



OPENSOURCE3D

0.1 ALPHA VERSION

This documentation is not maintained anymore. You can access to the new HTML documentation at :

<http://www.openspace3d.com/documentation/en>



Tutorial guide

I-MAGINER

8, rue Monteil

44000 NANTES

France

Tél. : + 33 (0)2 40 12 16 44

E-mail : contact@openspace3d.com

Website : <http://www.openspace3d.com>



Summary

| | |
|--|-----------|
| Foreword | 1 |
| SCOL technology presentation | 2 |
| SCOL's background | 2 |
| Behind simplicity, power..... | 2 |
| Multiuser Langage..... | 2 |
| Many opportunities for enrichment | 2 |
| The SO3Engine a 3D engine based on Ogre | 3 |
| Technical Features..... | 3 |
| OpenSpace3D presentation | 4 |
| 3D applications development platform..... | 4 |
| Fast and intuitive development..... | 4 |
| A simple concept, a tool accessible to all..... | 4 |
| Starting with OpenSpace3D..... | 5 |
| Pre-requisites | 5 |
| Organization of files in the SCOL partition | 5 |
| Installation of the scol Voy@ger..... | 6 |
| Installation of OpenSpace3D Editor | 10 |
| Launch..... | 13 |
| 3D Configurator | 14 |
| General ergonomics of the Platform | 16 |
| General Menu..... | 17 |
| Scene tree area and information on Resources | 17 |
| Open an OpenSpace3D scene (.xos)..... | 18 |
| Import an Ogre Max scene (.scene)..... | 20 |
| Save an OpenSpace3D scene..... | 24 |
| Exporting OpenSpace3D applications | 24 |
| As A Stand Alone Application | 25 |
| As a Web Page | 27 |
| Managing the Scene tree and informations on Resources | 29 |
| Scene Tree | 29 |
| Add a mesh : Wizard..... | 50 |
| Color Map..... | 53 |
| 3D Editing..... | 54 |
| Interface..... | 54 |
| Moving inside 3D | 56 |
| Selecting an object..... | 57 |
| Moving an object..... | 58 |
| Duplicate an Object | 59 |
| Play/Pause Mode | 61 |



| | |
|--|------------|
| Editing Functions | 62 |
| Interface..... | 62 |
| Links..... | 63 |
| PlugIT Documentation | 67 |
| « Navigation » plugIT | 72 |
| PlugIT : “Walkthrough“ | 72 |
| PlugIT : “Examine View “ | 73 |
| « Network » plugIT | 75 |
| PlugIT: “chat” | 75 |
| PlugIT: “connect”..... | 76 |
| PlugIT: “message” | 77 |
| PlugIT:” room item” | 78 |
| « Input » plugIT | 79 |
| PlugIT: “Keyboard” | 79 |
| PlugIT Joypad | 81 |
| PlugIT : “Wii tools” | 82 |
| PlugIT : ”Mouse” | 83 |
| PlugIT NeuroSky : | 84 |
| PlugIT Serial | 86 |
| « Interface » plugIT | 87 |
| PlugIT : « flash Interface »..... | 87 |
| « Material » plugIT | 89 |
| PlugIT : « Material color »..... | 89 |
| PlugIT change Material | 90 |
| PlugIT : « change texture » | 91 |
| PlugIT:” texture text “ | 92 |
| « Maths » plugITs | 93 |
| PlugIT Get vector..... | 93 |
| PlugIT Set Vector..... | 94 |
| PlugIT: “vector average” | 95 |
| PlugIT vector multiply | 96 |
| PlugIT : “vector permutation” | 97 |
| PlugIT operator | 98 |
| « Media » plugIT | 99 |
| PlugIT: “Sound” | 99 |
| PlugIT: “Flash” | 100 |
| PlugIT Speech | 102 |
| PlugIT Speech Recognition..... | 105 |
| All events created in the plugIT Speech recognition is then available at the output of the PlugIT | 106 |
| PlugIT : « Open Url »..... | 108 |
| PlugIT: “On init” | 109 |

| | |
|--|------------|
| PlugIT: “Close” | 110 |
| PlugIT: “Timer” | 111 |
| PlugIT: “Sequence” | 112 |
| PlugIT: “Var” | 113 |
| PlugIT: “Switch” | 114 |
| PlugIT ActiveXmessage | 115 |
| PlugIT « Switchcase » | 118 |
| PlugIT: “If” | 120 |
| PlugIT:”Dialog box “ | 121 |
| PlugIT: “Counter” | 122 |
| PlugIT: “Call Url “ | 123 |
| plugIT input Dispatcher | 124 |
| PlugIT output Dispatcher | 125 |
| PlugIT Random Output : | 126 |
| « Object» plugIT | 127 |
| PlugIT : « Get Object » | 127 |
| PlugIT : “Set Active camera” | 128 |
| PlugIT : “Animation” | 129 |
| PlugIT : Animation Transition | 131 |
| PlugIT : “Distance” | 132 |
| PlugIT: “Rotate” | 133 |
| PlugIT: “Light” | 135 |
| PlugIT: “Hide” | 136 |
| PlugIT: “Object link “ | 138 |
| PlugIT Get Object | 139 |
| PlugIT: “Get Camera “ | 140 |
| PlugIT: “objectFollow” | 141 |
| PlugIT: “objectGoto” | 142 |
| PlugIT : ”Object Control” | 143 |
| PlugIT: “target” | 144 |
| « Physics» plugITs | 145 |
| PlugIT: “Physic Contact” | 145 |
| PlugIT : « PhysicImpulse » | 146 |
| PlugIT: « physicTools » | 147 |
| PlugIT “Rendering” | 148 |
| PlugIT: « Stereo » | 148 |
| PlugIT: « fullscreen » | 149 |
| PlugIT Viewport | 150 |
| PlugIT Compositor | 151 |
| Debug et Log | 152 |
| Advanced use and Definitions | 154 |
| Graphic Resources (3D) | 154 |
| Group Resources | 157 |
| Group Meshes | 158 |
| Resources Directories | 159 |
| Import an Ogre resource (shaders, texture, material, mesh) | 160 |
| Group Export | 164 |

| | |
|--|------------|
| Definitions and advanced use of the parameters | 165 |
| SkyBoxes and scenes environnements | 173 |
| Physics Engine | 174 |
| Contact and Support | 178 |



Foreword

The OpenSpace3D project has born at I-maginer. The company wishes to democratize the creation of real-time 3D contents to a large public.

This project is under Open Source license that why it is easier to broadcast it in 3D production studios (graphists/developers) or in student schools.

However, OpenSpace3D could be used in big company to create projects for partners, in video game industry or for non-professional people (community).

We are happy to present you a new version with new functionalities.

This document aims to give you the possibility to use the platform with autonomy.

So, we made it to give the possibility to new users to learn it easily and progressively.

If you are a company: All questions about the use of the platform or about technical support could be post at: contact@openspace3d.com

The SCOLRing forum <http://www.scolring.org/> , is here to help you too.

Have a nice time discover it!

OpenSpace3D developer's team.



SCOL technology presentation

SCOL's background

SCOL is a programming language designed for network use, which gives websites interactivity between users, multimedia functions and 3D browsing.

Behind simplicity, power

It is an "interpreted" language, which leaves the designer of a SCOL site the opportunity to develop according to his desires and needs. However, learning the language is necessary only to Schoolmasters wanting to develop very specific applications. While it is true that SCOL is very flexible, it is also true that it is relatively complex.

Multiuser Langage

In the classical model of the Internet, users are not in direct touch. The server is divided into as many parts as there are online clients.

Scol allows several connected clients to communicate and interact; The Scol server does not split, it relays communications. Browsing is no longer individual but collective.

This feature is ideal in the field of 3D online gaming, virtual communities, e-commerce ...

Many opportunities for enrichment

Scol includes comprehensive development libraries (network, 3d engine, 2d interface, audio, video, SQL ...).



The SO3Engine a 3D engine based on Ogre

Since November 2008, I-maginer is actively working on a new 3D engine: the SO3Engine.

It is this engine that will be addressed in this document.

So rather than reinventing the wheel, we wanted to integrate a very powerful 3D engine "Ogre 3D" technology in the SCOL.

<http://www.ogre3d.org/>

Indeed, it enables application development with 3D photo-realistic and very efficient rendering. The SO3Engine features derived directly from Ogre engine features.

Technical Features

- Platform & 3D API support: DirectX / OpenGL compatibility, Hardware and Information System support
- Scenes management: Management of 3D objects (mesh, lights, cameras, skyboxes ...) to have control of the scene's graph, dynamic lighting, real time shadows.
- Materials / Shaders Support: Effects on Materials, vertex supports and fragment shaders Texture compression
- Animations: skeletal animation support, stage nodes animation
- Other features: dynamics resources, physics engine based on Newton, Exports for 3DS Max, Blender, Sketch up

OpenSpace3D presentation

3D applications development platform

With **OpenSpace3D** platform directly developed using the SCOL technology, all users can finally build full 3D interactive scenes with rich 3D rendering without entering a single line of code. This platform has many ready to use features, is highly flexible and easy to use. The realization of an OpenSpace3D application needs only to integrate different functions and define their reciprocal interactions.

Fast and intuitive development

OpenSpace3D allows you to create full applications by assembling functions without programming. Each of the available functions run a feature of the application, eg video, web link, animation ... From this library, select your functions and combine them freely to build your application by defining the relationship between functions.

A simple concept, a tool accessible to all

OpenSpace3D is a multi-uses platform with multiple levels depending on the skills of your teams.



3D computer graphist :

Integrate quickly and easily your own 3D productions by incorporating them in real-time in our platform and applying functions and interactions.



Integrator :

This mode already very complete allows to build complete application using the provided basic elements in our library.



Developer :

This level is aimed at developers who wish to go further by programming new features through the SCOL programming language because the platform is open.



Starting with OpenSpace3D

Pre-requisites

Organization of files in the SCOL partition

Reminder: Unless otherwise specified, the path of directories will be relative to the sub directory \ directory Partition_LocalUsr of my documents directory.

Example: *C:\Documents and Settings\User\Mes Documents\scol voyager\Partition_LocalUsr*

The files you use to develop an application must be placed in the directory *\Partition_LocalUsr* ->

example : *C:\Documents and Settings\User\Mes Documents\scol voyager\Partition_LocalUsr\ma_scene\monobjet.mesh*

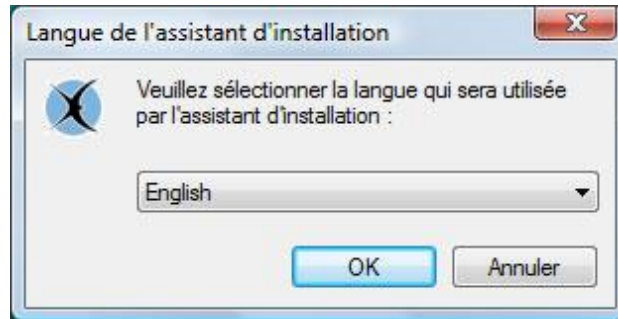
Openspace3D will only read the information **placed in this specific folder!**

It is important that the computer graphics produce resources directly from that directory otherwise it will be impossible to import their own 3D objects.

Ideally, the computer graphics exports its resources in a folder for the scene (see: "Ogre Export Documentation / SCOL").

Installation of the scol Voy@ger

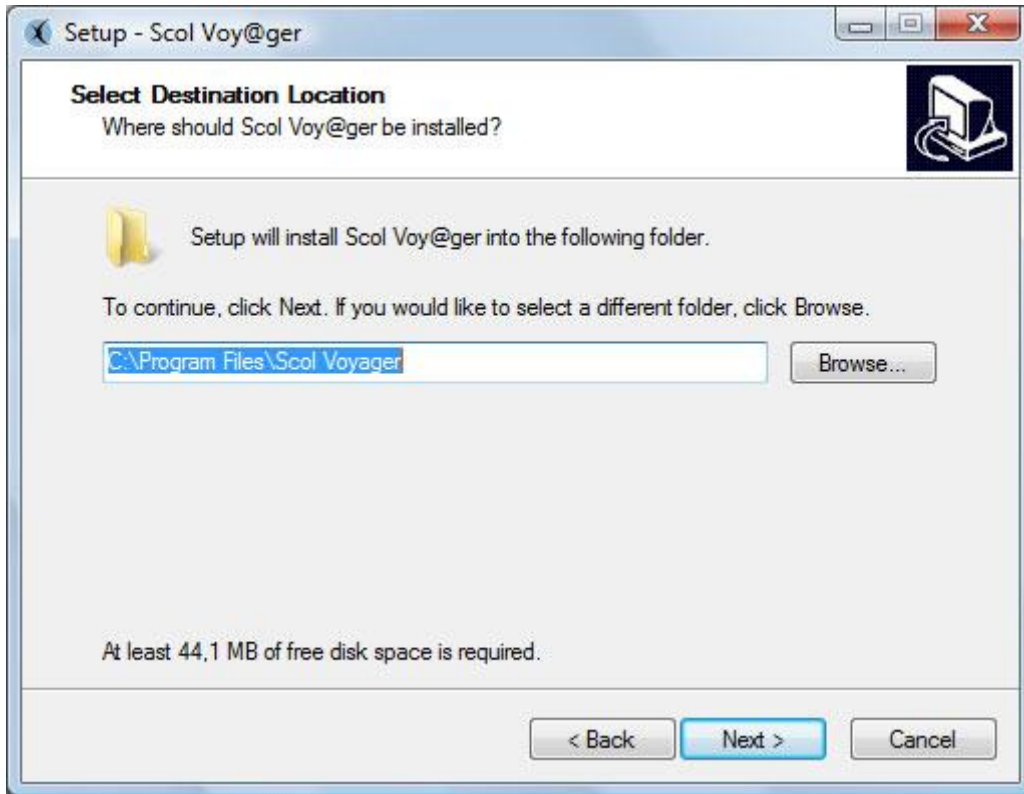
Run the file "scol_plugin.exe".



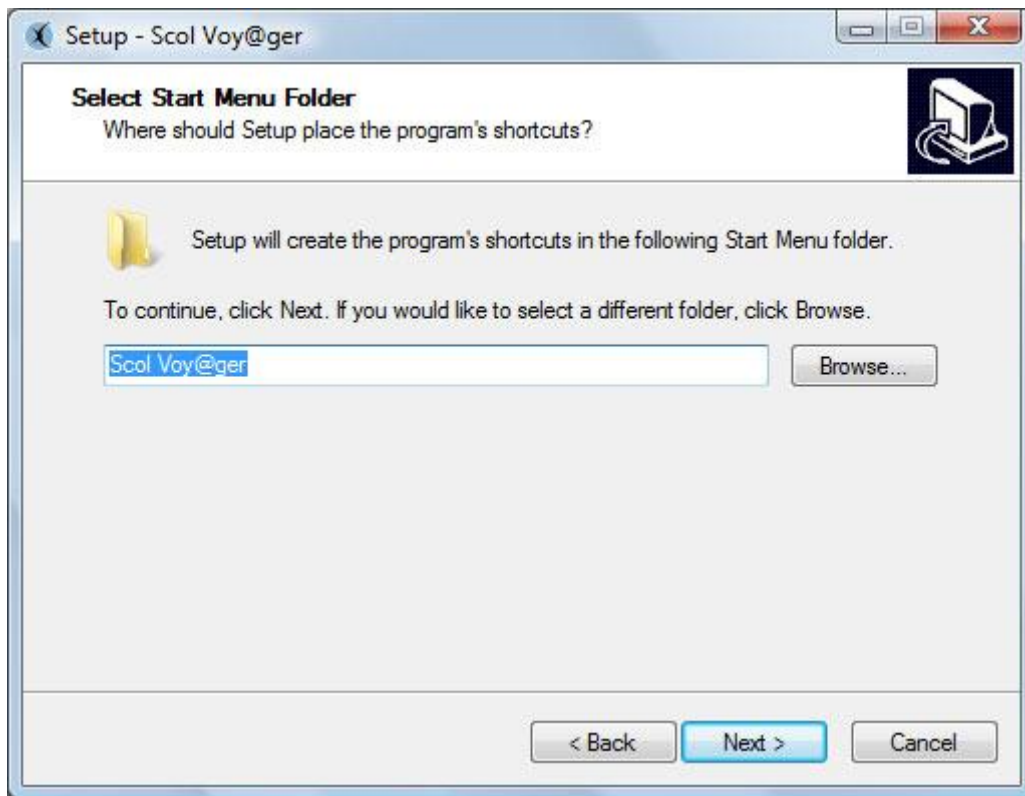
Select your language and click "Ok".



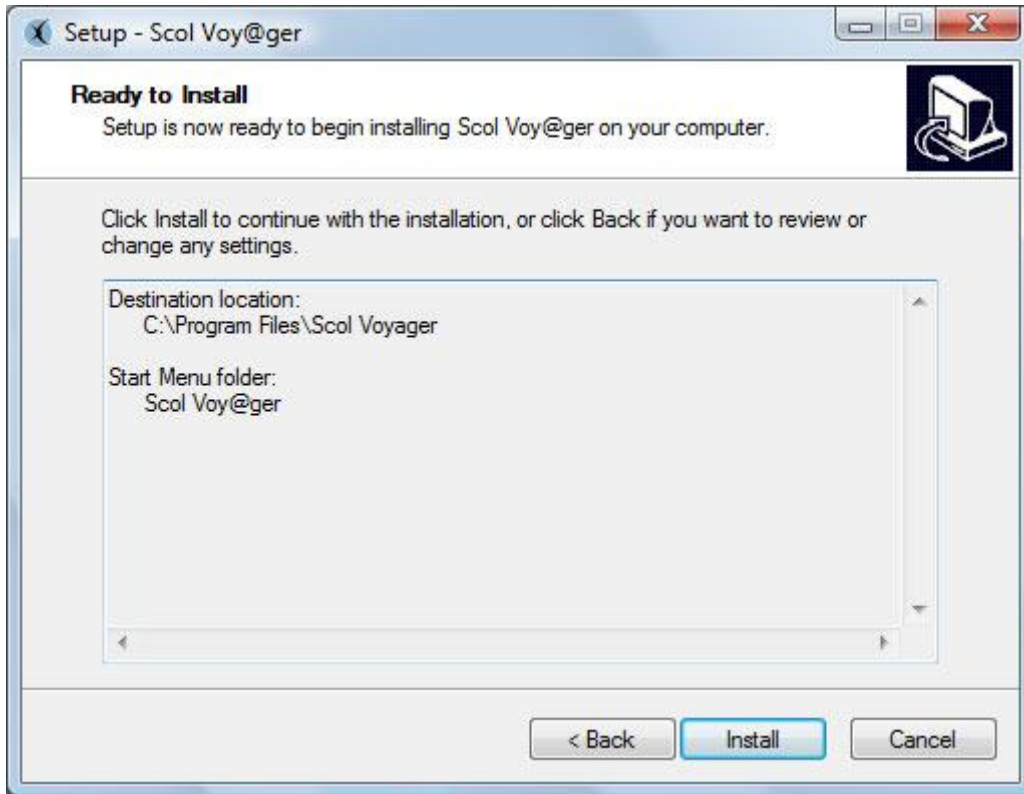
On the Welcome screen click "Next".



On the screen "Destination Folder", you can specify the installation path, by default "C: \ Program Files \ scol Voyager", then click "Next."



On the screen "Select Folder from the Start menu, you can specify the name of the program that contains shortcuts to the installation, by default" scol Voyager ", then click" Next. "



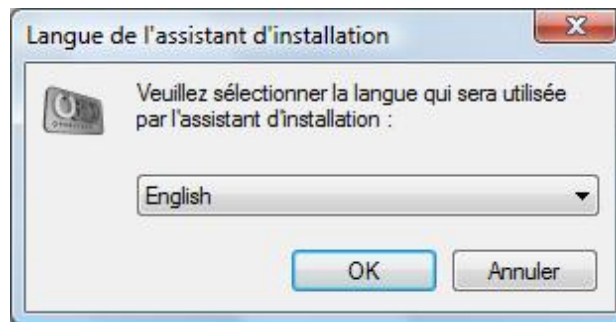
On the screen "Ready to Install", click "Install" to install the application.



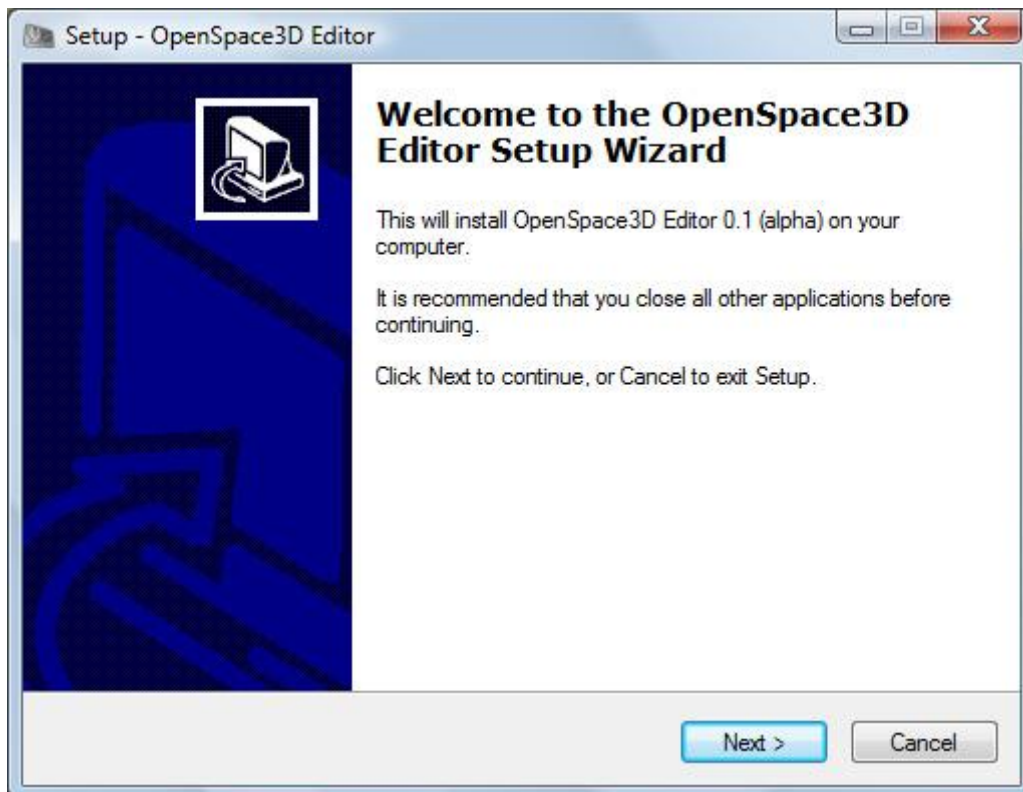
The Scol Voy@ger is now installed on your computer, click "Finish" to close.

Installation of OpenSpace3D Editor

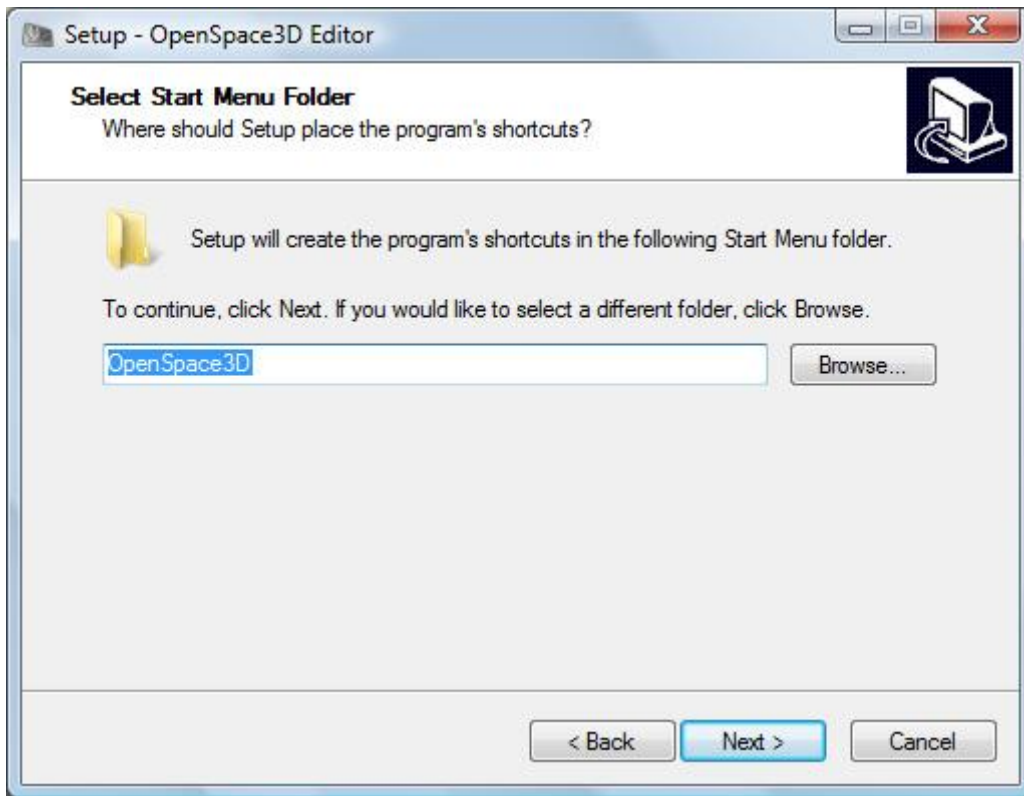
Run the file "openspace3d_editor_setup.exe".



