].values",1)
setbehavior("[..\reset color].values",0)
wait(t)
setbehavior("[..\set red color].values",0)
setbehavior("[..\reset color].values",1)
End Sub

myloop:
signal=getbehavior("[..\red light].currentvalue")
if signal=true then
while (signal=true)
signal=getbehavior("[..\red light].currentvalue")
flash(0.5)
wend
else
setbehavior("[..\set red color].values",0)
setbehavior("[..\reset color].values",1)
endif
goto myloop
```

Inputs/Outputs



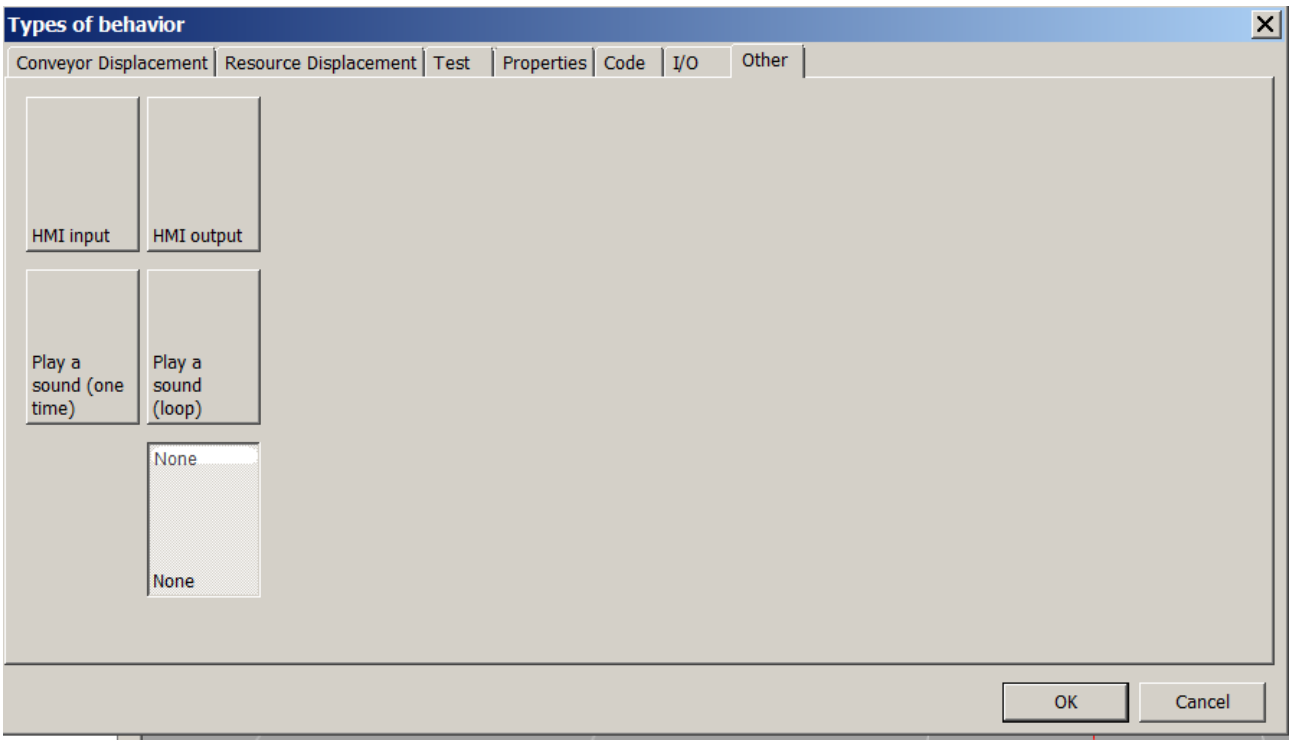
Generic read:

- Input variable for VIRTUAL UNIVERSE PRO and output for the external software/controller. This behavior will only read a variable from external software/controller.

Generic write:

- Output variable for VIRTUAL UNIVERSE PRO and input for the external software/controller. This behavior will only write a variable from the external software/controller.

Others



- HMI input:
  - Creates an input link to an HMI component (a push button, for example).

- HMI output:
  - Creates an output link to an HMI component (a light, for example).

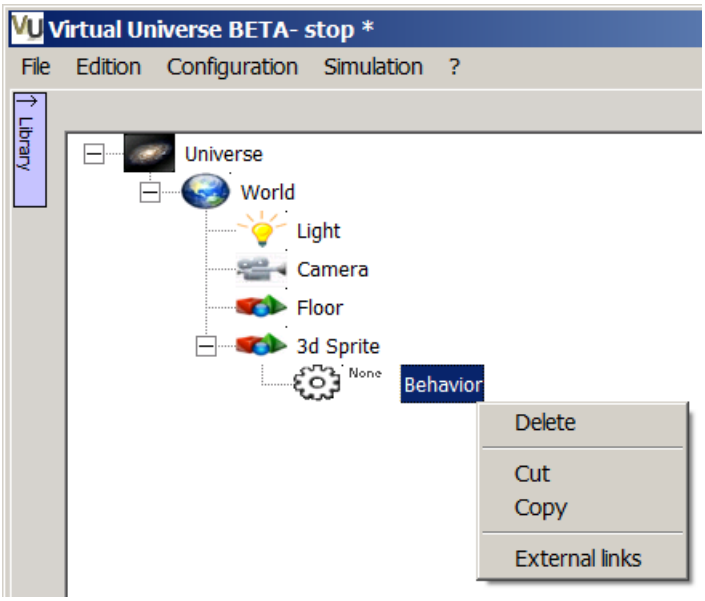
- Play a sound (one time):
  - Enables to play a sound file only once if the behavior current value is different from 0. The 3D sound will be heard as coming from the 3D parent Sprite. Moreover, the behavior current value can modulate the sound volume speed. This behavior allows, for example, to reproduce a motor noise as a function of its rotational speed.

- Play a sound (loop):
  - Same as above but the sound is played in a loop.

- None:

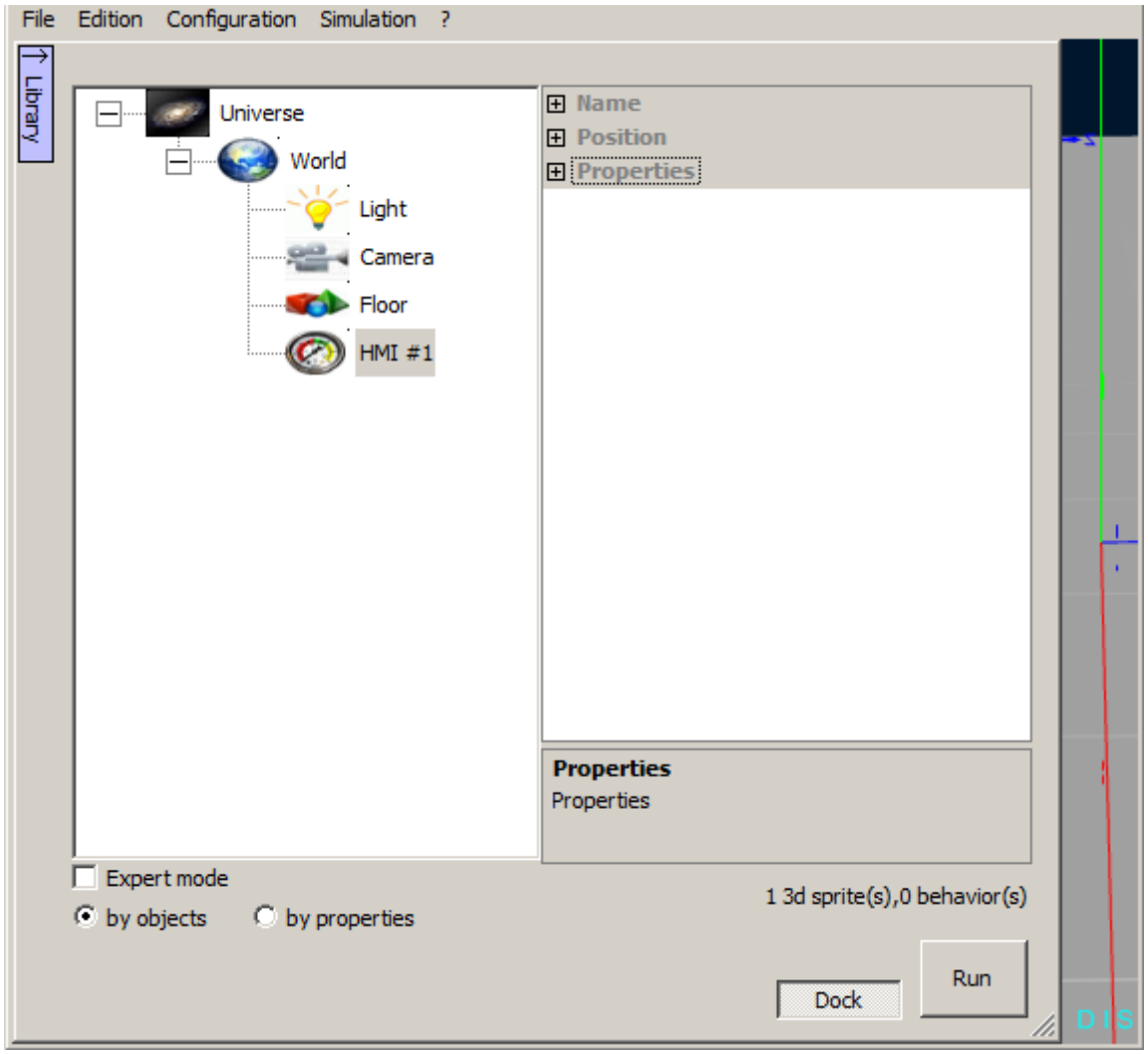
- Default type, to VIRTUAL UNIVERSE PRO represents an internal value. The Behavior “None” is inert.

Functionalities at the Behavior level

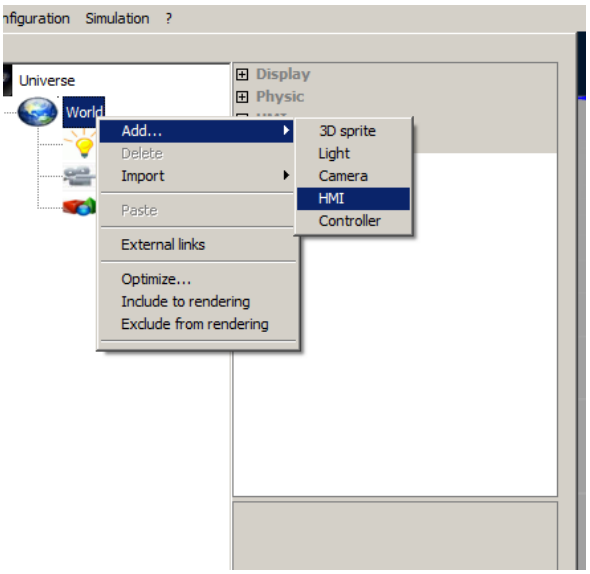


External links: Provides access to the behavior External Links window.

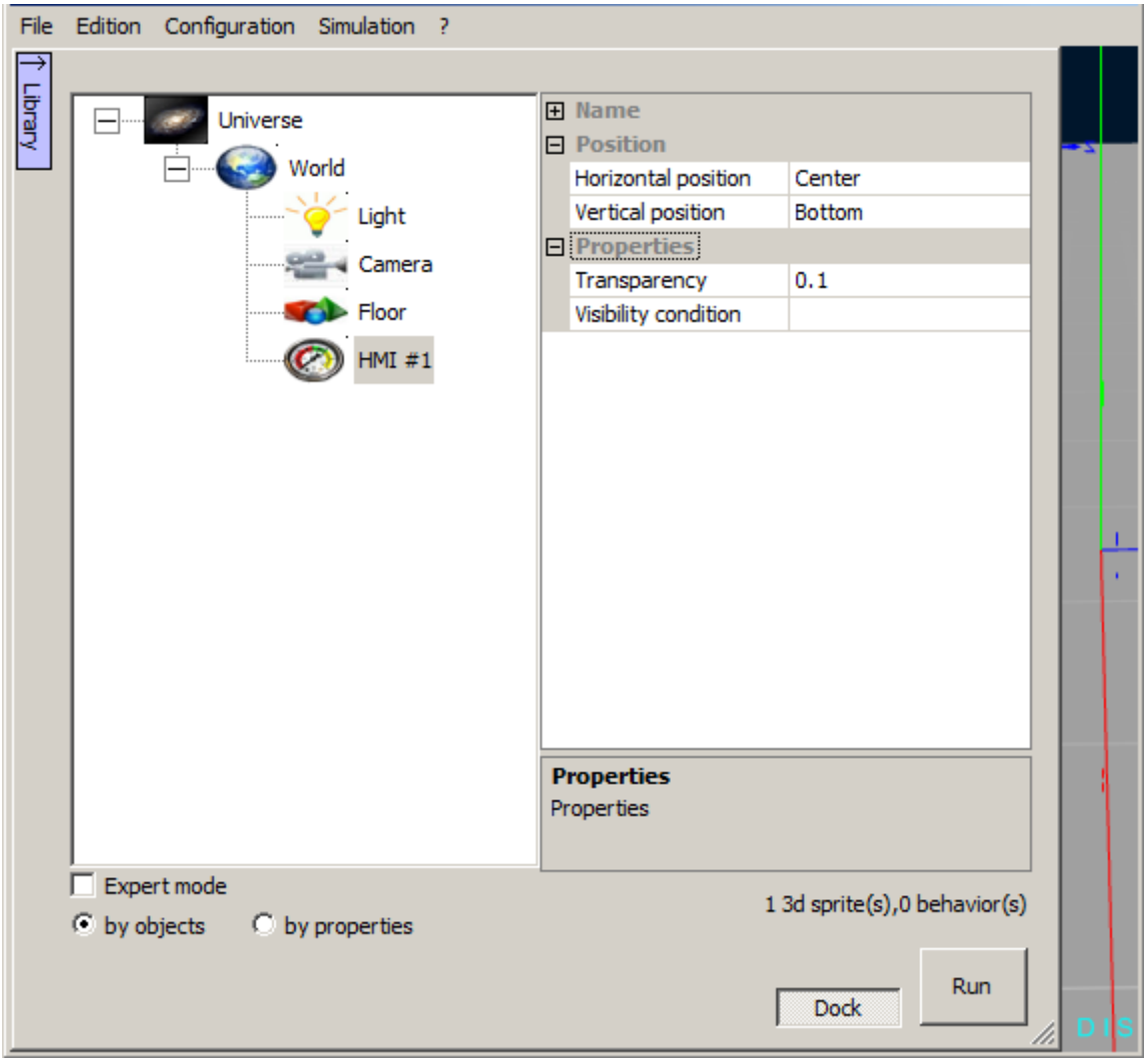
Properties of HMIs



The menu that opens when you click the right mouse button on the world allows the creation of a HMI :



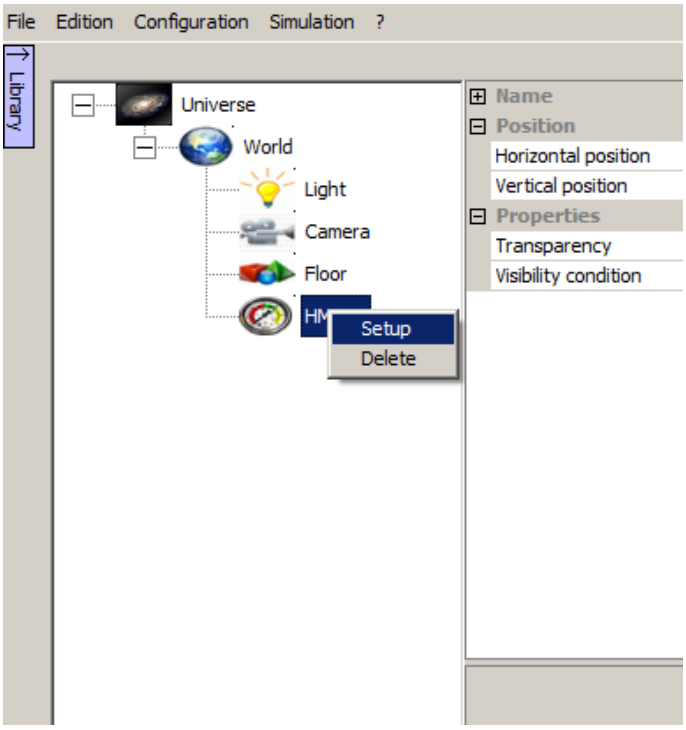
Detailed properties of a HMI



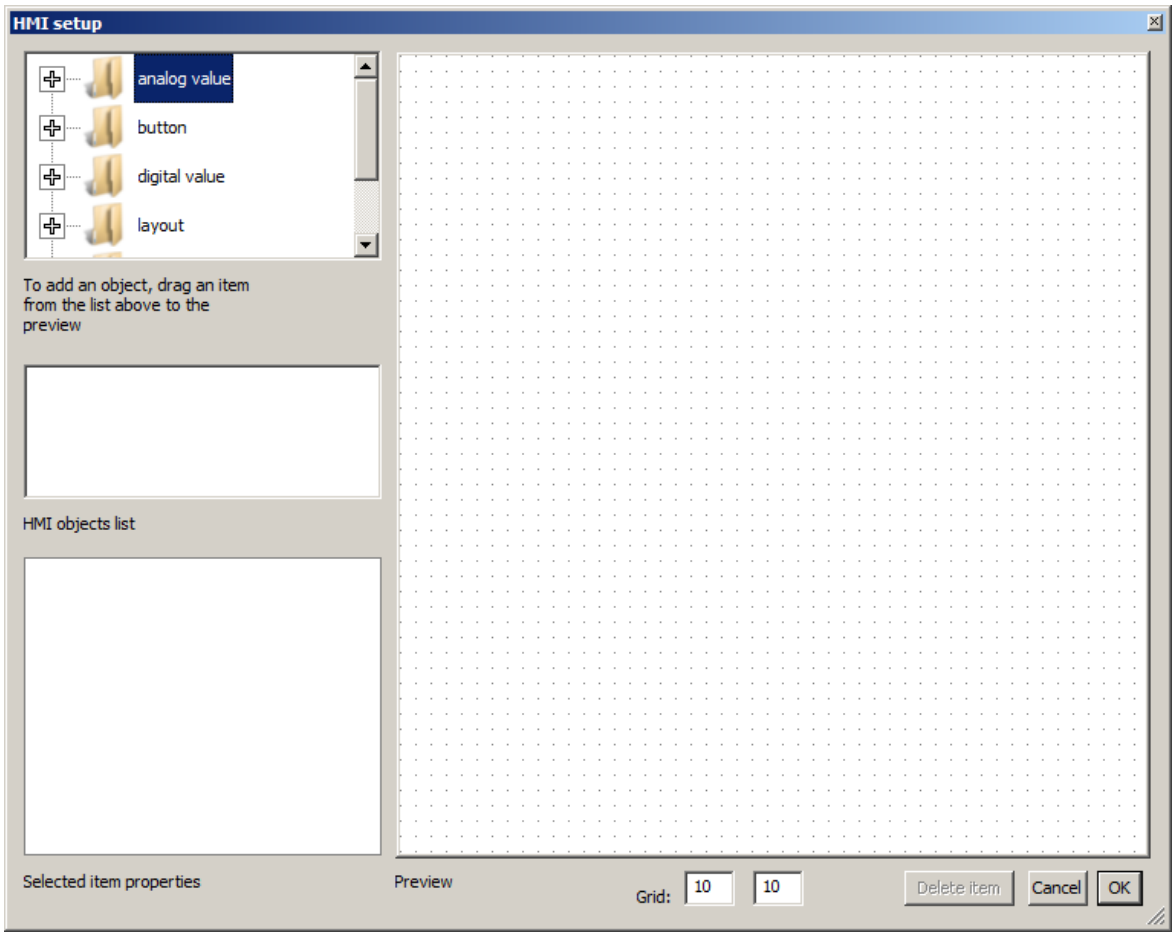
The parameters "positions" are used to define where the HMI will appear in the render window.

Settings "properties" are used to define the transparency of the HMI and a possible condition (state behavior) for display. If "condition" contains the valid name of a behavior, then the HMI will be displayed if the value of the behavior is different from 0 and hidden otherwise. This condition can display conditionally HMIs, for example to create a menu system.

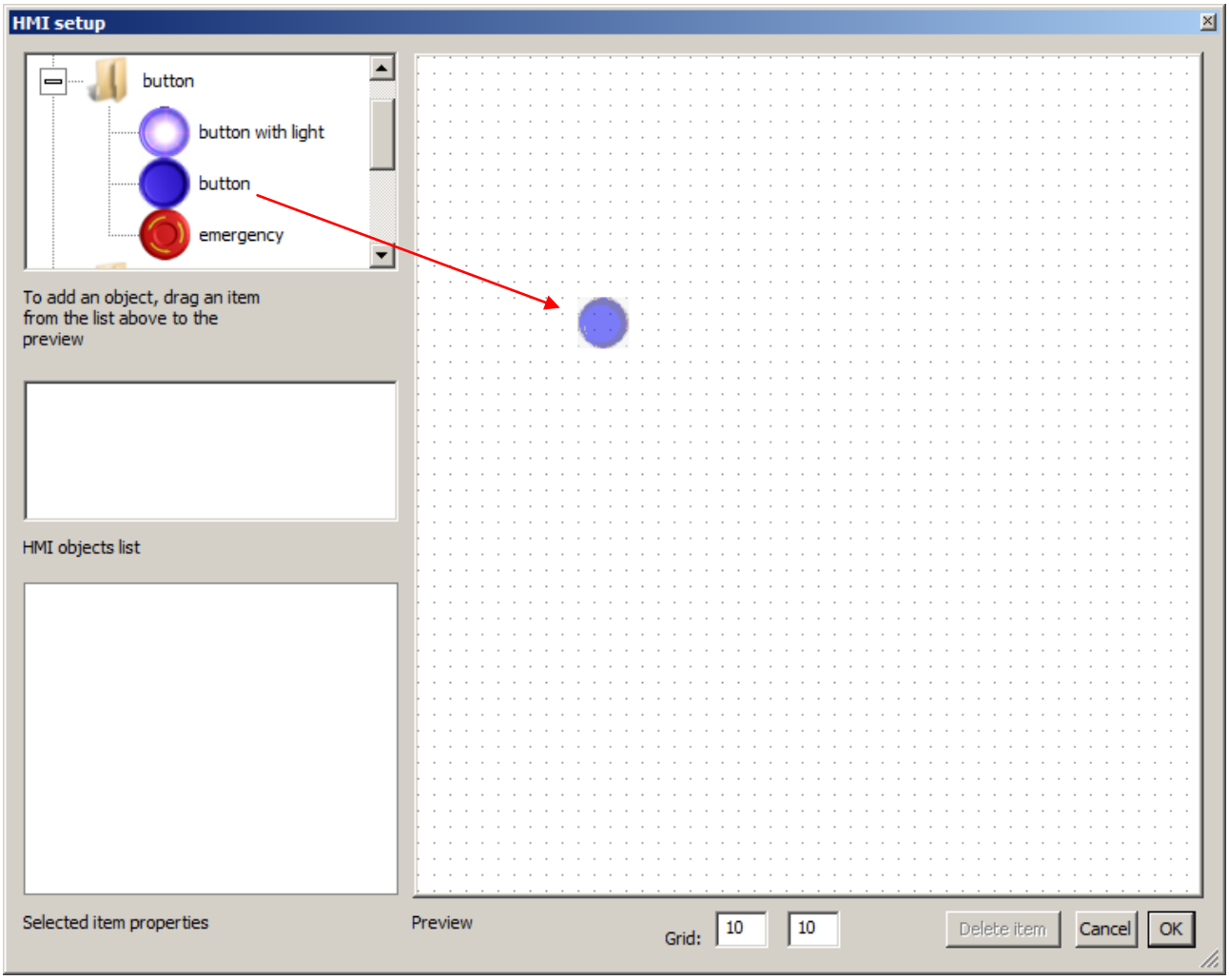
Creation or modification of a HMI

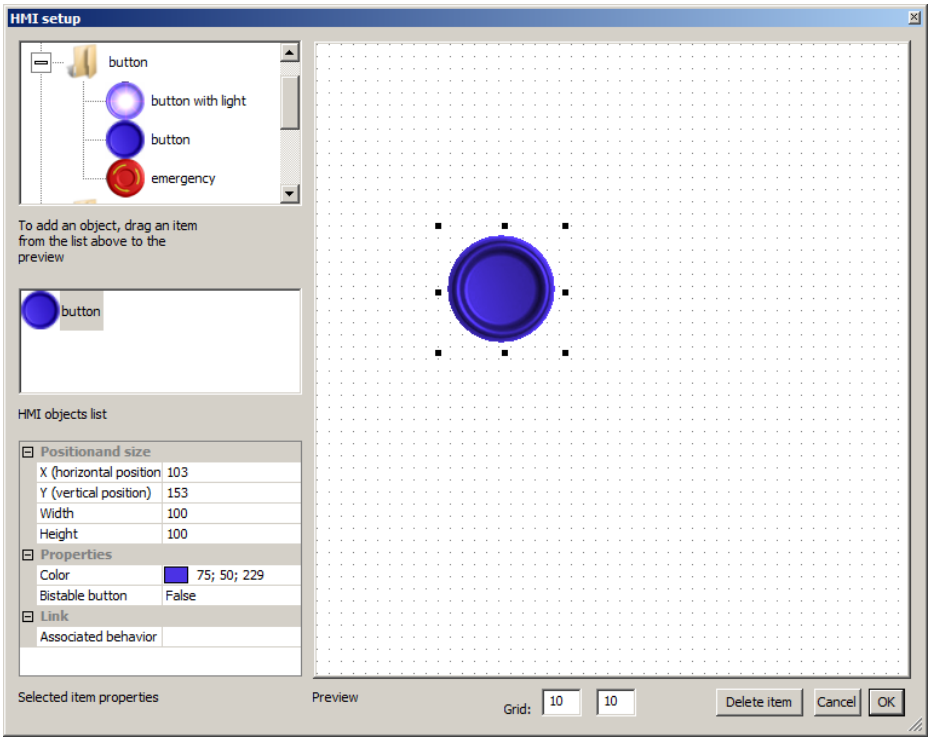


The context menu or double clicking on a HMI item provides access to the configuration window:



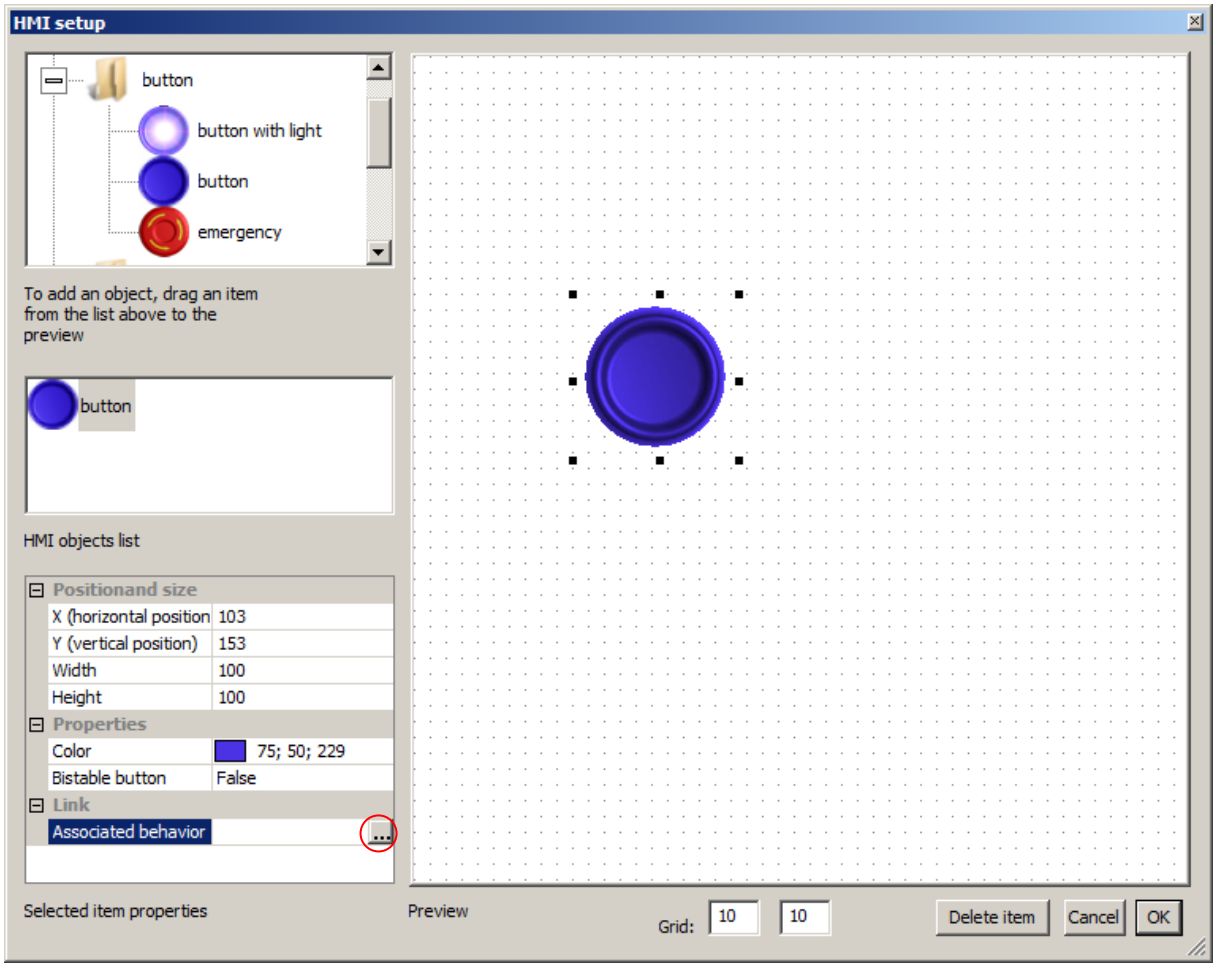
To define the elements of a HMI, drag and drop the available items from the upper left area to the right area.