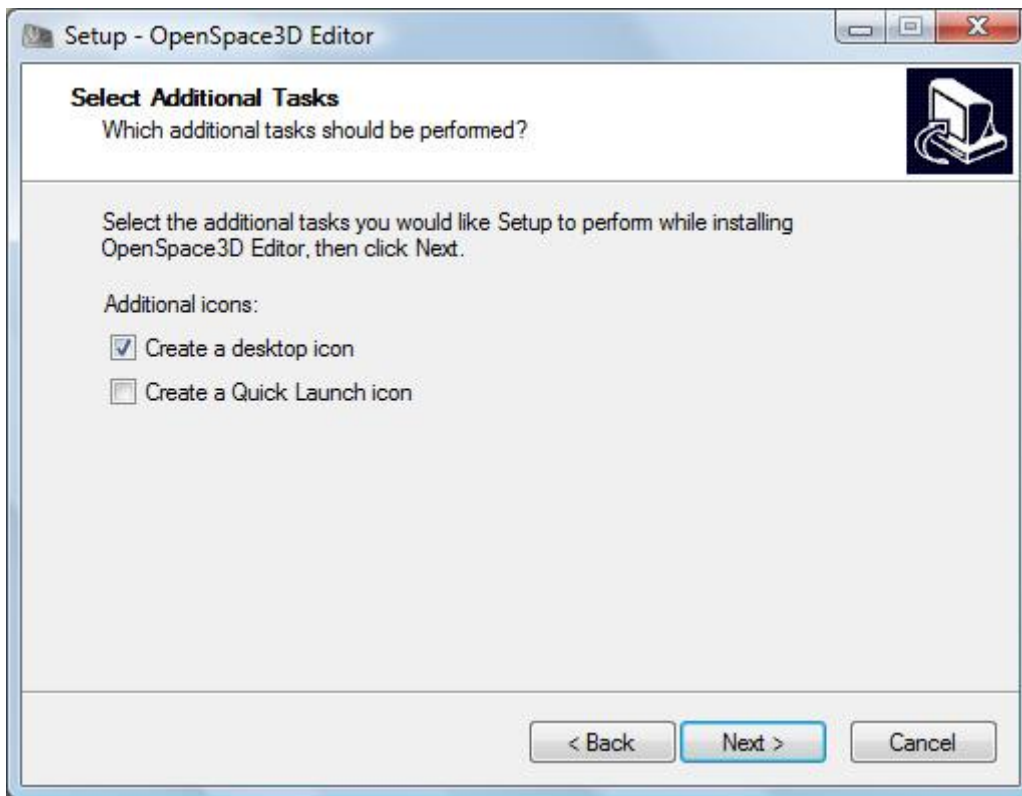
Select your language and click "Ok".



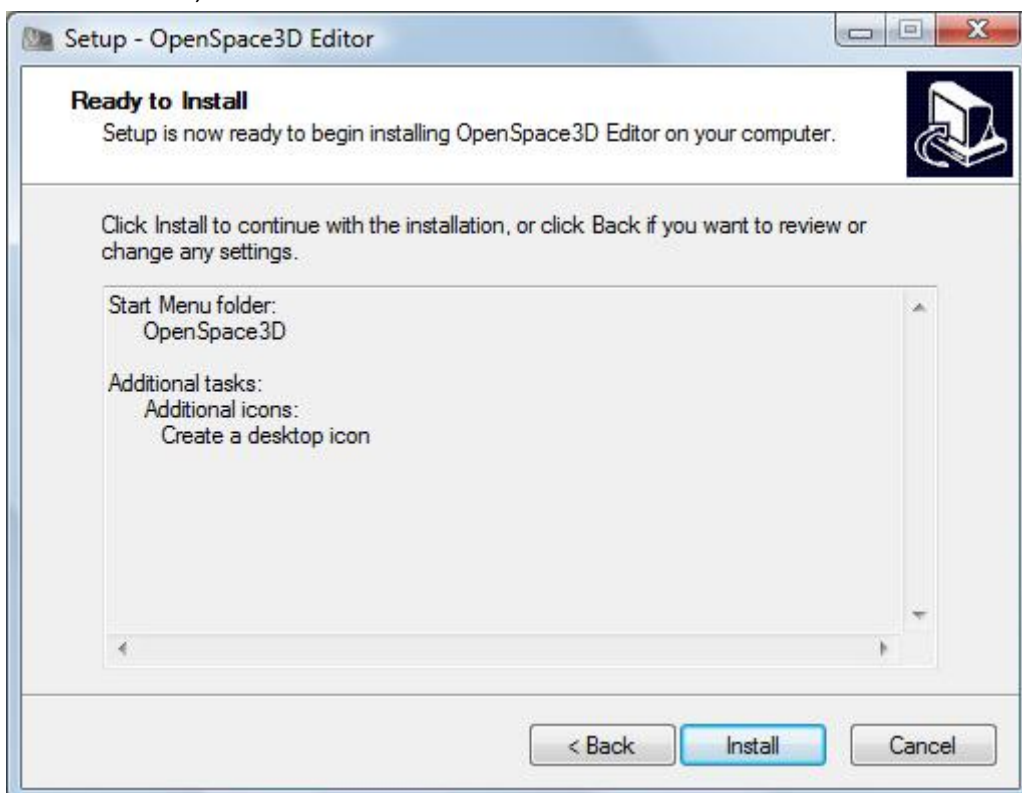
On the Welcome screen click "Next".



On the screen "Select Folder from the Start menu, you can specify the name of the program that contains shortcuts to the installation, by default" OpenSpace3D ", then click" Next. "

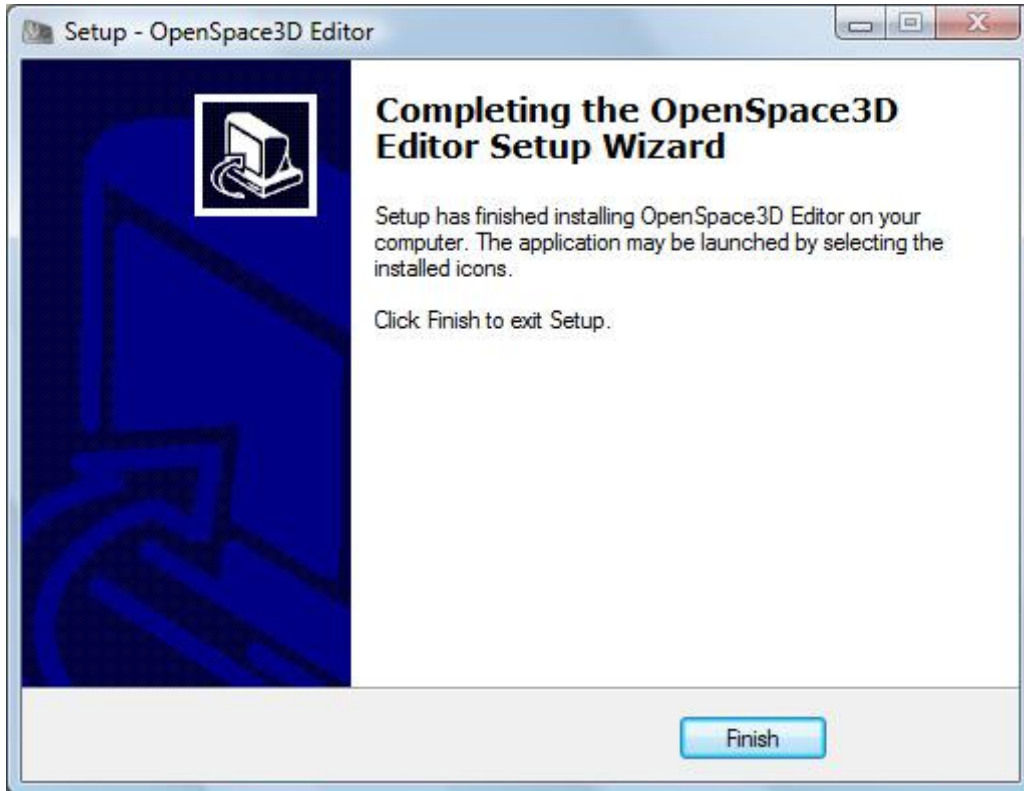


On the screen "Additional Tasks", you can enable or disable the creation of shortcuts on the desktop and the Quick Launch toolbar, then click "Next."





On the screen "Ready to Install", click "Install" to install the application.



OpenSpace3d Editor is now installed on your computer, click "Finish" to close.

Launch

To launch OpenSpace3D Editor click on the shortcut :

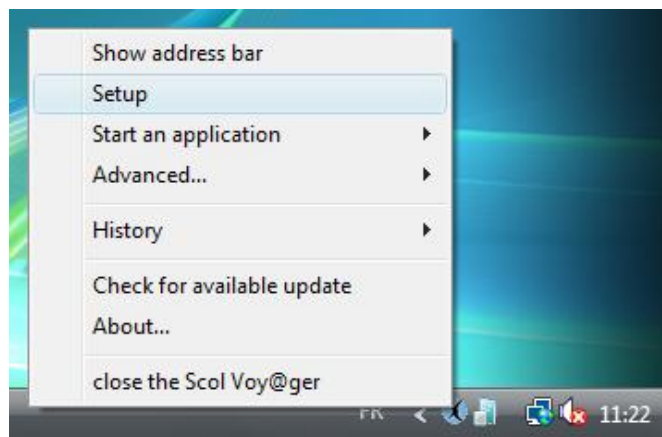


3D Configurator

Once the Scol Voy@ger installed, it is possible to configure the "SO3Engine" 3d engine parameters via Scol Voy@ger's configuration.

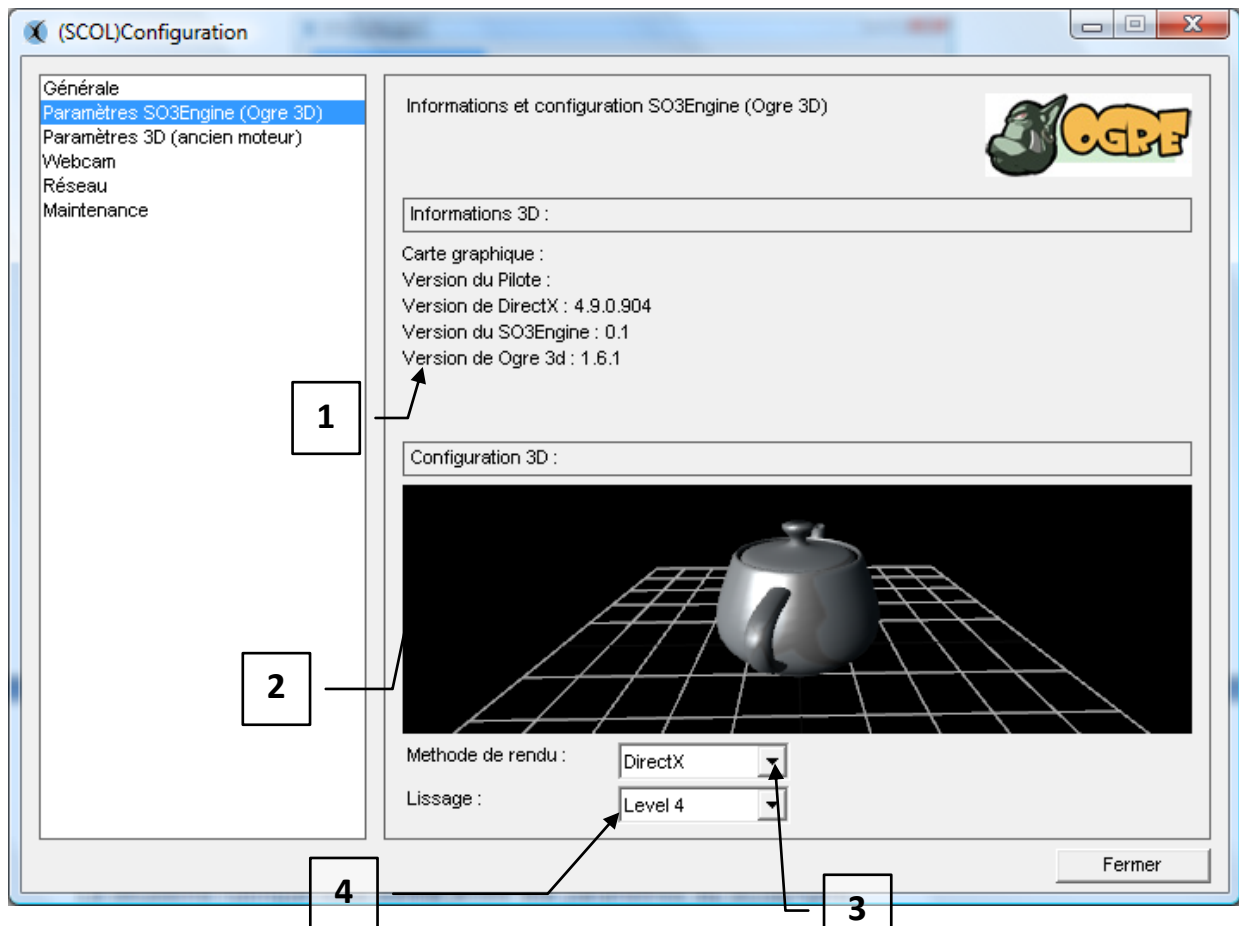


By right clicking on the scol Voy@ger's icon:



You can access the configuration menu:

The second section provides access to SO3Engine's settings.



1 / This is the section for 3D information on the hardware of the user and on the DirectX version installed and SO3Engine and Ogre 3D versions used.

2 / This is the area of 3D Test which will validate the proper functioning of the 3D

3 / Rendering Method: 2 "hardware" rendering methods can be used:

- DirectX
- OpenGL

N.B. 1: If you change these settings, the Scol Voy@ger must be restarted for the changes to take effect.

N.B 2: DirectX / OpenGL

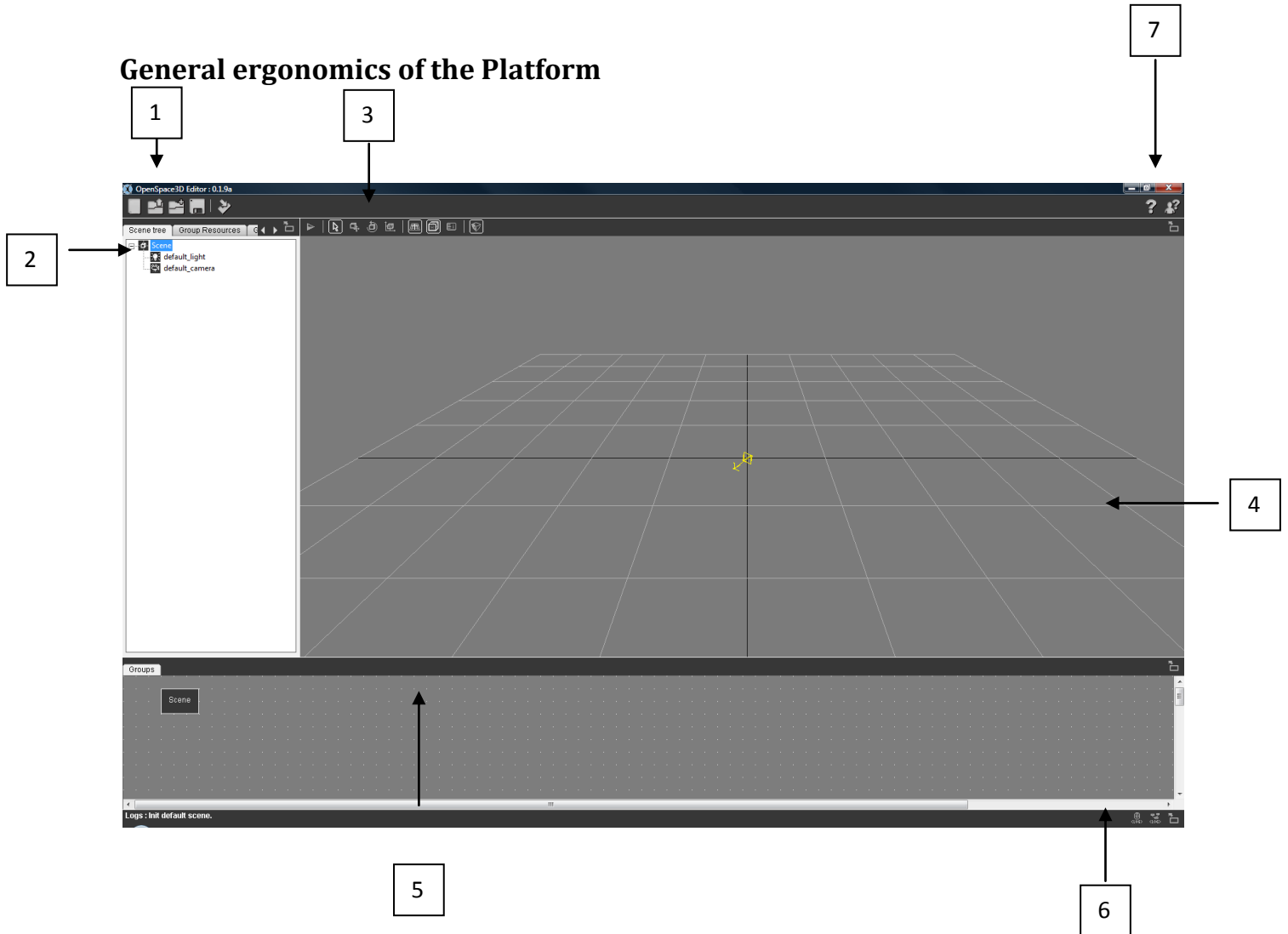
Under Windows Vista, it is strongly recommended to work in DirectX mode, to ensure the stability of 3D.

4 / Smoothing: These are the settings for Anti-Aliasing to provide a high 3D quality to avoid staircase effect on rendering.

It should be noted that the more Level (DirectX) or pass (OpenGL) number there is the better graphic quality will be.

However, if there is a heaviness in the display of 3D scenes, it may be useful to reduce the smoothing in order to optimize the 3D scenes.

General ergonomics of the Platform









| | |
|---|------------------------------------|
| 1 | Genral menu |
| 2 | Scene tree and resource management |
| 3 | 3D Edition menu |
| 4 | 3D Viewport |
| 5 | plugIT Edition |
| 6 | Log Information |
| 7 | Help and about |

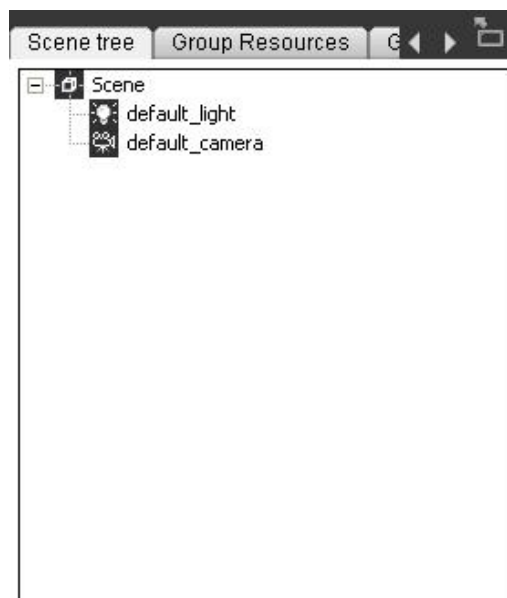


General Menu



| | |
|---|--|
|  | New: New scene, you can unload the current scene in OpenSpace3D and create a new one. |
|  | Open Scene: Open, you can browse your files to load a scene file OpenSpace3D (. XOS) in OpenSpace3D to edit it. |
|  | Import Scene: Import Ogre Max scene, you can browse your files to load a scene file (. Scene) in the current scene OpenSpace3D |
|  | Save: you can save the current scene in .XOS |
|  | Save as OpenSpace3D scene : you can save the current scene under another name nom in .xos |
|  | Export to OpenSpace3D Player : Create a file to launch the application |

Scene tree area and information on Resources

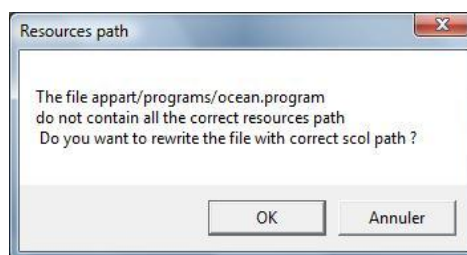
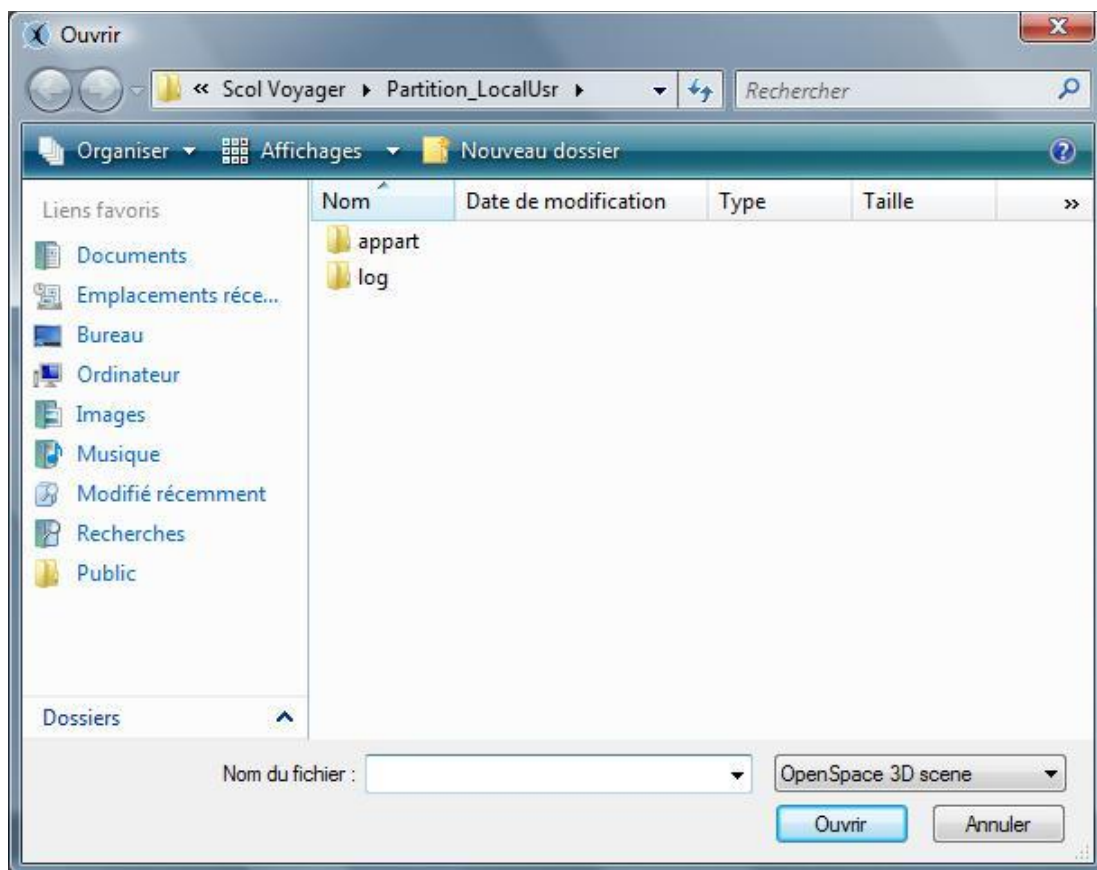


Open an OpenSpace3D scene (.xos)

In the Menu, click the icon « Open Scene »



Browse your folder \ **Partition_LocalUsr** go to select your scene file to open:

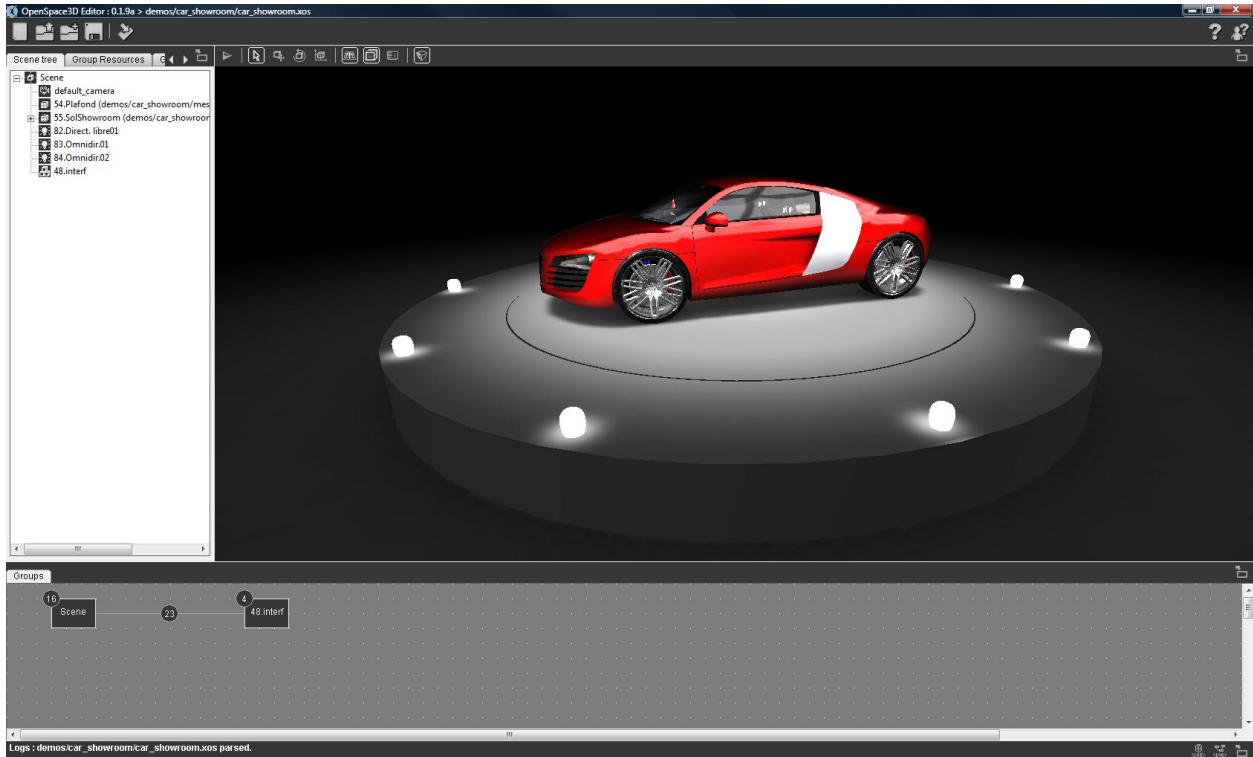




When you import Ogre graphic resources, it is possible that the resources paths in the *.material and *.program (shaders) do not correspond to the Scol path.


Accept the proposal by clicking "OK" to correct the paths from resource files.

Wait a few moments while OpenSpace3D loads your scene :

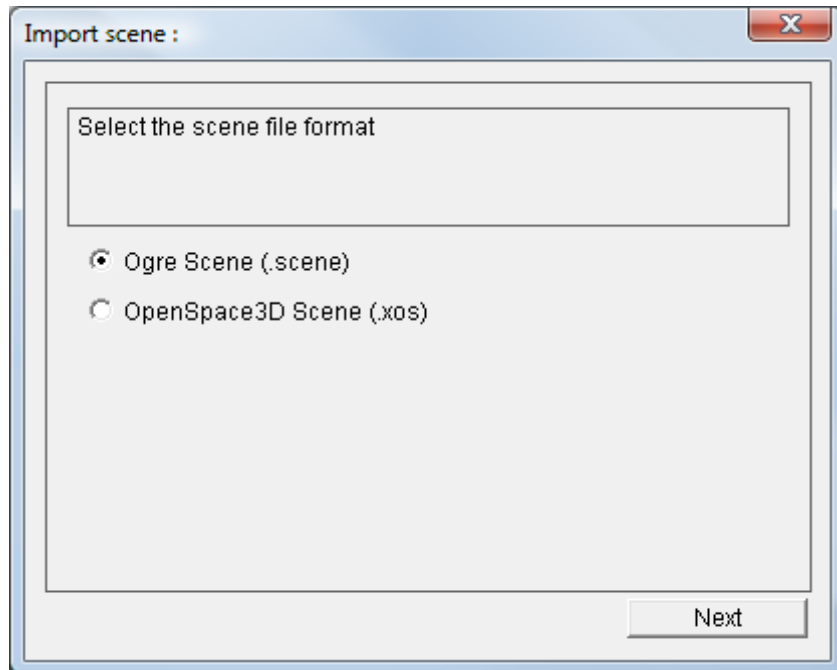


Your scene is now loaded into OpenSpace3D, the area of 3D Edition displays the scene resources and the area of information and resource management has been enriched.

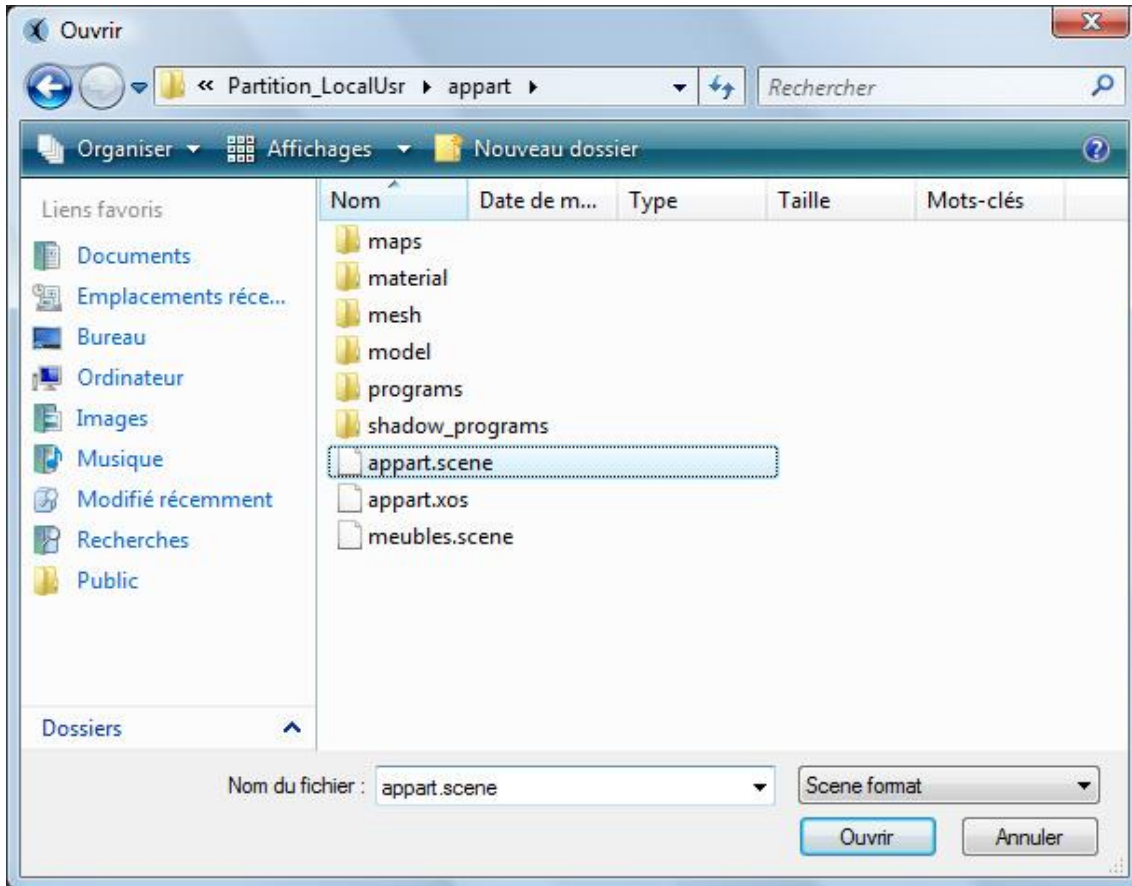
Import an Ogre Max scene (.scene)

Click on the icon « Import Scene » 

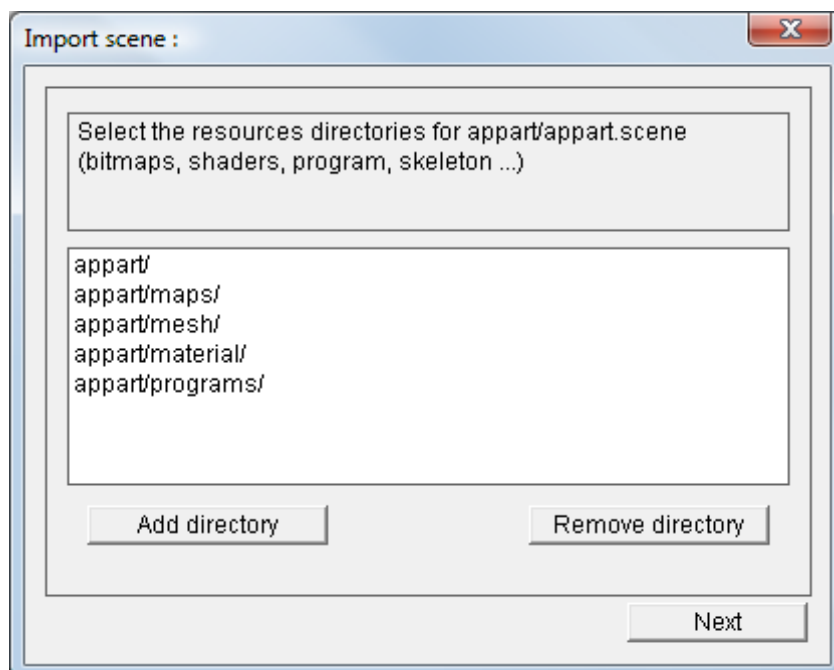
A wizard appears:



Click "Next" and then browse your folder \Partition_LocalUsr to select your Ogre scene file you want to open:



If the informations are available in the file the resources folders will be automatically added.

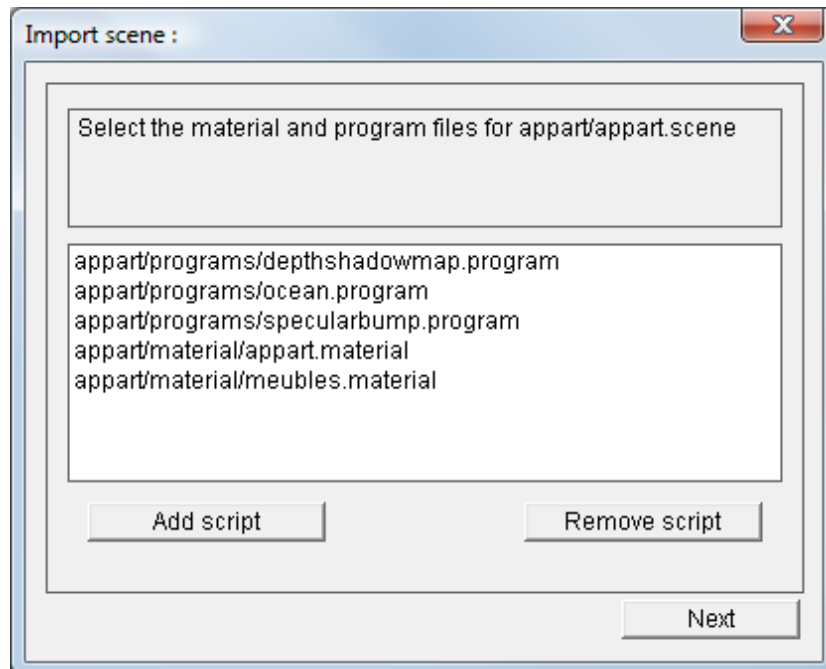


NB : you can also import your resources (.xos, .scene, .mesh) using Drag and Drop from your windows explorer to the 3D Edition zone of OpenSpace3D

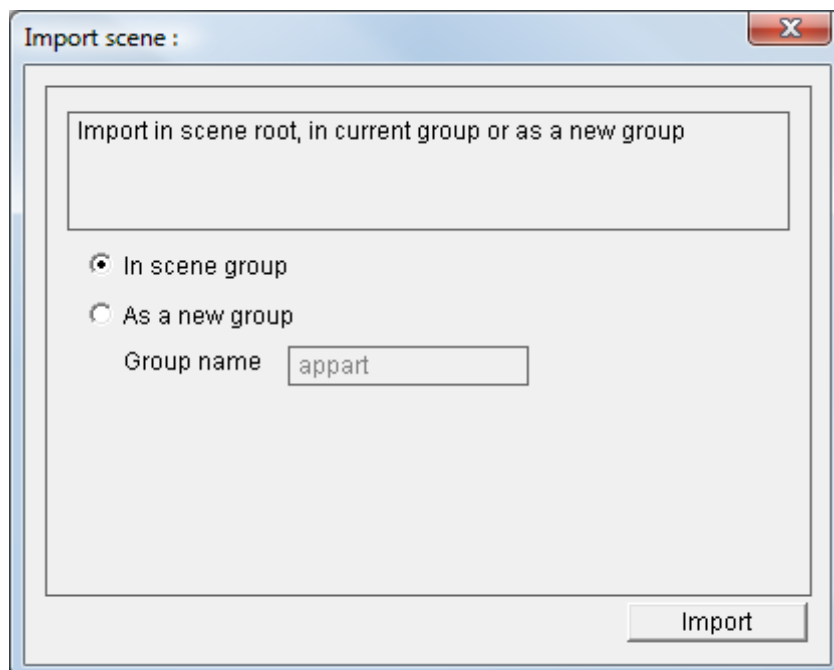


Otherwise you will need to specify the files where the resources are (textures, mesh, material, shaders ...)

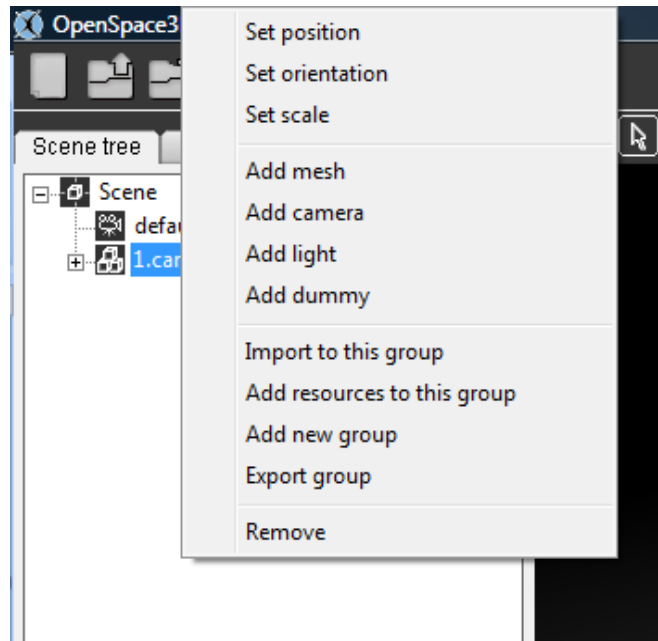
In the next step you'll have to tell the material and program script files necessary to your scene, the script found in the files will be automatically added.



You can then choose how your scene will be imported into your current scene. Either in the default group "Scene" or in a new group.



If you access the Import Wizard from the menu of a group its selection will also be possible.



Save an OpenSpace3D scene

Click the icon « Save »

Hot key: CTRL + S



Exporting OpenSpace3D applications

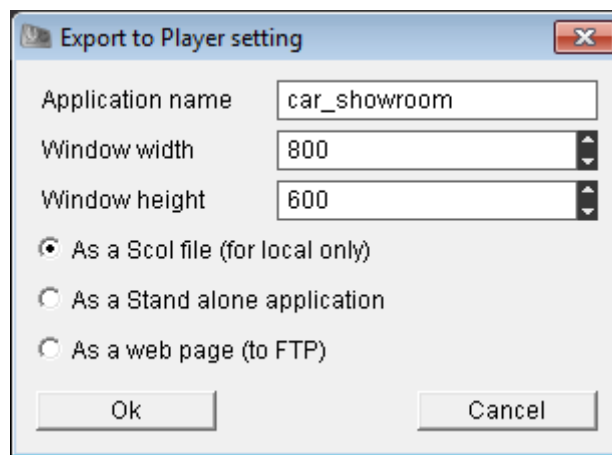
This part of the documentation concerns the different possibilities of export for openSpace3D applications.

Under OpenSpace3D, the interface is identified by the icon:



Click on the button « Export to OpenSpace3D Player »

A setting box is displayed



The "Application name" parameter is used to define the application name displayed in the OpenSpace3D Player window.

The "Window width" sets the width of the OpenSpace3D Player window.

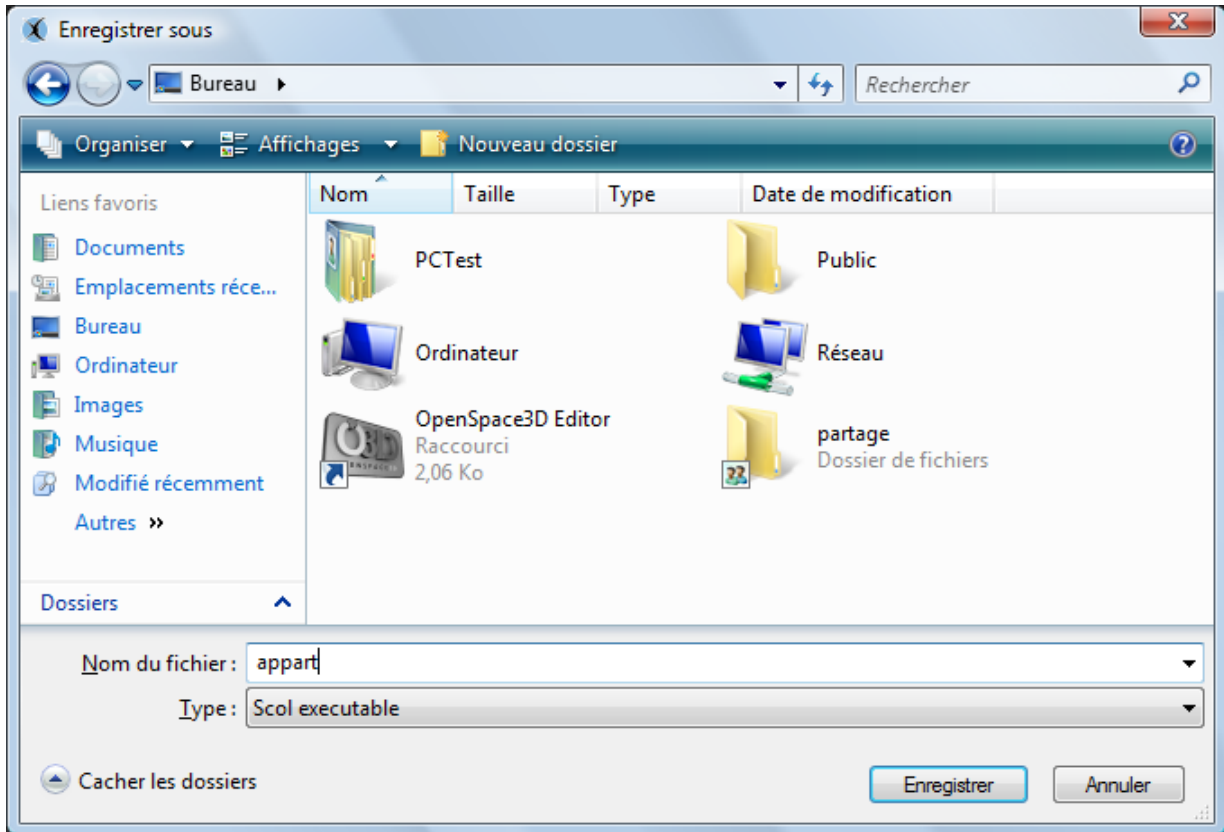
The "Window height" sets the height of the OpenSpace3D Player window.

Then, you have the option of creating an export from three possibilities

As a scol File

Executable file .scol allowing you to launch the application directly into the player (on the development machine only).

After validation of this type of export, if you click the OK button a file selection window appears.



When you click "Save" a file allowing you to launch your scene directly into the OpenSpace3D Player is created. (Note this file contains no resources of your scene).

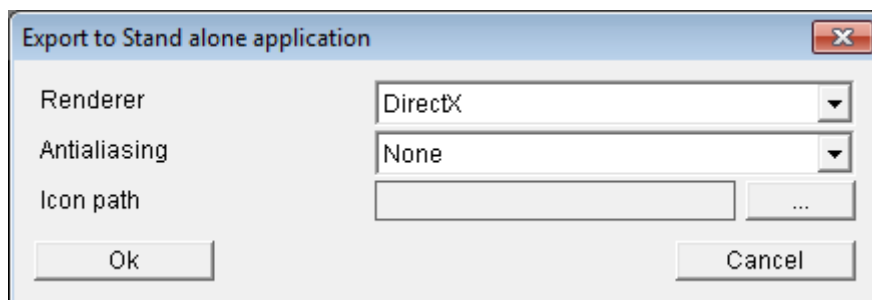
Then simply double click the file to launch your scene in the OpenSpace3D Player.

As A Stand Alone Application

In this case, OpenSpace3D assists you in order to export your application as a stand-alone executable. This will contain all the resources and the needed Voyager to launch your project.

This mode is very useful when developer wants to distribute an application on a CD,an USB stick or if he wants to create a setup installer.