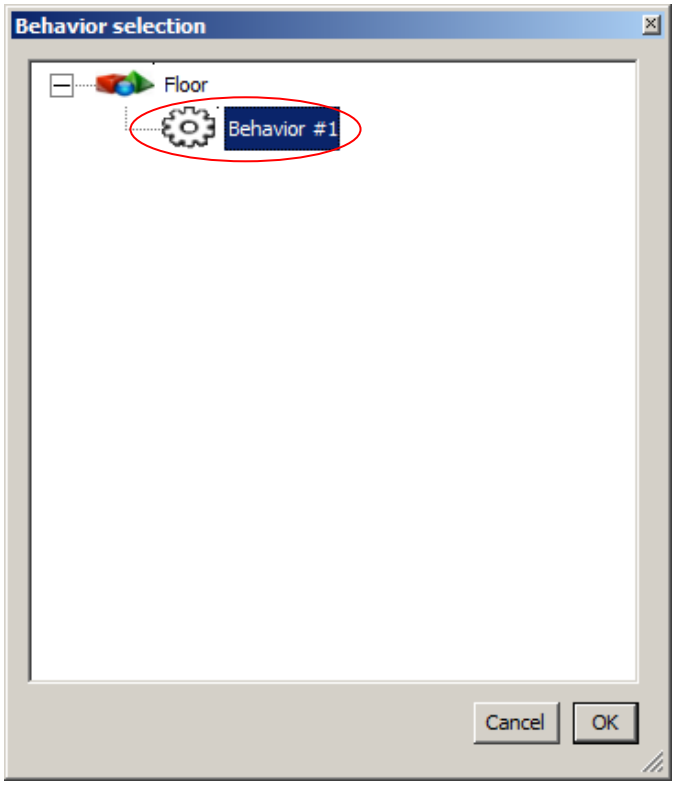


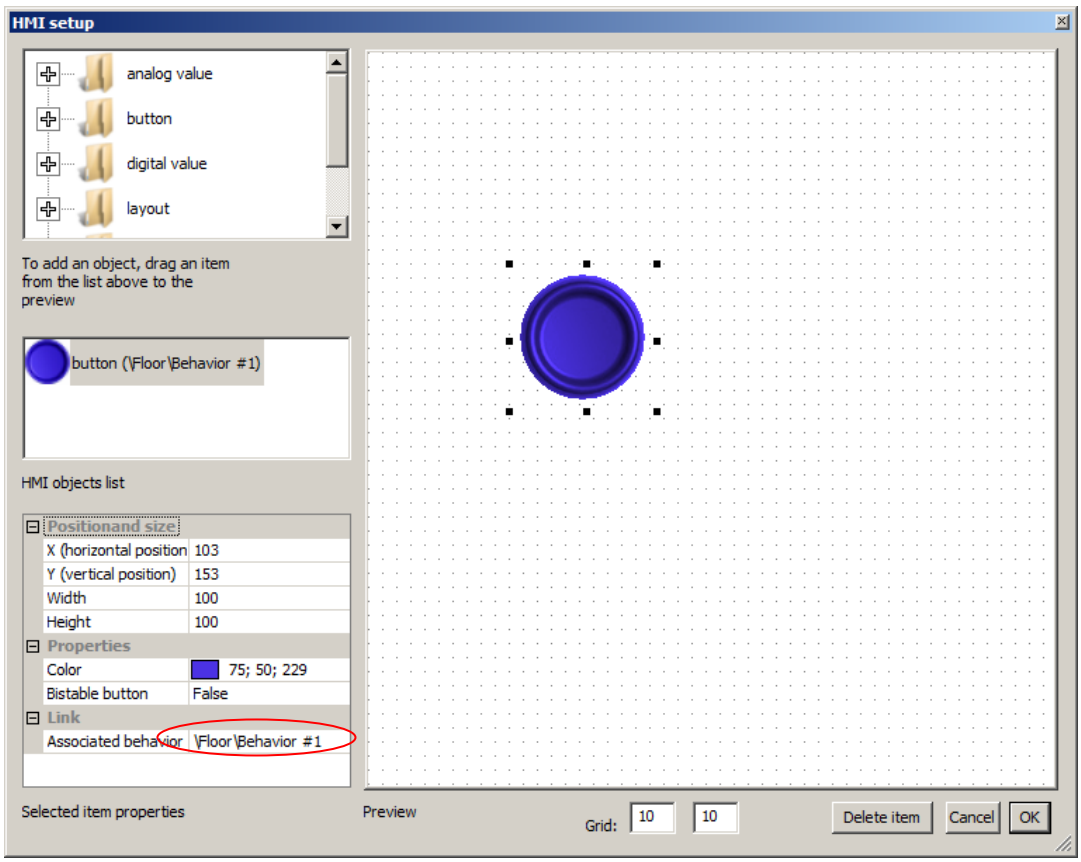


When a HMI element is selected, its properties are accessible in the lower left area.

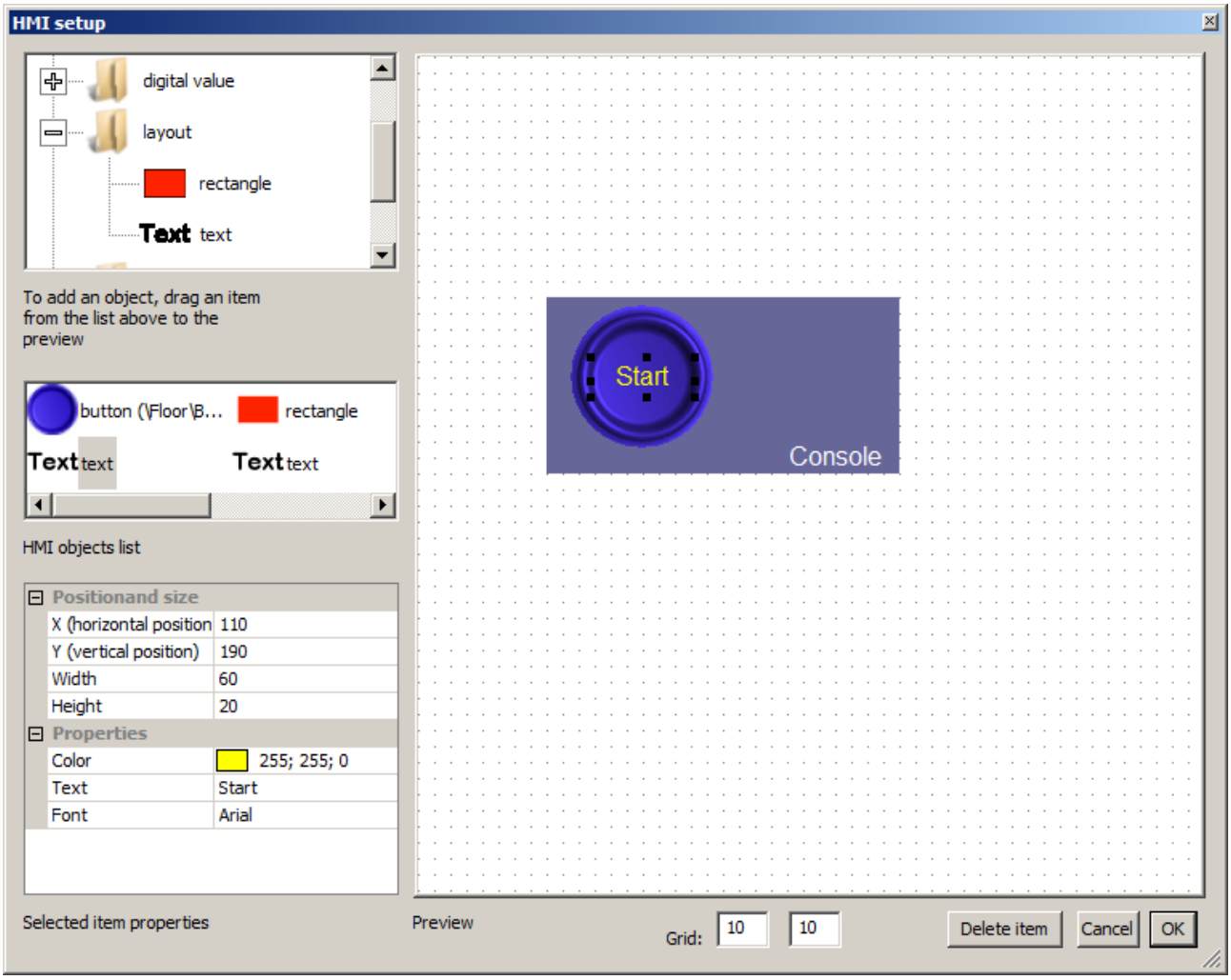
The "Links" section allows to define the relationship between the item and behavior:






HMI setup
analog value
button
digital value
layout
To add an object, drag an item from the list above to the preview
button (\Floor\Behavior #1)
HMI objects list
Positionand size
X (horizontal position 103
Y (vertical position) 153
Width 100
Height 100
Properties
Color 75; 50; 229
Bistable button False
Link
Associated behavior \Floor\Behavior #1
Selected item properties
Preview
Grid: 10 10
Delete item
Cancel
OK

Drawing elements (text, colored rectangles) can also be used to materialize the bottom of a console as well as indications on HMI elements:

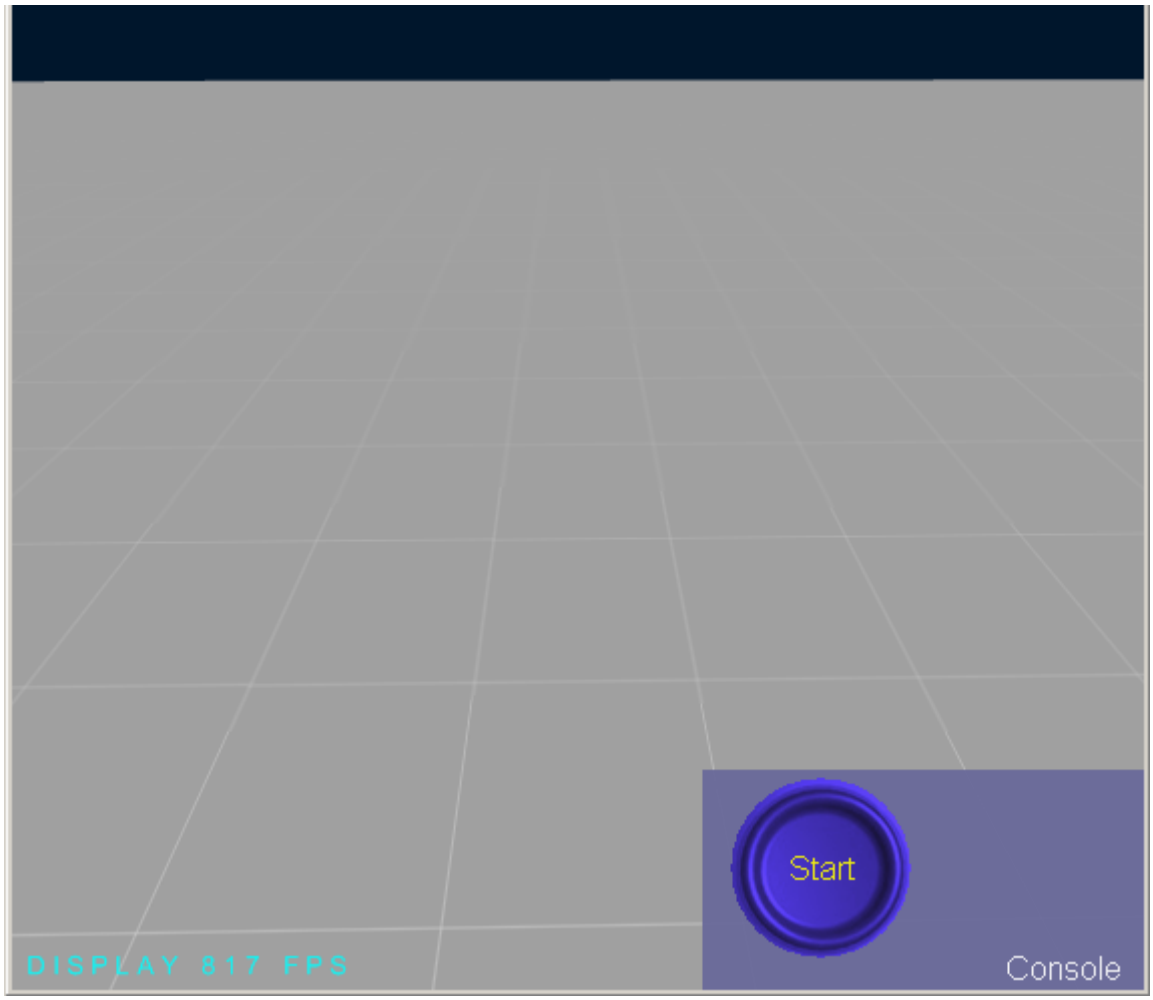


Remarks:

- The values of the grid allow you to easily position the elements, values below 2 disable the grid,

- If multiple items are in the same location, click multiple times with the left button on them alternately selects one of them.

HMI sample :

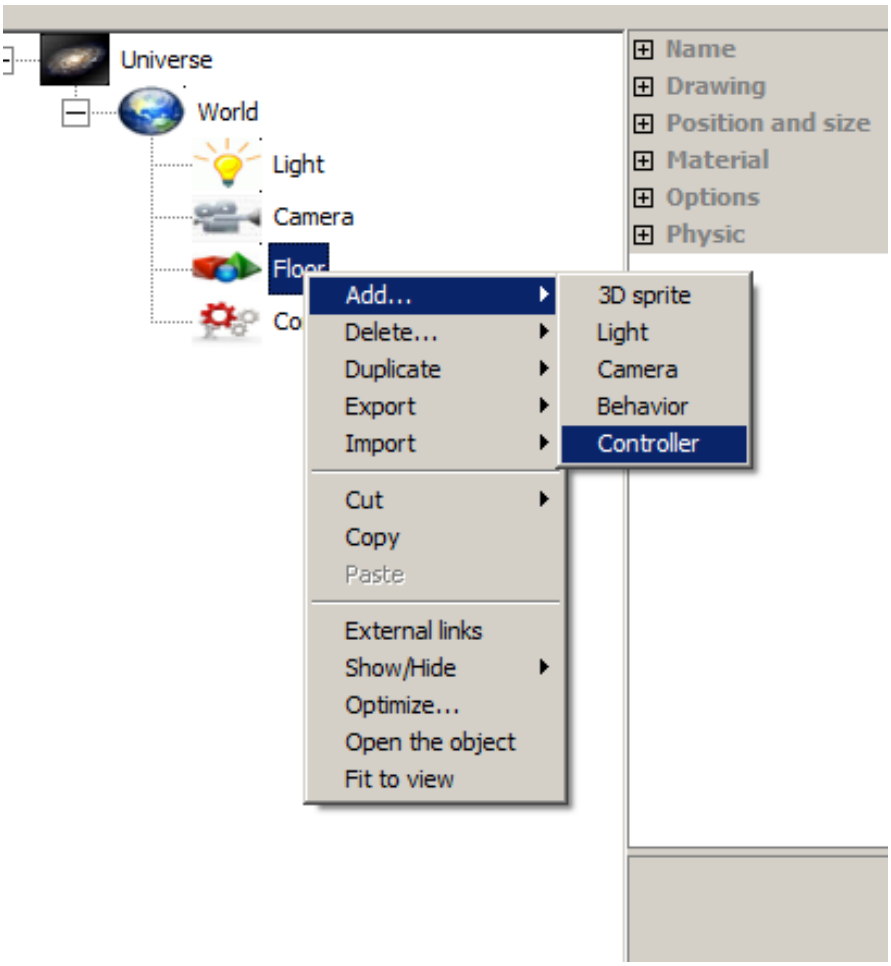


Properties of Controllers

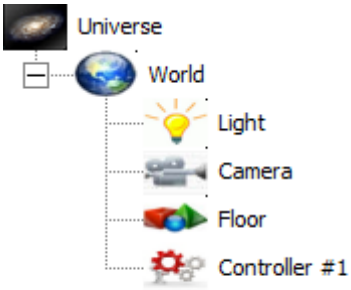
Programming Functions

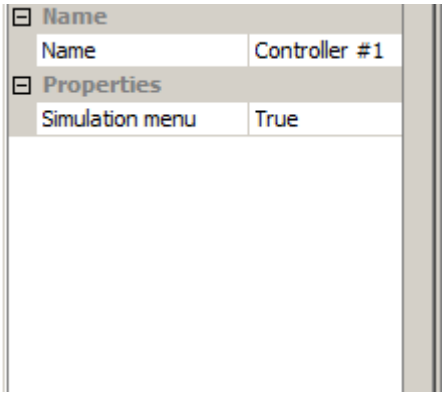
Programming features appear in the projects as one or more "controller" children of the "World" or "Sprite 3d" items. Each controller can contain one or more pages of program (areas not limited in size) written in Ladder or FBD / SFC (SFC and function blocks). Controllers can read and write values of behaviors of the project. Each controller can also read and write local variables local to each controller. Controllers perform their program pages in RUN mode.

Adding a controller is achieved by right clicking on the "World" or "Sprite 3d" items:



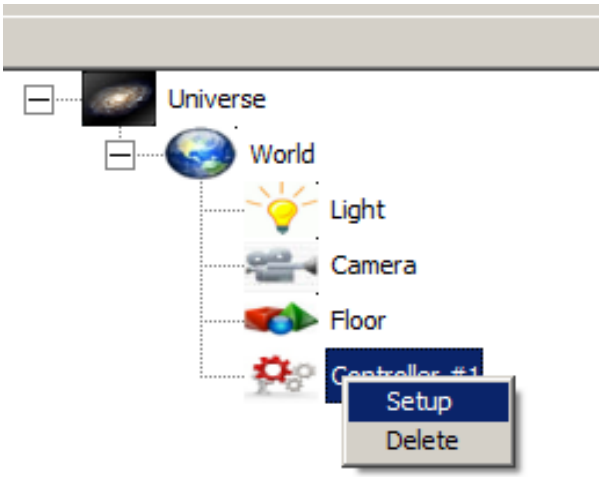
Detailed properties of a controller



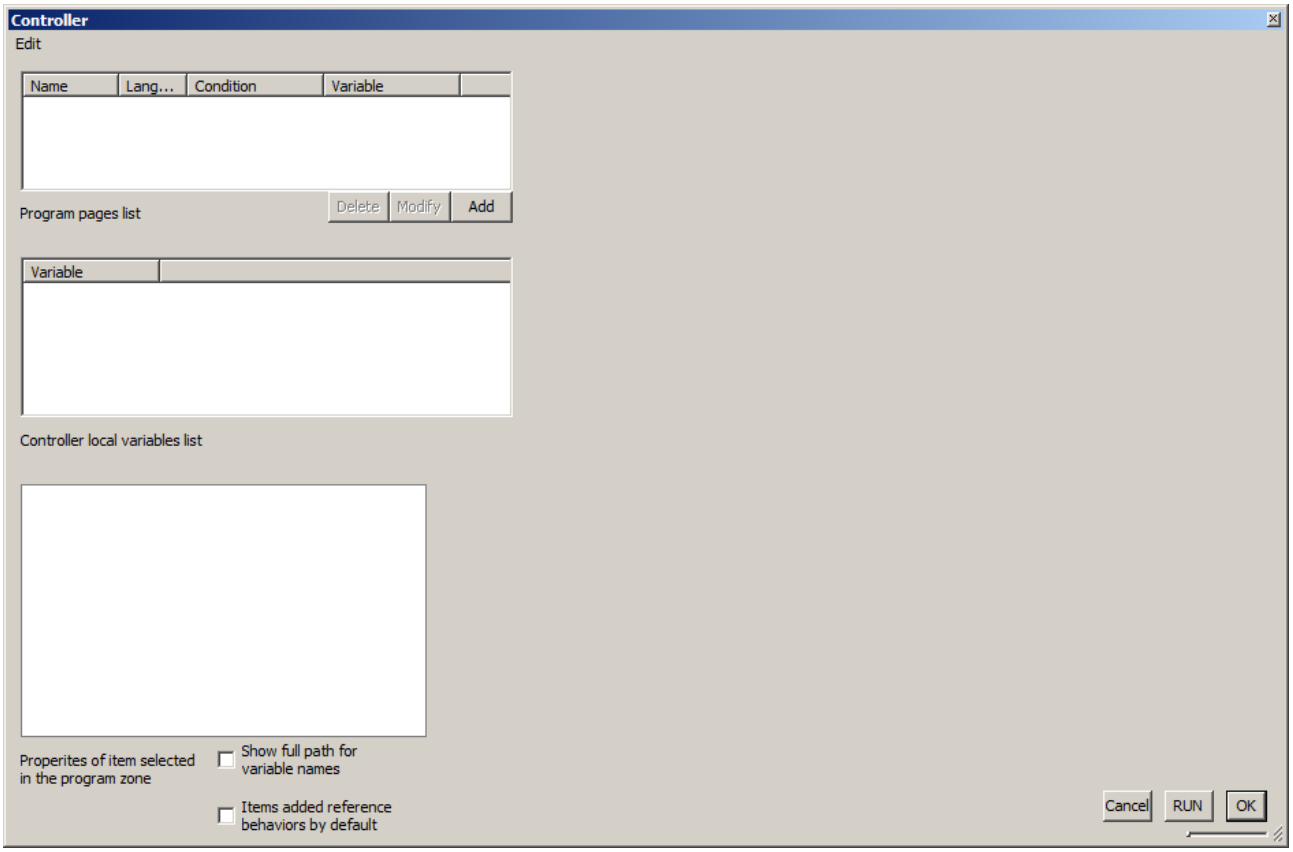


The parameter "Simulation menu" set the visibility of the controller (and the programs it contains) in the menu "Simulation / Debug / Program - Simulation" in RUN mode.

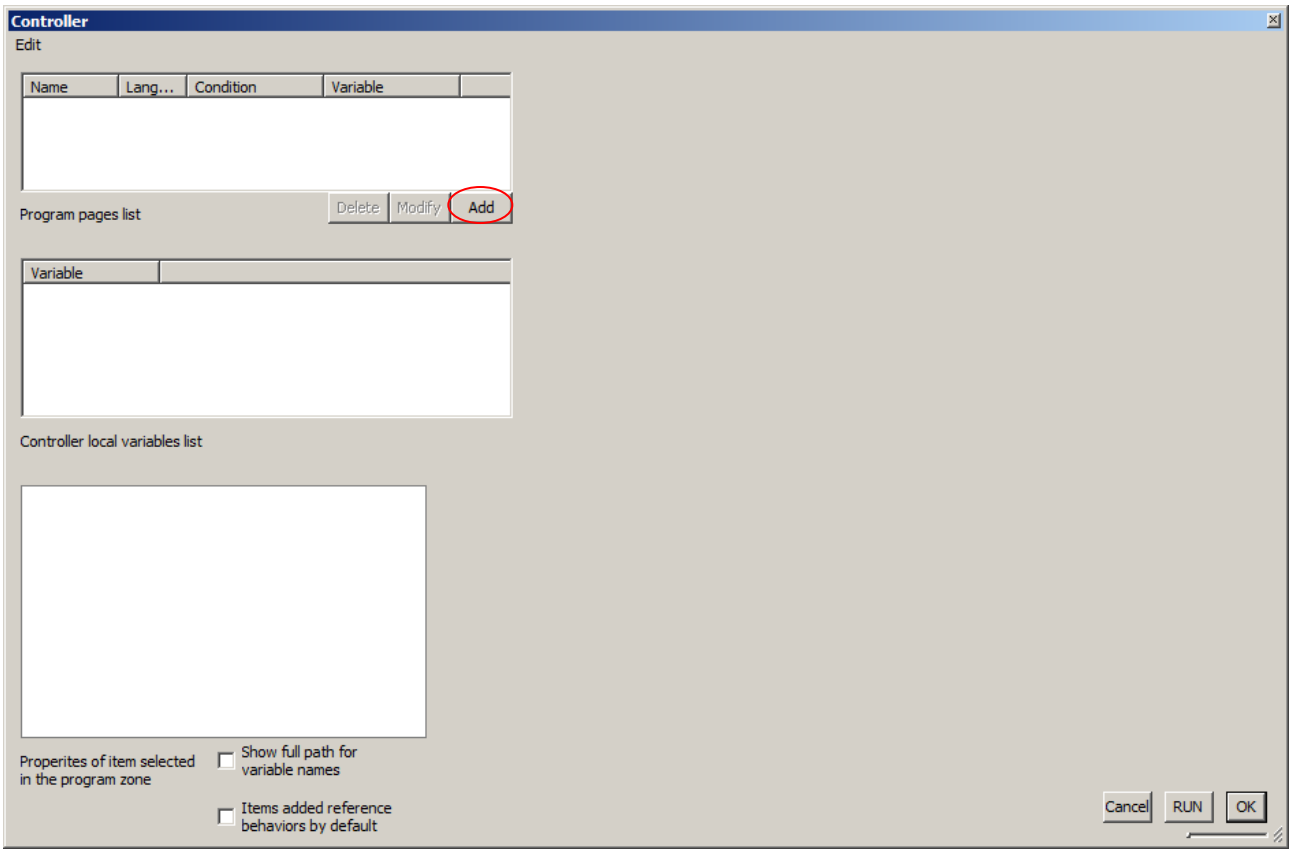
Programming a controller

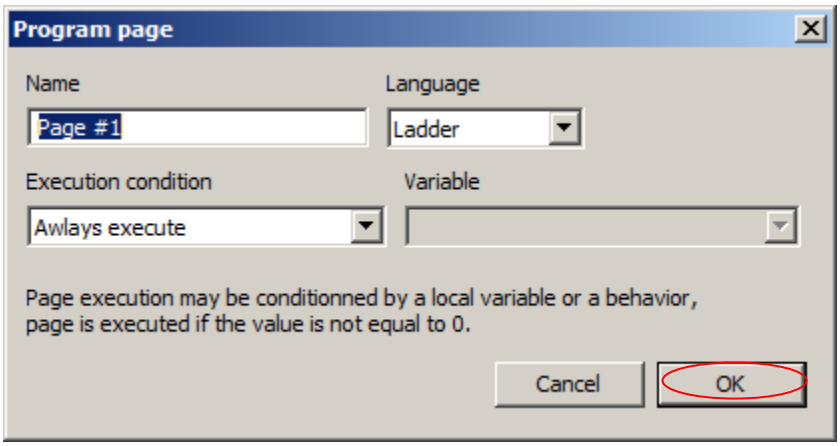


In STOP mode, use the context menu or double click on an item "Controller" to open the configuration window:



To add a program page, click on "Add":



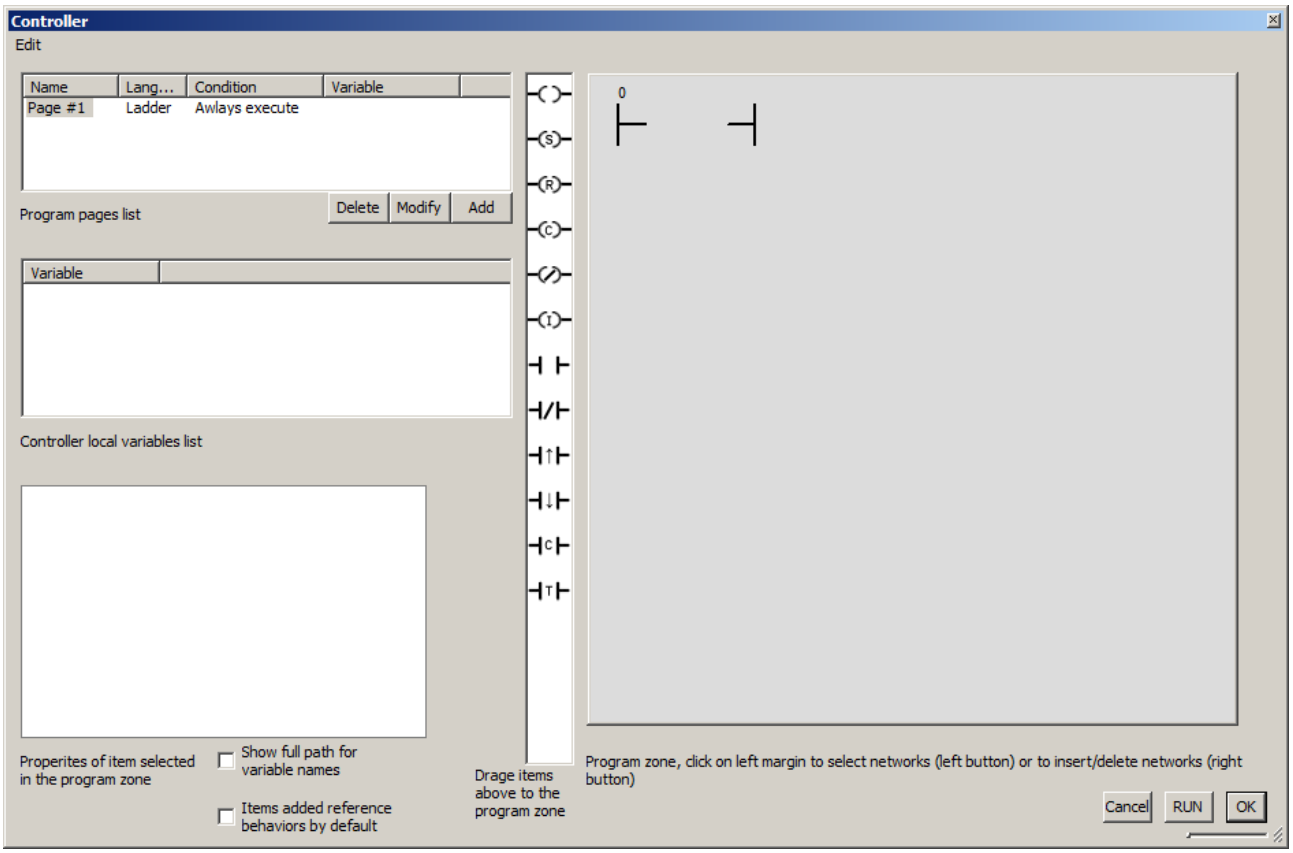


The name of the page, the language used and a possible execution condition can be completed.

Remarks:

- The chosen language can't be changed for the page thus created,

- The condition (may be a behavior or a local variable state) is used to validate or not the execution of each program page. If the value of local variable or behavior is 0, the page is not executed, it is otherwise.



Common elements

All languages use three types of variables:

- Behavior (behavioral state of the project), to reference a behavior, a full path or a relative can be used. A full path gives, starting from the world, the name of a parent sprite or a light. For example: "\ floor \ robot \ engine": behavior "engine" child of the sprite "robot" itself child of the sprite "floor". Relative paths, used in the case of a child controller specifies a sprite path from that sprite. For example: ". \ engine": behavior "engine" child of the sprite which is the parent of the controller. Using relative paths can create a group sprite(s) + controller(s) duplicable. Moreover, the check box "Show variable names with full path" allows to view or not a short format (without the full path) for the variable names making it more concise programs display.