After choosing this option a new window appears:



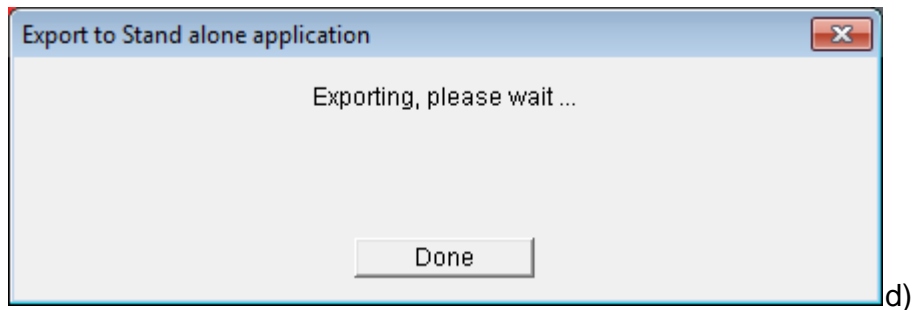
It allows you to set some options for your application:

« Renderer »: Allows you to choose the rendering middleware mode (opengl / DirectX)

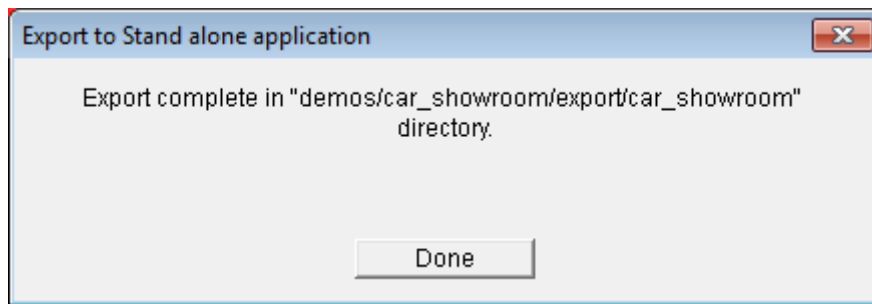
« Anti aliasing »: Allows you to choose the level for anti aliasing option

« Icon Path »: Allows you to choose an icon for your developed application

After validating the interface, OpenSpace3D tells you he is to export your application



When this operation is completed, OpenSpace3D displays:



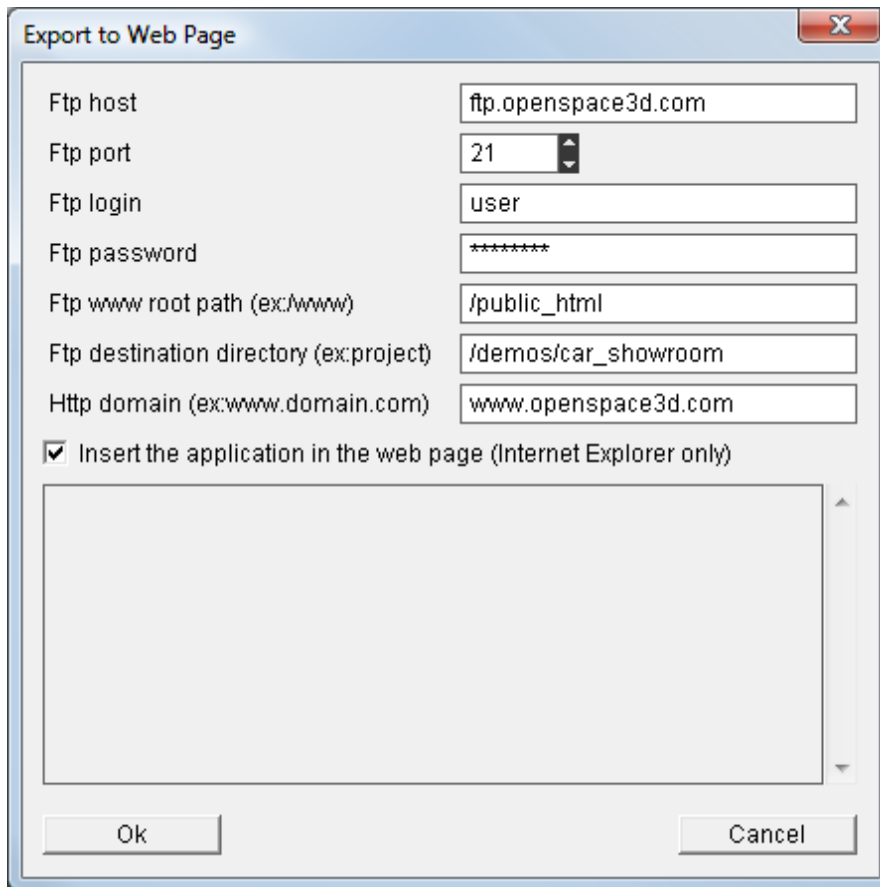
Your application developed has been exported to the folder specified by that interface, in this case:

« demos/car_showroom/export/car_showroom »

N.B: This path is relative to your working scol partition.

As a Web Page

When you click "OK" a setting window appears.



The "FTP host" parameter is the address of ftp server.

The "FTP port" parameter is the port of ftp server.

The "FTP login parameter is the login for connecting to the ftp server.

The "FTP password" parameter is the password for connecting to the ftp server.

The "www Ftp root path" parameter is the path from the ftp root folder containing the website. For example `"/public_html/mysite/"`.

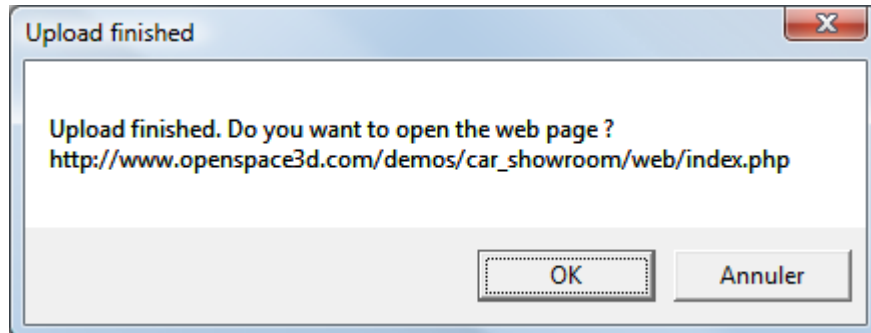
The "Ftp destination directory" parameter is the path where the application is downloaded. For example, `"/demos/my_application/"`.

The "Http domain" parameter is the http address of your site.

Select the "Insert the application in the web page if you want the 3D application to be embedded in the browser page (Internet Explorer only).

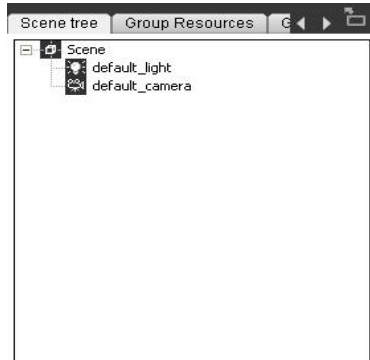
When you click the button "Ok", the login and downloading procedure starts.

At the end of the procedure a warning popup propose you to test the application.



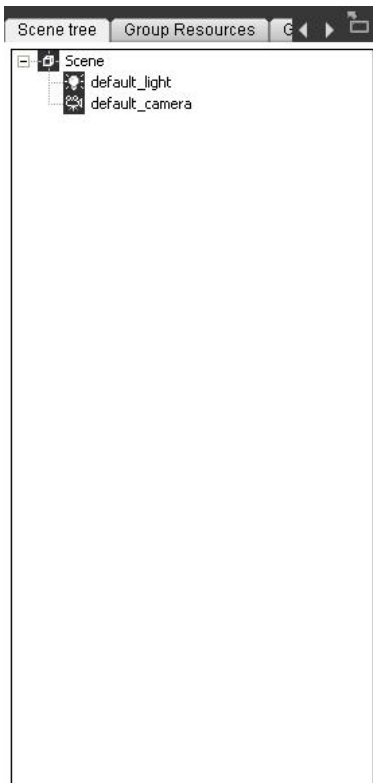
Managing the Scene tree and informations on Resources

The interface of this tool is:



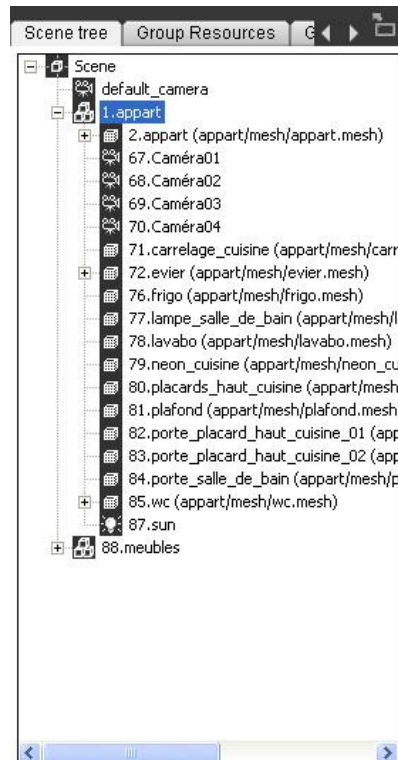
This interface consists of several tabs.

- Scene Tree
- Group Resources
- Group Meshes
- Ressources Directories



Scene Tree

Example of blank scene tree




Example of open scene tree



Each element of the tree in a scene loaded in OS3DEditor is represented by an icon that characterizes it:

NB: By right clicking on the various elements of the scene tree you can access the different settings for each object in the tree



Scene: "This is the representation of the main node of the 3D application"

- Set background color
- Set ambient color
- Set shadow methode
- Set fog setting
- Set material scheme
- Set physic setting

- Show scene grid
- Show scene helpers
- Show scene infos
- Auto fit on select

- Add sky box
- Add sky dome
- Add sky plan
- Add compositor
- Add mesh
- Add camera
- Add light
- Add dummy
- Add particle system

- Import to scene
- Add resources to scene
- Add new group

Set Background color: Sets the background color of the 3D window (See Color Map)

Set Ambient color: Sets the color of the ambient light in the scene. (See Color Map)

Set Shadow Method: Options on real-time shadows in the scene (see 24 Advanced Use and definitions)

Set Fog setting : Options on 3D scene fog .



Set physic setting : Physic parameters of the 3D scene (see 24 : advanced use : Physics Engine)

Show grid scene: To show or hide the grid in the 3D scene.

Show scene helpers : To show or hide icon helpers for camera, lights and dummies.

Show scene infos : To show or hide the render scene infos.

Auto fit on select : Enable or disable the auto fit of the camera on object selection.

Add sky dome : To add a dome sky type.

Add sky box : To add a box sky type.

Add sky plan : To add a plan sky type..

Add compositor : allows the addition if a compositor in the scene

Add mesh: Allows the addition of a mesh in the scene (see 24 Advanced Use and definitions)

Add Camera: Allows the addition of a camera

Add Light: Allows the addition of a light

Dummy Add: Allows the addition of a scene node in the scene (see 24 Advanced Use and definitions)

Add particle system : allows the addition of a particle sytem in the scene

Import to stage (see 2.7 Advanced Usage and definition)

Add to resources scene: Allows the addition of a resource in the scene graph (see 2.7 Advanced Usage and definition)

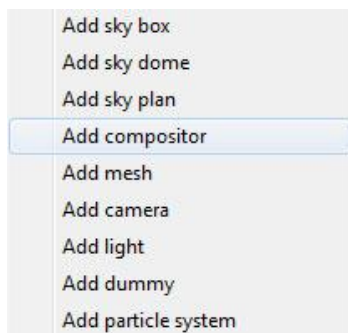
Add new Group (see 24 Advanced Use and definitions)

Compositors : "Représentation d'un compositor"

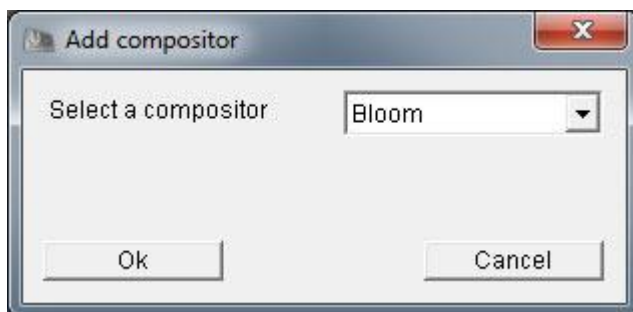
Pour charger un système de particule dans votre scène vous devez préalablement avoir mis vos fichiers ".material", ".compositors" et les ressources associées (shaders) " dans votre dossier Partition_LocalUsr sous le répertoire Scol Voyager de vos documents

Une fois cette opération effectuée charger vos ressources dans la scène : clic droit "Scene" puis "Add ressources to scene"

Vous pouvez désormais ajouter un compositor grâce au menu "Add Compositor" depuis le menu clic droit sur le noeud de Scene



Sélectionnez le compositor que vous souhaitez ajouter depuis la liste défilante :

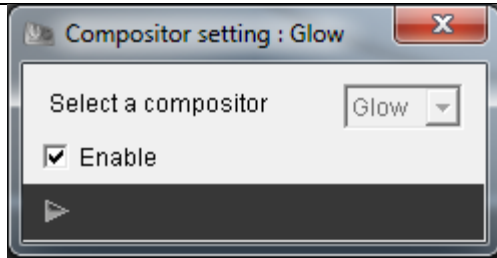


Renommez le pour mieux repérer vos différents compositors dans la scène puis cliquez sur "Ok"

Pensez à activer le compositor en effectuant un clic droit sur le compositor dans l'arbre de scène



Utilisez le PlugIt Compositor pour faire interagir le compositor avec d'autres PlugITs



Compositors Explications

The Compositors can make filters made by adding post-processing on the final image.

This allows to obtain particular effects such as rendering black and white, sepia or a night made camera.

In OpenSpace3D, a library is provided to allow you to test, modify or re-create effects in your applications.

Notions :

// Black and white effect

compositor B&W

```
{
  technique
  {
    // Temporary textures
    texture rt0 target_width target_height PF_A8R8G8B8

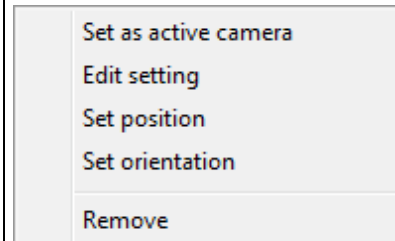
    target rt0
    {
      // Render output from previous compositor (or original scene)
      input previous
    }

    target_output
    {
      // Start with clear output
      input none
      // Draw a fullscreen quad with the black and white image
      pass render_quad
      {
        // Renders a fullscreen quad with a material
        material Ogre/Compositor/BlackAndWhite
        input 0 rt0
      }
    }
  }
}
```

```
}  
}  
}
```

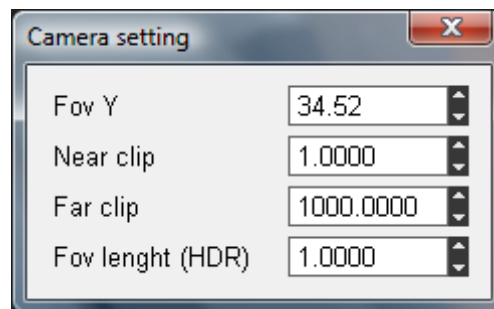
Like the material, the compositors authorize the use of several techniques on the final rendering.

The 3D engine loads in a texture image rendering application and apply the texture on a material that achieves the desired effect.

**Camera : « 3D camera Representation »**

Set as Active Camera: Allows you to make the active camera selected in the 3D

Edit Setting: (see 24 Advanced Use and definitions)

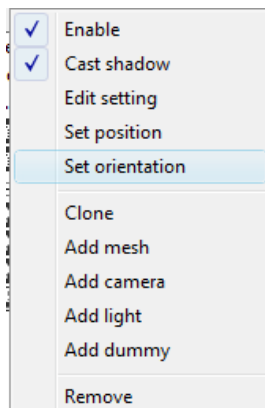


By entering a value in the text box + Left click on the arrow moving up / down
It immediately changes the value.

For more information about these parameters refer to Part 24 Advanced Use and definitions.



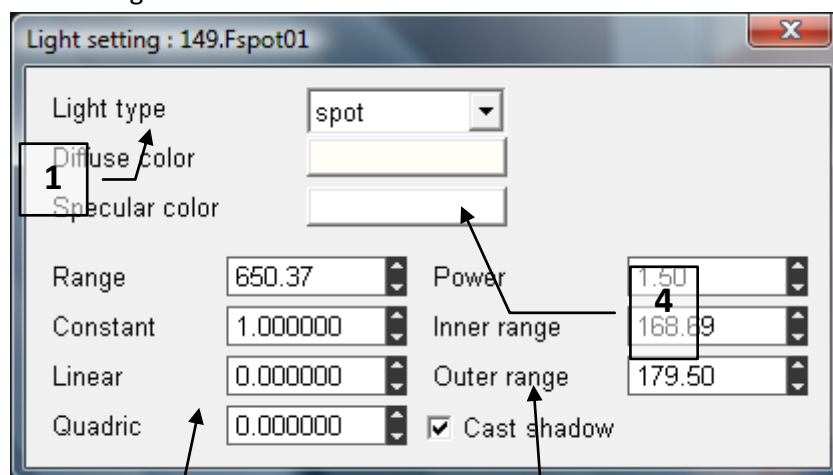
Light : « 3D Light Representation »



Enable: Enable or disable the light.

Cast Shadow: When this option is checked the light will project shadows in the scene

Edit Setting:



2

3

1. Light Type:

The drop-down menu allows you to select the desired type of light in the scene
Three types: Spot, Directional, Point. (see 24 Advanced Use and definitions)

2. Parameters of light

By entering a value in the text box + Left click on the arrow moving up / down, it immediately modifies the value.

(see 24 Advanced Use and definitions).


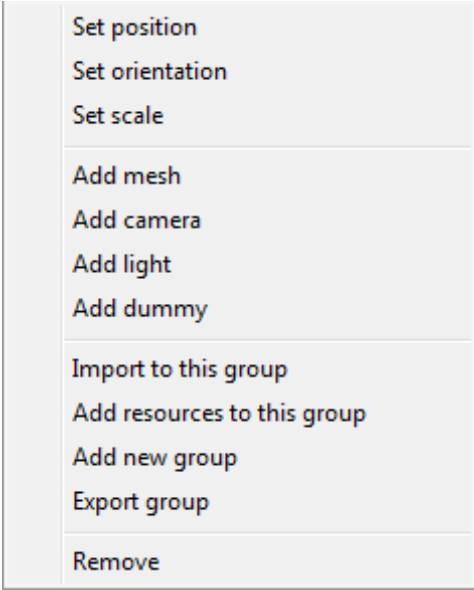

Note: Settings (Inner and Outer Range) are accessible only in the event of a light spot because they define the opening of this light.

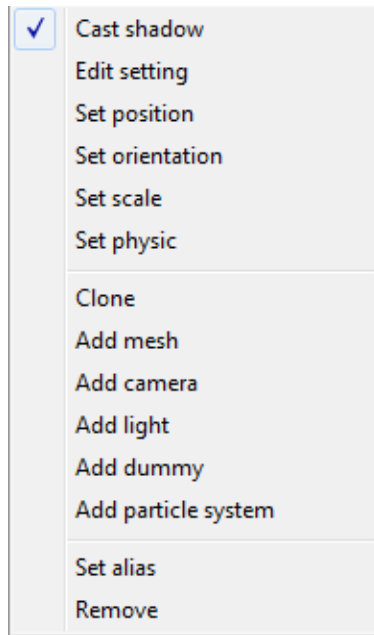
3. Cast Shadows

When this option is checked the light will project shadows in the scene

4. Diffuse and Specular color

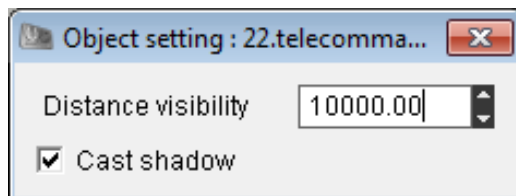
Both options allow you to define the color of the light

| | |
|---|--|
|  | <p>Group: "This is the representation of a group" (see 24 Advanced Use and definitions)</p>  <p>Set Position: Changes the position of a group of objects</p> <p>Set Orientation: Allows you to change the direction of a group of objects</p> <p>Set Scale: To change the scale of a group of objects</p> <p>Add mesh: Allows the addition of a mesh in the scene (see 24 Advanced Use and definitions)</p> <p>Add Camera: Allows the addition of a camera</p> <p>Add Light: Allows the addition of a light</p> <p>Dummy Add: Allows the addition of a node in the group stage (see 24 Advanced Use and definitions)</p> <p>Import to stage (see 24 Advanced Use and definitions)</p> <p>Add resources to scene: Allows the addition of a resource in the scene graph (see 24 Advanced Use and definitions)</p> <p>Add new Group (see 24 Advanced Use and definitions)</p> <p>Export Group (see 24 Advanced Use and definitions)</p> <p>Remove: Remove the group and its content</p> |
|  | <p>Mesh: "the representation of a 3D Object"</p> |



Cast Shadow: Option to specify if the mesh will cast shadow

Edit setting : To set options on the object



Distance Visibility: Distance from which the object is no longer visible (All or nothing LOD)

Cast Shadow: To allow object to cast shadows

Index materials: Add index on materials names and make them unique.

Set Position: Changes the position of the object

Set Orientation: Allows you to change the orientation of the object

Set Scale: To change the scale of the object

Set Physic : see 24 Advanced Use and definitions

Clone : To duplicate an object from scene tree

Mesh Add: Add a mesh to the object (see 24 Advanced Use and definitions)

Camera Add: Adds a son camera to the object

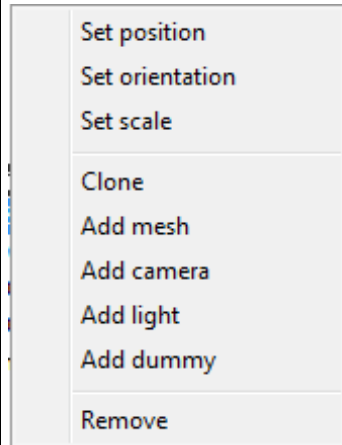
Light Add: Adds a son light to the object

Dummy add: Adds a node object on the scene

Remove: Remove the object and his son



Node: " representation of a scene node " (see 24 Advanced Use and definitions)



Set Position: Changes the position of node

Set Orientation: Allows you to change the orientation of node

Set Scale: To change the scale of the node

Clone : duplicate a node and its hierarchy

Mesh Add: Adds a mesh node (see 24 Advanced Use and definitions)

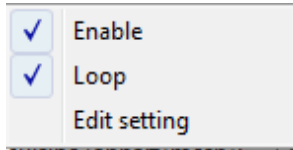
Camera Add: Adds a son camera to the node

Light Add: Adds a son light to the node

Dummy add: Adds a scene node on the node

Remove: Remove the node and its son

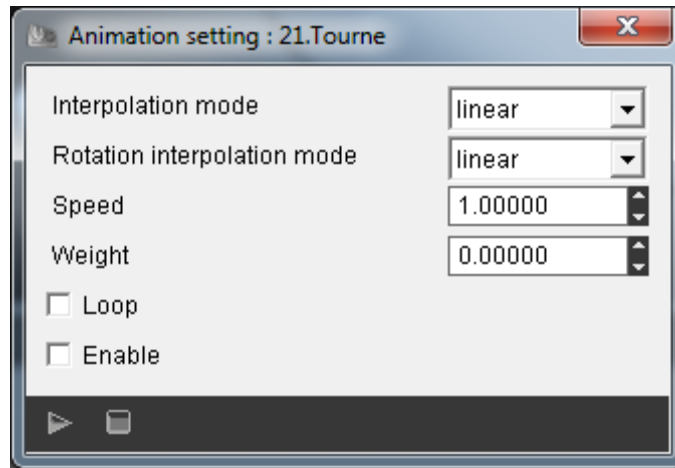
 Animation : « Representation of an animation »



Enable: Specifies that the animation will be active when launching the scene

Loop: Specifies that the animation will be played in loop

Edit Setting: Set the options for playing the animation



Interpolation Mode: Sets the type of interpolation mode between the animation keys

Rotation interpolation mode: Allows you to define the types of interpolation between animations concerning rotations

Speed: Sets the playback speed of animation ie the time to be spent between each animation key

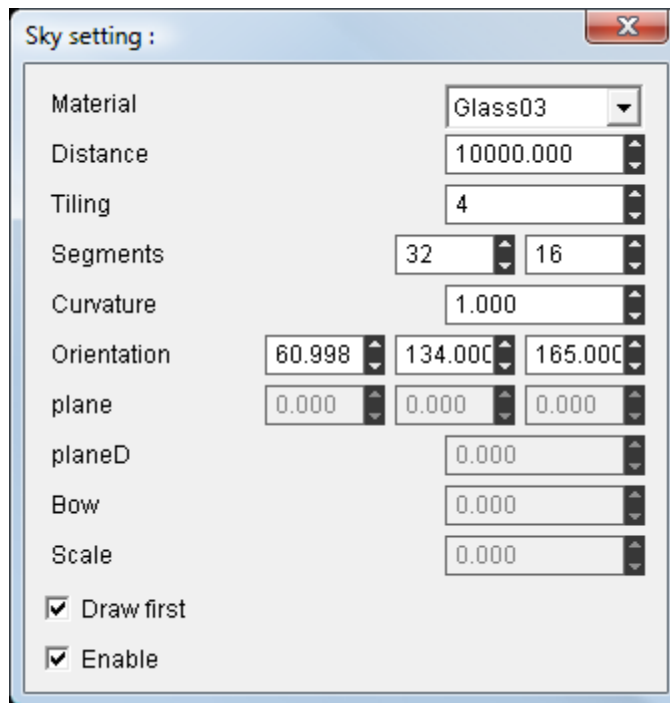


Sky : « Representation of a sky »

Edit setting
Remove

Edit Setting : sky options

Remove: remove the sky



Material: Allows you to select the material for the sky (to add materials you will have to add the resources manually in the scene)

Distance: distance from the sky

Tiling: The number of repetition of the texture

Segments: Number of segments XY on the created object

Curvature: Amount of curvature of the sky

Orientation: Orientation of the sky

Plane: vector for the plane type

Planned: Distance from the sky for the plane type

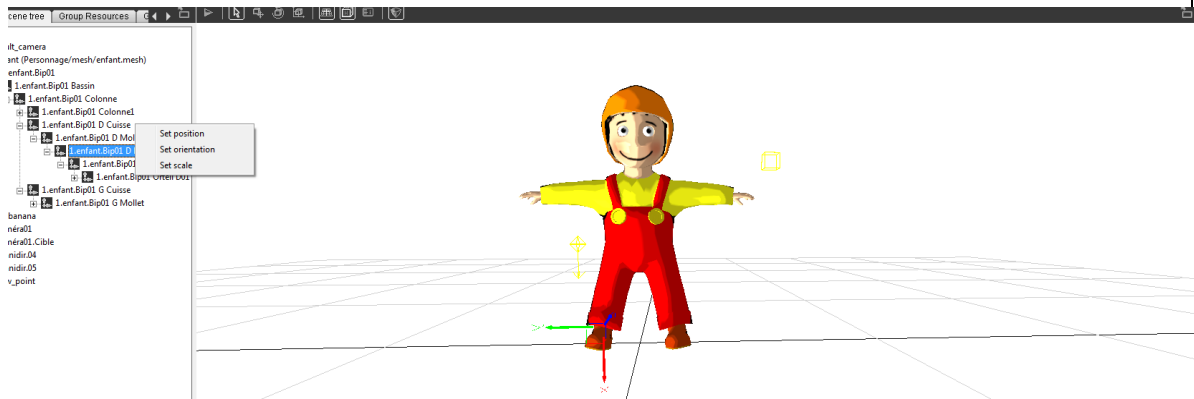
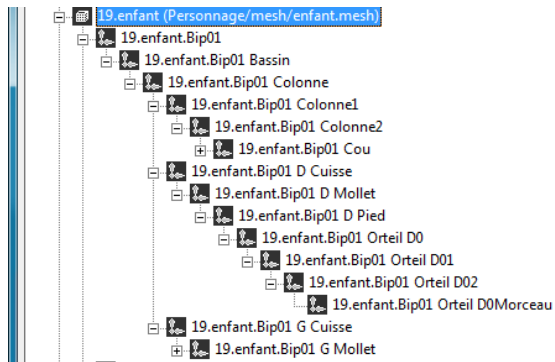
Bow: Amount of curvature for the type map

Scale: Scale of the sky for the plane type

Draw first: Set if the sky is displayed before the objects in the scene

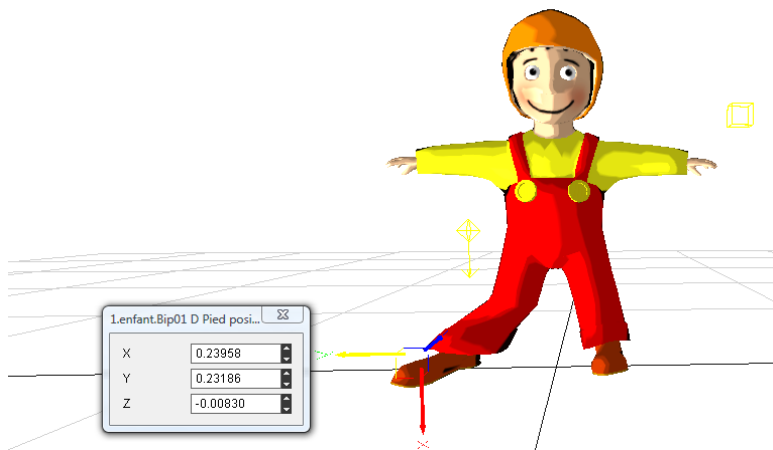
Enable: Enables or disables the sky

Bones : "Representation of the bones of an objet"

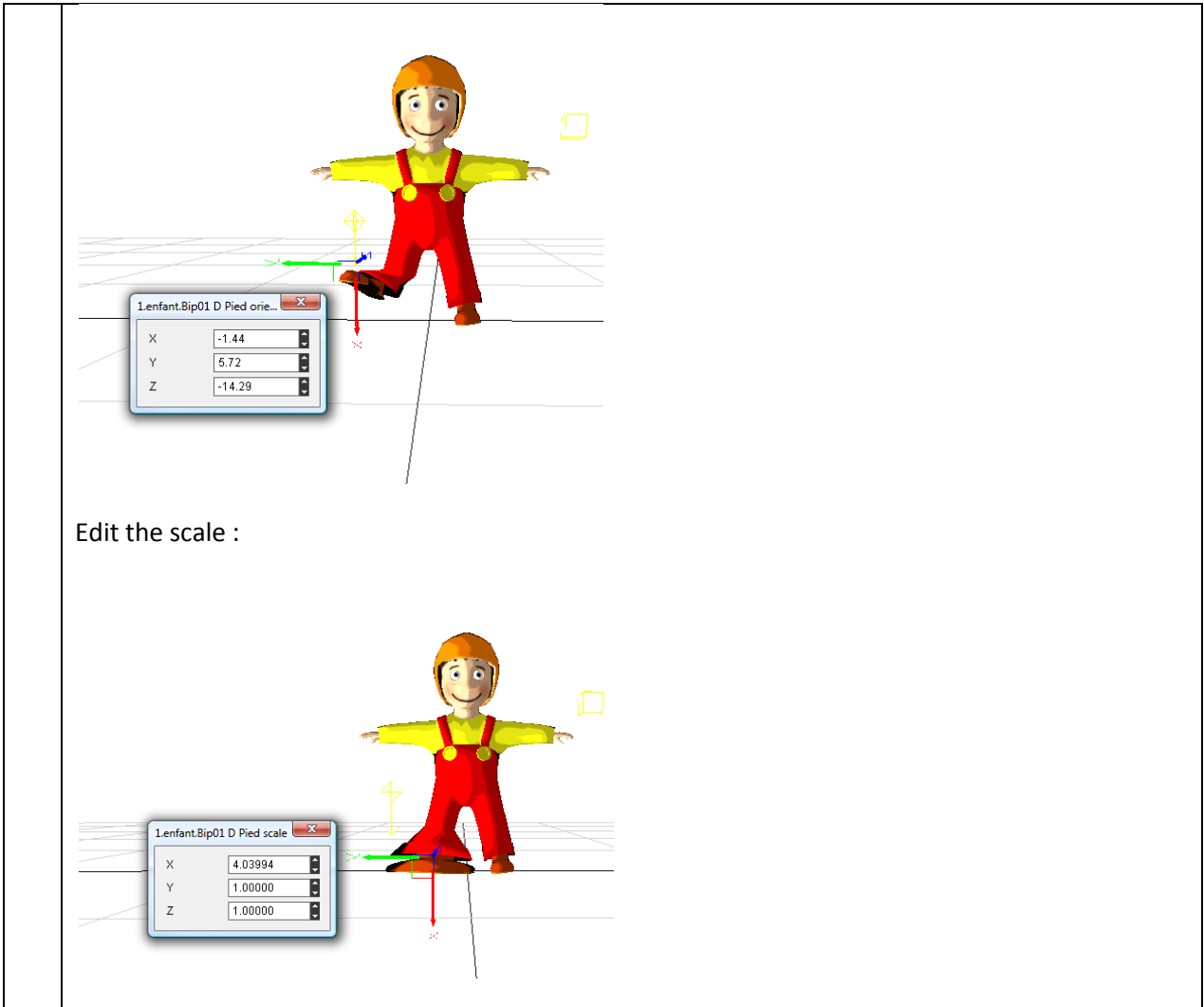


On each bones you can :

Edit the position :



Edit orientation :

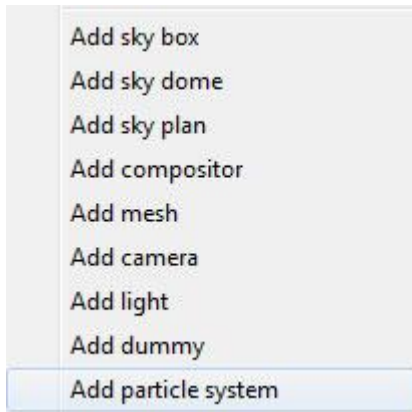




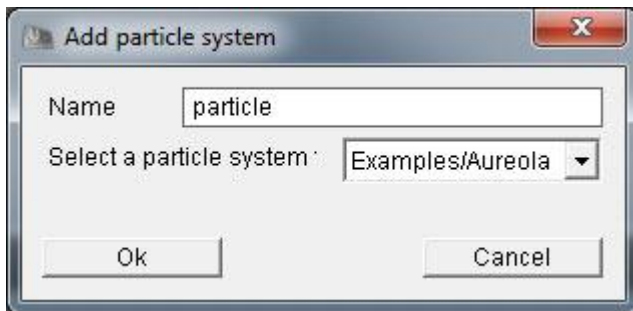
Particle system

Before loading a particlescript in your scene, you had to add all resources files from your directory : Partition_LocalUsr

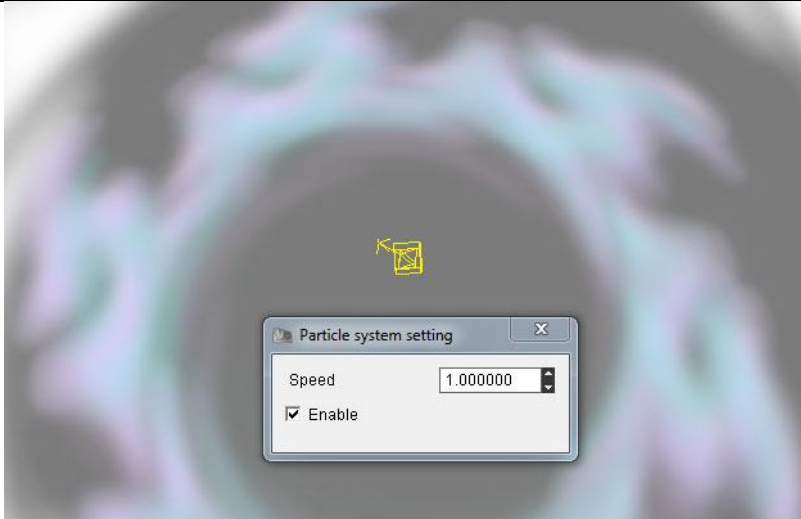
When it is done, by a right click on "scene", choose "Add particle system" from the scene node of the scene Tree.



Select the particle system you want from the list :



Give it a name and apply with the « ok » button.



Choose the speed and enable it.

The particle script

The use of particle script in OpenSpace3D is based on particle system from Ogre3D engine. Indeed, Ogre3D manage the use of particle resources with particle script. In the OpenSpace3D resources, several scripts are given. It gives you the possibility to modify or re-create some particle systems. This operation could be done by using templates.

The particle scripts are identified by files own by « .particle » extension. We detailed above the use of particle scripts and the associated vocabulary.

```
// Exudes aureola particles which around the model float upwards
```

```
particle_system Examples/Aureola
{
    material    Examples/Aureola
    particle_width 200
    particle_height 200
    cull_each    false
    quota       100
    billboard_type perpendicular_common
    common_direction 0 1 0
    common_up_vector 0 0 1
}
```

```
// Area emitter
```

```
emitter Box
{
    angle       30
    emission_rate 4
    time_to_live 5
}
```

```

position    0 -100 0
direction   0 1 0
velocity_min    0
velocity_max    30
colour_range_start 0.3 0.3 0.3 0.0
colour_range_end 0.8 0.8 0.8 0.0
width        10
height       10
depth        30
}

// Make em float upwards
affector LinearForce
{
    force_vector    0 70 0
    force_application add
}

// Fader
affector ColourFader2
{
    red1 +0.4
    green1 +0.4
    blue1 +0.4
    alpha1 +0.7

    red2 -0.25
    green2 -0.25
    blue2 -0.25
    alpha2 -0.3333

    state_change 3.5
}

// Rotater
affector Rotator
{
    rotation_range_start 0
    rotation_range_end 360
    rotation_speed_range_start 0
    rotation_speed_range_end 180
}
}

```

Particle scripts explications :

Particle system :

The visualization of a particle script in OpenSpace3D is based on the particle systems notion. A particle system contains several types of objects, we describe some of the following concepts must be understood for proper handling of particulate systems :

- **Material:** Sets the name of the material which all particles in this system will use. All particles in a system use the same material, although each particle can tint this material through the use of its color property.
- **particle_width:** Sets the width of particles in world coordinates. Note that this property is absolute when billboard_type (see below) is set to 'point' or 'perpendicular_self', but is scaled by the length of the direction vector when billboard_type is 'oriented_common', 'oriented_self' or 'perpendicular_common'..
- **particle_height:** Sets the height of particles in world coordinates. Note that this property is absolute when billboard_type (see below) is set to 'point' or 'perpendicular_self', but is scaled by the length of the direction vector when billboard_type is 'oriented_common', 'oriented_self' or 'perpendicular_common'.
- **Quota:** Sets the maximum number of particles this system is allowed to contain at one time. When this limit is exhausted, the emitters will not be allowed to emit any more particles until some destroyed (e.g. through their Time_to_live running out). Note that you will almost always want to change this, since it defaults to a very low value (particle pools are only ever increased in size, never decreased).
- **Cull each:** All particle systems are culled by the bounding box which contains all the particles in the system. This is normally sufficient for fairly locally constrained particle systems where most particles are either visible or not visible together. However, for those that spread particles over a wider area (e.g. a rain system), you may want to actually cull each particle individually to save on time; since it is far more likely that only a subset of the particles will be visible. You do this by setting the cull_each parameter to true.
- **Sorted:** By default, particles are not sorted. By setting this attribute to 'true', the particles will be sorted with respect to the camera, furthest first. This can make certain rendering effects look better at a small sorting expense.
- **Local space:** By default, particles are emitted into world space, such that if you transform the node to which the system is attached, it will not affect the particles (only the emitters). This tends to give the normal expected behavior, which is to model how real world particles travel independently from the objects they are emitted from. However, to create some effects you may want the particles to remain attached to the local space the emitter is in and to follow them directly. This option allows you to do that.
- **Billboard_type :** This is actually an attribute of the 'billboard' particle renderer (the default), and is an example of passing attributes to a particle renderer by declaring them directly within the system declaration. Particles using the default renderer are rendered using



billboards, which are rectangles, formed by 2 triangles which rotate to face the given direction. However, there is more than 1 way to orient a billboard. The classic approach is for the billboard to directly face the camera: this is the default behavior. However this arrangement only looks good for particles which are representing something vaguely spherical like a light flare. For more linear effects like laser fire, you actually want the particle to have an orientation of it's own.

Options for those parameters are:

point: The default arrangement, this approximates spherical particles and the billboards always fully face the camera.

oriented_common: Particles are oriented around a common, typically fixed direction vector (see `common_direction`), which acts as their local Y axis. The billboard rotates only around this axis, giving the particle some sense of direction. Good for rainstorms, star fields etc where the particles will traveling in one direction - this is slightly faster than `oriented_self` (see below).

Oriented_self

Particles are oriented around their own direction vector, which acts as their local Y axis. As the particle changes direction, so the billboard reorients itself to face this way. Good for laser fire, fireworks and other 'streaky' particles that should look like they are traveling in their own direction

- **Perpendicular_common:** Particles are perpendicular to a common, typically fixed direction vector (see `common_direction`), which acts as their local Z axis, and their local Y axis coplanar with common direction and the common up vector (see `Common_up_vector`). The billboard never rotates to face the camera, you might use double-side material to ensure particles never culled by back-facing. Good for aureolas, rings etc where the particles will perpendicular to the ground - this is slightly faster than `perpendicular_self`.

Particle emitters :

Particle emitters are classified by 'type' e.g. 'Point' emitters emit from a single point whilst 'Box' emitters emit randomly from an area. New emitters can be added to Ogre by creating plugins. You add an emitter to a system by nesting another section within it, headed with the keyword 'emitter' followed by the name of the type of emitter (case sensitive). Ogre currently supports 'Point', 'Box', 'Cylinder', 'Ellipsoid', 'HollowEllipsoid' and 'Ring' emitters'.

- **Angle:** Sets the maximum angle (in degrees) which emitted particles may deviate from the direction of the emitter (see `direction`). Setting this to 10 allows particles to deviate up to 10 degrees in any direction away from the emitter's direction. A value of 180 means emit in any direction, whilst 0 means emit always exactly in the direction of the emitter
- **Colour:** Sets a static color for all particles emitted. Also see the `Colour_range_start` and `colour_range_end` attributes for setting a range of colors. The format of the color parameter is "r g b a", where each component is a value from 0 to 1, and the alpha value is optional (assumes 1 if not specified).
- **Colour_range_start** and **colour_range_end:** As the 'color' attribute, except these 2 attributes must be specified together, and indicate the range of colors available to emitted particles. The actual color will be randomly chosen between these 2 values.
- **Emission_rate:** Sets how many particles per second should be emitted. The specific



emitter does not have to emit these in a continuous burst - this is a relative parameter and the emitter may choose to emit all of the second's worth of particles every half-second for example, the behavior depends on the emitter. The emission rate will also be limited by the particle system's 'quota' setting.

- Position: Sets the position of the emitter relative to the Scene Node the particle system is attached to.
- Velocity: Sets a constant velocity for all particles at emission time. See also the velocity_min and velocity_max attributes which allow you to set a range of velocities instead of a fixed one.
- Time_to_live: Sets the number of seconds each particle will 'live' for before being destroyed. NB it is possible for particle Affectors to alter this in flight, but this is the value given to particles on emission. See also the time_to_live_min and time_to_live_max attributes which let you set a lifetime range instead of a fixed one.
- Duration: Sets the number of seconds the emitter is active. The emitter can be started again, see Repeat_delay. A value of 0 means infinite duration. See also the duration_min and duration_max attributes which let you set a duration range instead of a fixed one,
- Repeat_delay: Sets the number of seconds to wait before the emission is repeated when stopped by a limited duration. See also the repeat_delay_min and repeat_delay_max attributes which allow you to set a range of repeat_delays instead of a fixed one.

Particle Affectors :

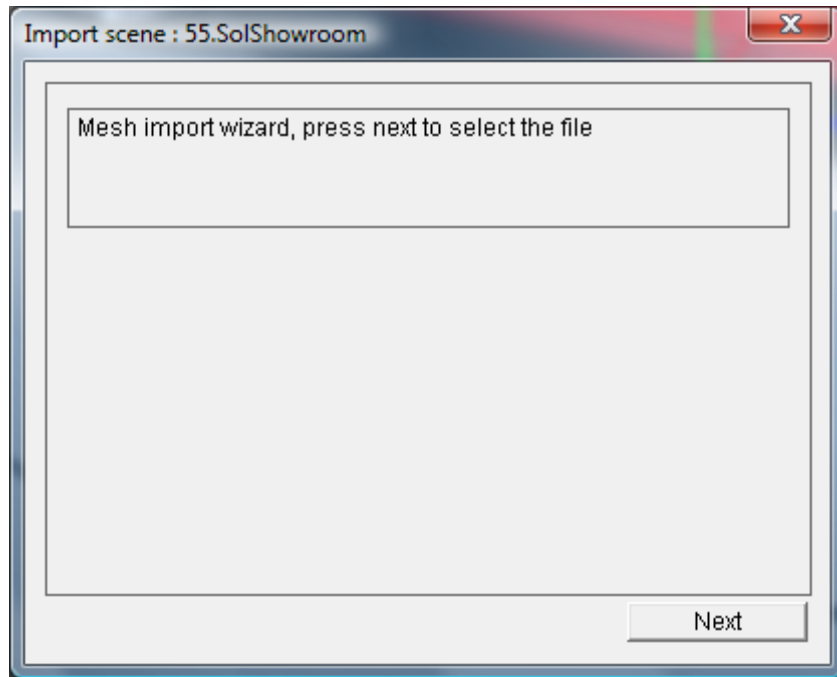
Particle affectors modify particles over their lifetime. They are classified by 'type' e.g. 'LinearForce' affectors apply a force to all particles, whilst 'ColourFader' Affectors alter the color of particles in flight. New affectors can be added to Ogre by creating plugins. You add an affector to a system by nesting another section within it, headed with the keyword 'affector' followed by the name of the type of affector (case sensitive). Ogre currently supports 'LinearForce' and 'ColourFader' affectors.

For further informations : see : <http://www.ogre3d.org/>

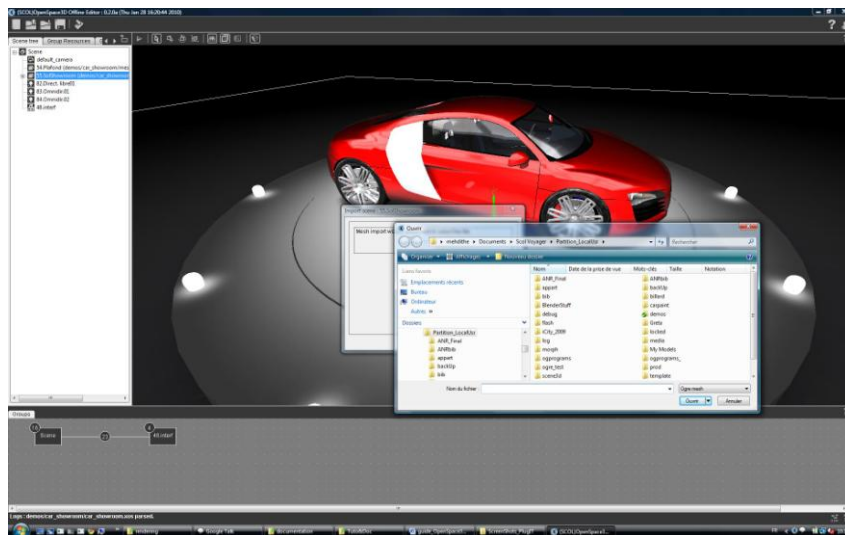
Add a mesh : Wizard

A very useful functionality in OpenSpace3D is the possibility to add a mesh using a wizard which made treatment of files automatically.

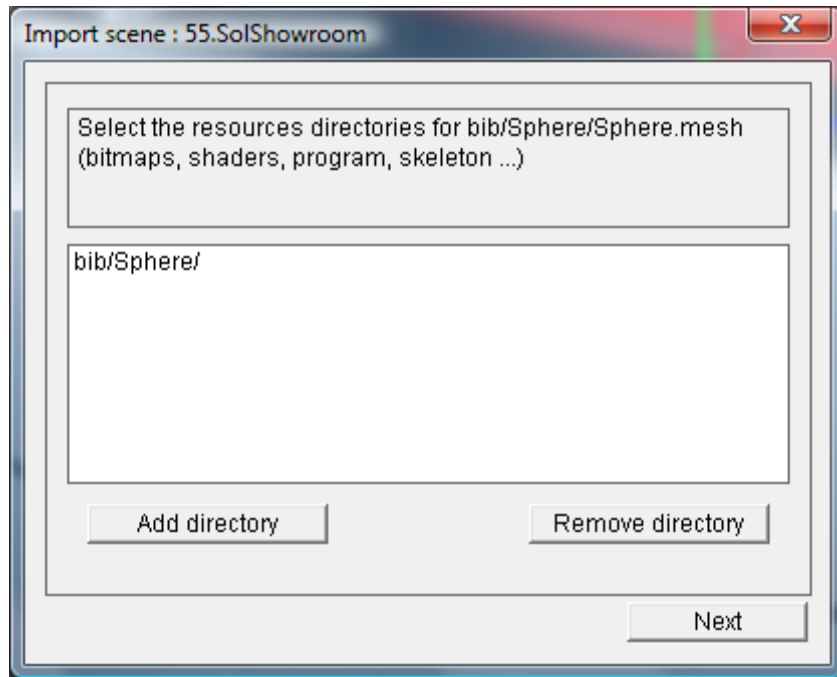
With a right click on a group of the scene Tree in add mesh:



Next :

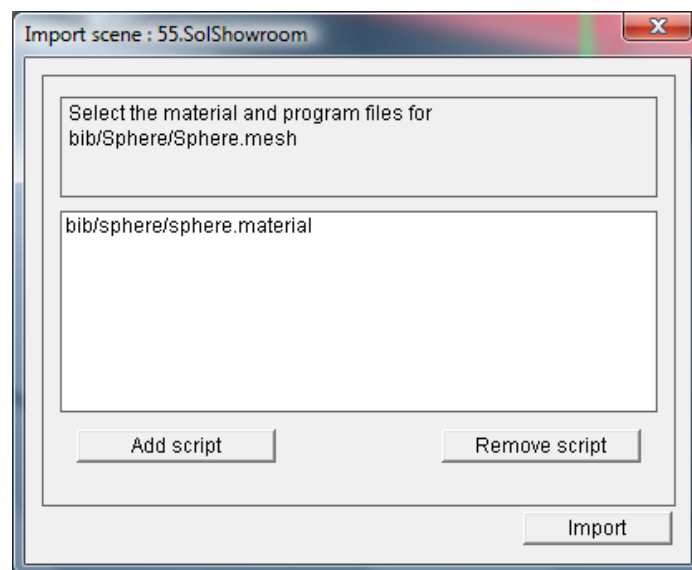


The opened window gives the possibility to load the desired mesh to import it in the scene.