- Local variables (variables local to each controller and common to all pages of the same controller), local variables are initialized to 0 at the transition to RUN,

- System variables:

- BLINK500MS: variable that changes state false / true each 500MS, typically used to make lights blinking,

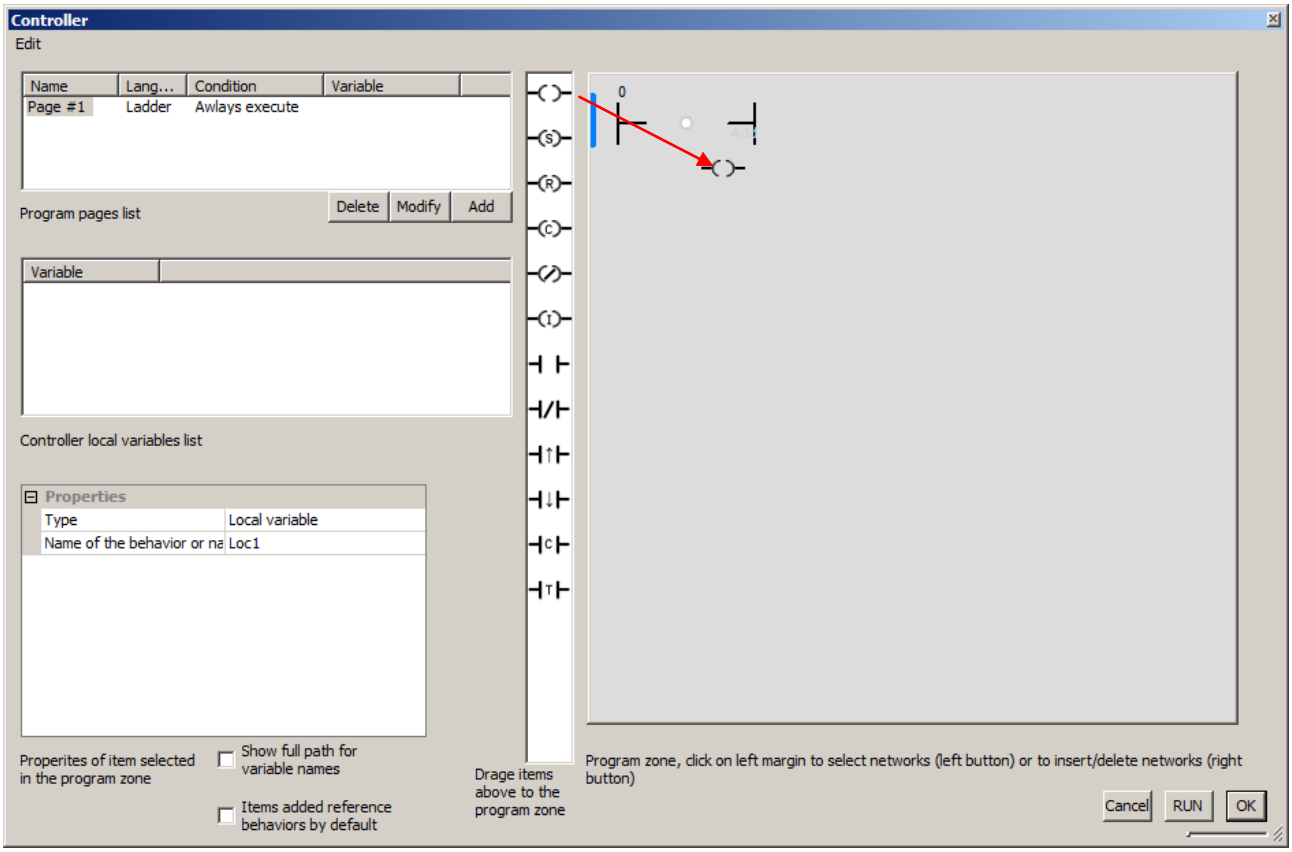
- FIRSTCYCLE: true for the first execution cycle of the program, then false. Typically used to perform initialization. This variable is true only if the page is executed to transition to RUN (not conditioned by a false condition in the transition to RUN).

- ELLAPSEDTIME: time elapsed since the last scan in seconds.

All these variables are real numeric types of 8 bytes. By convention, for processing Boolean, they are considered false if equal to 0 and true if not equal to 0.

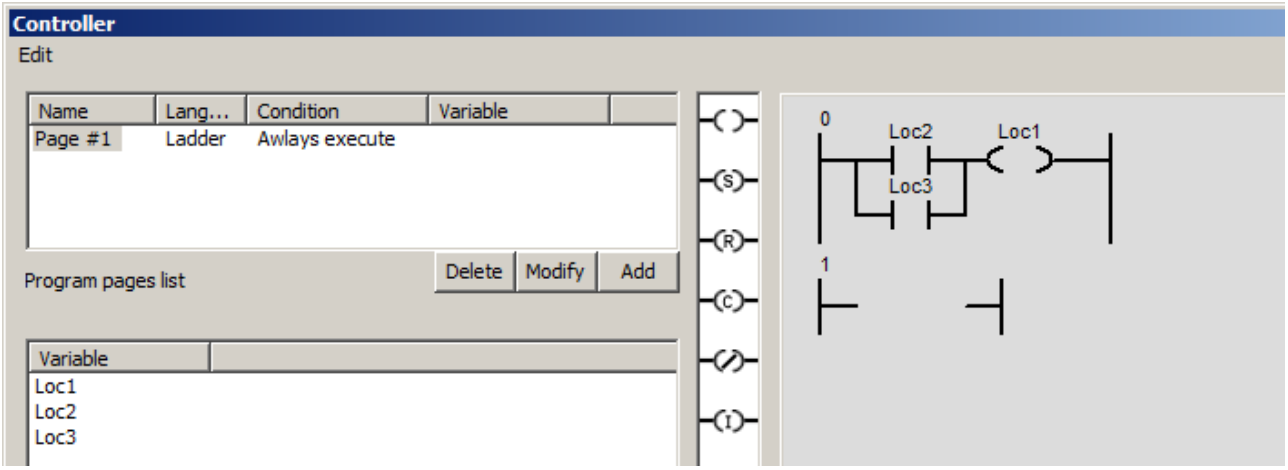
Ladder language

Programs creation:

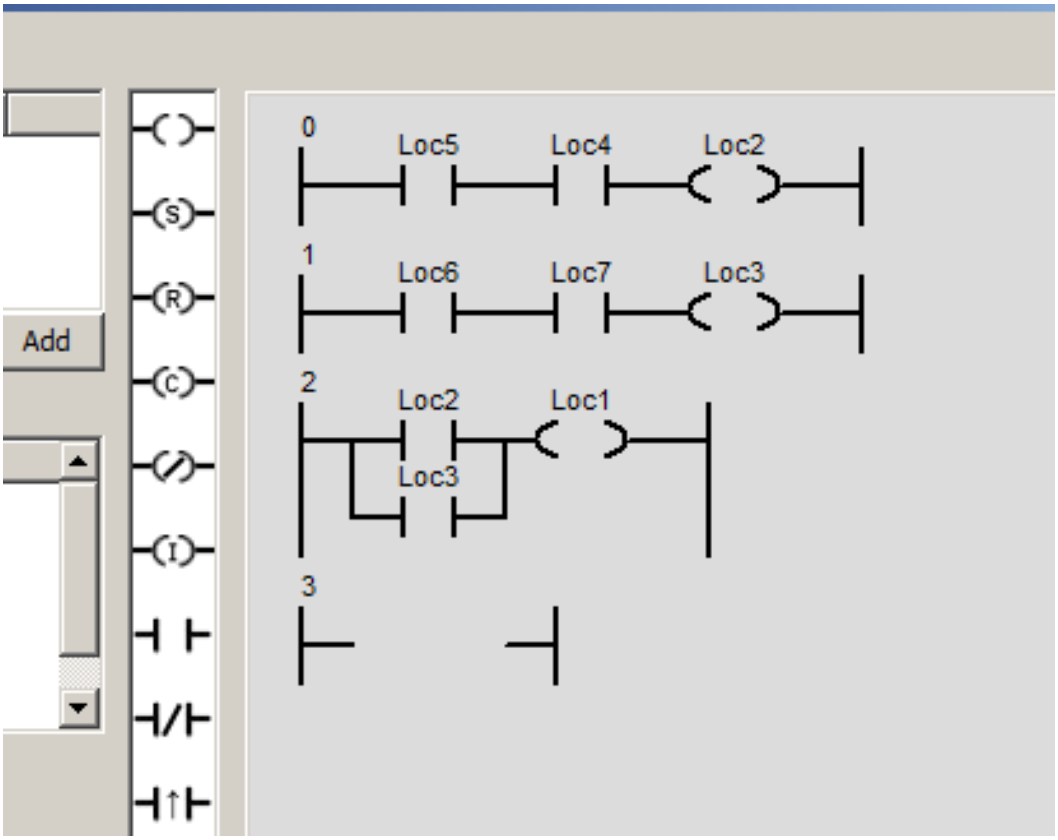


The creation of the program is done by dragging the items to the programming area. The first element that can be deposited is one of the coils available. A new local variable is automatically associated with each new item created.

Contacts can then be deposited to form a network.

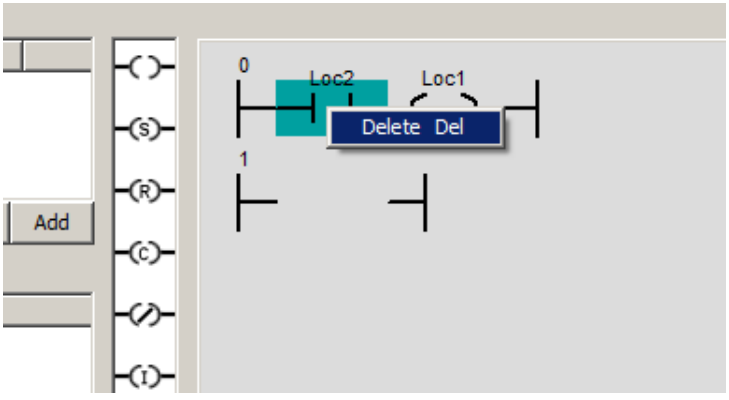


Only simple networks can be created, if more complex networks are needed, they must be broken down into simple networks using local variables. Example:



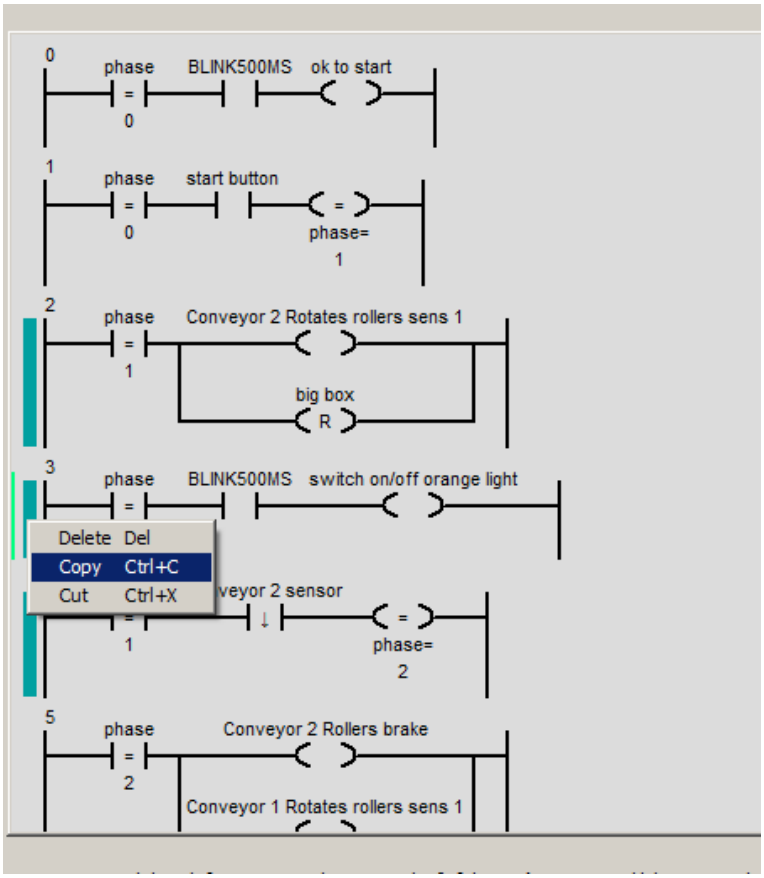
Editing functions

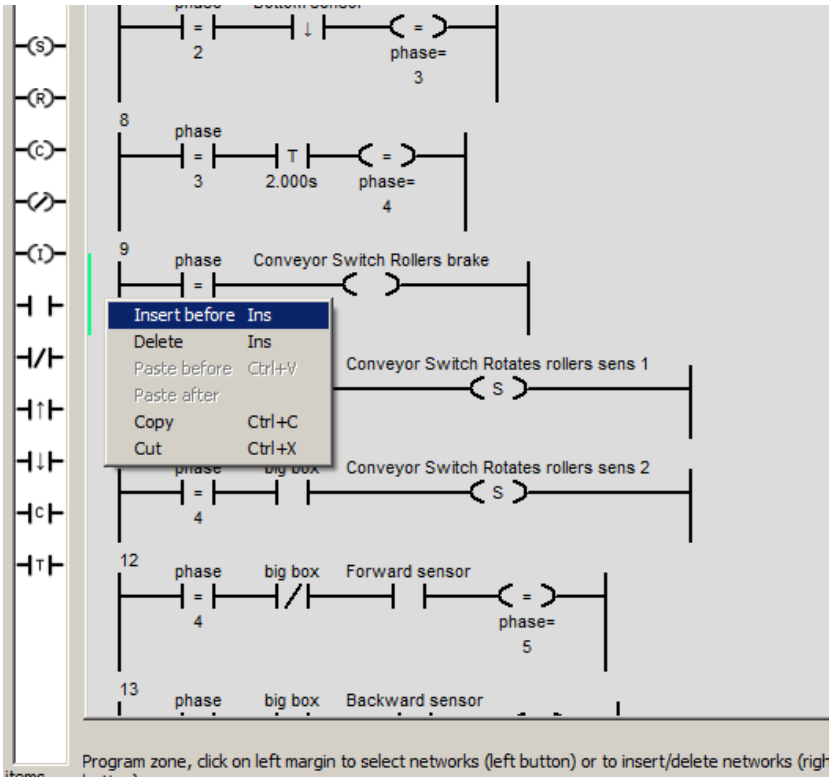
- To delete an item, select it (left click) and then open the context menu (right click on the item) and select "Delete":



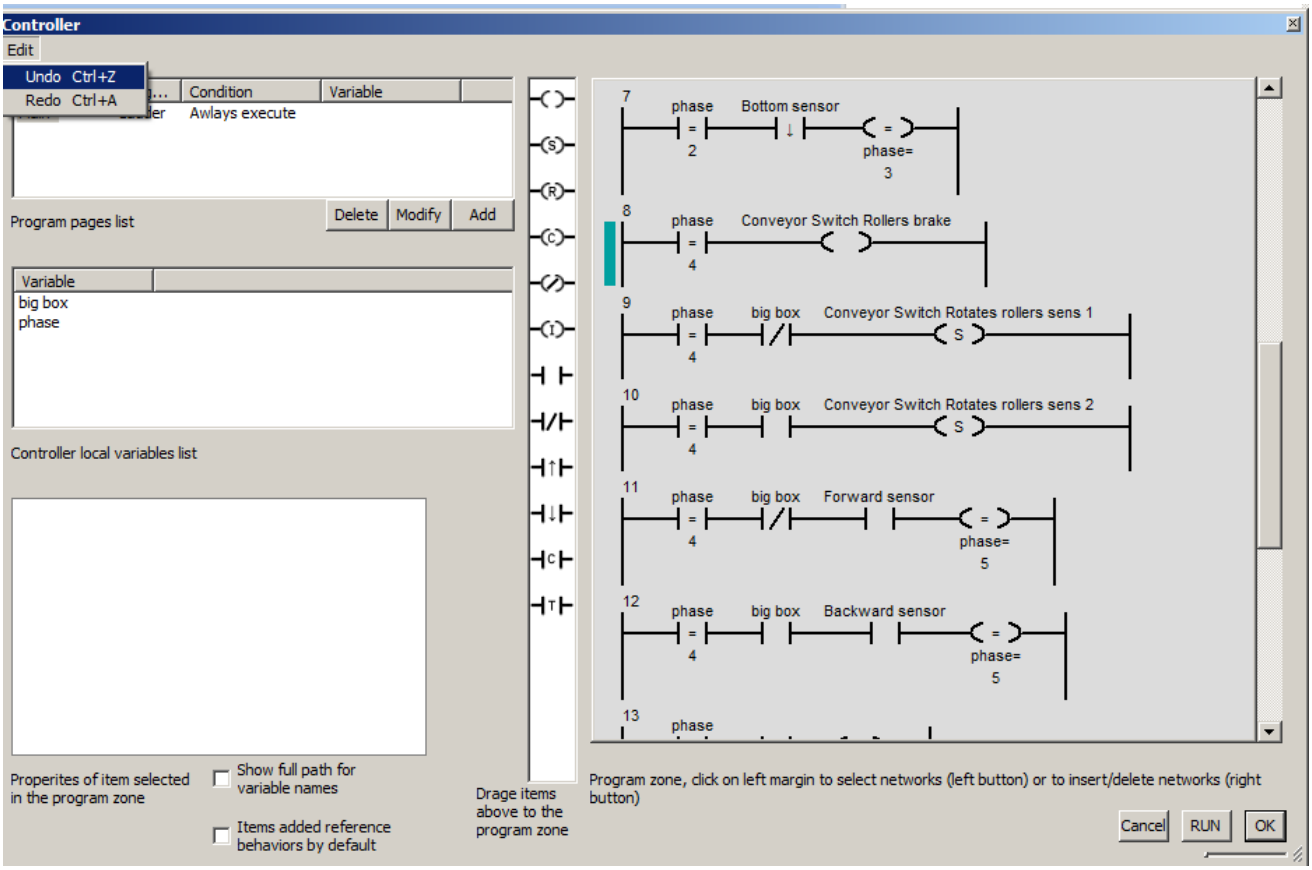
Remark: the last coil item of a network can be deleted only if there is no more contact item.

- Delete / insert / copy or paste a network or multiple networks, select the networks (left click on the left margin) and then open the context menu (right click on the left margin):

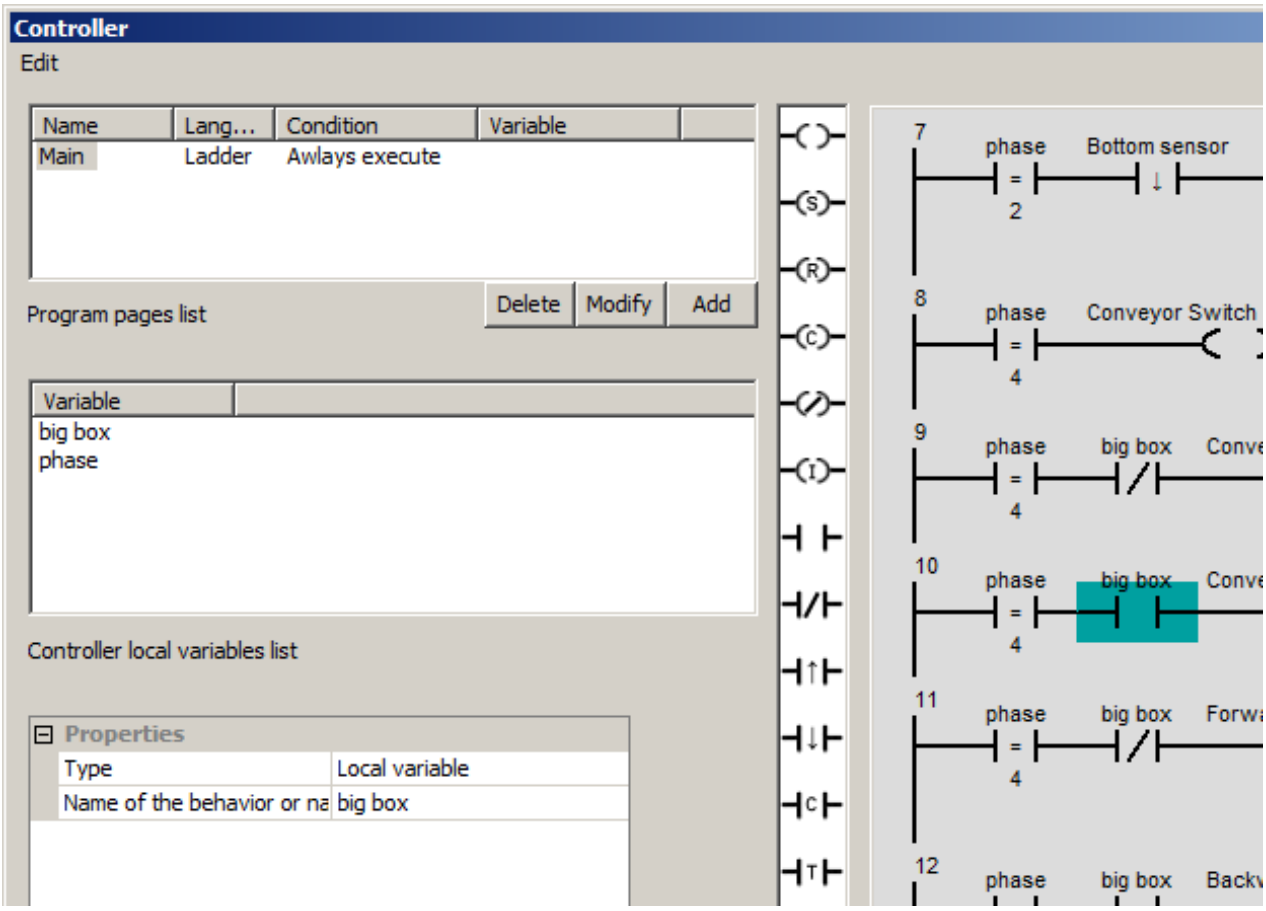




- To undo or redo the latest changes, use the Edit menu in the main window:



When an item is selected, you can access its properties in the lower left area:



One can thus select the variable associated with the item.

Items list

 coil: the associated variable is written with value 0 or 1 depending of the state defined by the network boolean equation (0 if false, 1 if true)

 not coil: same than coil but written value is 1 if false and 0 if true

 set coil: the variable is set to 1 if the network is true

 reset coil: the variable is reset to 0 if the network is true

 inverting coil: the variable is inverted if the network is true. This invertion is performed every page execution, the use of the "raising edge" contact allow to perform this invertion only one time.

 calculation coil: perform a mathematical calculation or a copy between variables or constants and variables if the network is true.

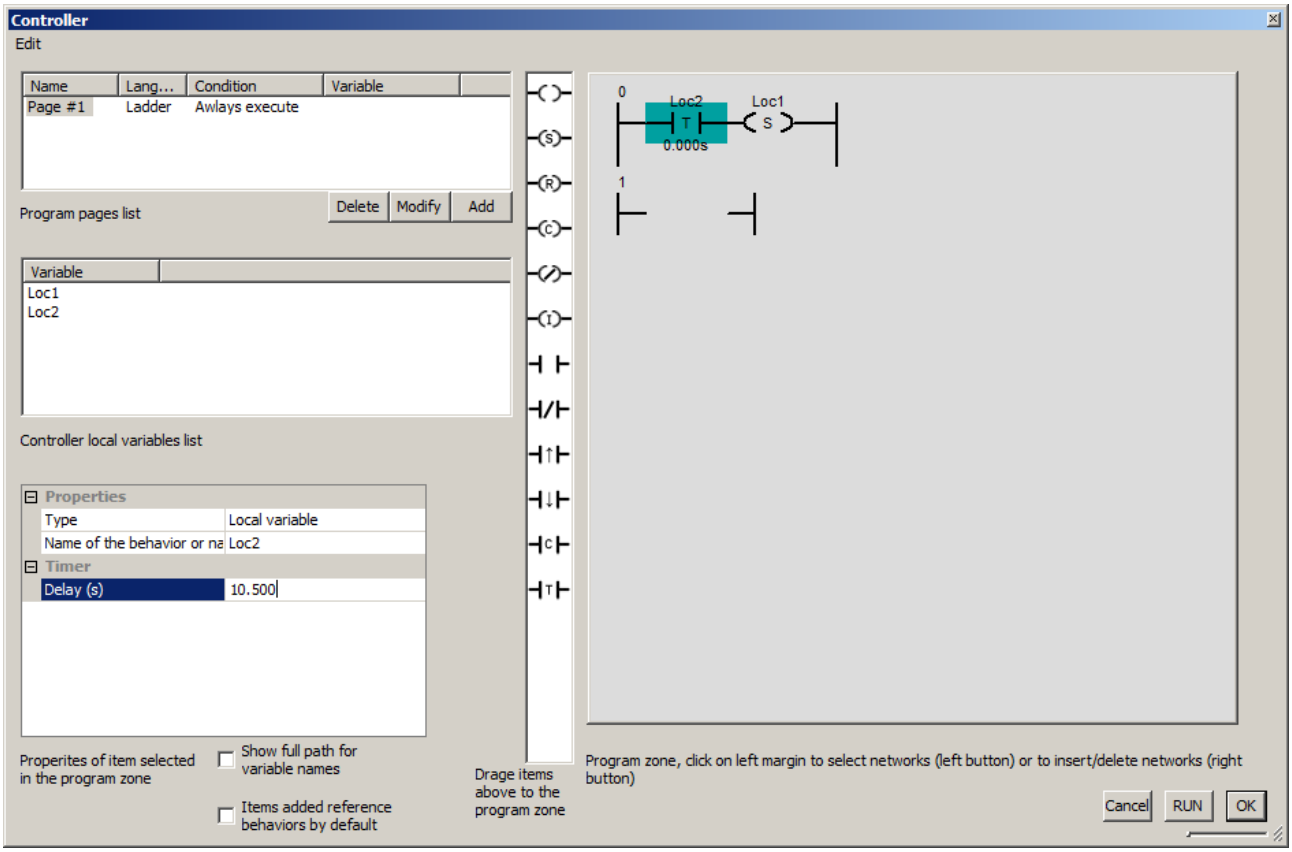
⊣ ⊢ Normally Open contact : true if the state of the associated variable is true (different from 0)

⊣/⊢ Normally Closed contact:true if the state of the associated variable is false (equal to 0)

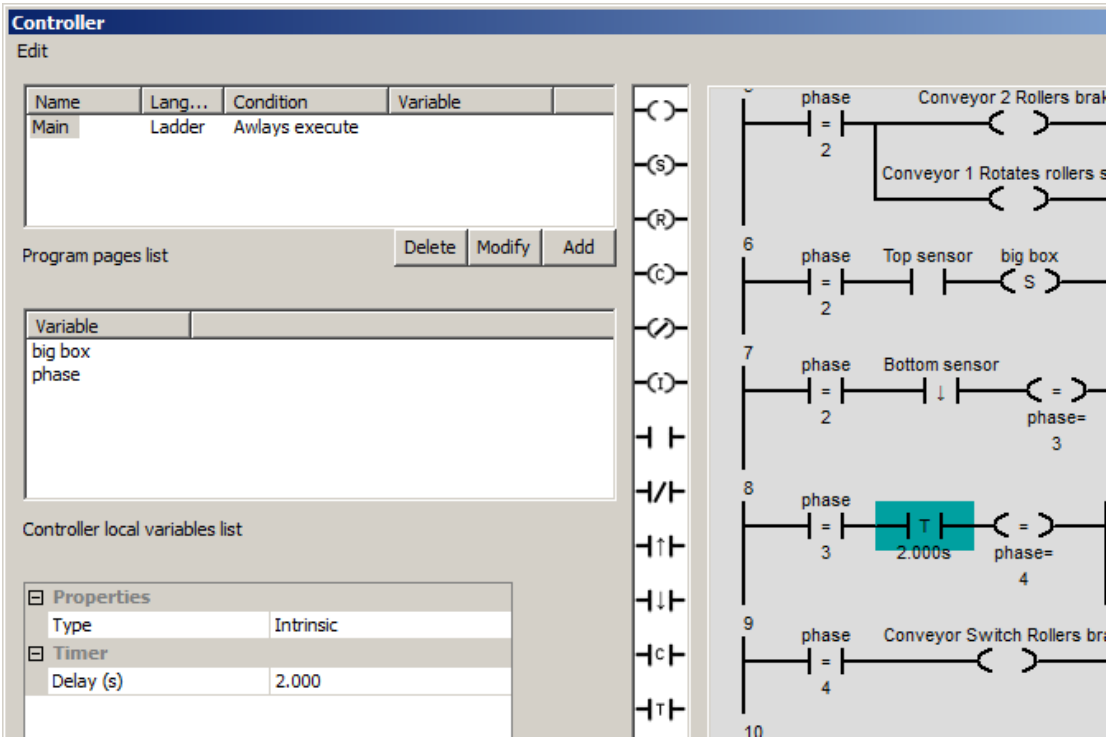
⊣↑⊢ Raising Edge contact: true during one execution cycle if the state of the associated variable is changing from false (equal to 0) to true (different from 0).

⊣↓⊢ Falling Edge contact: true during one execution cycle if the state of the associated variable is changing from true (different from 0) to false (equal to 0).