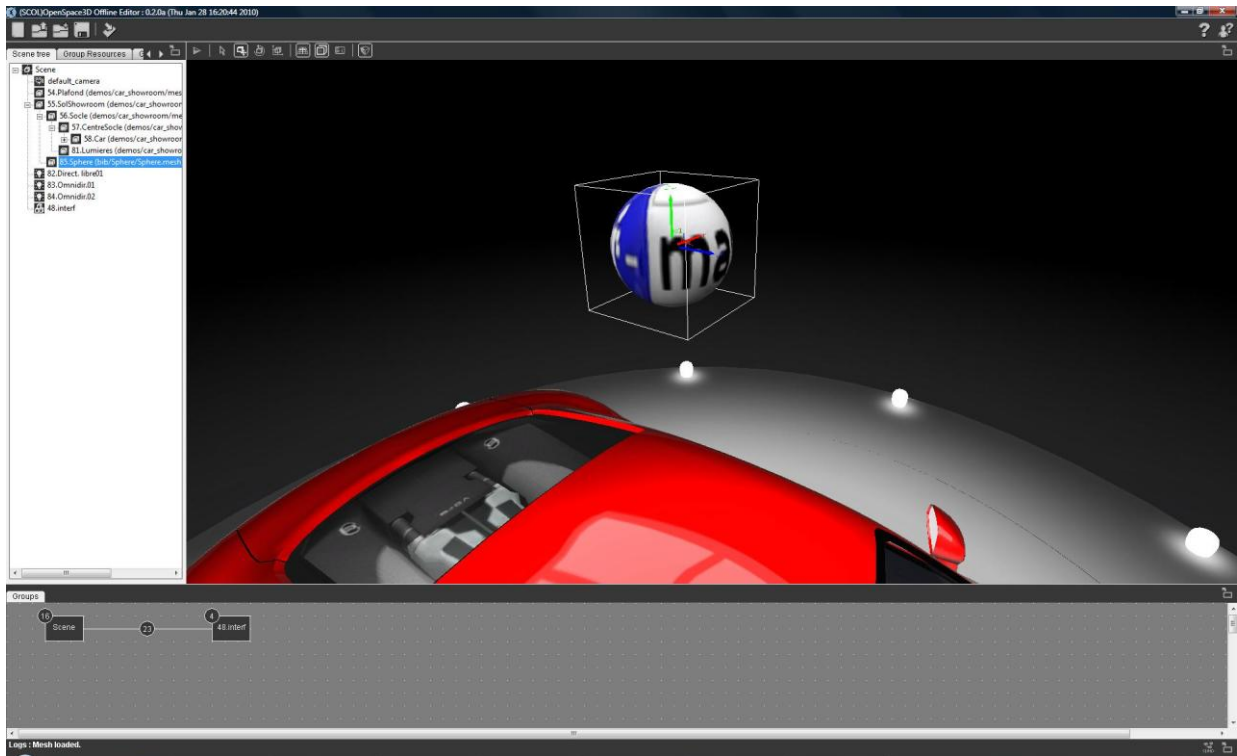


Here for example: We add sphere.mesh. Automatically all the resources repertories will be add at the same time. (see 24 Advanced Use and definitions)

Now, we had to add directories containing all our resources (material script, textures, shaders...)



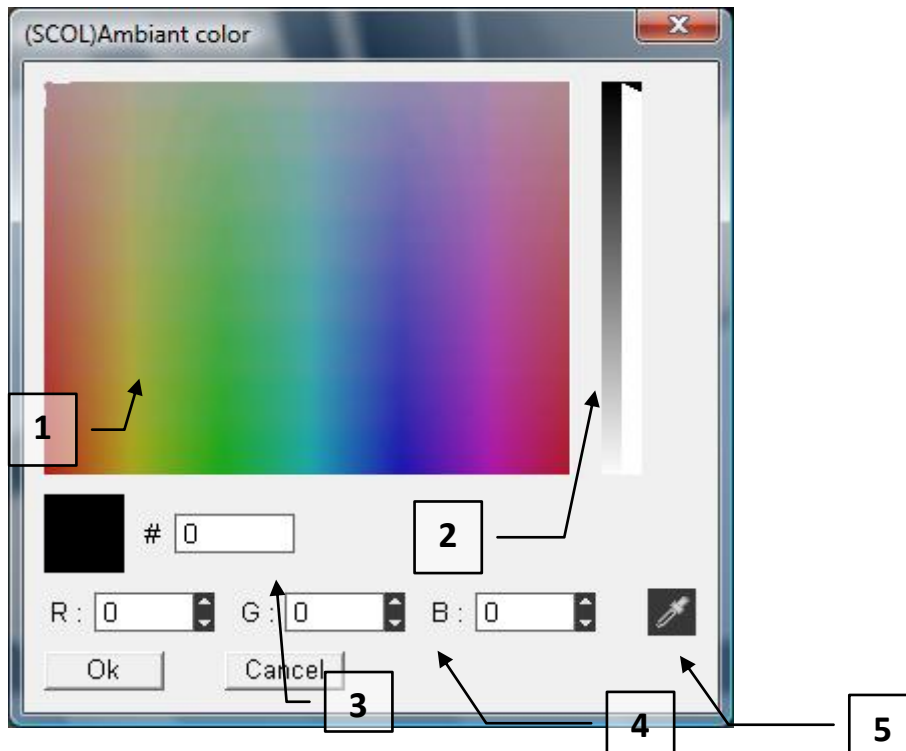
This last step, permits to « Add script » and find the material script associated with our mesh



Here, the sphere has been imported with success

Color Map

Left click on the color box to open a colormap window:



- 1 Selection of the color directly into the colormap
 - 2 Select the color saturation
 - 3 Enter the color directly in Hexadecimal
 - 4 Enter the color in RGB
 - 5 Select a color anywhere on the screen
- 'OK' for validation

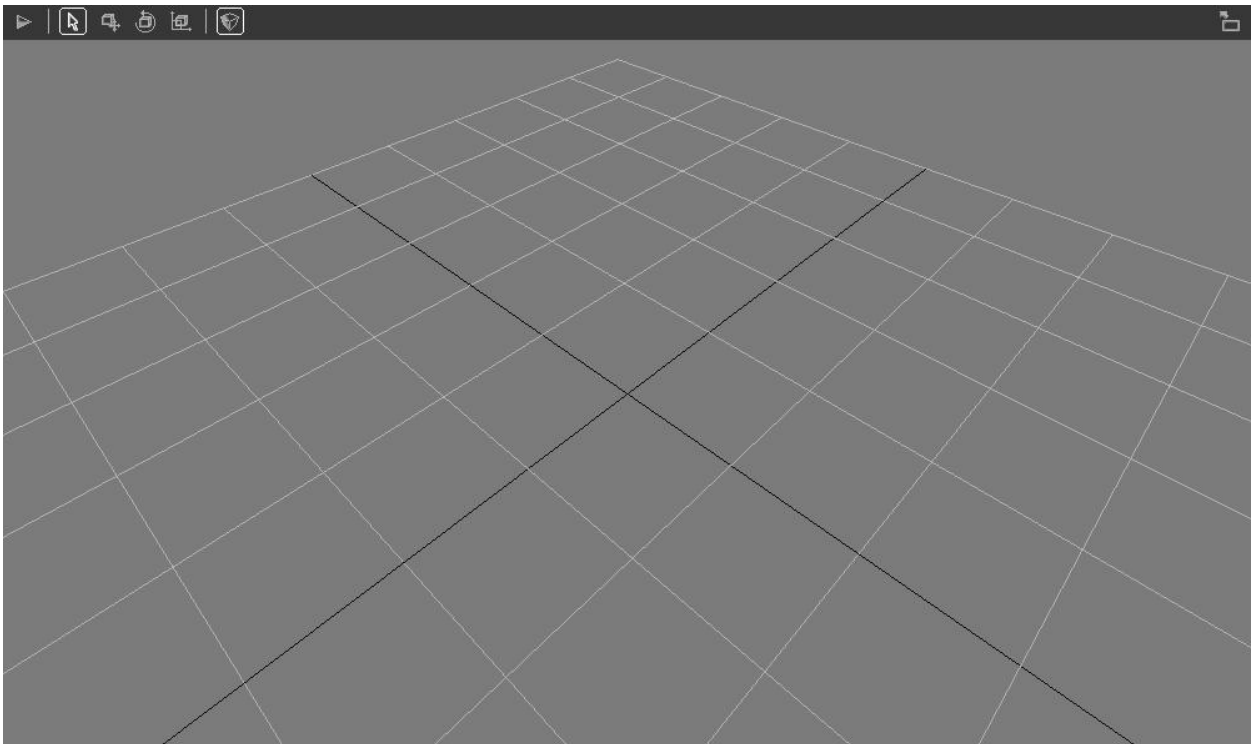
3D Editing

Interface





This is the 3D area to visualize the edited 3D scene.

Thus, It is in this area that the various objects will appear.








This area also allows to adjust the imported objects by altering their spatial coordinates relative to each other regarding to different groups of loaded objects in the scene and to the hierarchy.



The image above shows a blank scene containing only the main camera for the 3D view, a light and a grid whose center is the point of coordinates (0, 0.0) of the scene.

| | |
|---|---|
|  | <p>Play: "Allows the launch of the 3D application" (see Part editing functions) Pause: "Stop the scene from the application mode to edit mode"</p> |
|  | |
|  | <p>Move: Allows you to change the position of the object of the selected scene "</p> |
|  | <p>Rotate "Allows you to change the orientation of the object of the selected scene"</p> |



| | |
|---|--|
|  | Scale "Allows you to change the scale of the object of the selected scene" |
|  | Show / Hide grid : Show or hide the 3d grid |
|  | Show / Hide helpers : Show or hide 3d helpers icones for cameras, lights and dummies |
|  | Show / Hide 3d infos : Show or hide the render infos on screen |
|  | Show / Hide Polygons : allows you to show or hide polygons of the scene |
|  | Navigate Mode "button indicating the 3D browsing mode of navigation in edit mode |
|  | Walk Mode "button to move to a 3D browsing mode in 3rd person" view. |

The toolbar located above this 3D zone includes editing tools for this scene and viewing options of the scene



Moving inside 3D

The button furthest to the right of the toolbar allows you to choose the browsing mode in the 3D area. This button has two states (two browsing modes), left click on this button allows you to switch from a browsing mode to another



Default moving in the 3D in viewer mode:

You just need the mouse in this browsing mode:

- Left Click + Move: Allows the rotation of the view
- Wheel Click + shift: Sets the translation of view
- Wheel front / rear: Zoom in / Zoom Out

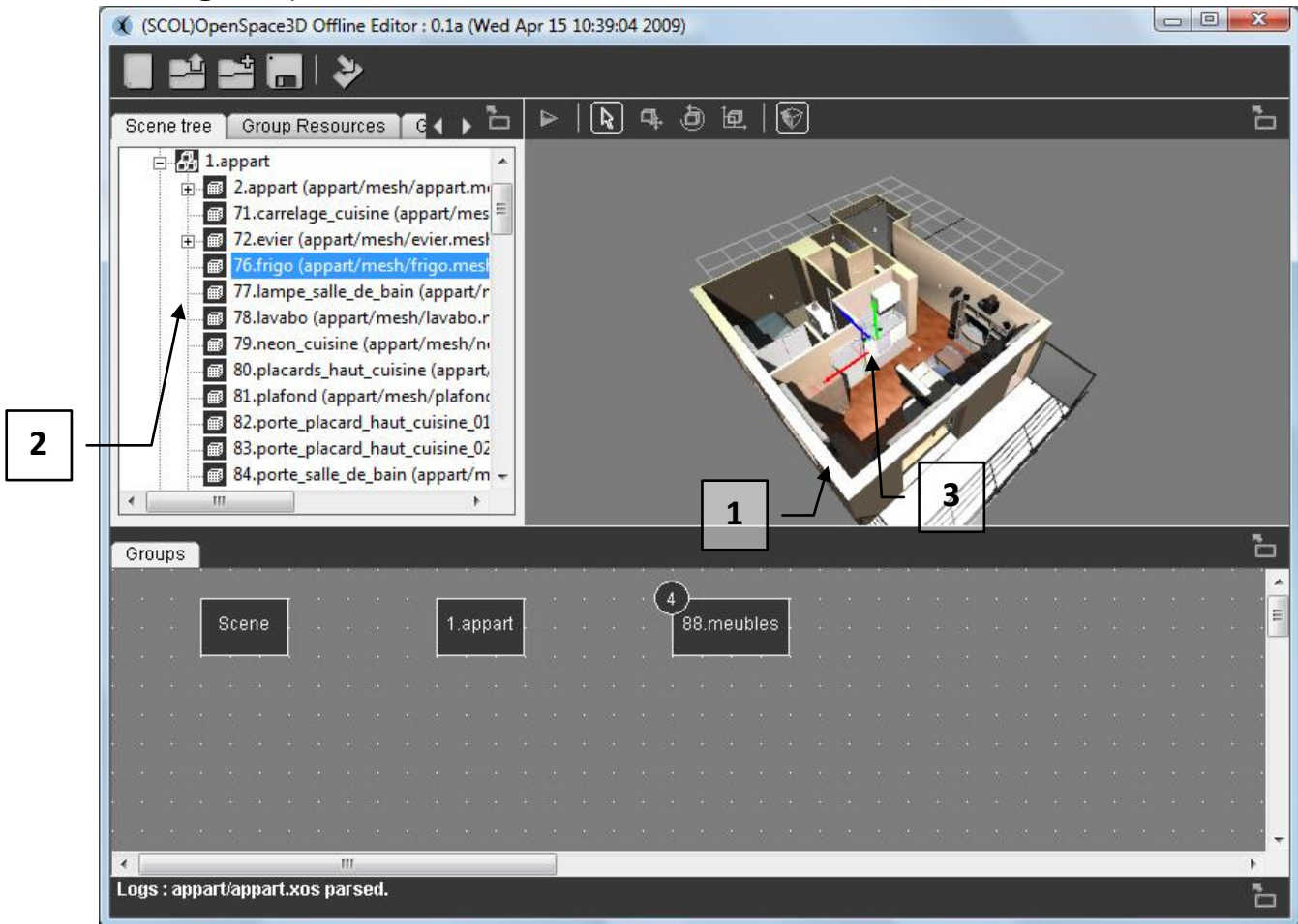


Moving in 3rd person view:

As for the default browsing mode you just need the mouse here:

- Left Click + Move Left or left arrow on the keyboard: To rotate the active camera (I turn the head to the left)
- Left Click + Move Right or right arrow keyboard: To rotate the active camera (I turn the head to right)
- + Left Click Move Up or up arrow on the keyboard: To move forward
- Left Click + Move Down or down arrow on the keyboard: To move backwards
- Shift key pressed + move: To turn on the camera itself (I turn the head)

Selecting an object



1 / The selection of an object is made inside the 3D by left clicking or in the "tree scene area.

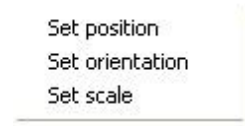
2 / When an object is selected in the 3D, it is automatically selected in the "scene tree" on the left and the camera is automatically positioned in front of this object.

3 / upon selection, a marker allows you to visualize the center of this object
This marker is a representation of the axes according to the directions X (red) Y (green) and Z (blue)
A right click on an object in the 3D allows you to display its associated menu.

Moving an object

There are several ways to move or change an object or group of objects in the 3D scene.

- Thus, by right clicking on an object selected in the "scene tree" you can access the features:



For each of these changes a dialog box opens and you can enter values to change the position, orientation or scale of an object in 3D.

- These same features will be accessible by the buttons:



When an object is selected in the 3D and one of these buttons is then triggered by the action of the mouse left-click on the axes of markers + mouse move cause a transformation on the object.

Example: I choose my object, I left click on the position icon, in the 3D I click on the X axis, and then I move my mouse still clicked. My object will then move along the axis X.

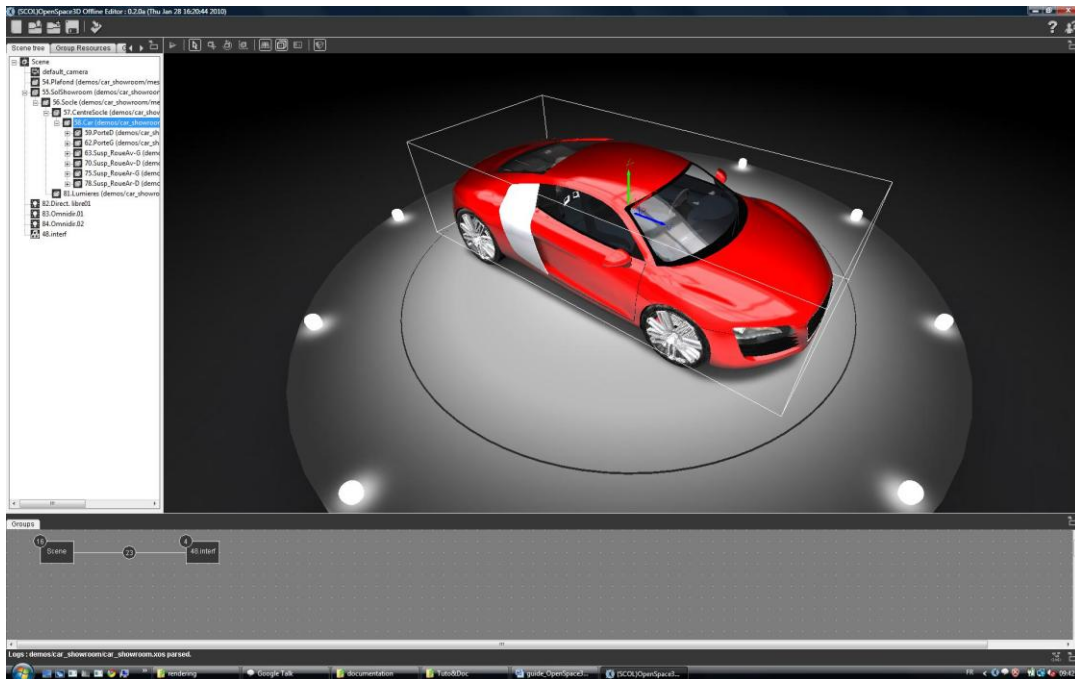
It is the same on the other axes as well as changes in rotation and scale when their respective buttons are switched.

Finally, by right clicking on these buttons, the same dialog box changes on the selected object.

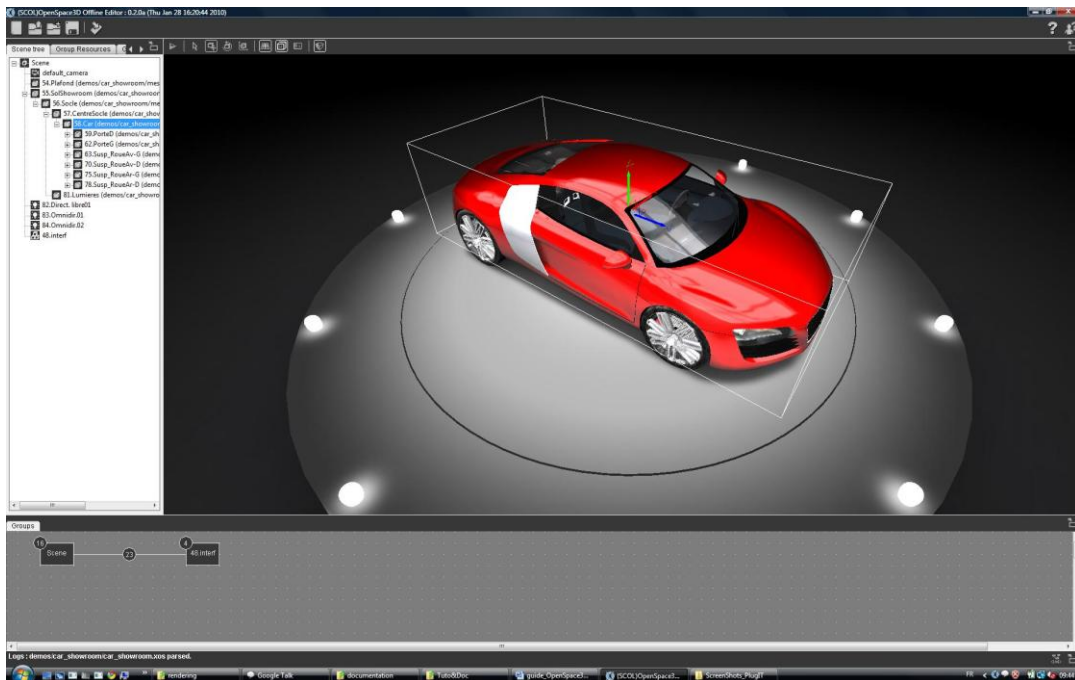
Duplicate an Object

In the 3D viewport, it is possible to duplicate or copy an Object dynamically

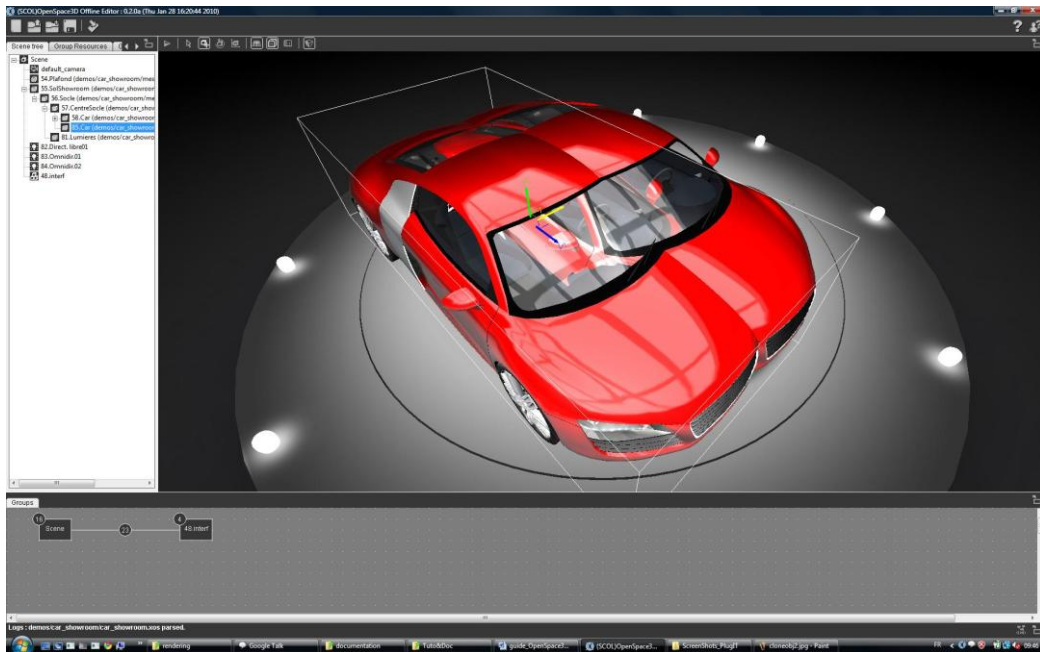
- Select the Object :



- Chose a dis placement option (translation, rotation, scale) :



- Now, by maintaining the « shift » key and displacing the object, we see the duplication of it.





Play/Pause Mode



This button will allow the launch of the scene in application mode.

The application mode is defined by the functions that create the different interactions in the 3D scene.

To create these interactions use the 'Functions Editor'.