⊣T⊢ Timer contact: true after the state of the associated variable has been true during a delay. The delay is defined in seconds unity and has to be setting up in the contact properties:

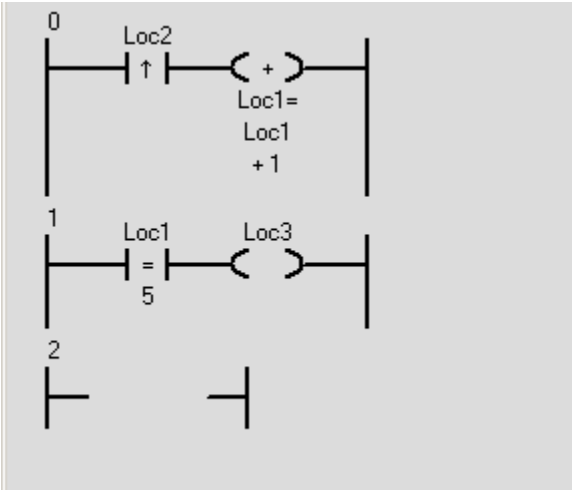


For this contact, a pseudo variable type "Intrinsic" can be selected, this means that the delayed signal is the one coming from the left part of the contact:



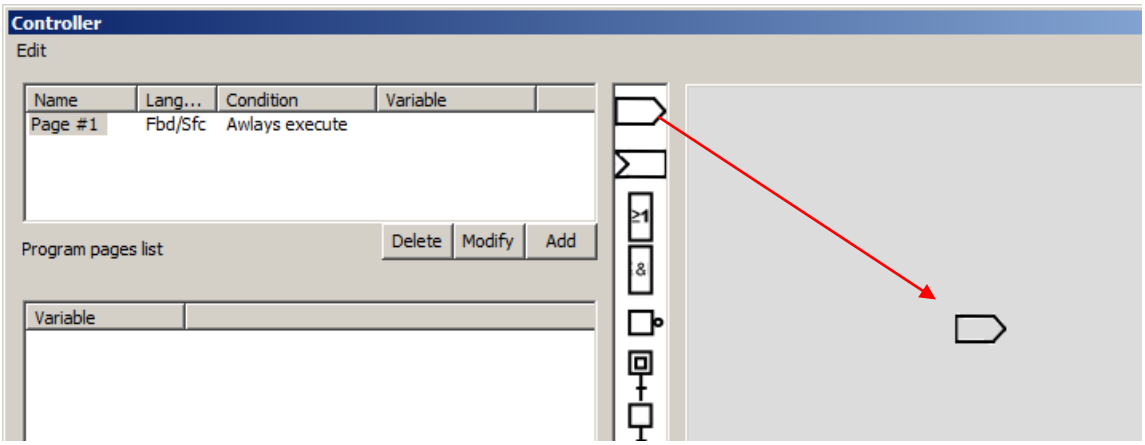
Comparison contact: true if the numerical comparison is true

Example (activating Loc3 after 5 raising edges on Loc2. Loc1 is used as counter):



Fbd/Sfc language

Programs creations



The creation of programs is performed by dragging the items to the programming area. A grid aligns the elements for a better presentation.

The elements are then connected by creating links. Objects connections are represented by colored circles:

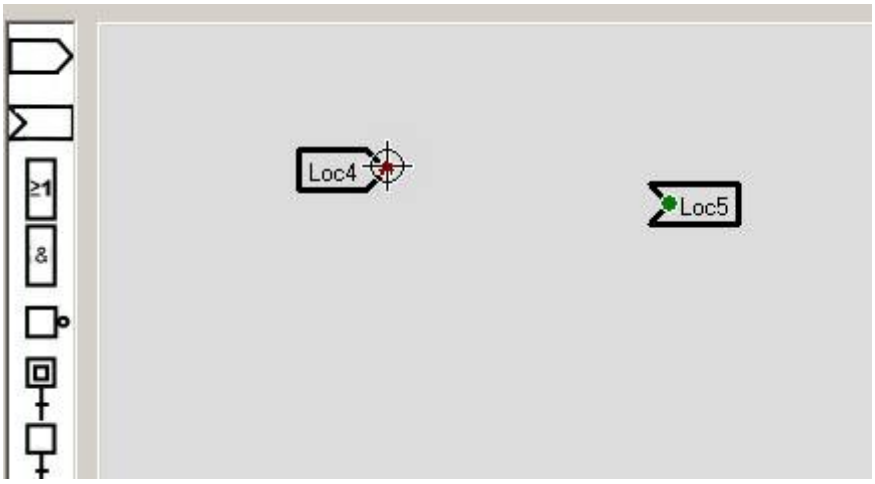
- Red: output connection, can be connected to one or more input connections,

- Green: input connection, can be connected to one output connection,

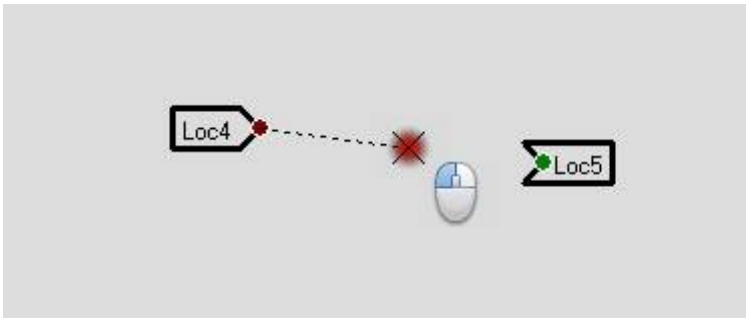
- Blue: SFC connection, can be connected to another SFC connection.

Links creation:

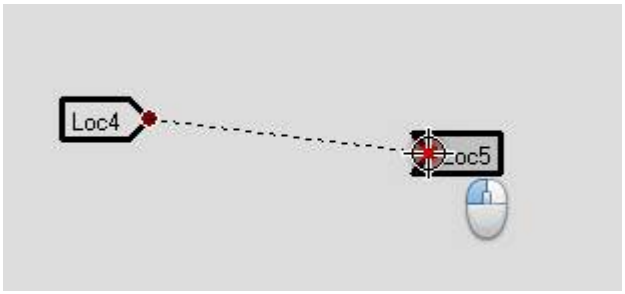
1 - move the cursor over a connection (the cursor changes to a target) and press the left mouse button (without releasing the left button):



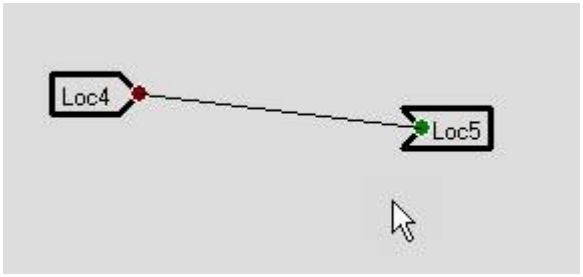
2 - move the cursor (the link which is being created is displayed):



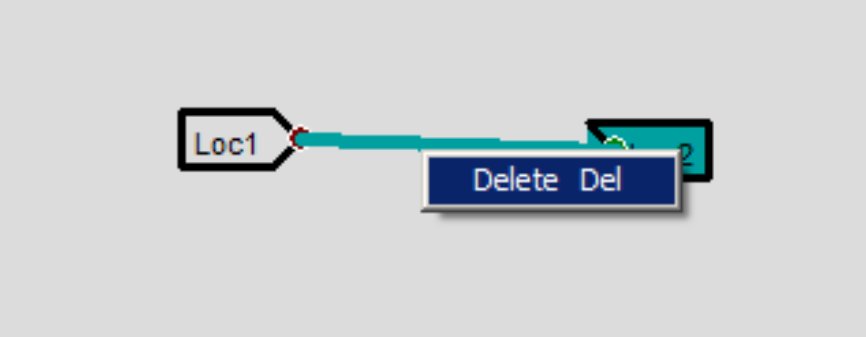
3 - Move the cursor over the target connection:



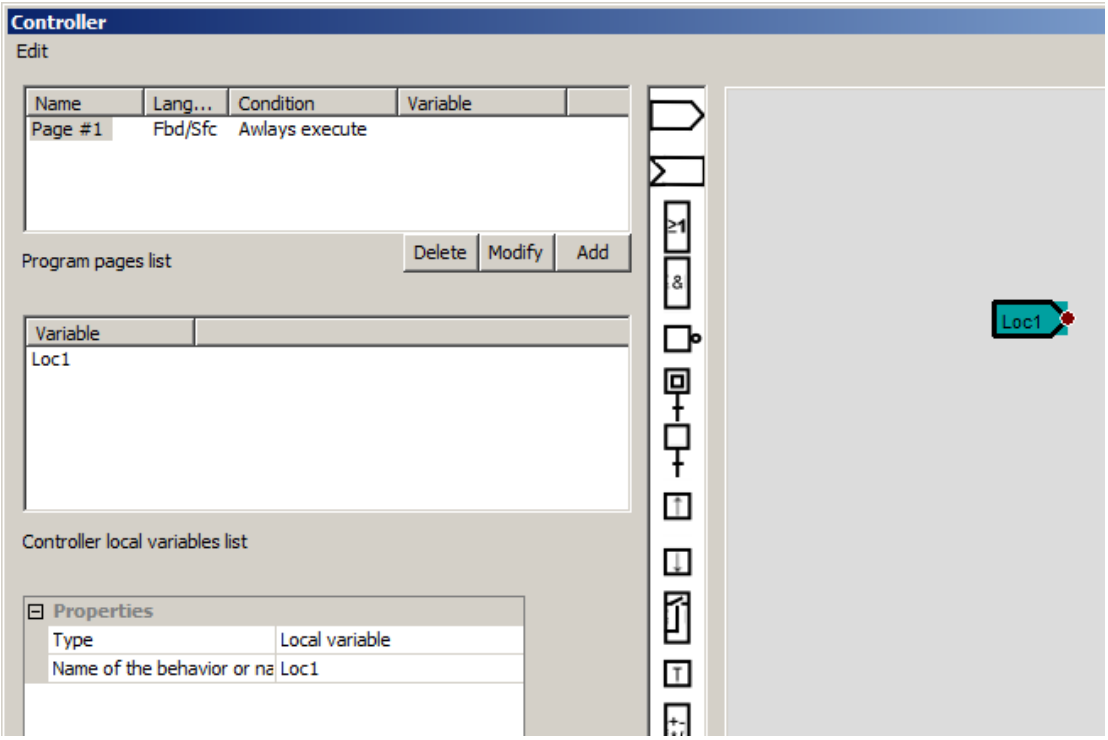
4 - release the left mouse button:



Deleting a link: click right mouse on the link and choose "Delete":



Selecting an item: left click on an item selects it. When an item is selected, its properties are accessible at the bottom left:

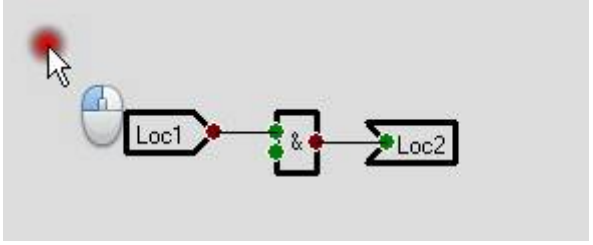


If Shit key is pressed, the item is added to any existing selection.

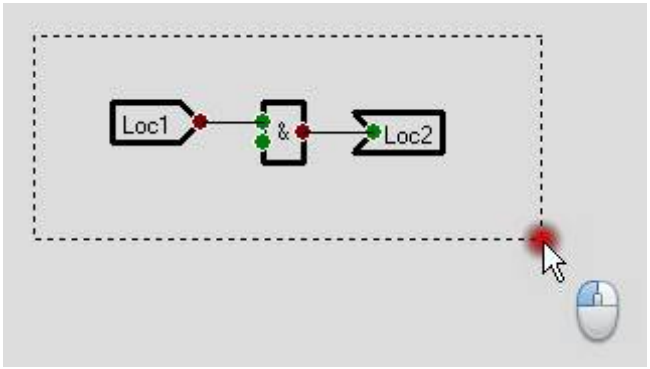
If Control key is pressed, the item is added or removed from the selection.

Selecting a set of elements:

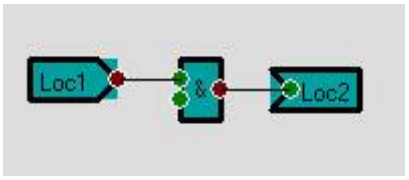
1 - press the left button of the mouse when the cursor is on an empty area of programming (let the button):



2 - move the mouse to select items:



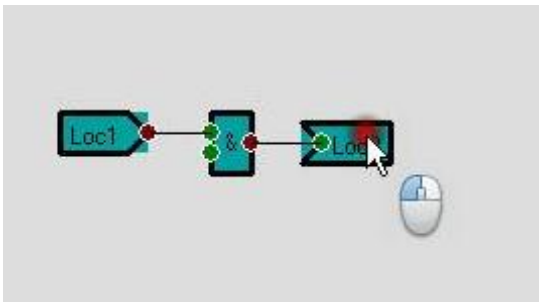
3 - release the left mouse button:



If the Shift key is pressed, the items are added to any existing selection.

Move one or more elements:

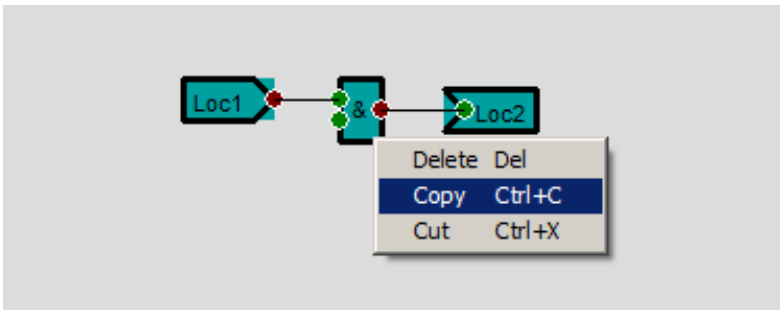
1 - press the left button of the mouse when the cursor is on one of the selected items (keep the button pressed):



2 - move the object or objects,

3 - release the left mouse button.

Deleting or copying of one or more elements: click right mouse on an item and choose in the menu:



Remarks:

- Links connected to a deleted item are also deleted.

- Links connected to two elements copied are also copied.

Paste an item: click right mouse on an empty area of programming and choose "Paste" from the menu.

Items list

 input: input signal associated to a variable (reading of the state of a variable)

or a constant

 output: output signal associated to a variable (writing of the state of a variable)

 logical not boolean function : the output signal is the negation of the input signal

 logical AND boolean function: the output signal is a boolean AND between the 2 input signals

 logical OR boolean function: the output signal is a boolean OR between the 2 input signals

 raising edge: the output signal is true during one program execution cycle when the input signal is changing from state false to true

 falling edge: the output signal is true during one program execution cycle when the input signal is changing from state true to false

 timer: the output signal is true after a delay This delay is defined in seconds unity in the item properties

 assignment: if the bottom input is true, the state of the top input is copied to the state of the output, otherwise, the output state stays unchanged

 calculation: performs a mathematical calculation between the states of the two inputs and writes the result to the state of the output

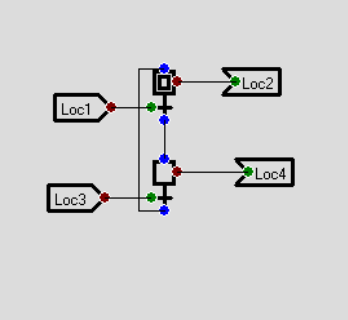
 comparison: compares the states of the two inputs and writes the boolean result (false or true) in the state of the output

 SFC initial steps + transition

 SCF step + transition

 Encapsulation

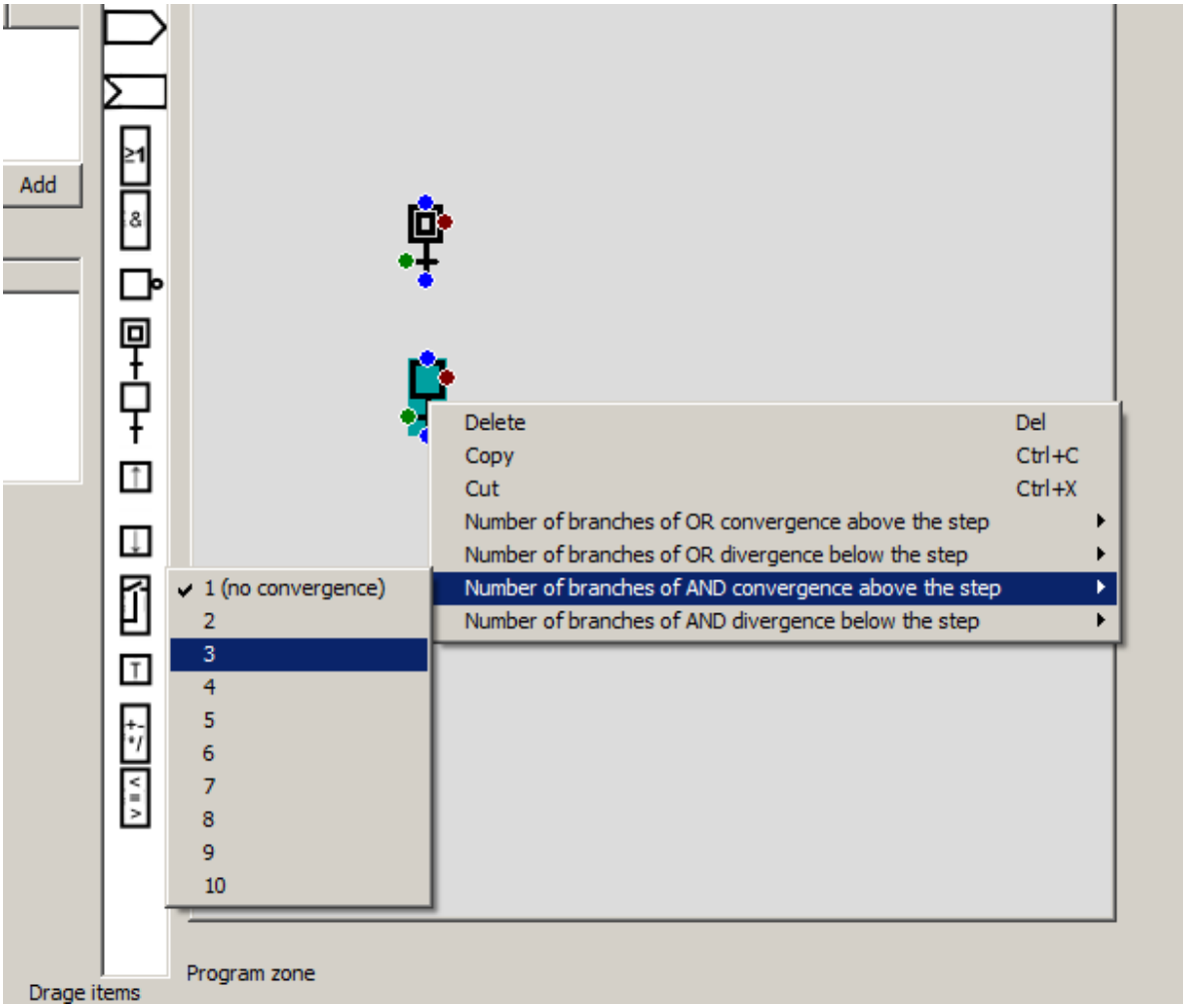
SFC

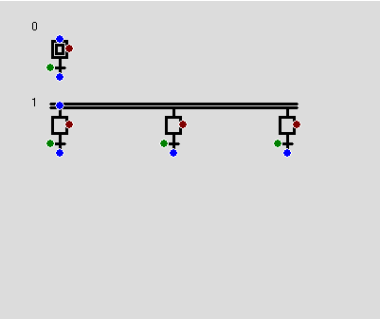


Step + transition blocks have a green connection to connect the condition of the transition and a red connection to connect to a possible action.

Divergence / convergence in AND and OR

Clicking the right mouse button on a selected step (step should be the only one selected), a context menu allows you to set the number of branches for the convergences or divergences in AND or OR:

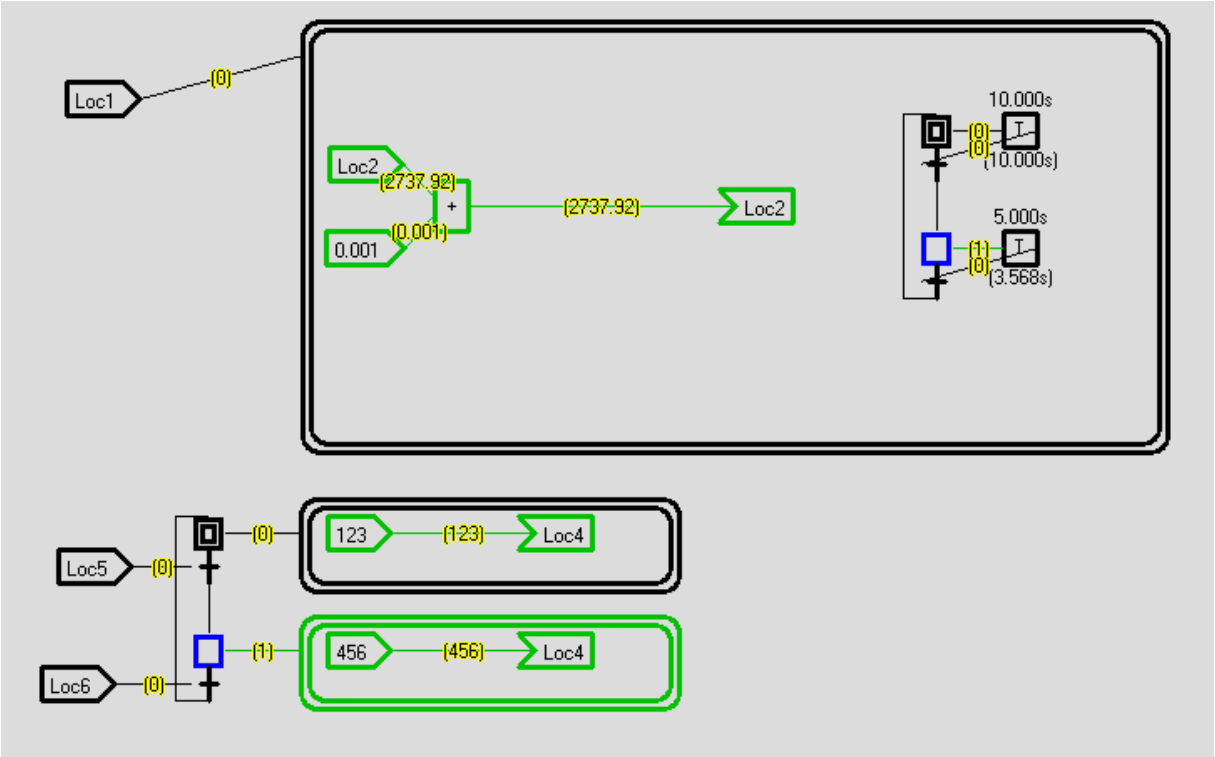



0
1

Encapsulation

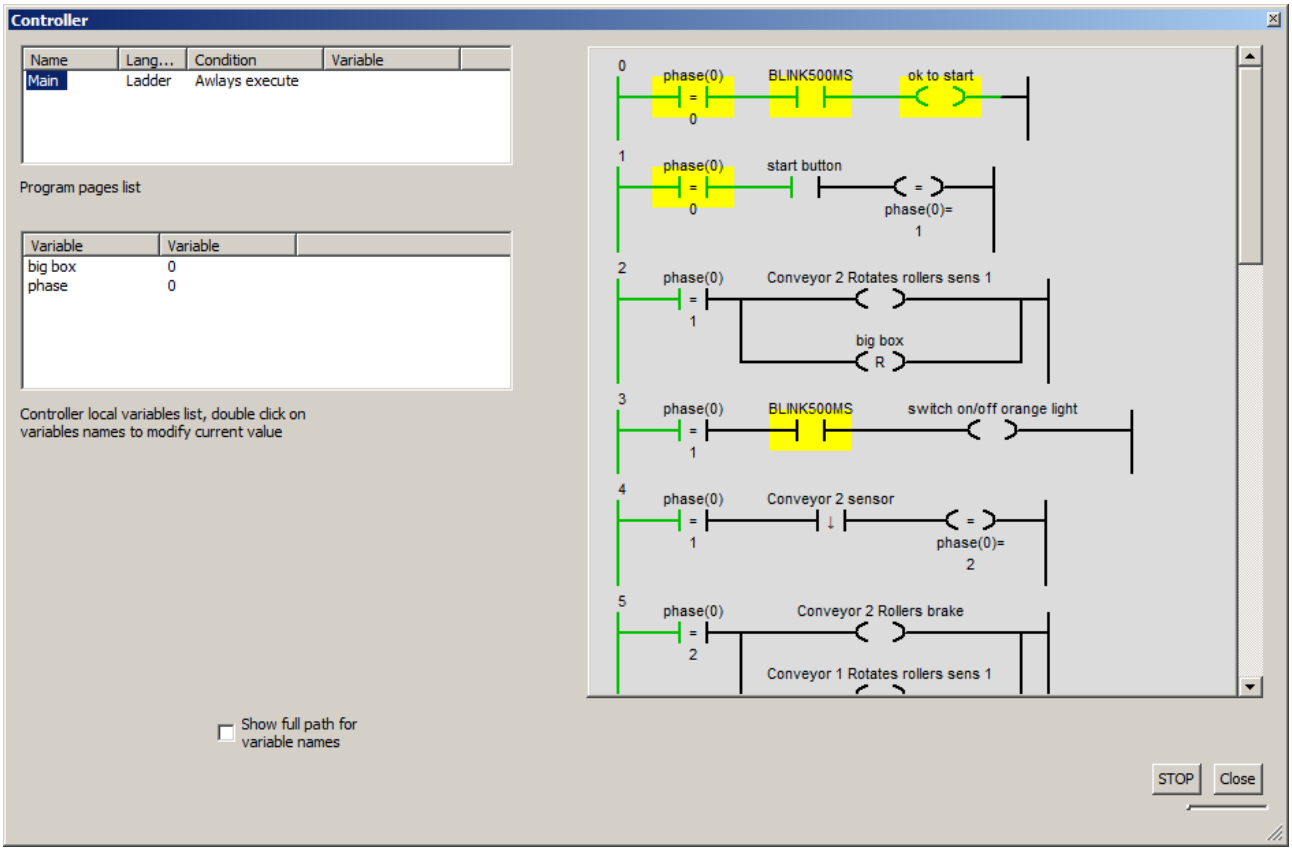
The  block encapsulates a part of a program. This block has one input whose state determines the execution of blocks in the encapsulation. The size of this block is editable by clicking the left mouse button on the black squares around him: let the left mouse button pressed, move the cursor to change the size and release the left button. If one or more blocks or SFCs is in the encapsulation, their evolutions are frozen if the state of the encapsulation block input is false.

Example :

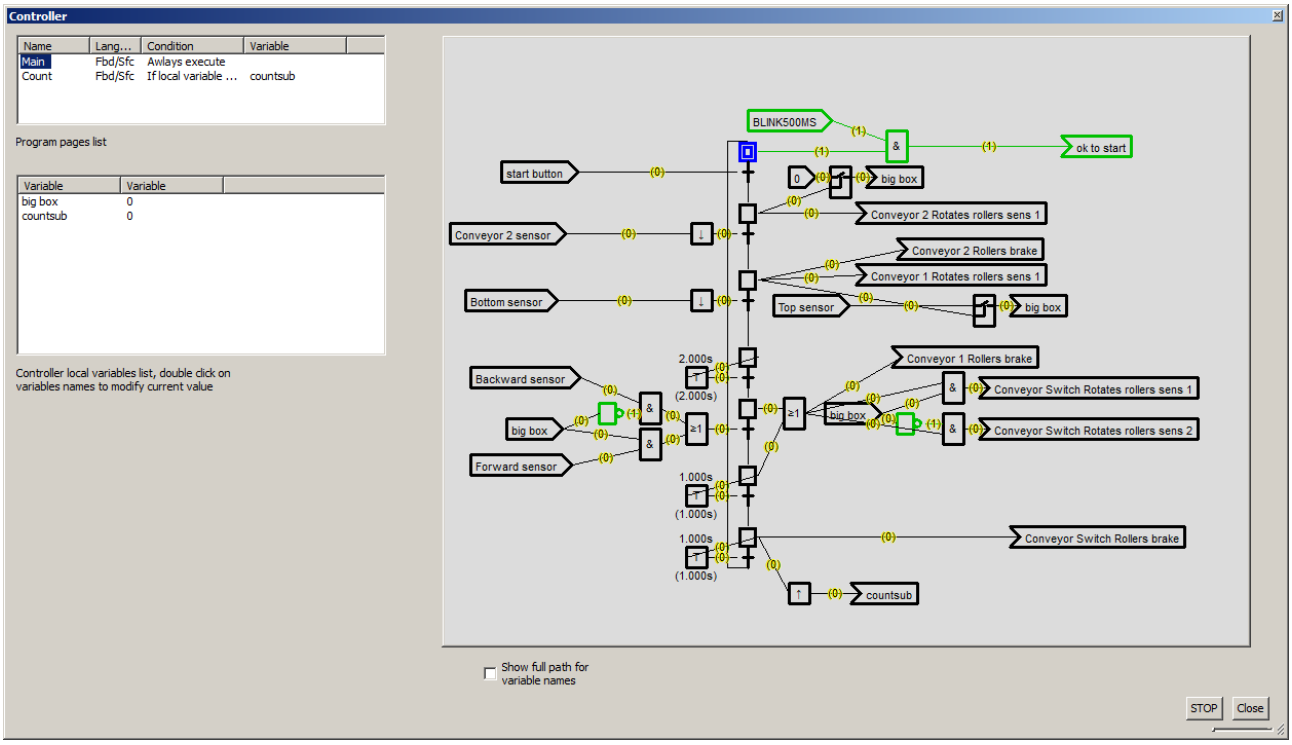


RUN mode

In RUN mode, the status of programs are displayed and the status of local variables in the list in the left area:



Double click on the name of a local variable can change its state. Right click reverses the state (considering the variable as a boolean variable).

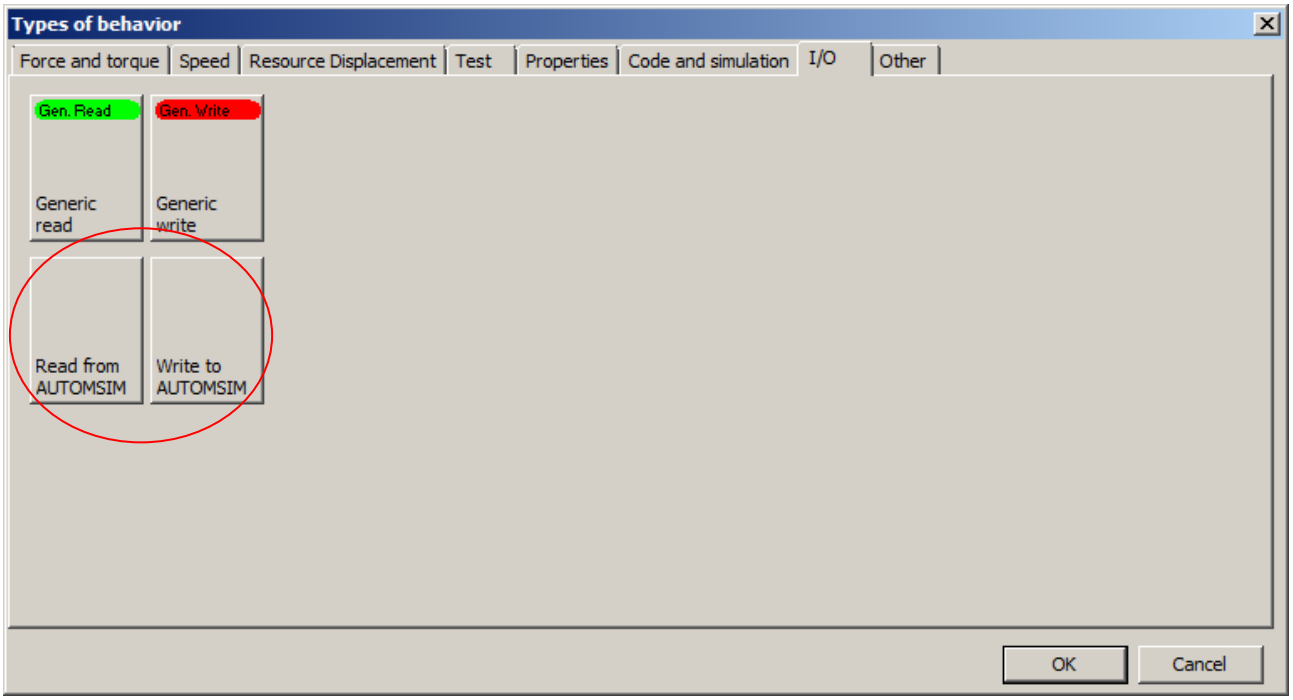


Access to the pages of the controller program is possible in the configuration window or menu "Simulation / Debug / Program - Simulation" if "simulation Menu" is enabled in the controller properties.

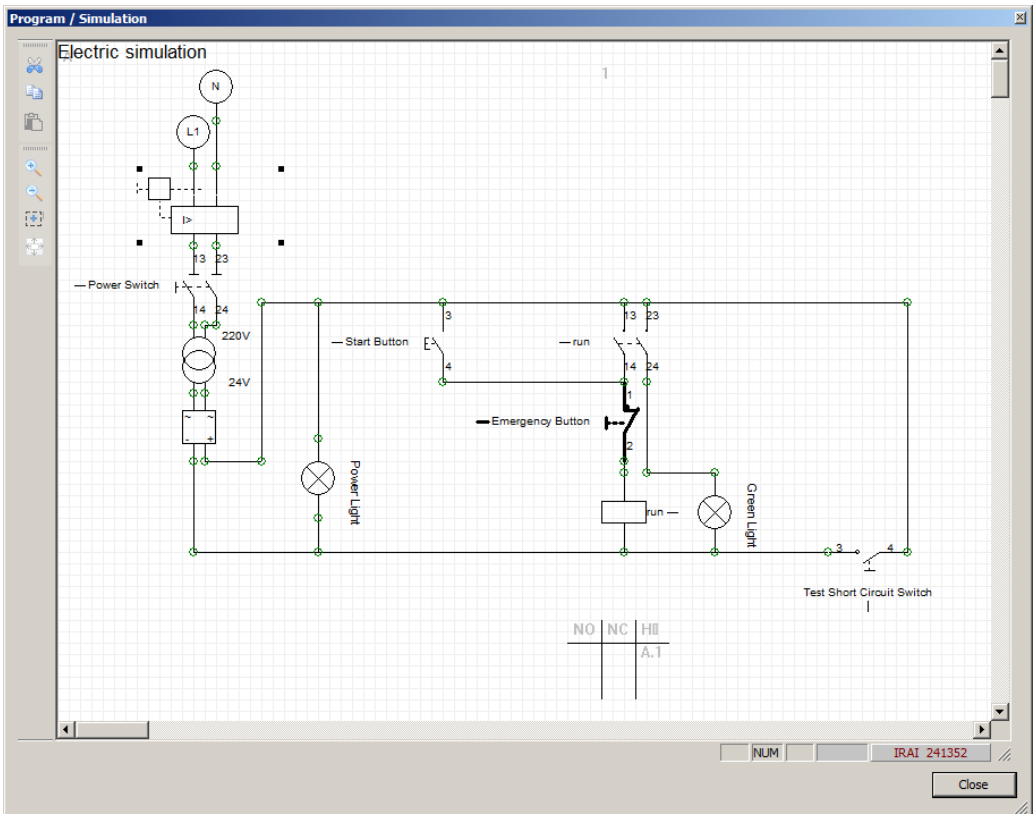
Diagrams simulation

The diagrams editor integrated to Virtual Universe Pro allows to create and simulate electric, pneumatic, hydraulic and digital electronic diagrams linked to 3D items.

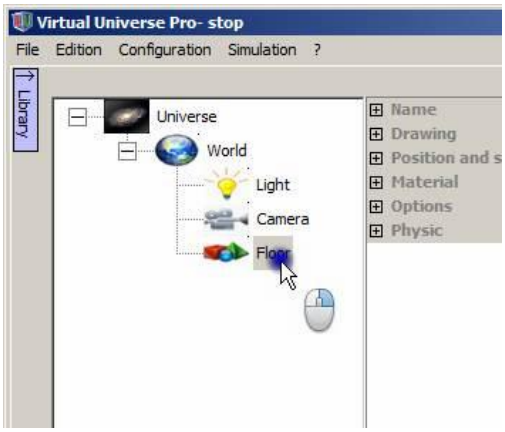
The behaviors called "read from AUTOMSIM" and "write to AUTOSIM" permit to read or write a stat from or to a simulation folder.

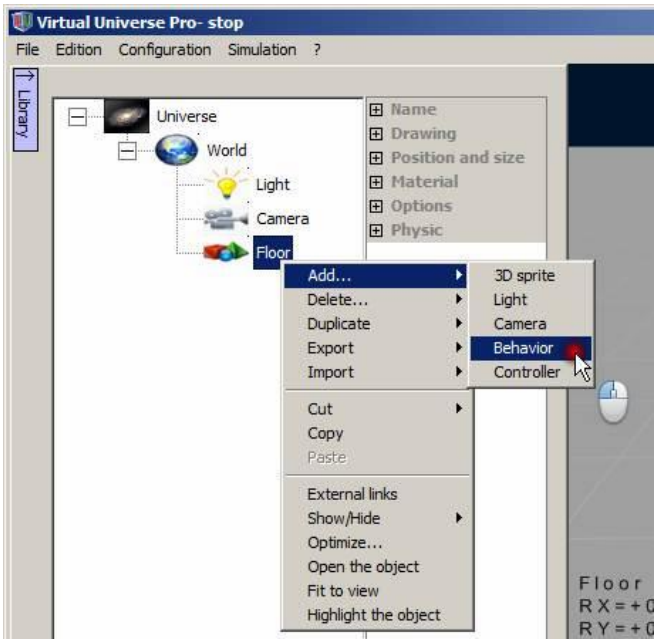


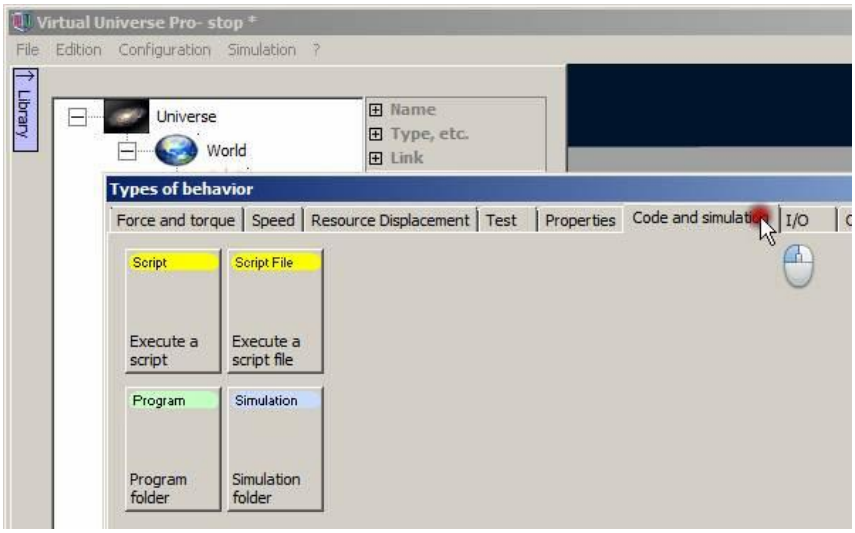
Example of an electric enclosure associated to a diagram:

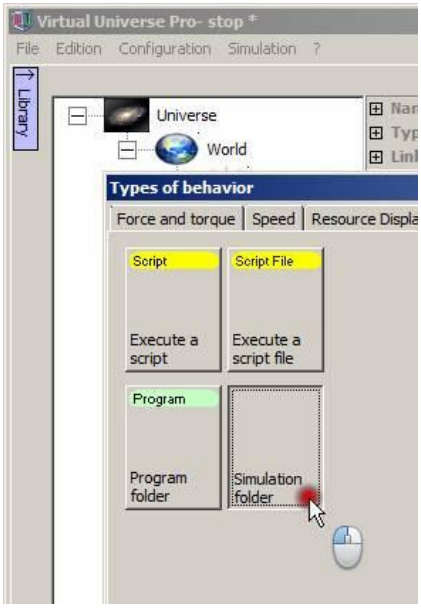


Creation of a simulation folder

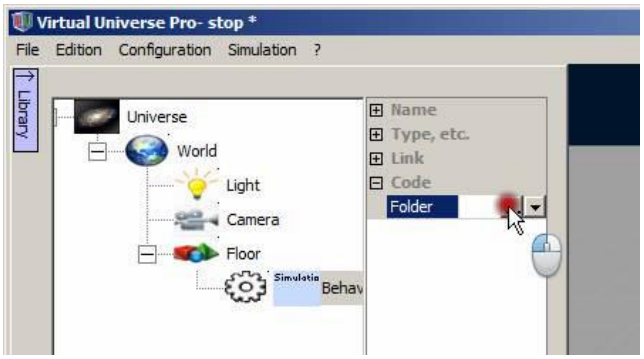






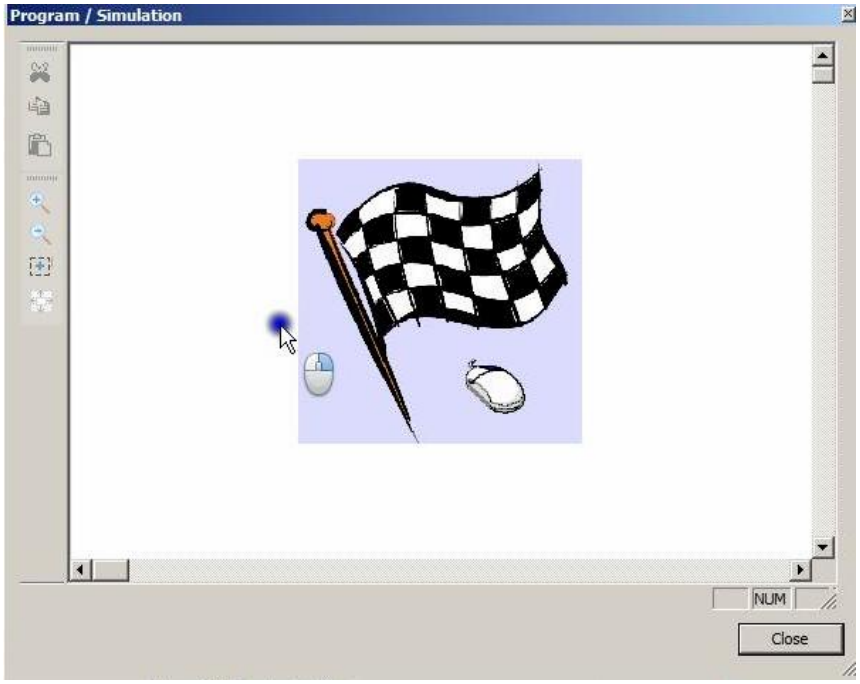


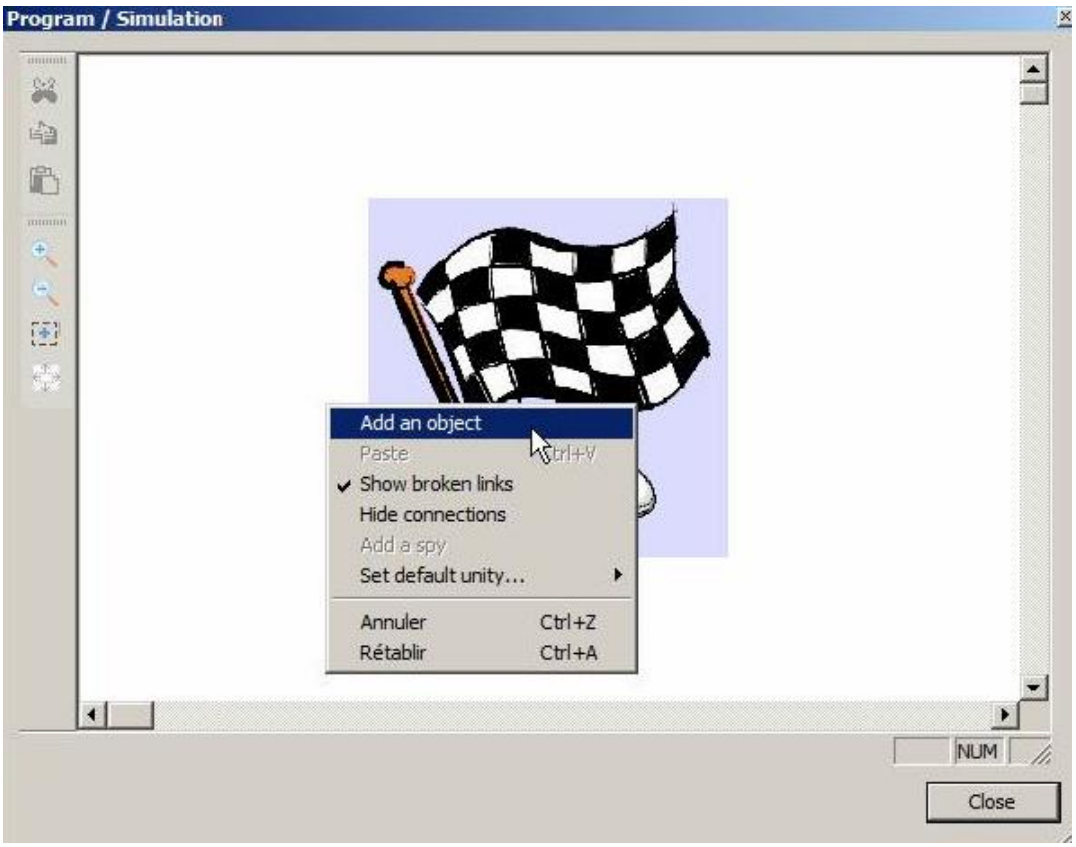
Modifying a simulation folder

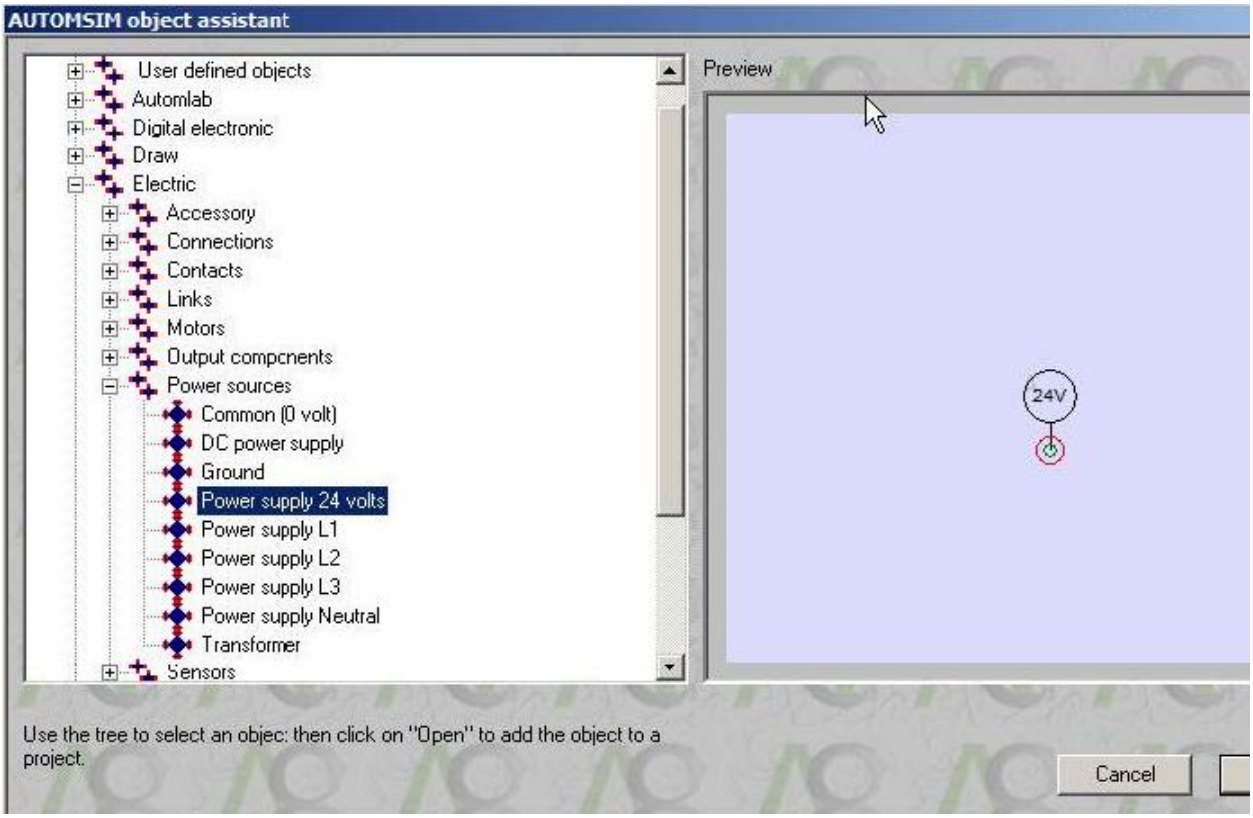


A double click on the behavior name also permits the modification of a somulation folder.

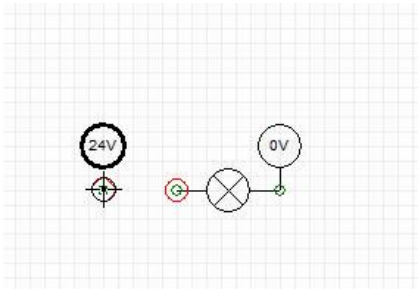
Adding an object on a simulation folder

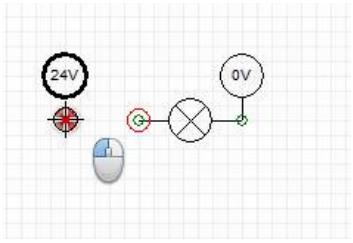


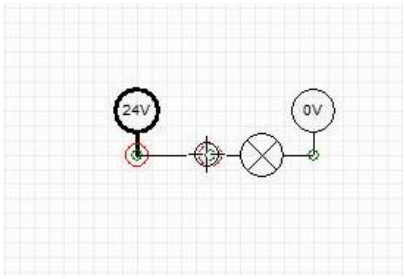

Program / Simulation
Add an object
Paste Ctrl+V
Show broken links
Hide connections
Add a spy
Set default unity...
Annuler Ctrl+Z
Rétablir Ctrl+A
NUM
Close

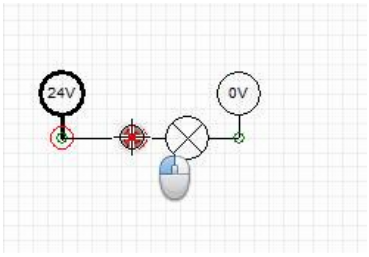

AUTOMSIM object assistant
User defined objects
Automlab
Digital electronic
Draw
Electric
Accessory
Connections
Contacts
Links
Motors
Output compcnents
Power sources
Common (0 volt)
DC power supply
Ground
Power supply 24 volts
Power supply L1
Power supply L2
Power supply L3
Power supply Neutral
Transformer
Sensors
Preview
24V
Use the tree to select an objec: then click on "Open" to add the object to a project.
Cancel

Drawing a link on a simulation folder

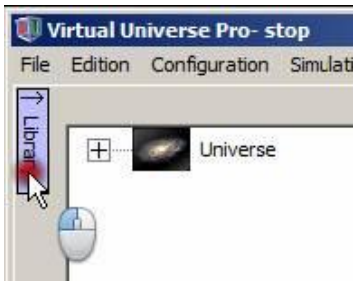


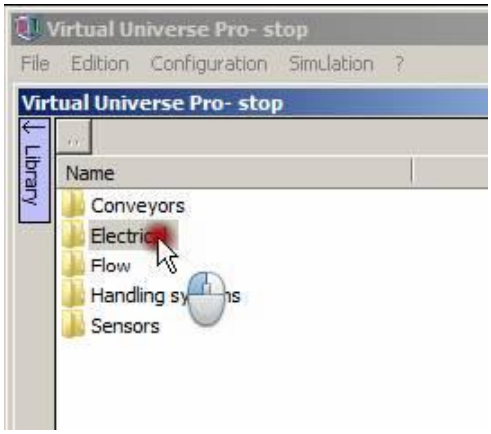






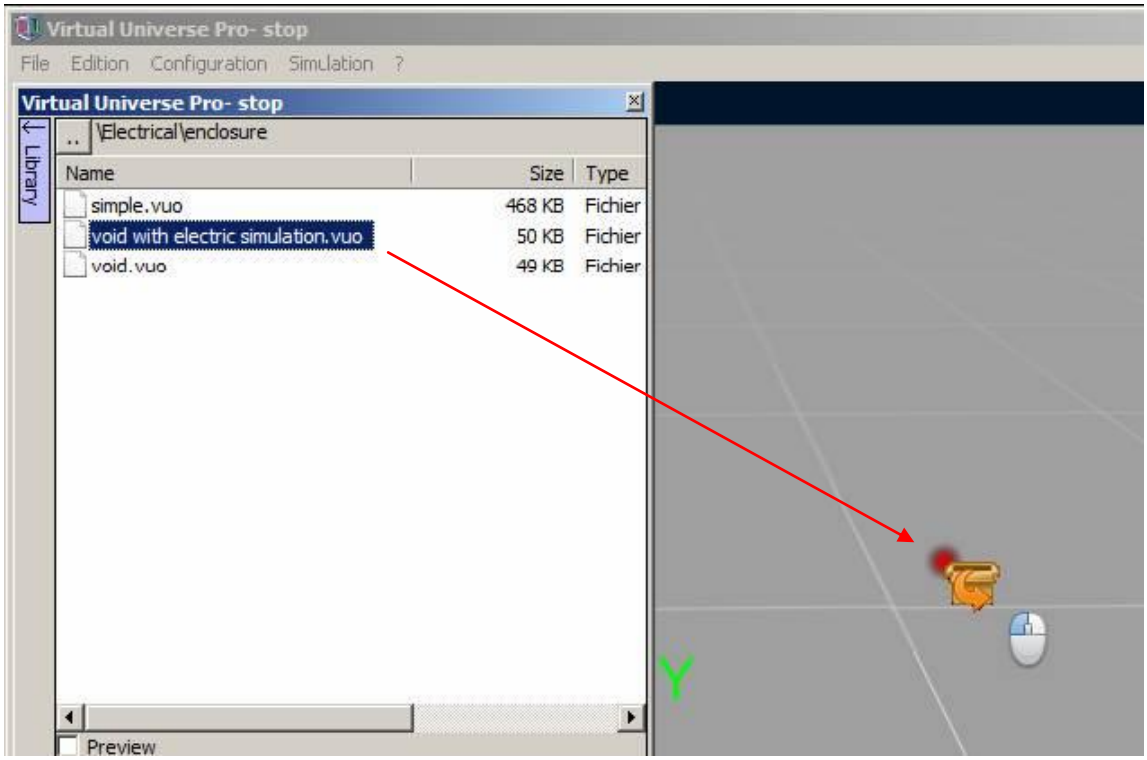
Automatic generation of a diagram

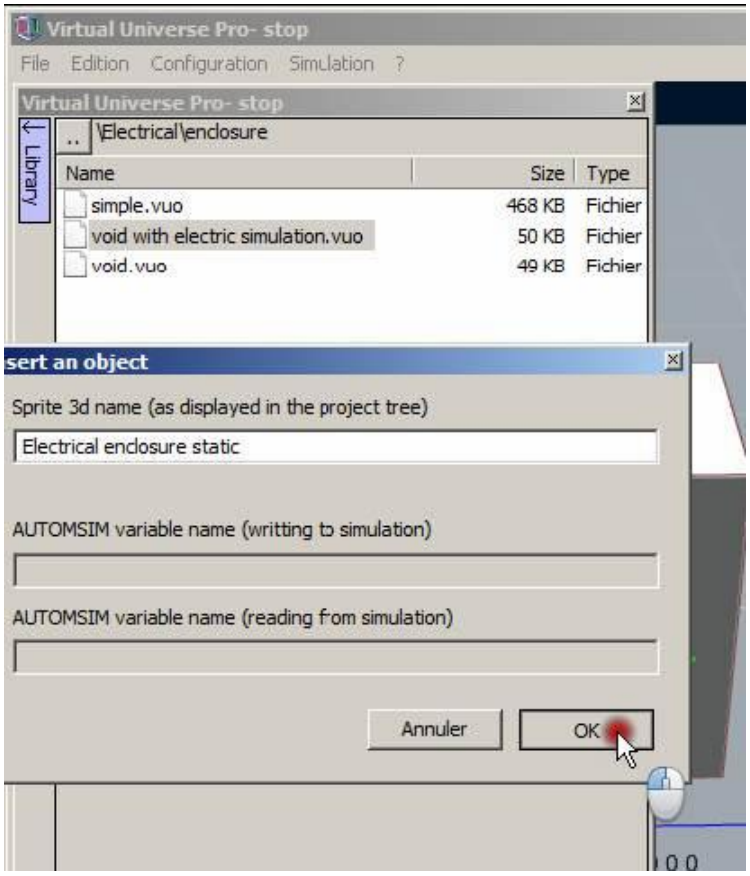


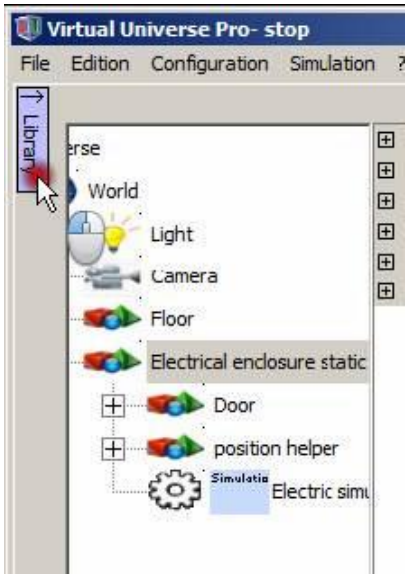

Virtual Universe Pro- stop
File Edition Configuration Simulation ?
Virtual Universe Pro- stop
Library
..
Name
Conveyors
Electric
Flow
Handling sy ns
Sensors

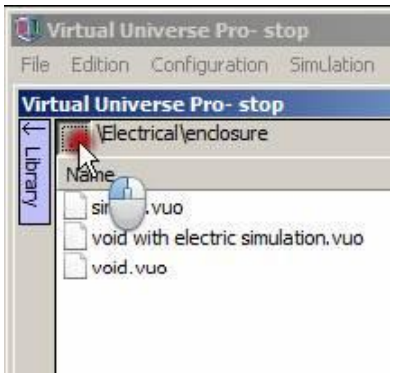

Virtual Universe Pro- stop
File Edition Configuration Simulation
Virtual Universe Pro- stop
Library
..
\Electrical
Name
circuit breaker
emergency
enclosu
light
plc
push button and light
switch and light

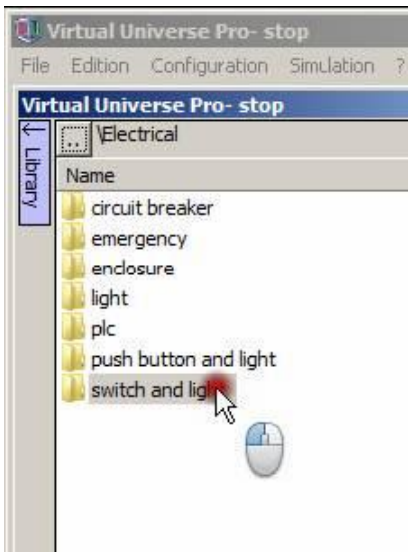

Virtual Universe Pro- stop
File Edition Configuration Simulation ?
Virtual Universe Pro- stop
Library
..
\Electrical\enclosure
Name
Size
simple.vuo
468 KB
void with electric simulation.vu
50 KB
void.vuo
49 KB

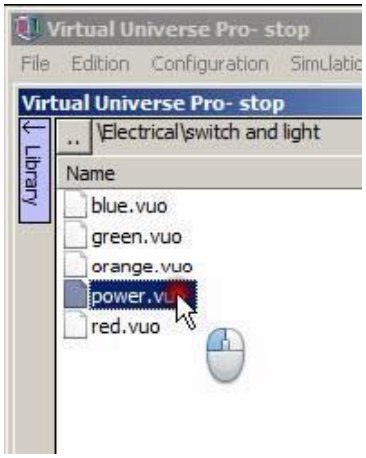

Virtual Universe Pro- stop
File Edition Configuration Simulation ?
Virtual Universe Pro- stop
Library
..