N.B : Try with hot keys : CTRL+Back to play/pause your application

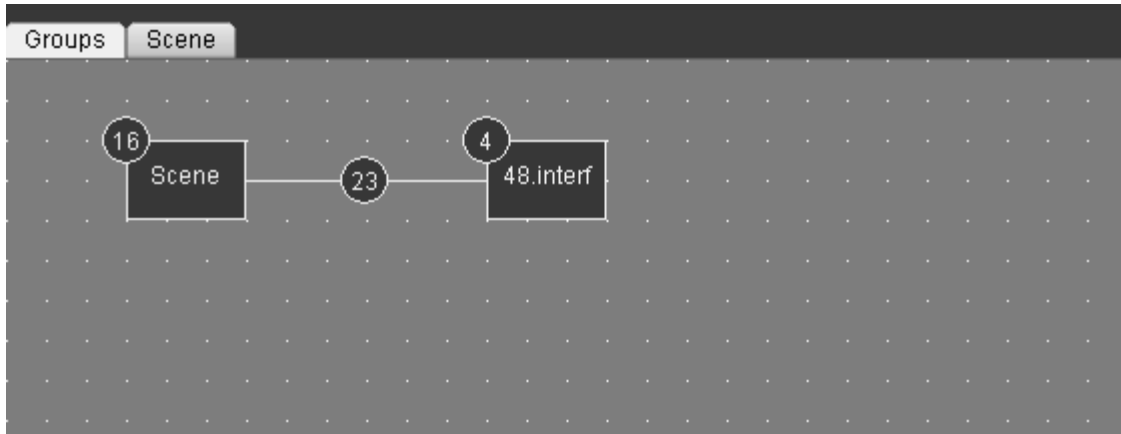
Editing Functions

Interface

Editing Functions Interface organizing resources by groups of resources:

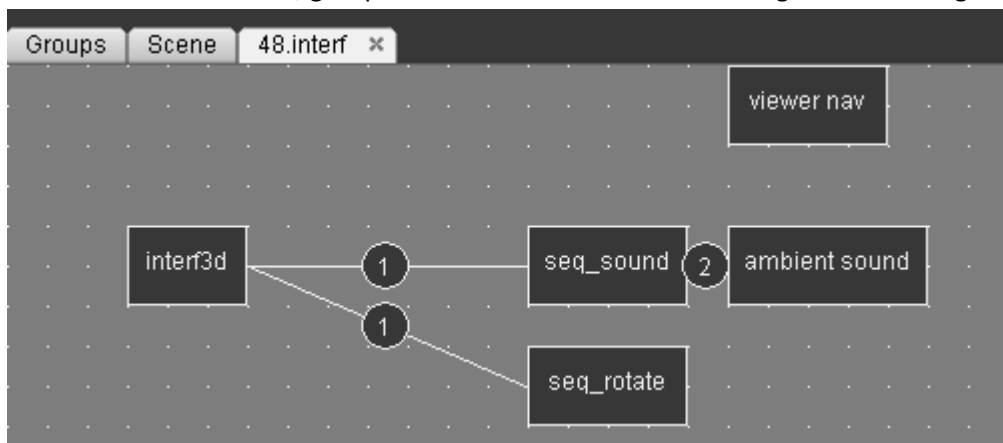
Three groups are represented:

- Scene
- 48.interf



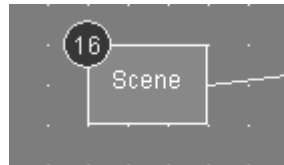
Access to the functions of a group is done by double clicking on their representation (the rectangle).

When open in the functions editor, groups of resources are available through the browsing tabs.

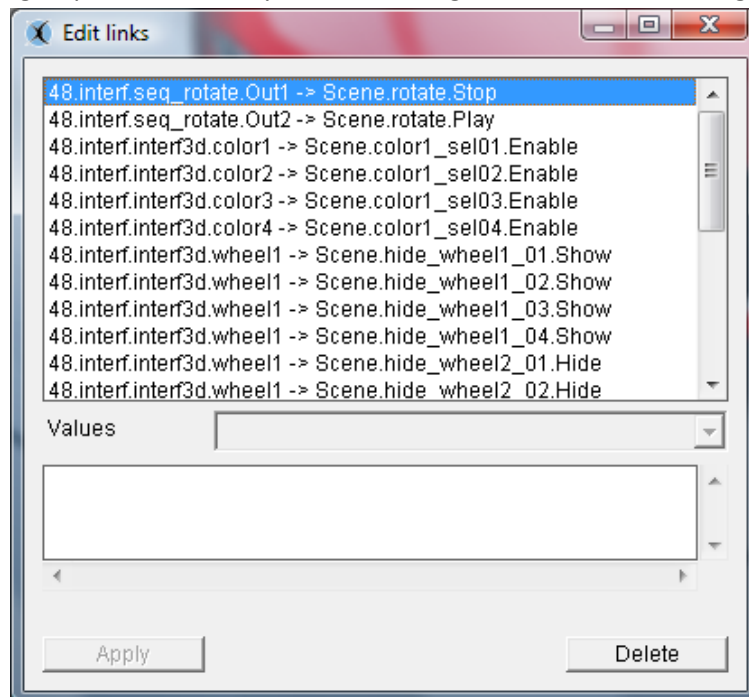


Links

In the "Groups" tab all resource groups are represented and the number of links (interactions) is shown.



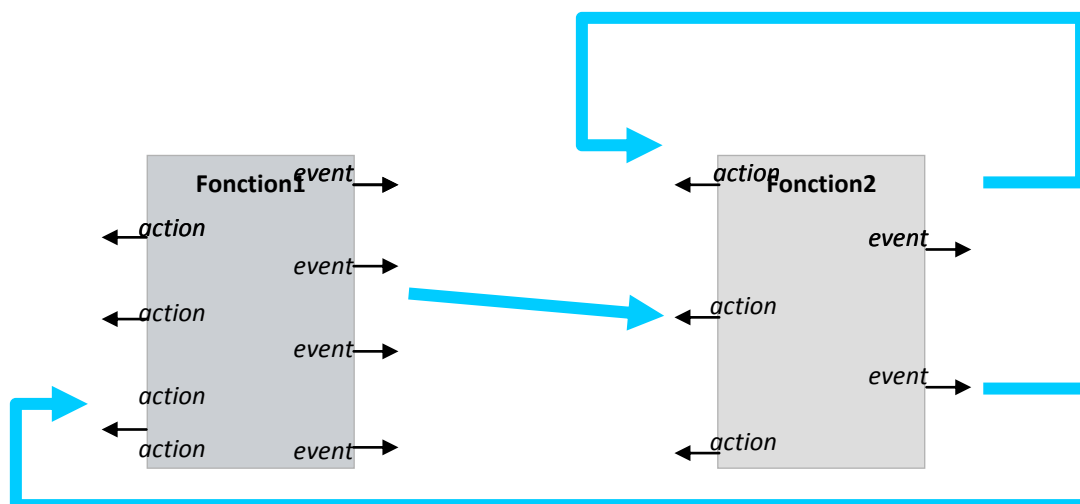
The list of links in the group is accessible by double-clicking on the circle containing the number of links.



Explanations of the concept of "link":

A link is the connection of an event function to an action of another or the same function.

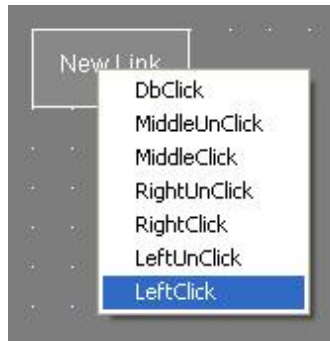
Some links give the possibility to select predefined values in the link editor.



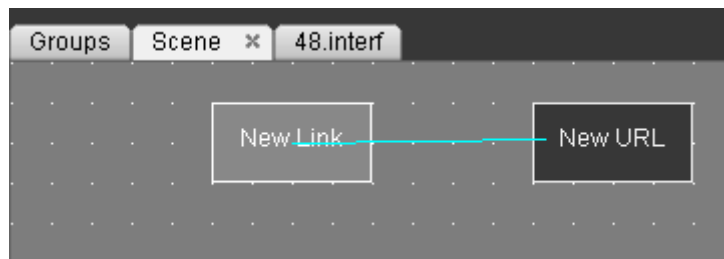
In the remainder of this tutorial, the link description will be the following:

Group.Fonction.Event > Group.Fonction.Action

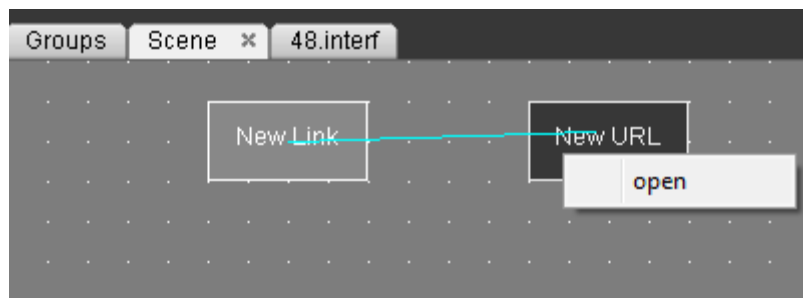
To create links between functions, right click on the source function, select the desired event here "Left Click".



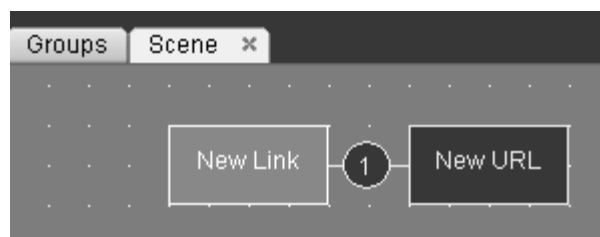
Move the blue link from the source to the destination function



Right click on the destination function and then click on the desired action, here "open"

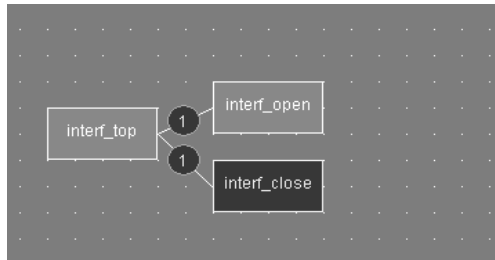


The link is now created.



When you are drawing a link, you could remove its creation by pressing the "echap" key.

To select several plugITs, do a Ctrl+click on the plugITs to select



To duplicate a plugITs group, select them and by a right click choose “copy” (Ctrl+C). After, a right click to choose « paste » (Ctrl+V) in the edition window allows you to paste the group or Cut (Ctrl+X) then Paste (Ctrl+V)

plugITs

Reload (F5)

Copy (Ctrl + C)

Cut (Ctrl + X)

Paste (Ctrl + V)

anr ▶

input ▶

interface ▶

material ▶

maths ▶

media ▶

misc ▶

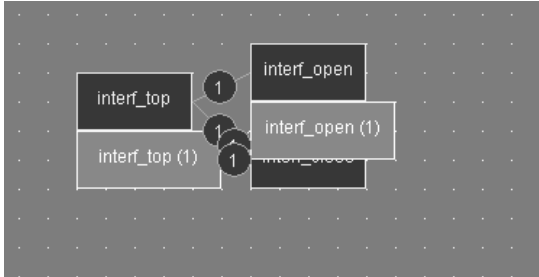
navigation ▶

network ▶

object ▶

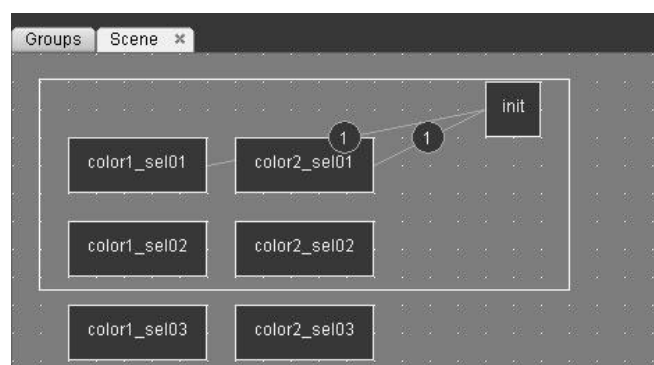
physics ▶

rendering ▶

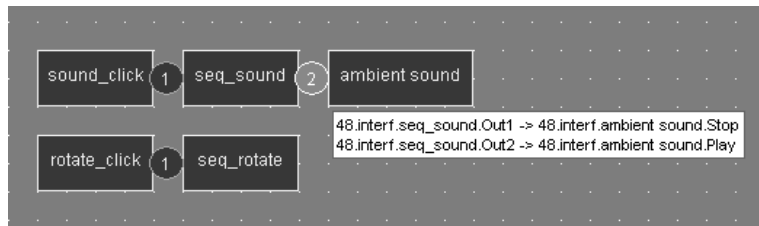


Now, you could delete or edit your plugITs.

You can also make a multi-selection of PlugITs by clicking in the editor area then holding the left mouse button and moving your mouse



When mouse is over a link or a group of links, a bubble let you know the link between plugITs and the properties of these links.



Summary Table of ergonomics in the PlugITS Editor

| | |
|---|---|
| Add PlugIT | Right click in the editor area |
| Create a link | Right click sur le PlugIT |
| Select a PlugIT | Clic gauche sur le PlugIT |
| Select several PlugITs | Left Click then Ctrl+ Left Click to select the others |
| Select a link or a group of link | Left Click on the number of link |
| Select a group PlugITs and their links by area | Left Click in the editor then select |
| Cut one or several PlugITs and their links | Right click "Cut" or Ctrl+X |
| Copy one or several PlugITs and their links | Right click "Copier" or Ctrl+C |
| Paste one or several PlugITs and their links | Right click "Paste" or Ctrl+V |
| Duplicate one or several PlugITs and their links | Ctrl + left click + Move |
| Remove a PlugIT, a link or a group of PlugITs | Suppr |
| Refresh PlugITs and reset | F5 |

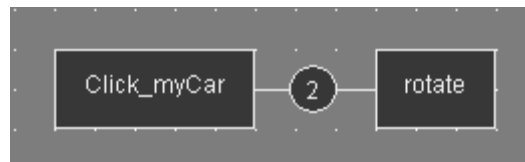
PlugIT Documentation

PlugIT are classified by categories, depend of their functionality:

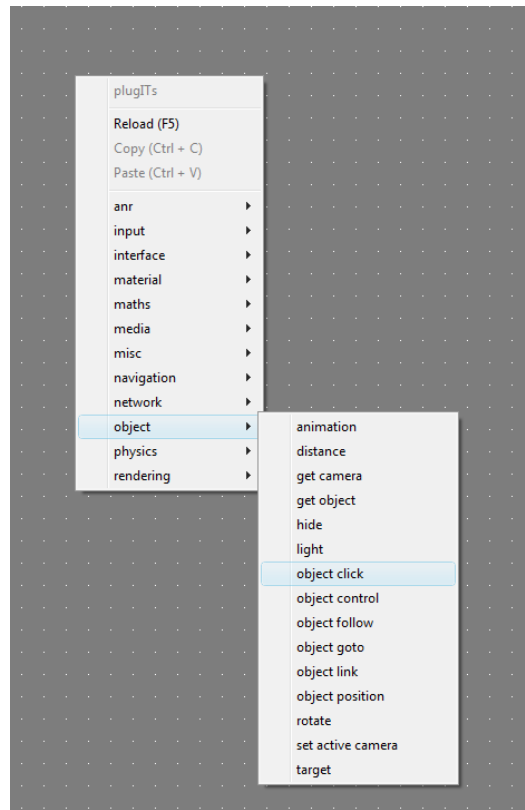
| | |
|------------------|---|
| plugITs | |
| Reload (F5) | |
| Copy (Ctrl + C) | |
| Cut (Ctrl + X) | |
| Paste (Ctrl + V) | |
| anr | ▶ |
| input | ▶ |
| interface | ▶ |
| material | ▶ |
| maths | ▶ |
| media | ▶ |
| misc | ▶ |
| navigation | ▶ |
| network | ▶ |
| object | ▶ |
| physics | ▶ |
| rendering | ▶ |

Detailed example PlugIT: "Object click "

Object click PlugIT allows you to create a click on an object



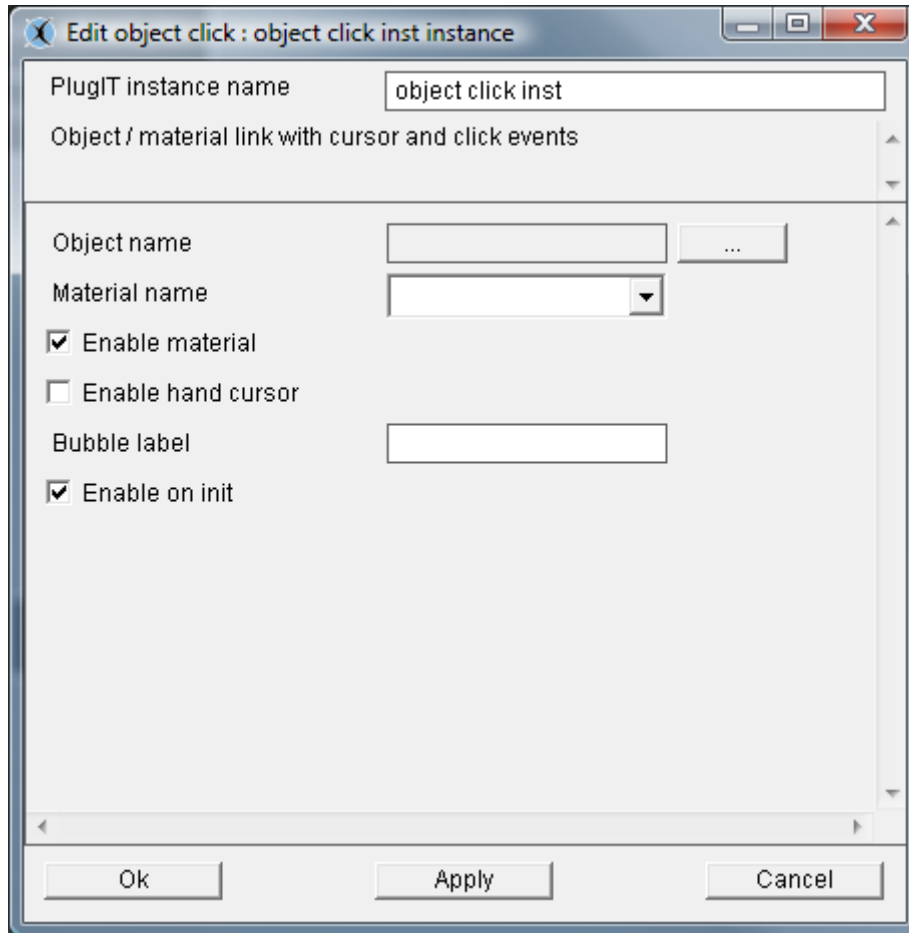
To add this PlugIT (in the furniture group here), just do a right click in an empty area, select "Object" and select "object click"



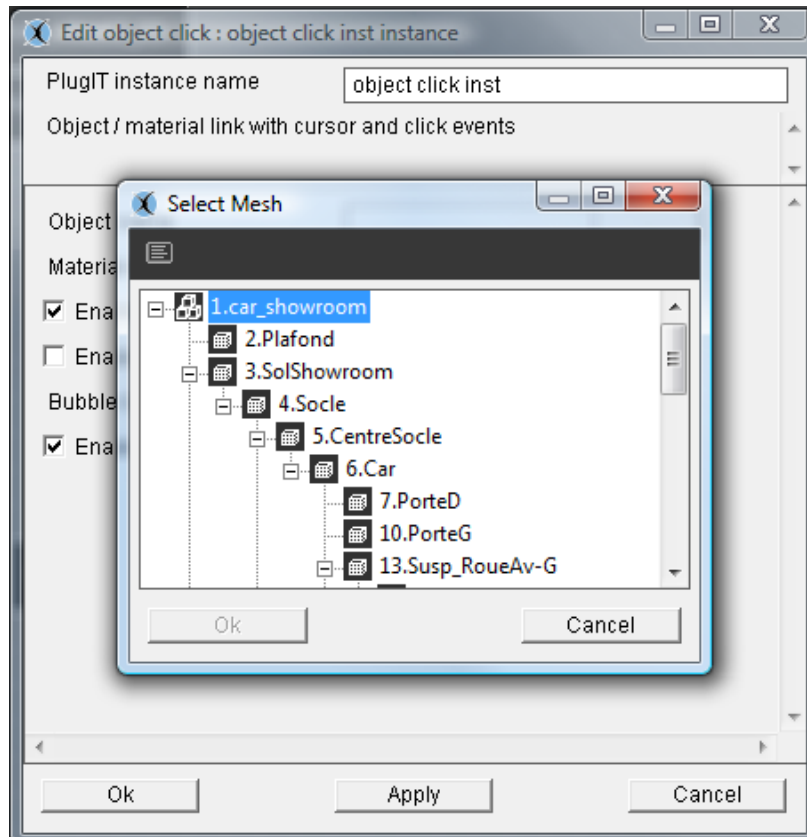
It is recommended to name precisely instances in order to facilitate the reediting.



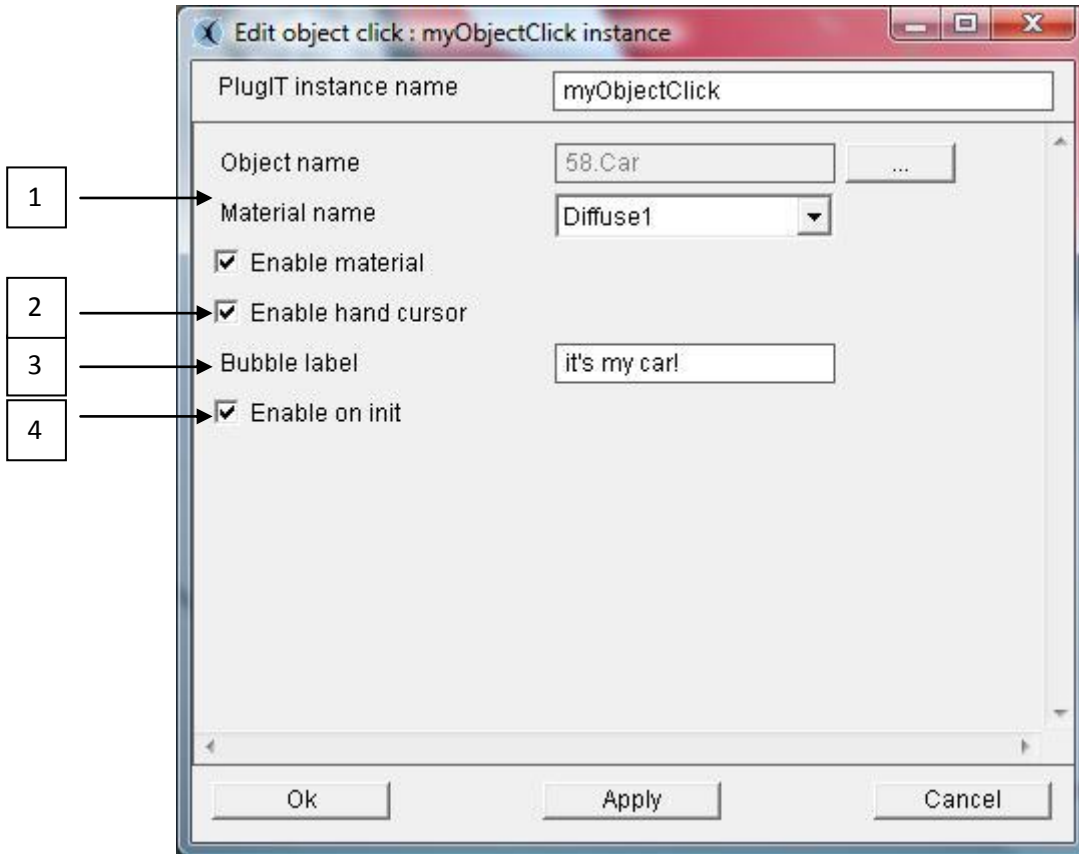
The instance has been added correctly, you must now edit the settings, to do so, double click on the instance, it will open the settings window of the instance.



The settings are blank, meaning that for now, the instance is applied to any resource. To select a resource (mesh / object name) of the group, click on the icon "Browse", browse the tree and click "OK" once the selected resource is highlighted.



The requested resource is "58.Car", here a single material is available. If we had an object with several materials, we could choose the one on which we want the function to apply from the "material name" area.



1 ° The "Link" function can be applied to an entire object or a defined material, in the latter case, you must check "Enable material".

2 ° The "Enable Hand cursor" permit to change the mouse cursor when it pass on the object and the selected material.

3 ° The "Bubble label" gives the possibility to have a flash bubble when passing on the selected object.