\Electrical\enclosure
Name
Size
Type
simple.vuo
468 KB
Fichier
void with electric simulation.vuo
50 KB
Fichier
void.vuo
49 KB
Fichier
Preview
Y

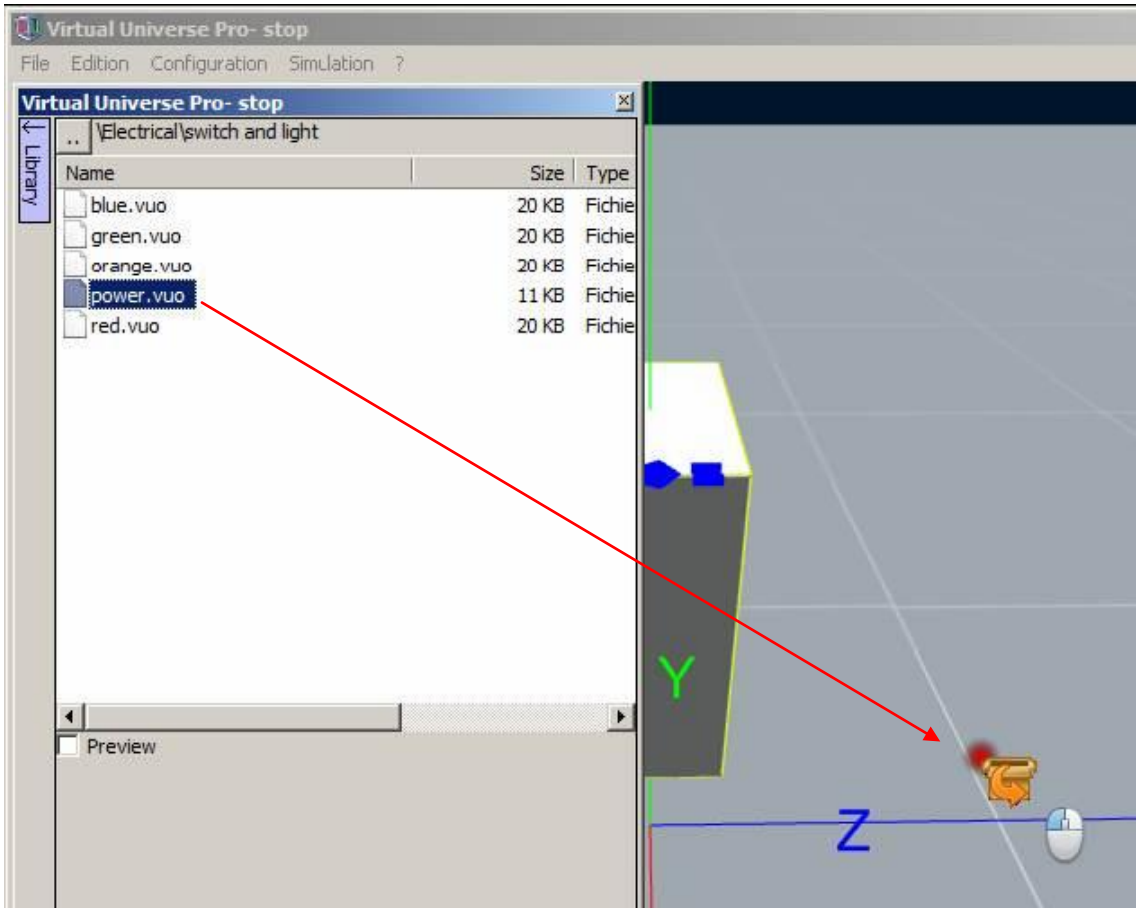

Virtual Universe Pro- stop
File Edition Configuration Simulation ?
Virtual Universe Pro- stop
Library
..
\Electrical\enclosure
Name
Size
Type
simple.vuo
468 KB
Fichier
void with electric simulation.vuo
50 KB
Fichier
void.vuo
49 KB
Fichier
sert an object
Sprite 3d name (as displayed in the project tree)
Electrical enclosure static
AUTOMSIM variable name (writting to simulation)
AUTOMSIM variable name (reading from simulation)
Annuler
OK

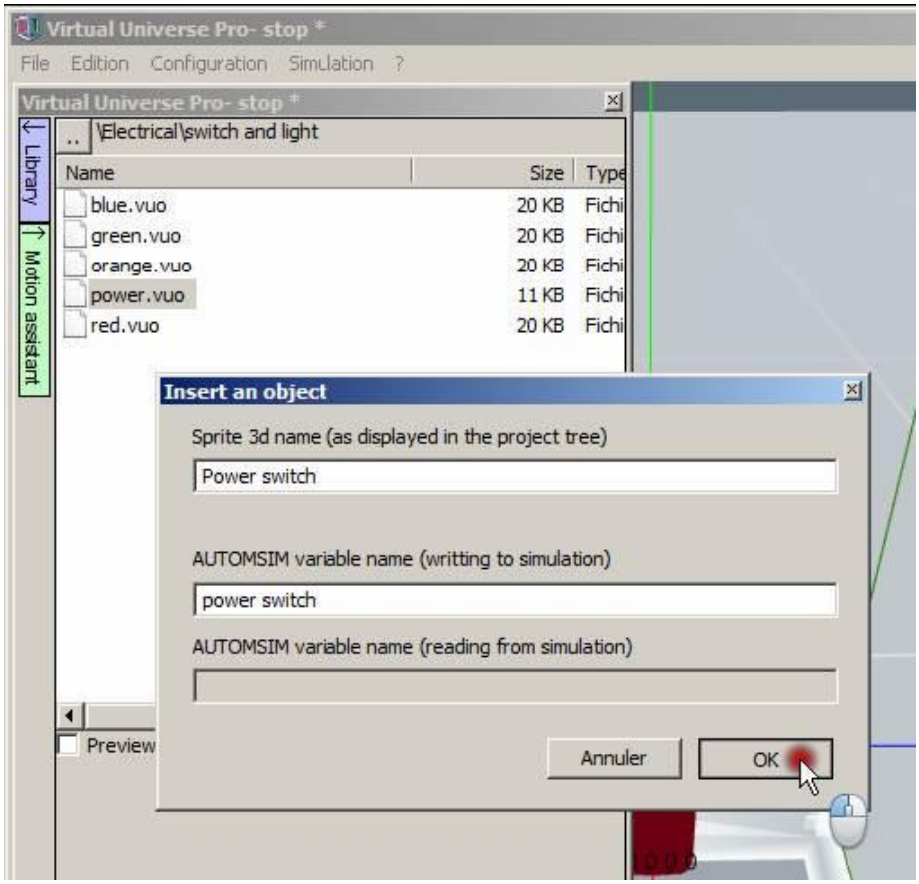

Virtual Universe Pro- stop
File Edition Configuration Simulation
Library
World
Light
Camera
Floor
Electrical enclosure static
Door
position helper
Electric simu

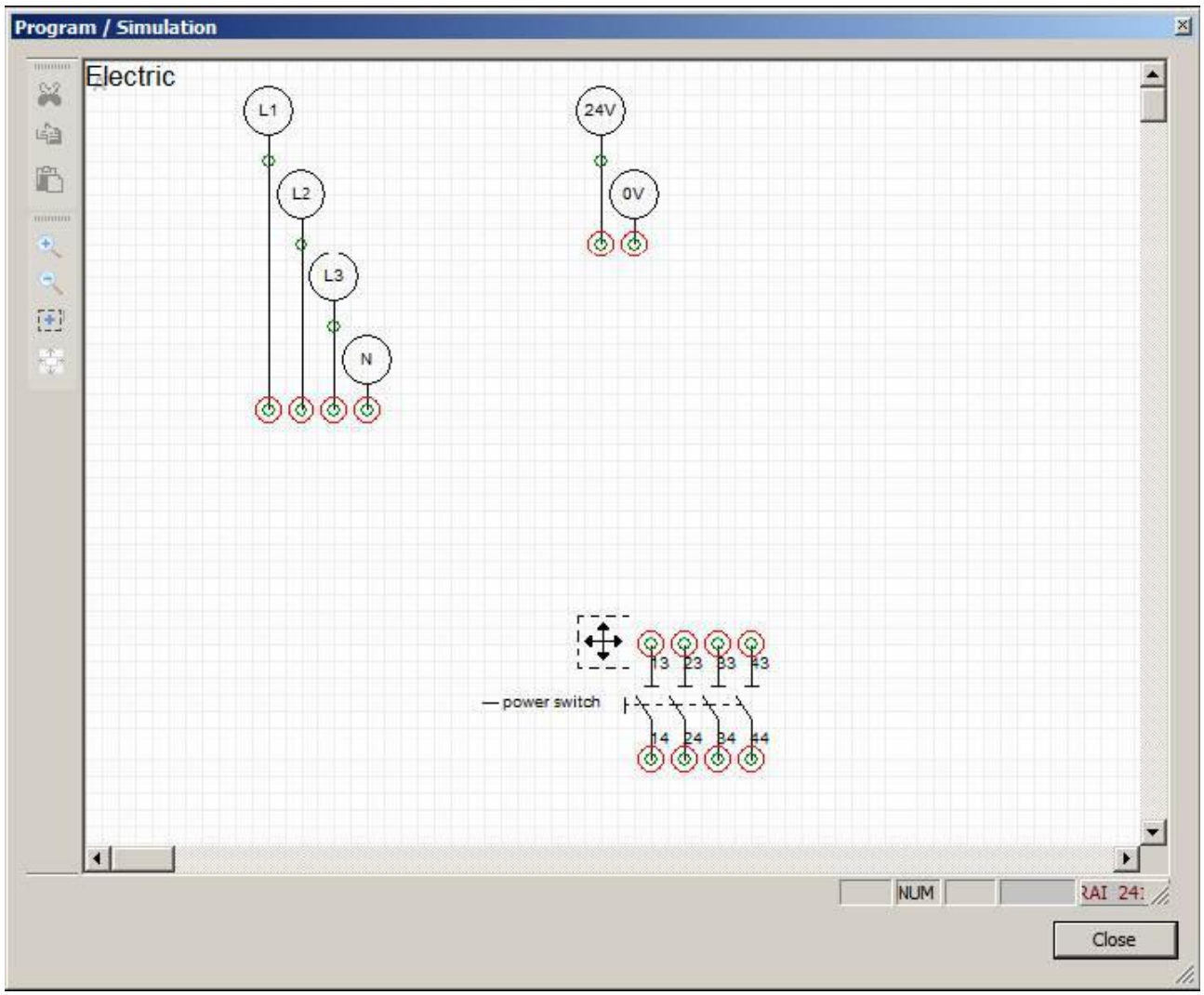

Virtual Universe Pro- stop
File Edition Configuration Simulation
Virtual Universe Pro- stop
Library
\Electrical\enclosure
Name
.vuo
void with electric simulation.vuo
void.vuo

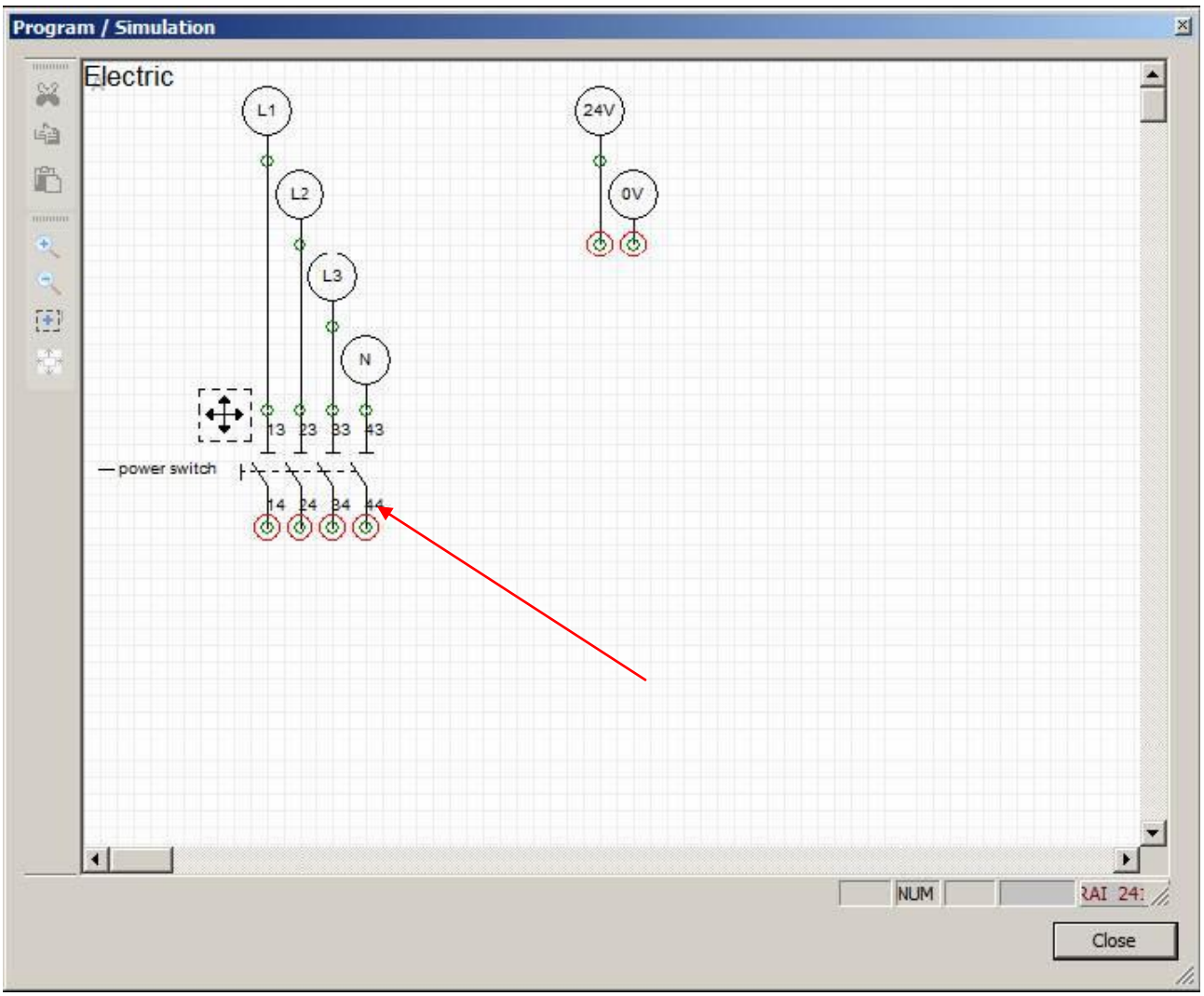

Virtual Universe Pro- stop
File Edition Configuration Simulation
Virtual Universe Pro- stop
Library
\Electrical
Name
circuit breaker
emergency
enclosure
light
plc
push button and light
switch and

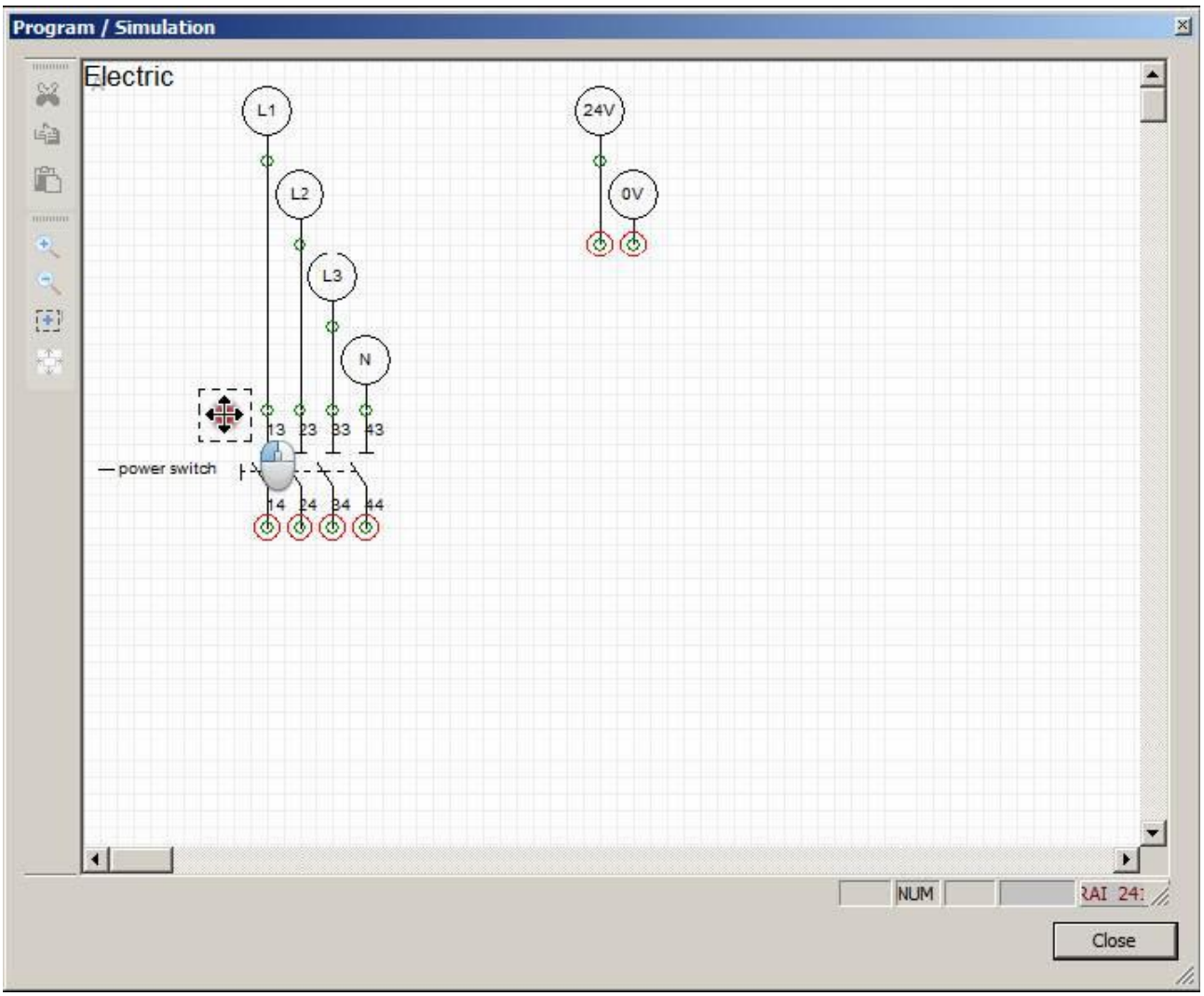

Virtual Universe Pro- stop
File Edition Configuration Simulatic
Virtual Universe Pro- stop
Library
..
\Electrical\switch and light
Name
blue.vuo
green.vuo
orange.vuo
power.vu
red.vuo

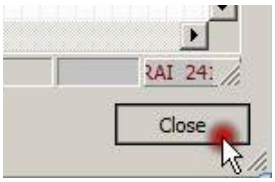

Virtual Universe Pro- stop
File Edition Configuration Simulation ?
Virtual Universe Pro- stop
Library
..
\Electrical\switch and light
Name
Size
Type
blue.vuo 20 KB Fichie
green.vuo 20 KB Fichie
orange.vuo 20 KB Fichie
power.vuo 11 KB Fichie
red.vuo 20 KB Fichie
Preview
Y
Z

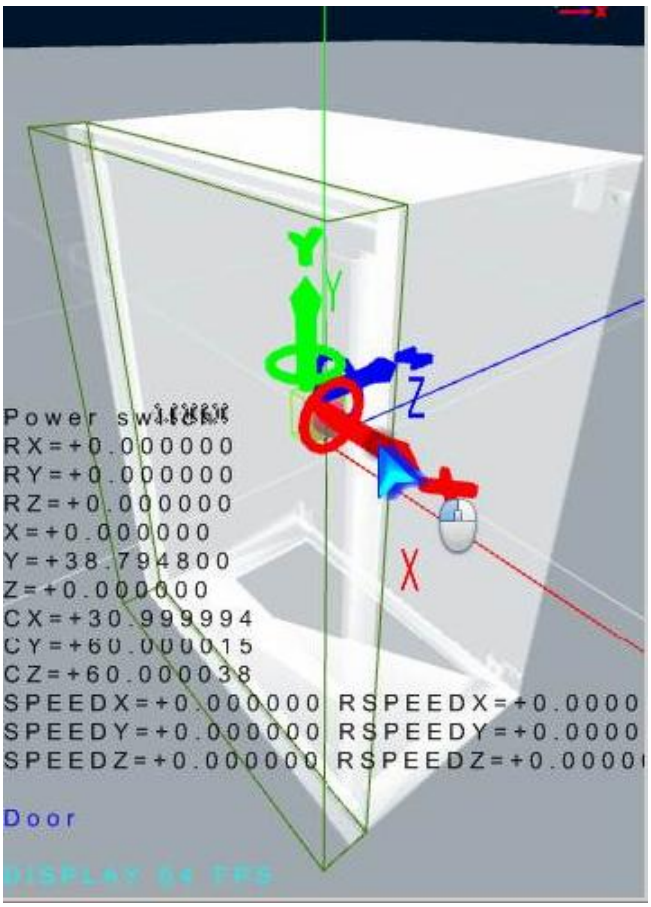

Virtual Universe Pro- stop *
File Edition Configuration Simulation ?
Virtual Universe Pro- stop *
Library
Motion assistant
..
\Electrical\switch and light
Name
Size
Type
blue.vuo 20 KB Fichi
green.vuo 20 KB Fichi
orange.vuo 20 KB Fichi
power.vuo 11 KB Fichi
red.vuo 20 KB Fichi
Insert an object
Sprite 3d name (as displayed in the project tree)
Power switch
AUTOMSIM variable name (writting to simulation)
power switch
AUTOMSIM variable name (reading from simulation)
Preview
Annuler
OK

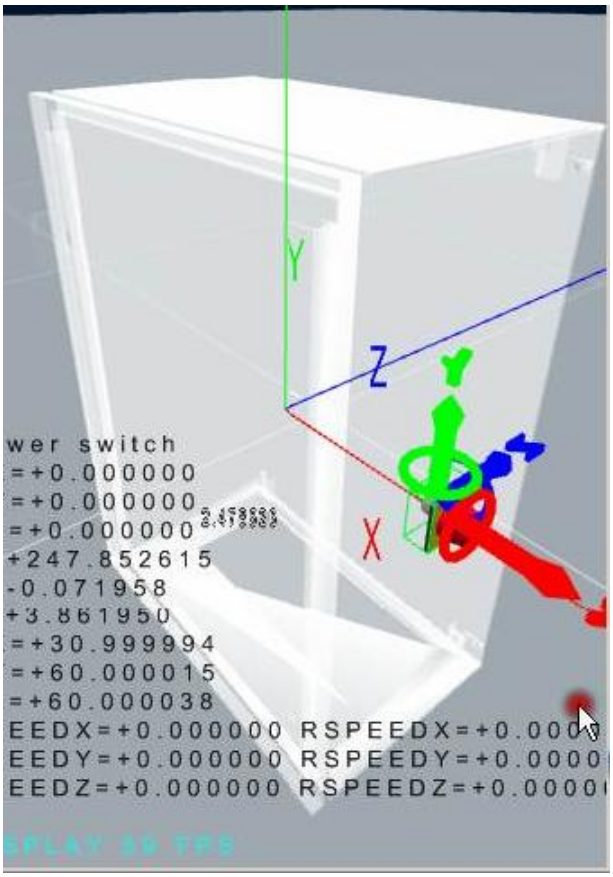

Program / Simulation
Electric
L1
L2
L3
N
24V
0V
power switch
13
23
33
43
14
24
34
44
NUM
Close

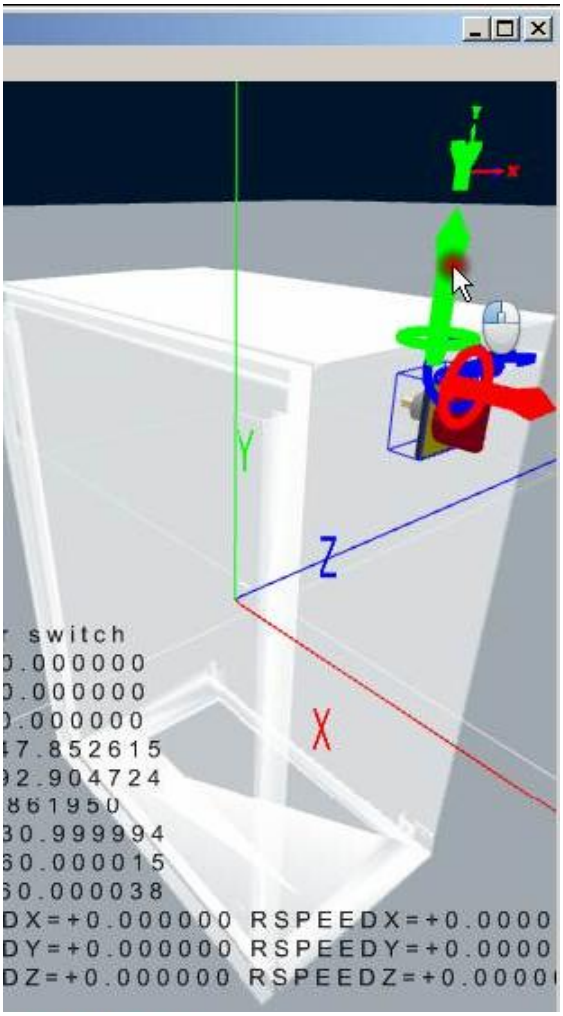

Program / Simulation
Electric
L1
L2
L3
N
24V
0V
13
23
33
43
power switch
14
24
34
44
NUM
Close


Program / Simulation
Electric
L1
L2
L3
N
24V
0V
13 23 33 43
power switch
14 24 34 44
NUM
Close

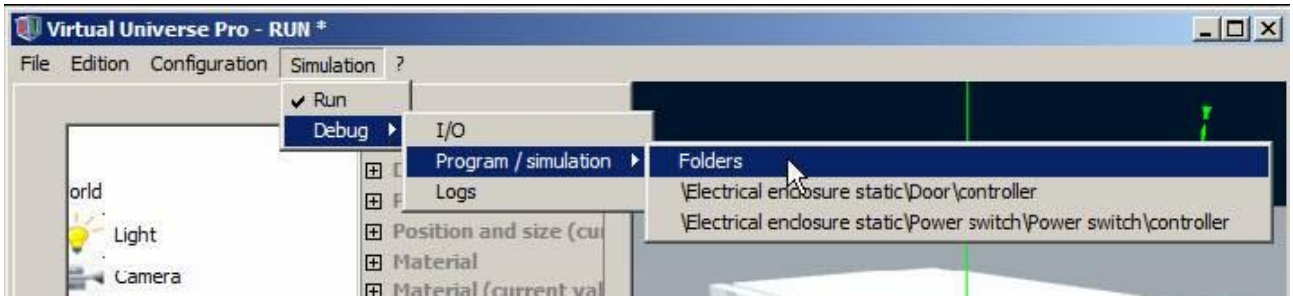

Close

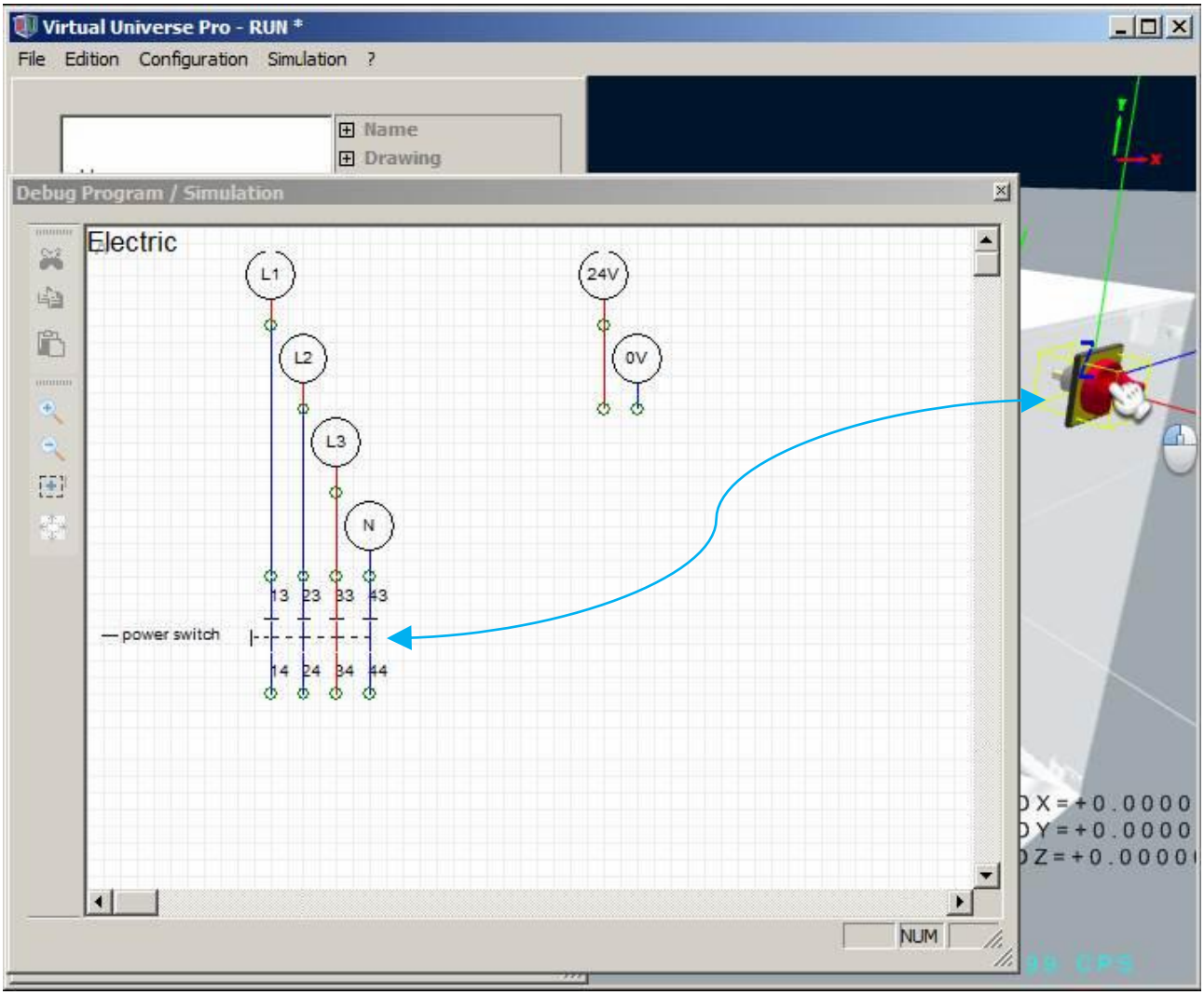

Power switch
RX=+0.000000
RY=+0.000000
RZ=+0.000000
X=+0.000000
Y=+38.794800
Z=+0.000000
CX=+30.999994
CY=+60.000015
CZ=+60.000038
SPEEDX=+0.000000 RSPEEDX=+0.0000
SPEEDY=+0.000000 RSPEEDY=+0.0000
SPEEDZ=+0.000000 RSPEEDZ=+0.0000
Door
Y
Z
X


wer switch
=+0.000000
=+0.000000
=+0.000000
+247.852615
-0.071958
+3.861950
=+30.999994
=+60.000015
=+60.000038
EEDX=+0.000000 RSPEEDX=+0.000
EEDY=+0.000000 RSPEEDY=+0.0000
EEDZ=+0.000000 RSPEEDZ=+0.0000
Y
Z
X



Hold down the ALT key to enable free moving of the object.



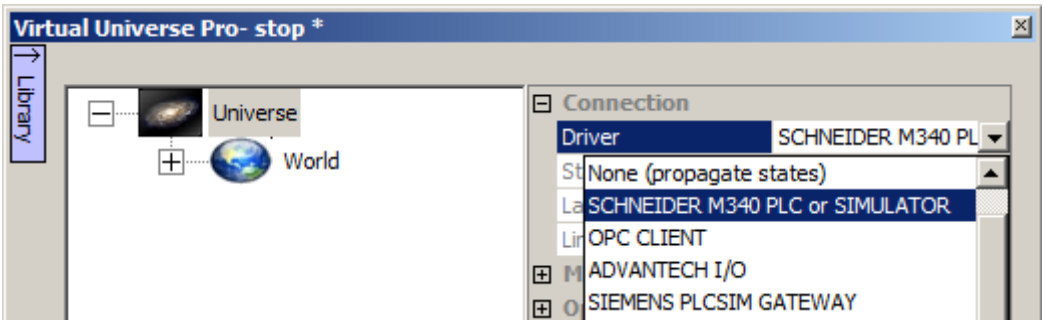


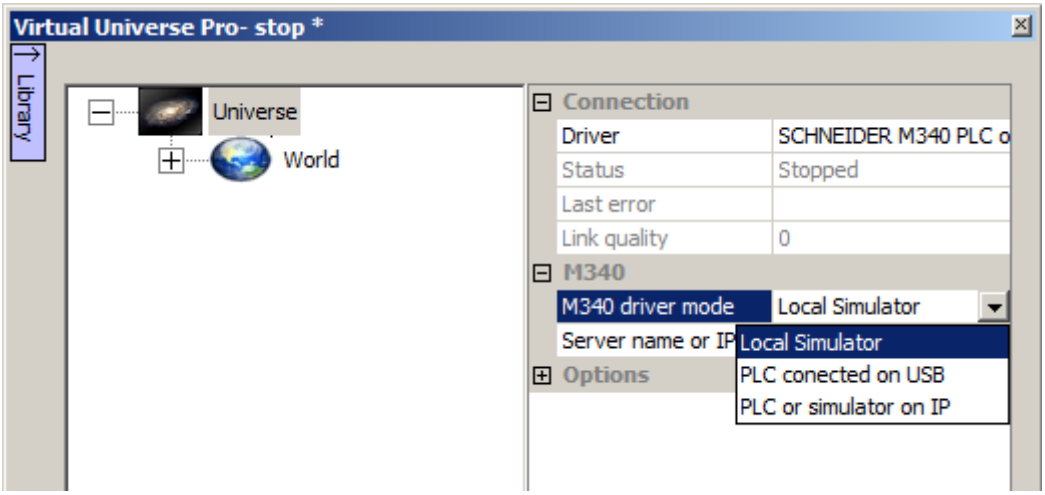
The examples which can be found on the "Electric and pneumatic" sub directory of the Virtual Universe Pro installation directory show the diagram simulation functionalities.

External connections

The "PLC connection samples" directory of the Virtual Universe Pro example directory contains Virtual Universe Pro examples and also programs created with the corresponding PLC workshops.

Connection with a Schneider Electric M340 PLC or Unity Pro simulator.





"Local simulator" allows the communication with Unity Pro V4 or greater running on the same PC.

"PLC connected on USB" allows the communication with a M340 PLC on USB port.

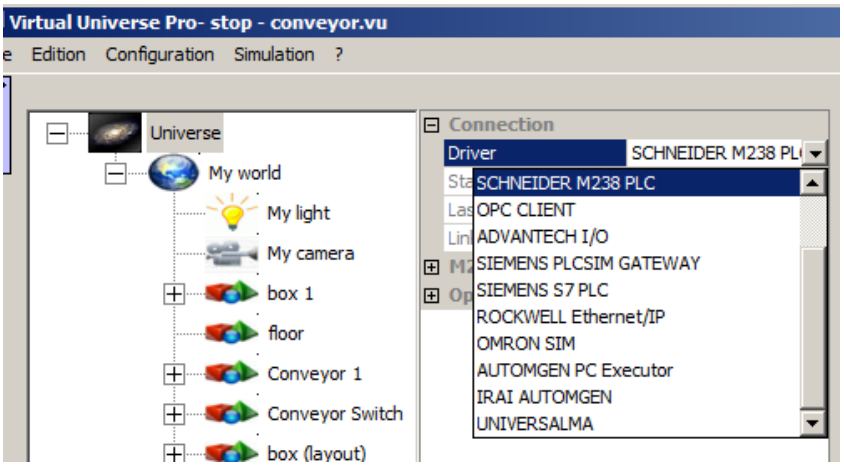
"PLC or simulator on IP" allows the communication with a M340 PLC connected on Ethernet or with Unity Pro simulator running on another PC connected over Ethernet. For this mode, the network name or IP address have to be filled in.

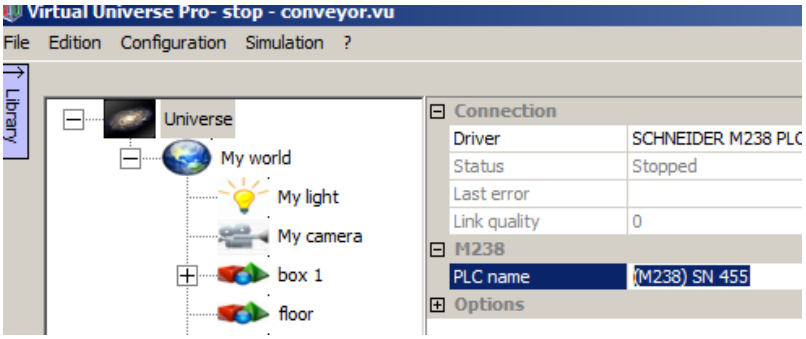
The usable variables names for links are inputs, outputs and internal variables. Bit, 16 bits words, 32 bits words and floating types can be used. Examples: %I0.0.1, %MW10, %QW0.12, %MF15

Limitation: the Unity Pro simulator doesn't allow access to I/O variables. In this case, you must use internal variables.

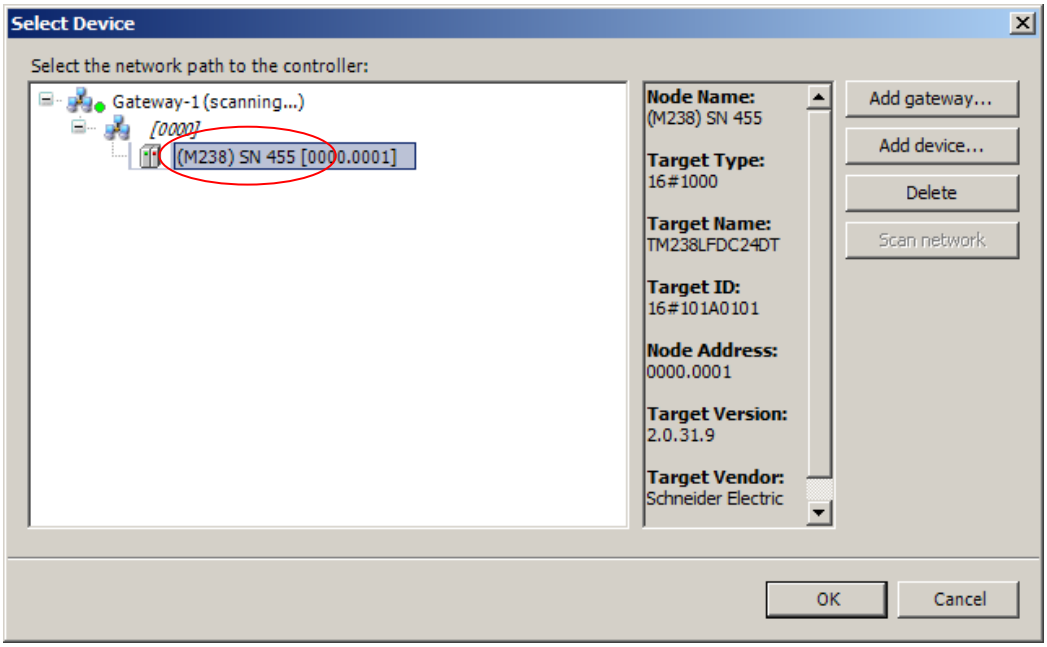
Connection to a Schneider Electric m238 PLC

The SoMachine Schneider Electric software must be installed on the PC: Virtual Universe Pro uses SoMachine gateway for communicating with m238 PLC. The PC must be connected to the PLC. SoMachine and Virtual Universe Pro can communicate with the PLC at the same time.

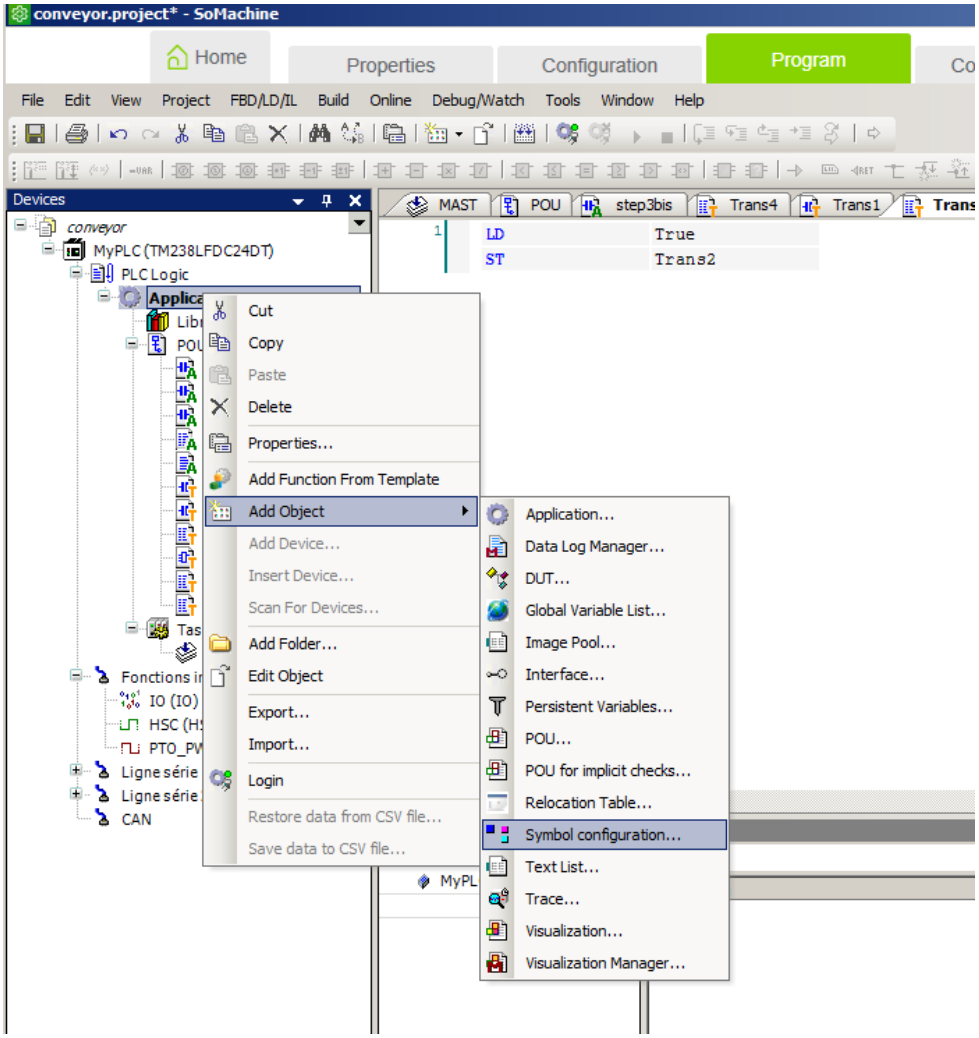




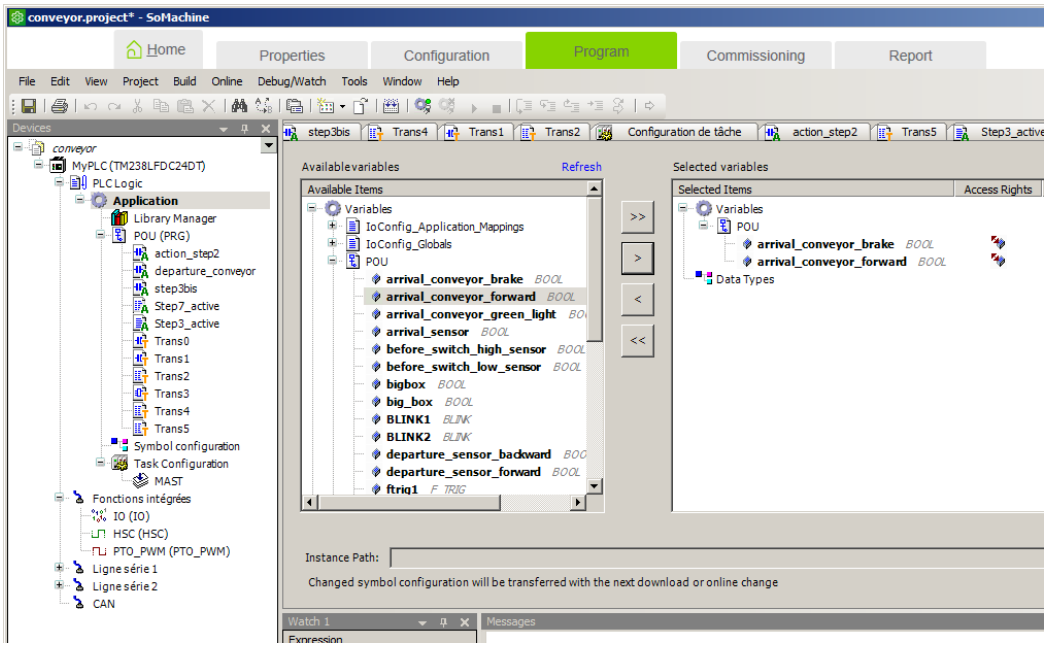
The PLC name have to be the same than the one selected in SoMachine:



The variables of the Somachine project which must be read or written by Virtual Universe Pro have to be added in a Symbol Configuration item:



Select the application variables and copy them to the right area:



Then, close the "Symbol configuration" windows, rebuild the application and download it to the PLC.

The syntax to use for referencing m238 PLC variables in Virtual Universe Pro is as following:

<application name>.<POU name or GVL for global variables>.<variable name>

Example:

Application.GVL.myvariable

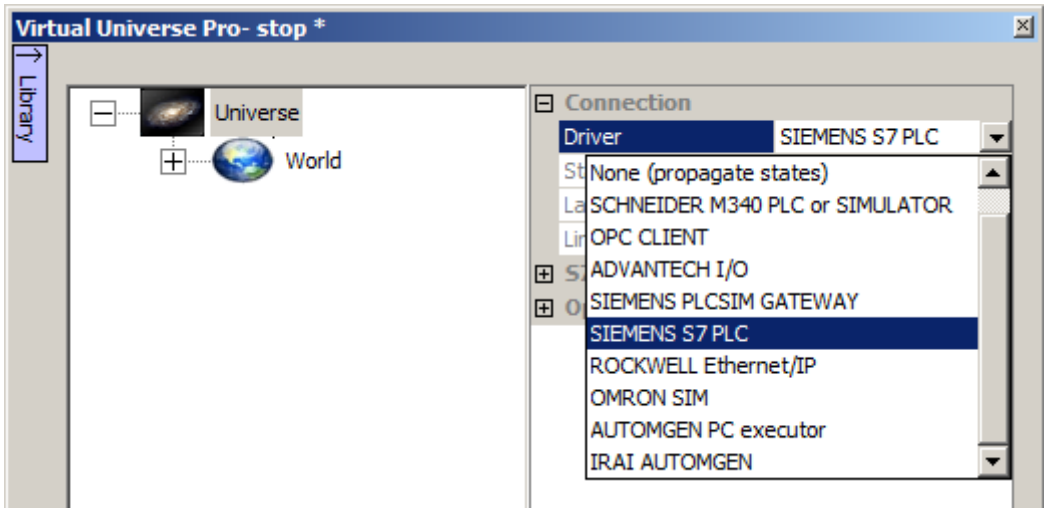
means: global variable called "myvariable" of the application called "Application".

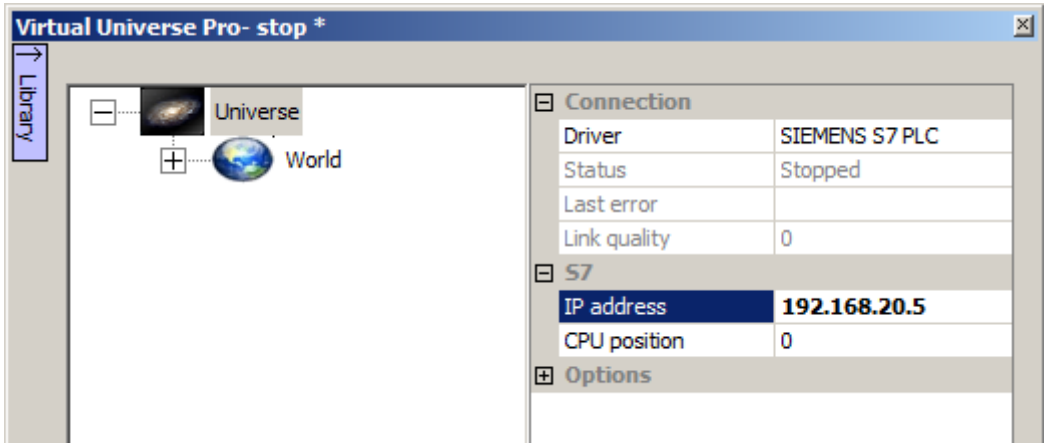
MyApplication.MyPOU.anothervariable

means: variable "anothervariable" of the module "MyPOU" of the application "MyApplication".

Connection to a Siemens PLC S7-1200, S7-300 or S7-400

The PC and the PLC must be connected to a network.

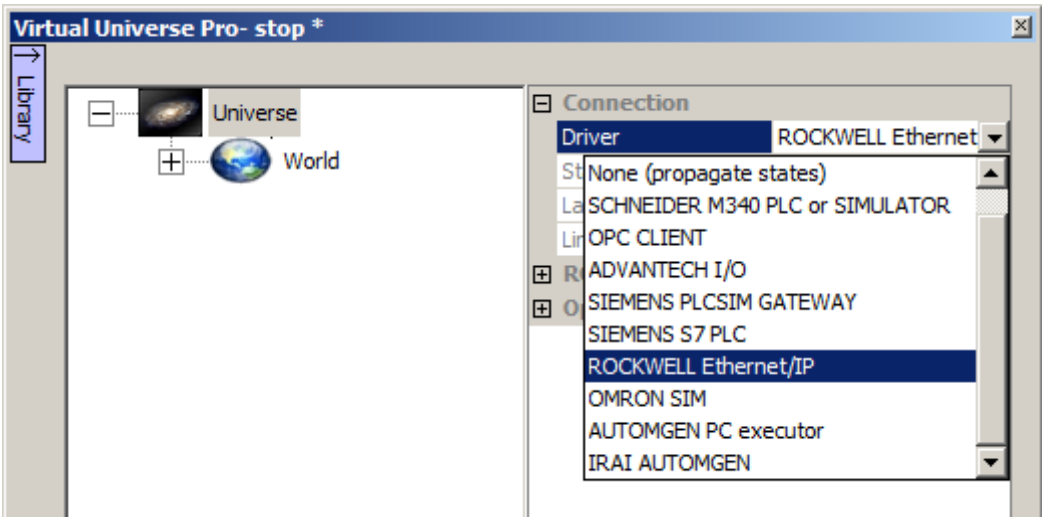


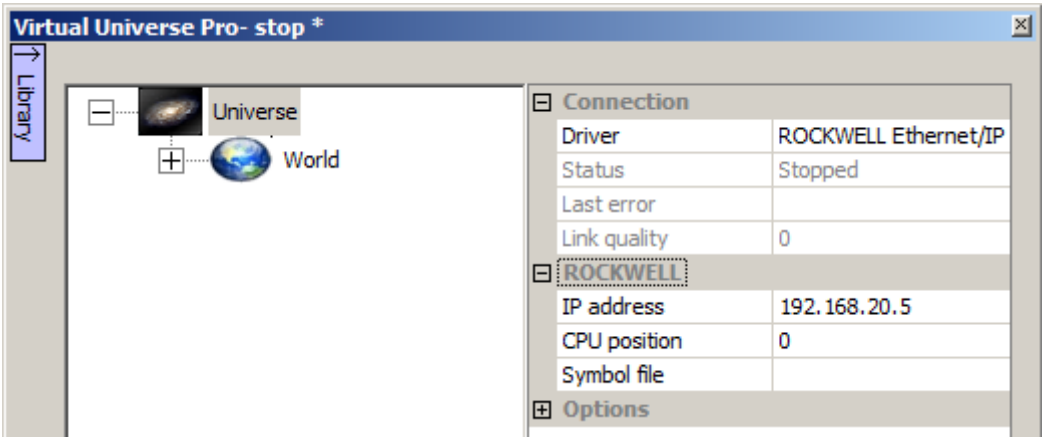


The IP address of the PLC must be filled in the S7/IP address item. For the S7-400 PLC, the PLC CPU position in rack must also be filled in the S7/CPU position item.

The usable variables for link names are inputs, outputs and internal variables. Bit, 16 bits words, 32 bits words and floating types can be used. Examples: %I0.0, %MW10, %QW0, %MF15

Connection to a Rockwell Compact Logix PLC, Control Logix PLC or SoftLogix emulator

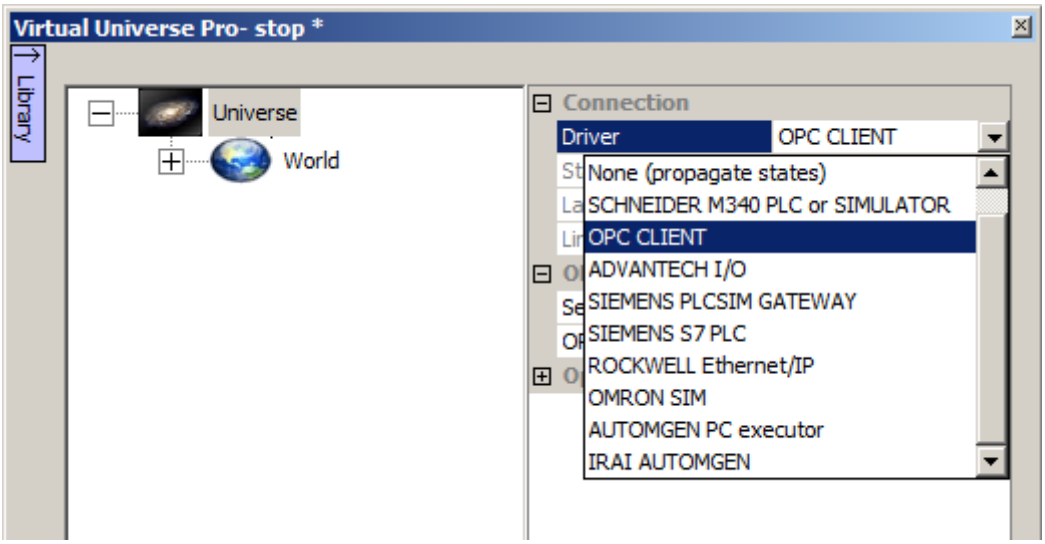


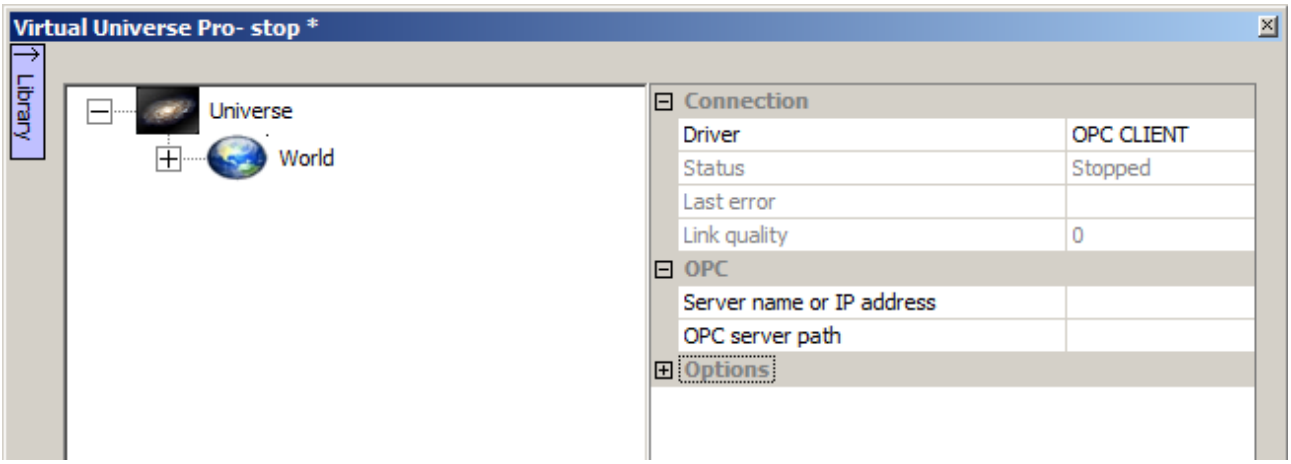


The PLC or PC (SoftLogix) IP address and CPU position must be filled in.

The usable variables for the links are Tags used in the Rockwell program running in the PLC or the Emulator. For a Tag belonging to a program, the following syntax must be used: PROGRAM:<program name>:<tag>, example: PROGRAM:MYPROGRAM:MYTAG

Connection to an OPC server



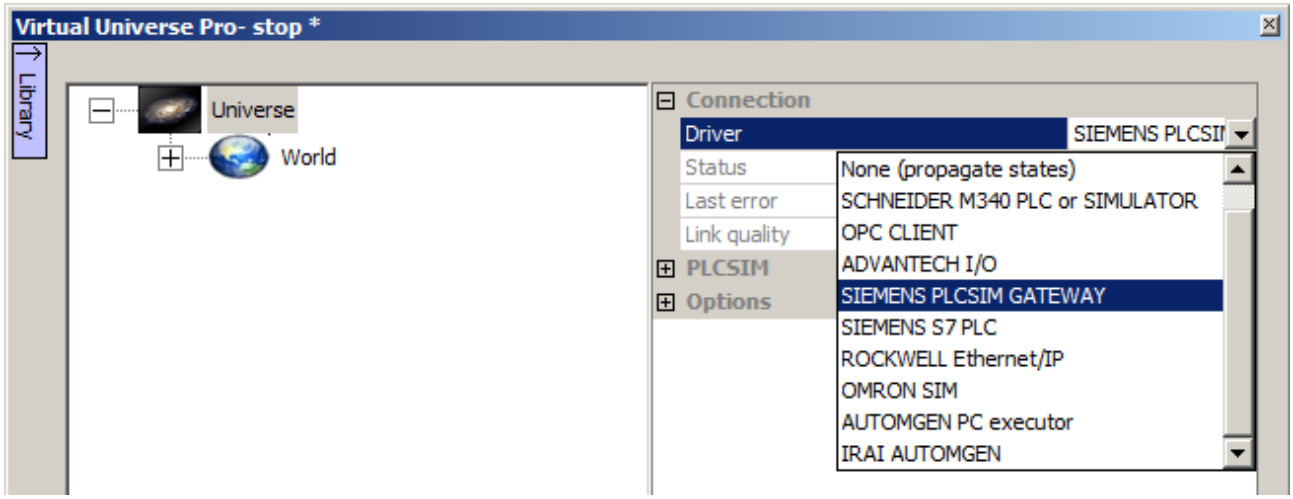


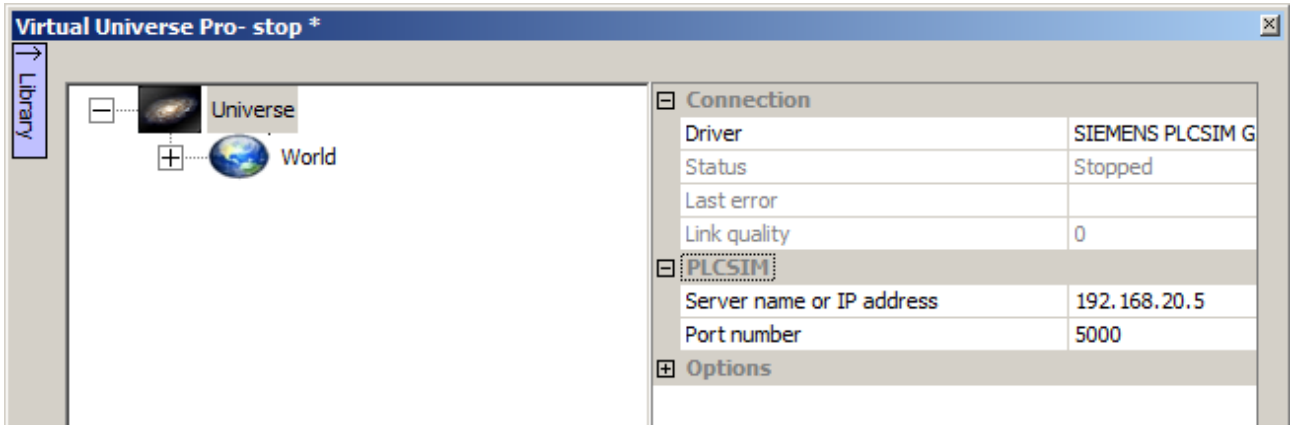
OPC server name must be filled in. If this server is running on the same PC then the OPC server path item must be left blank. For a distant server, the network name or IP address of the server rmust be filled in.

Usable variable names for links are the OPC variables names.

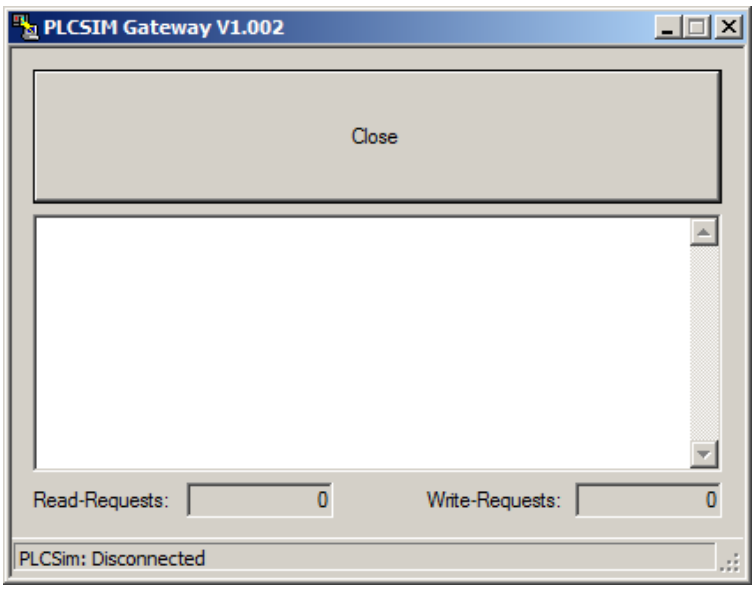
Connection to Siemens PLC-SIM

case of PLCSIM 5.4 SP<5





The "PLCSimGateway.exe" program which is present in the Virtual Universe Pro installation directory must be launched. The "Server name or IP address" must be filled in with the network name or IP address of the PLC where PLCSimGateway.exe is running (use "localhost" as server name if PLCSimGateway.exe runs on the same PC).



Case of PLCSIM 5.4 SP>=5

In this case, PLC Sim emulator is considered as a Siemens PLC.

The NetToPlcSim.exe program which is on the Virtual Universe Pro installation directory must be launched, it creates a gateway allowing to communicate with PLCSIM in the same way than a S7300 or S7400 PLC using a network adapter.

The STEP project must be defined with a CPU using a network adapter.

The selected PG/PC interface must be PLCSIM TCP-IP.

NetToPlcSim.exe automatically sets up and activates a communication link between a network adapter managed in PLCSIM with a 192.168.0.1IP address and a 127.0.0.1 local address. Virtual Universe Pro can then be connected with the S7 PLC driver using the 127.0.0.1 IP address.

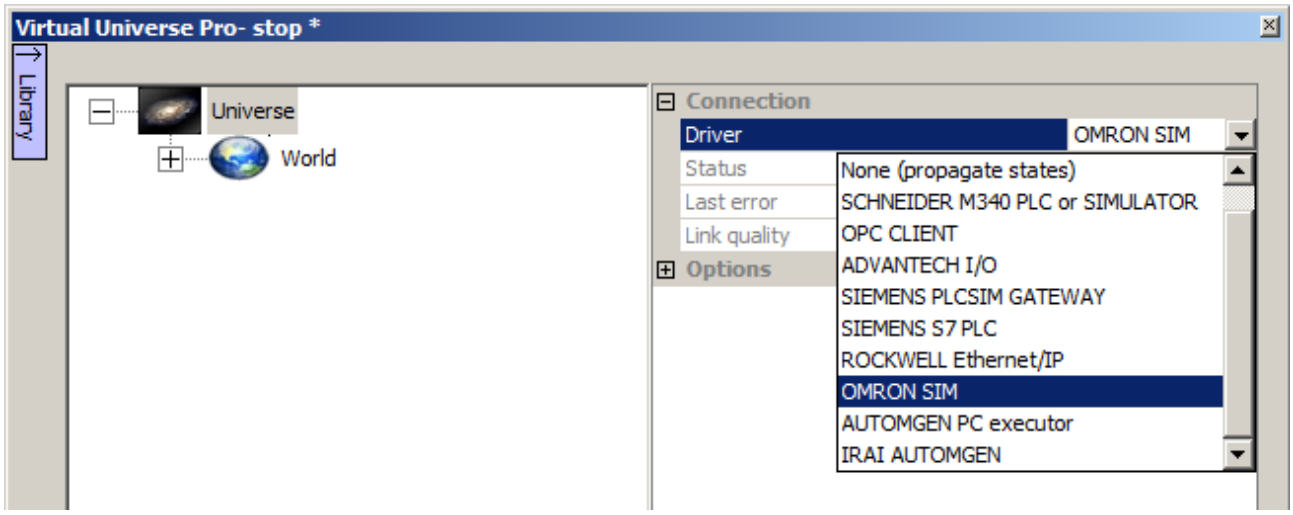
The usable variables for link names are inputs, outputs and internal variable. Bit, 16 bits words, 32 bits words and floating types can be used. Examples: %I0.0, %MW10, %QW0, %MF15

Connection to OMRON CX-Simulator

The driver for OMRON CX-Simulator emulator use a direct memory access technology for reading and writing DM variables. If other OMRON variables types must be read or written then copy to or from this variables must be written in the Omron CX-Programmer program (Mov instruction can be used for doing this). The localization of the DM variables needs that 2 DM words have to be preset with specific values at the beginning of the exchange area. These 2 values are used as signature and permits the driver to found the exchange table. The signature is only used at connection time. The 2 signature words are the words number 0 and 1 of the table, the next word is word number 2, etc.

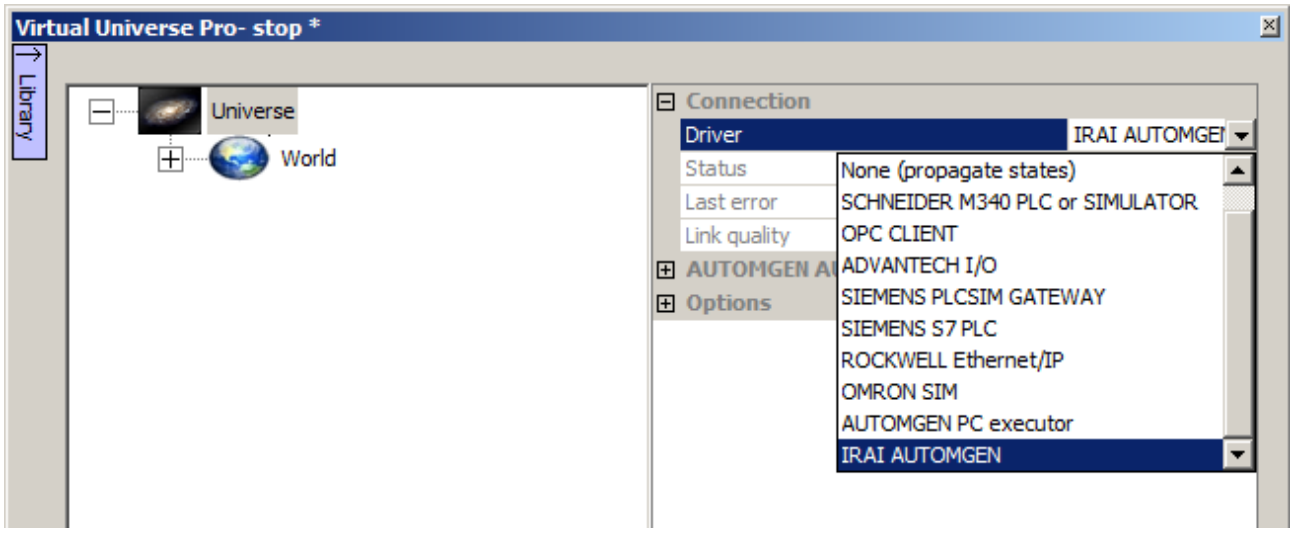
First signature word (position 0 of the table) (value hex 4952)	Second signature word (position 1 if the table) (value hex 4149)	Third word (position 2 of the table)			

Depending on the Windows version, this technology may needs Virtual Universe Pro to be run in administrator mode. For launching Virtual Universe Pro in administrator mode, right click on the Virtual Universe Pro launching shortcut and select "Run as administrator".

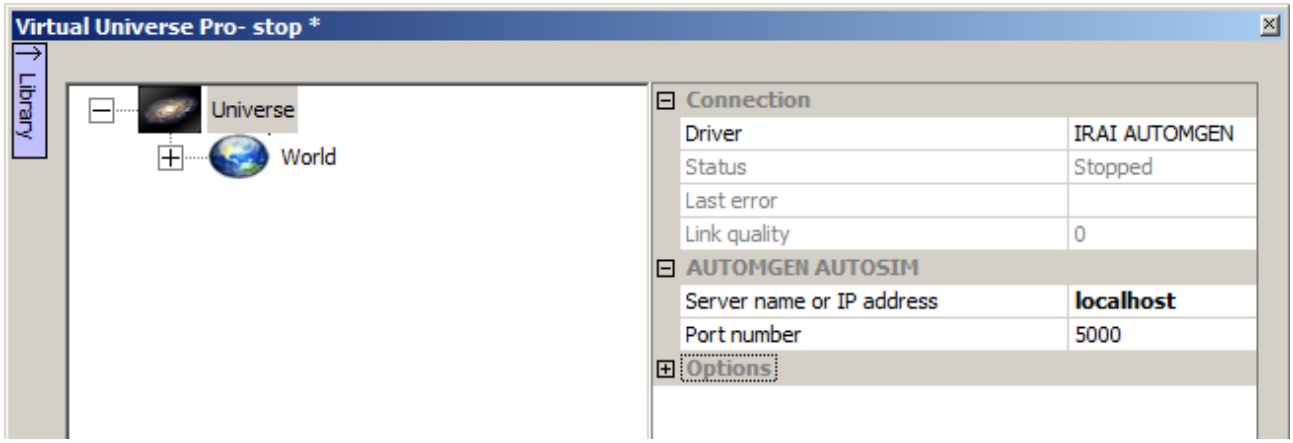


The configuration of the links is made by selecting the number of the DM word from the beginning of the exchange table and the access type. The access type specifies if the read or write will be processed to a bit of the word, to the whole word (16 bits integer) or to a group of 2, 4 or 8 consecutive DM words used as 32 bits integer or 32 or 64 bits float (in this case the number is the number of the first DM word of the group).

Connection to AUTOMGEN



The Execution/Connection TCP-IP/Server of the AUTOMGEN project properties must be checked.

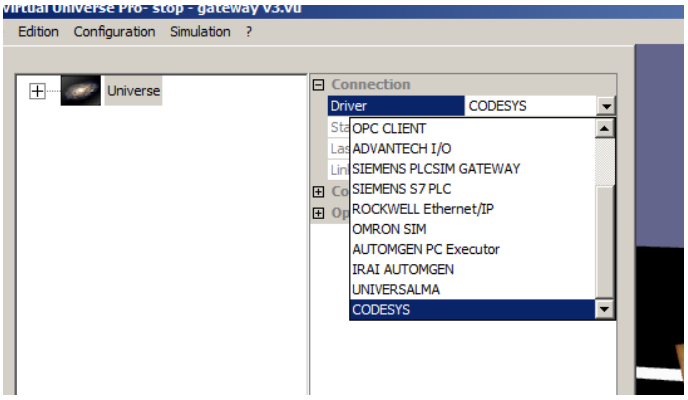


If AUTOMGEN is running on a distant PC, the network name or the IP address must be filled in the "Server name or OP address" item. If AUTOMGEN is running on the same PC then "localhost" must be used.

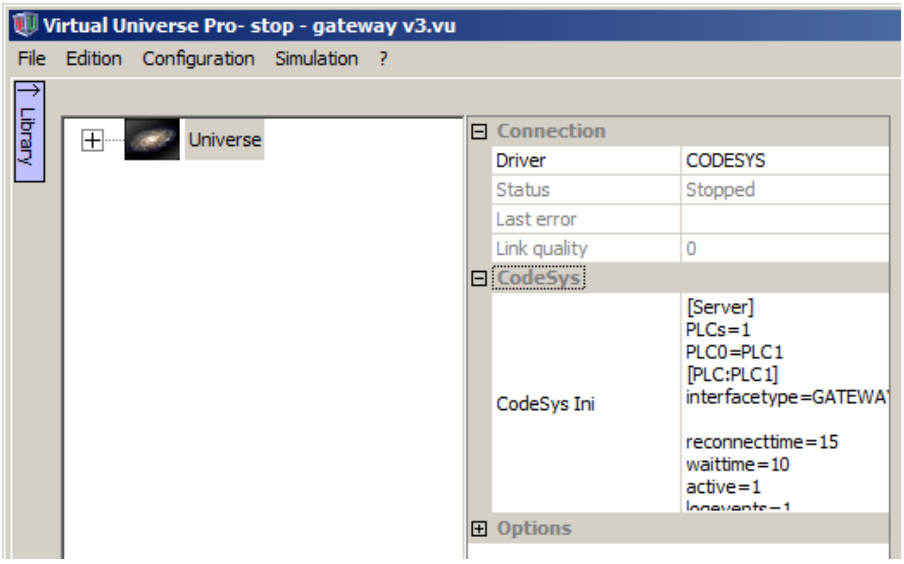
The usable variable names for the links are the whole AUTOMGEN variable names.

The "AUTOMGEN PC executor" driver permits a faster connection with the PC executor of AUTOMGEN.

Connection to CoDeSys

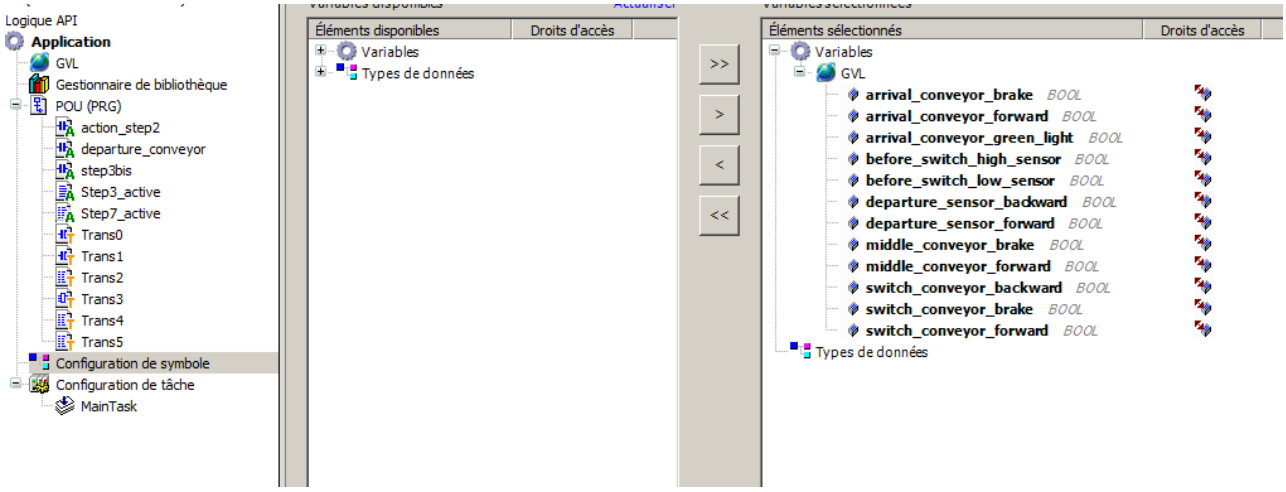


This connection permits an access to the CoDeSys compatible targets. The "CodeSys Ini" item have to be filled with the CoDeSys connection setup as defined in the INI CodeSys files.



An example of connection with Gateway 2 and Gateway 3 and associated CoDeSys projects can be found in the Virtual Universe Pro examples directory.

The variables of the CoDeSys project which must be read or written by Virtual Universe Pro have to be added in a Symbol Configuration item:



The syntax to use for referencing CoDeSys variables in Virtual Universe Pro is as following:

<application name>.<POU name or GVL for global variables>.<variable name>

Example:

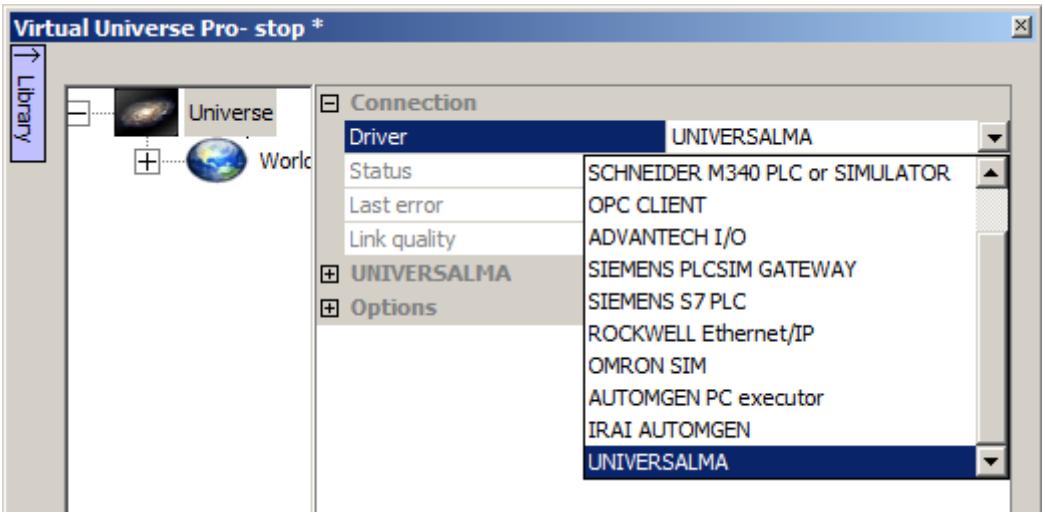
Application.GVL.myvariable

means: global variable called "myvariable" of the application called "Application".

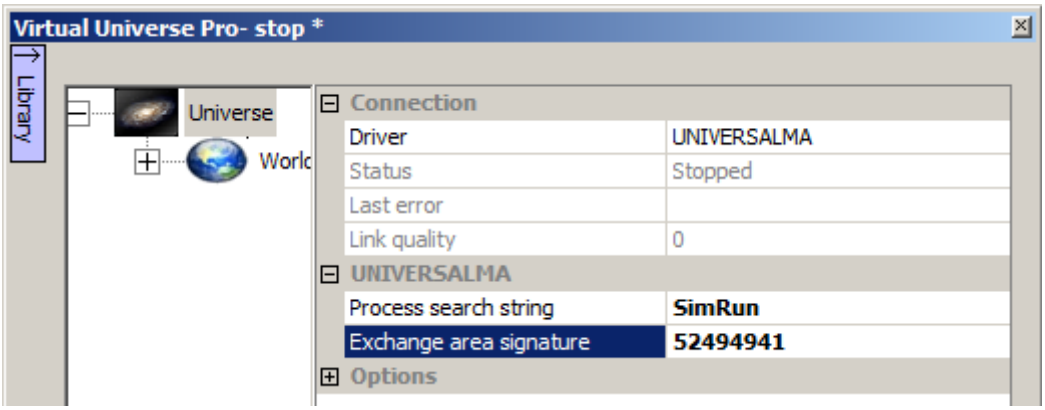
MyApplication.MyPOU.anothervariable

means: variable "anothervariable" of the module "MyPOU" of the application "MyApplication".

Universal connection



This connection permits an universal access to the variables stat of a PLC emulator running on the same PC. The main concept is to identify the PLC emulator by the name of its process and to localize an exchange area with a signature.



The "Process search string" parameter allow to identify a process. The first process found with an exe file name associated containing the search string will be used. Warning, this string is case sensitive. The Windows Task manager tool can be used to list the running processes.

The "Exchange area signature" parameter allows to localize in the process memory the beginning of an exchange area which will be used to read and write variables stats to and from Virtual Universe Pro.

This parameter is defined as two hex digits per character. By example, 52494941 will define a 4 characters string "RIIA".

In the PLC emulator, the signature must be written at the beginning of the required exchange area zone. The easiest way is usually to do this by adding some code to the project which is running inside the PLC emulator.

Remarks :

- it is recommended to use a signature with at least 4 characters to avoid a false localization,

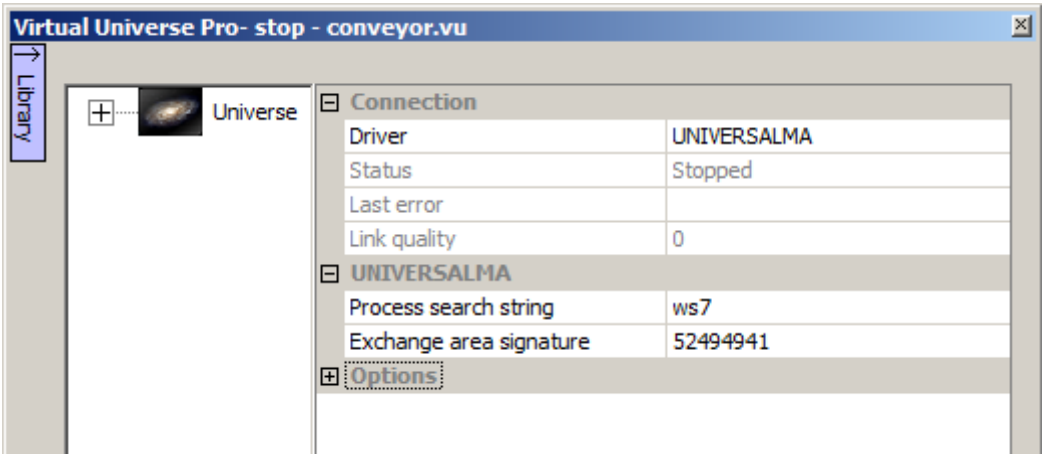
- it is recommended to make tests to obtain a successful work.

The links defined in Virtual Universe Pro use a position from the beginning of the exchange area and an access type. The position is arbitrary specified in number of 16 bits words. The access type specifies if the read or write will be processed to a bit of the word, to the whole word (16 bits integer) or to a group of 2, 4 or 8 consecutive words used as 32 bits integer or 32 or 64 bits float (in this case the number is the number of the first word of the group).

This technology permits to read and write words in the exchange area. If other PLC emulator specific variable types have to be written or read, then copy instructions must be added in the program (see example below).

Example: using universal connection with PLC emulator of the WinSPS-S7 software (MHJ-Software)

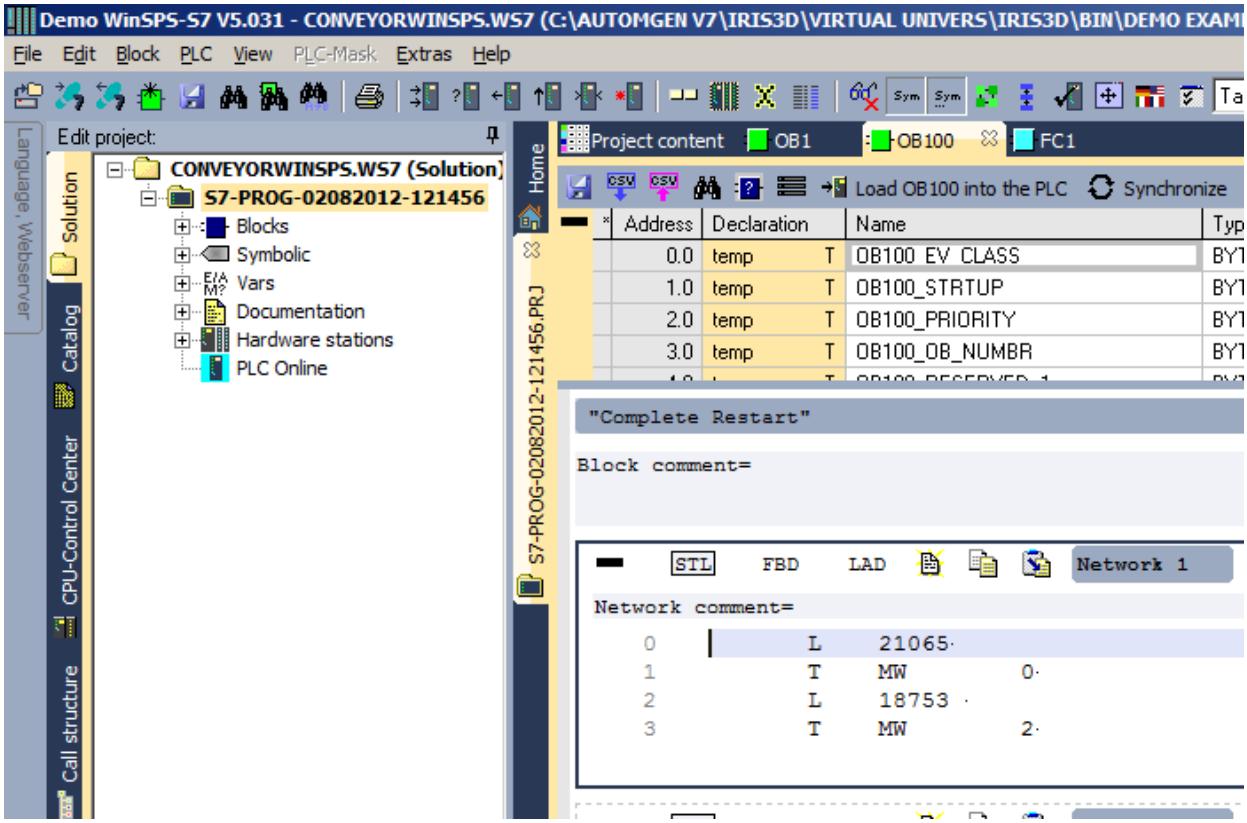
1- setting up connection in Virtual Universe Pro



The exe file name of the process which is the PLC emulator for the WinSPS-S7 V5 software is "w7sv5.exe". The "ws7" search string will identify this process.

Arbitrary, the signature is defined to 52494941: RIIA string.

2- defining the signature at the beginning of the exchange area in the emulator



The exchange area is arbitrary localized from MW0 STEP7 variable.

The following program lines

L 21065

T MW0

L 18753

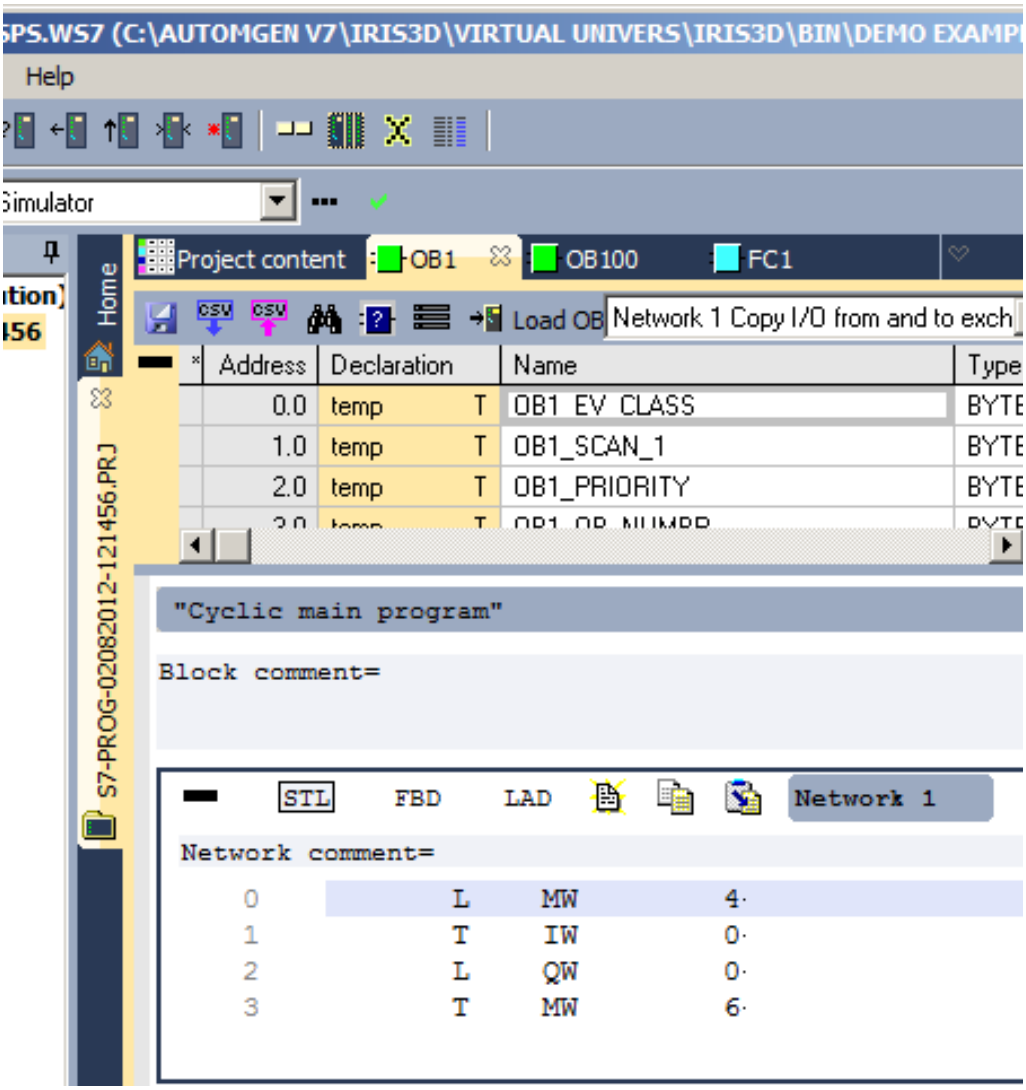
T MW2

write the 4 characters RIIA in the STEP7 words MW0 and MW2. In the following case (STEP7 language), these 2 words are mapped to 4 consecutive bytes in memory. This initialization is made only one time at program startup (use of OB100 STEP7 block).

3- copy from and to the exchange area

Our sample use 5 digital inputs and 15 digital outputs. Arbitrary, inputs I0.0 to I0.4 and outputs Q0.0 to Q0.14 are used.

The code for copying variables is as following:



For convenience and for this example, the variables are copied by group of 16. The MW4 STEP7 word matches the inputs, MW6 STEP7 word matches the outputs.

Summary of the use of the PLC emulator memory:

Byte 0 "R" 52	Byte 1 "I" 49	Byte 2 "I" 49	Byte 3 "A" 41	Byte 4	Byte 5	Byte 6	Byte 7
Word STEP7 MW0		Word STEP7 MW2		Word STEP7 MW4		Word STEP7 MW6	
Echange word #0		Exhange word #1		Exhange word #2		Exhange word #3	
				Inputs STEP7 IW0		Outputs STEP7 QW0	

3- Virtual Universe Pro links

Name	Alias	External variable
Conveyor 1 Rotates rollers sens 1		WORD3.0
Conveyor 1 Rotates rollers sens 2		WORD3.1
Conveyor 1 Rollers brake		WORD3.2
Bottom sensor		WORD2.0
switch on/off red light		WORD3.3
switch on/off orange light		WORD3.4
switch on/off green light		WORD3.5
Top sensor		WORD2.1
Conveyor Switch Rotates rollers sens 1		WORD3.6
Conveyor Switch Rotates rollers sens 2		WORD3.7
Conveyor Switch Rollers brake		WORD3.8
Backward sensor		WORD2.2
Forward sensor		WORD2.3
Conveyor 2 Rotates rollers sens 1		WORD3.9
Conveyor 2 Rotates rollers sens 2		WORD3.10
Conveyor 2 Rollers brake		WORD3.11
Conveyor 2 sensor		WORD2.4
switch on/off red light		WORD3.13
switch on/off orange light		WORD3.12
switch on/off green light		WORD3.14