4 ° The "Enable on Init" give the possibility to activate the plugIT at the beginning of the application

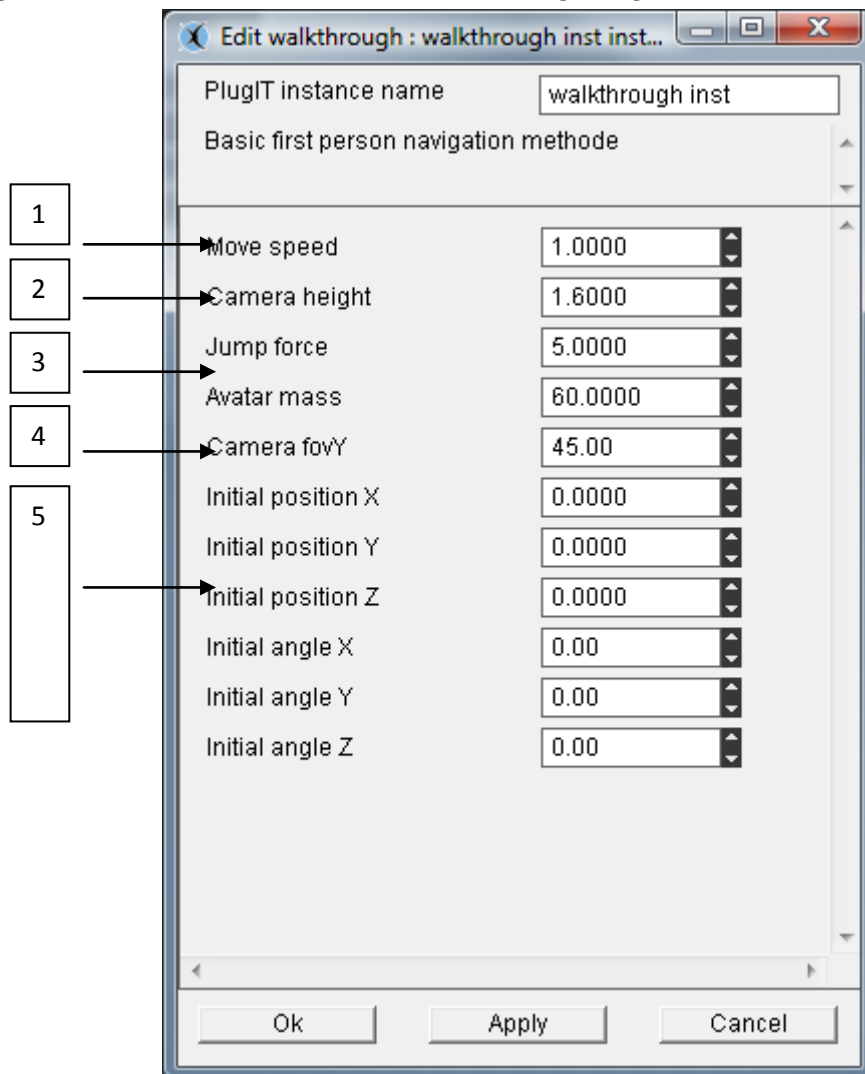
« Navigation » plugIT

PlugIT : “Walkthrough”

Walkthrough PlugIT allows you to activate the movement of the camera in first person mode.

This function should be added in the "Scene" group, as a single instance is enough and as no particular resources are needed.

Following the same method as before, add a Walkthrough PlugIT and edit the instance.



1 °/ The « Move Speed » parameter is used for the speed of displacement of the camera

2 °/ The « Camera height» parameter give the possibility to define the height of view for the camera.

3 °/ Physic parameters in the case physics are activated on our scene (« jump » : Straight of jump ("space" key), « avatar mass » : mass of the avatar)

4 °/ The « Camera fovY» parameter defines the focal angle of the camera .

5 °/ The « Initial position » parameters define the entering position of the viewer camera.

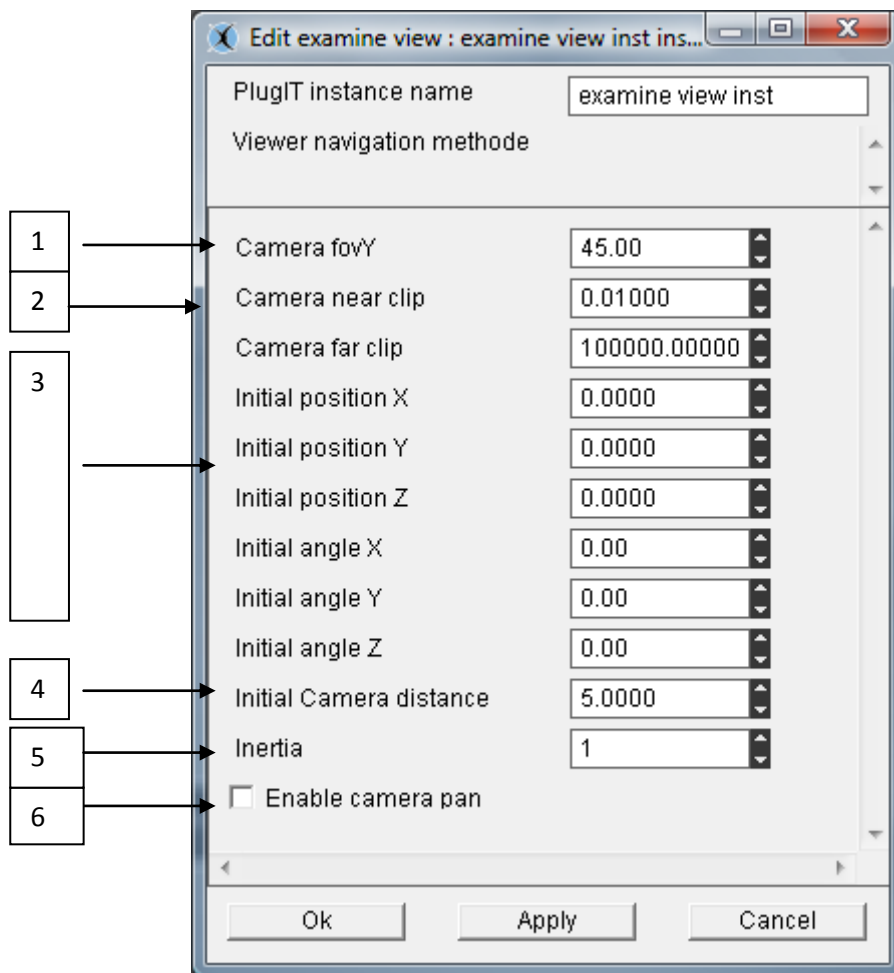
The « Initial position » parameters define the entering orientation of the viewer camera.

PlugIT : "Examine View "

The Viewer Navigation PlugIT allows to activate the navigation using the same ergonomics that when you are in edit mode.

In general this function should be added in the "Scene" as a single instance will suffice and that no resources are needed.

Following the same method as before, add an Examine View PlugIT, then edit the instance.



1 °/ The "Camera fovY" parameter sets the angle of the camera focal length.

2°/ The « Camera clip » defines the camera distance from view

3°/ The "initial position" parameters defines the position of entry into the scene YX Z.

The "Initial angle" parameters defines the direction of entry into the scene YX Z.

4 °/ The "Initial Remote Camera" parameter sets the default distance of the camera compared to the starting position.

5 °/ The « Inertie » parameter defines the inertia value when physics is activated

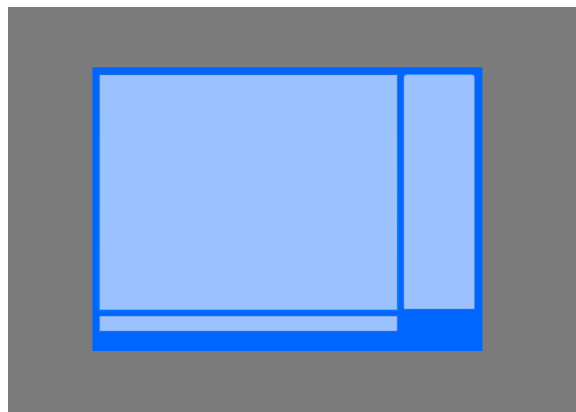
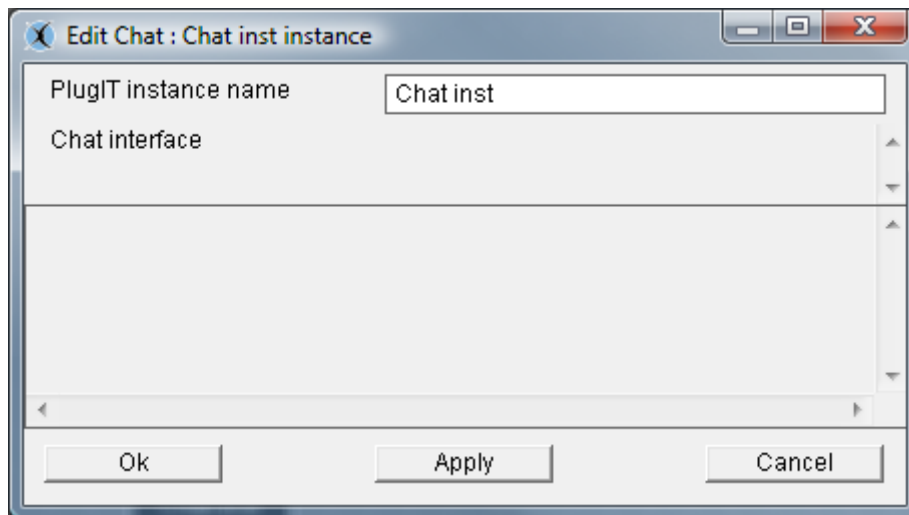
6°/ The " Enable camera pan " parameter enables or disables the translation of the camera

« Network » plugIT

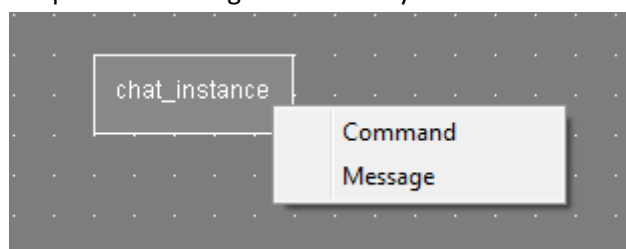
PlugIT: "chat"

(See example given with the OpenSpace3D installation named chat.xos)

This plugIT allows you to add a flash interface to manage a chat.

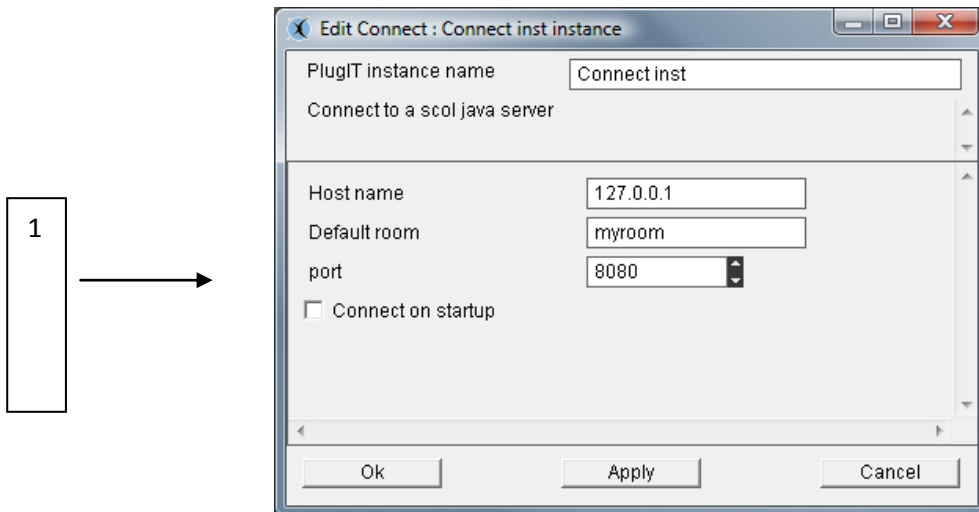


The associated events correspond to messages received by the flash or the commands sent to this one:



PlugIT: “connect”

This plugIT allows you to manage chat room for the java server in OpenSpace3D.



1 °/ « host name » : address of the server

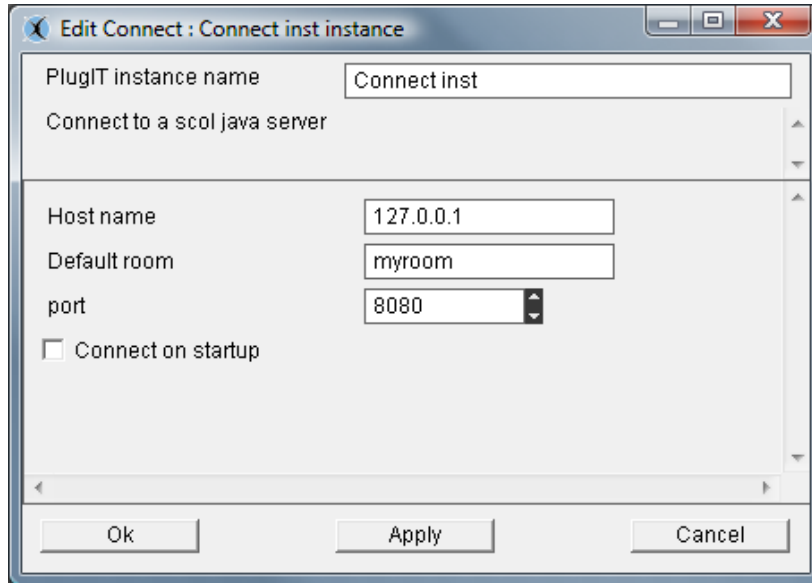
« Default room »: Name of the chat room

« Port »: associated server port

« Connect on Start up »: Automatic connection

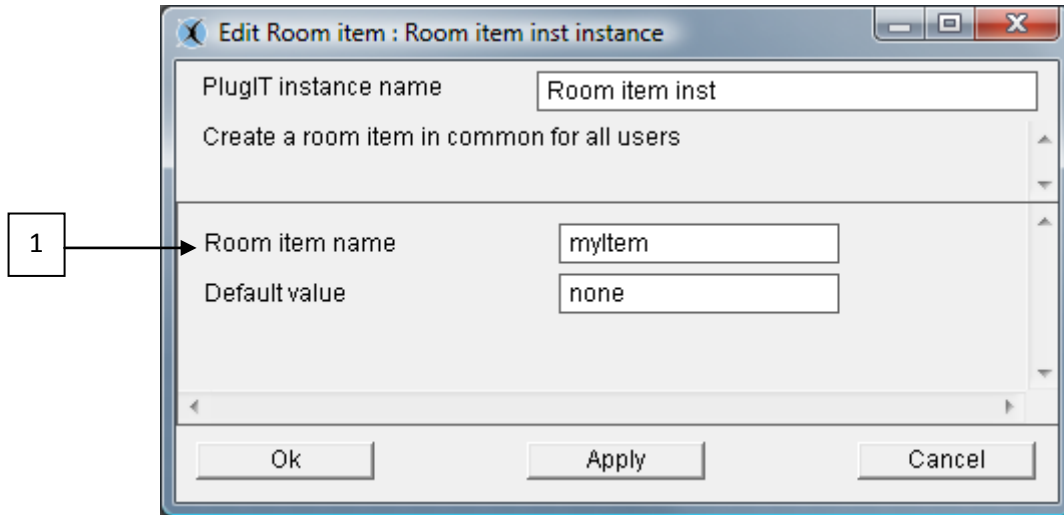
PlugIT: “message”

This PlugIT allows you to send a message.



PlugIT: "room item"

This plugIT allows you to create a room item

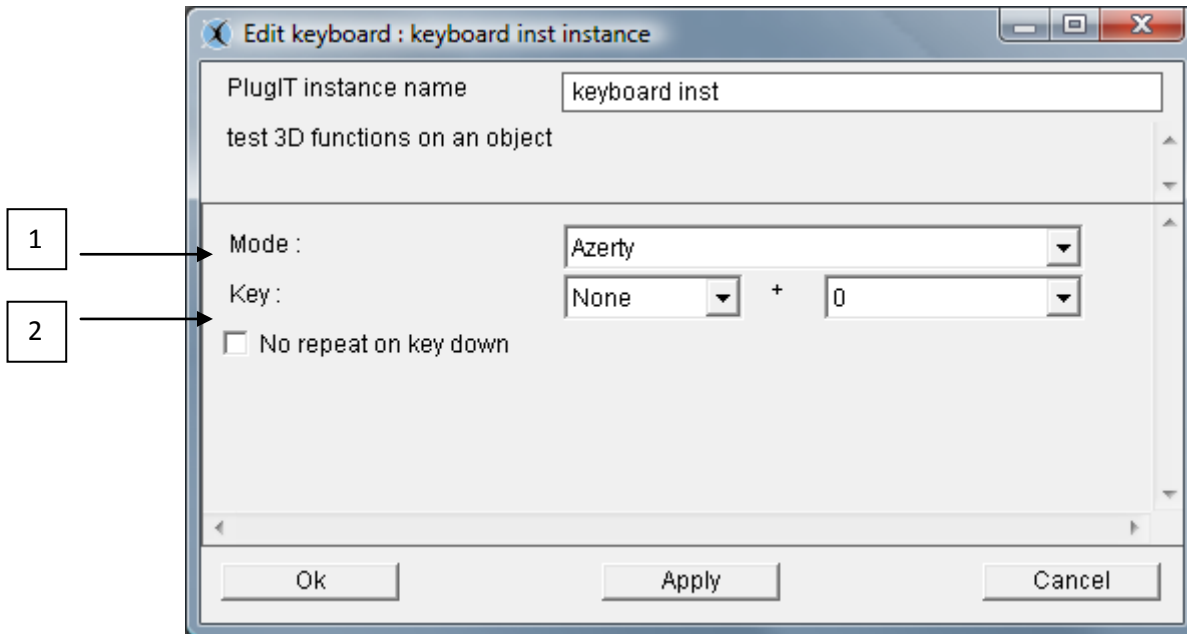


1 °/ Give the name for the Item and its value.

« Input » plugIT

PlugIT: “Keyboard”

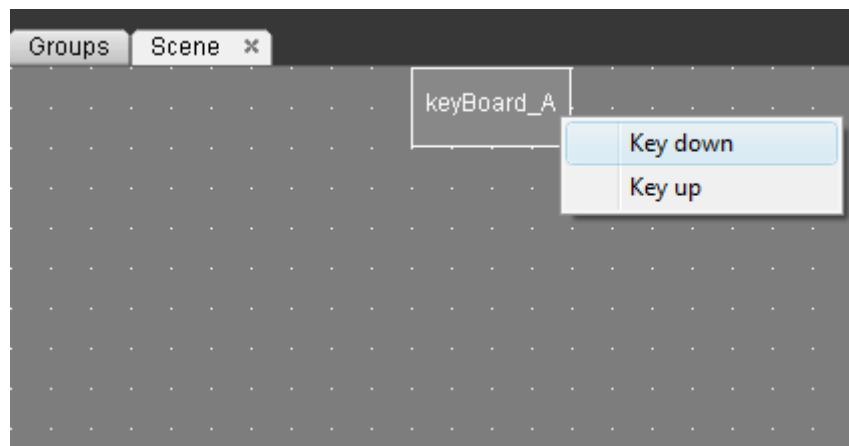
This PlugIT gives the possibility to define a key to send event when it is pressed.



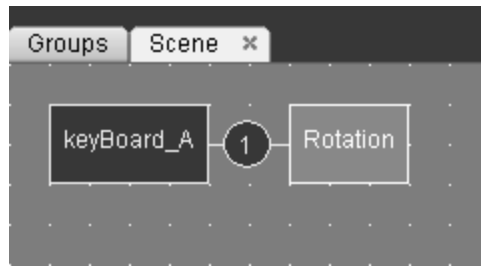
1 °/ to choose the type of keyboard used (AZERTY or QWERTY)

2 °/ the second parameter gives the possibility to choose a key. The first one permit to define the current state of these keys: CTRL, SHIFT, CTRL+SHIFT or NONE.

Exemple :



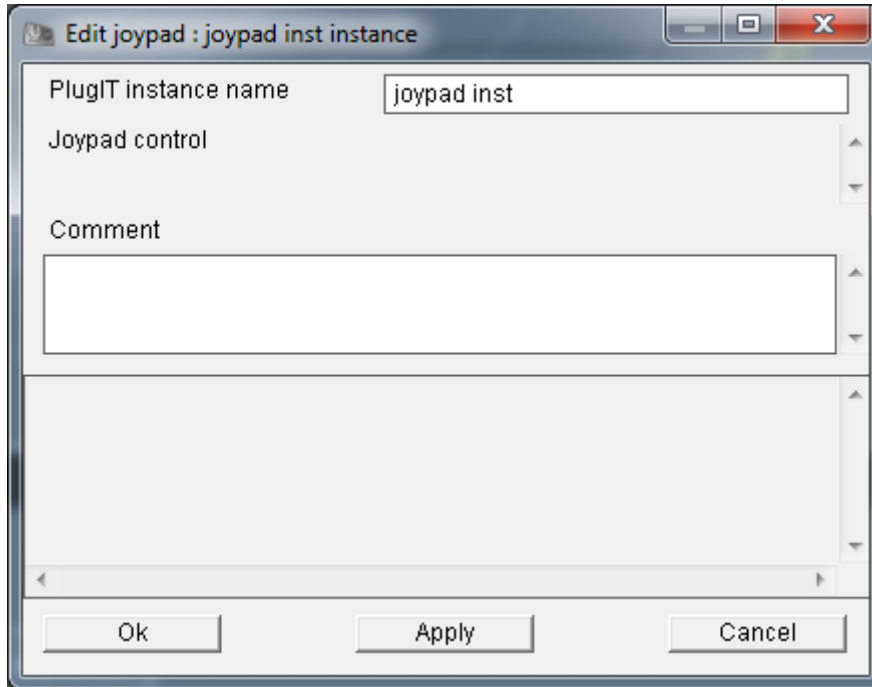
In events, we find our so parameterized keyboard, what allows us to create an action by a new link on another plugIT according to the state of the key.



Here, we run the rotate action when « A » is presses.

PlugIT Joypad

The joypad PlugIT lets you use a joypad controls to interact with the 3D application.

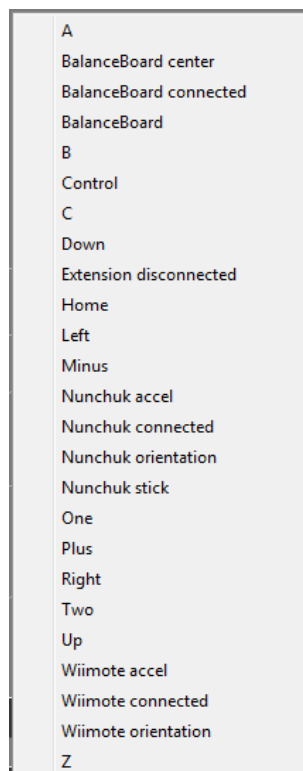
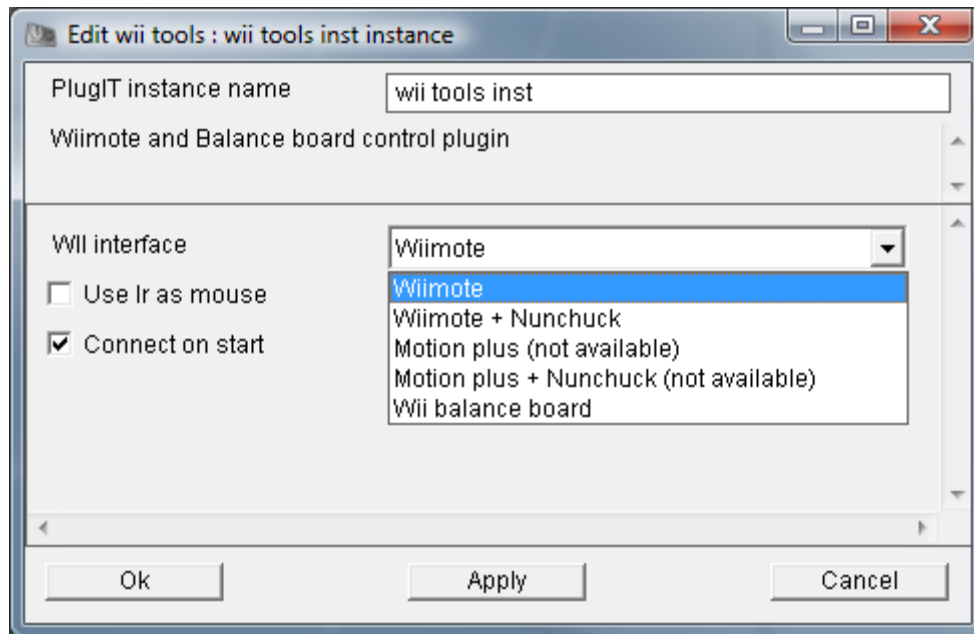


This PlugIT doesn't contain specific parameters, it's by connecting with another PlugIT that you can select the control to use in input and connected it to an output action



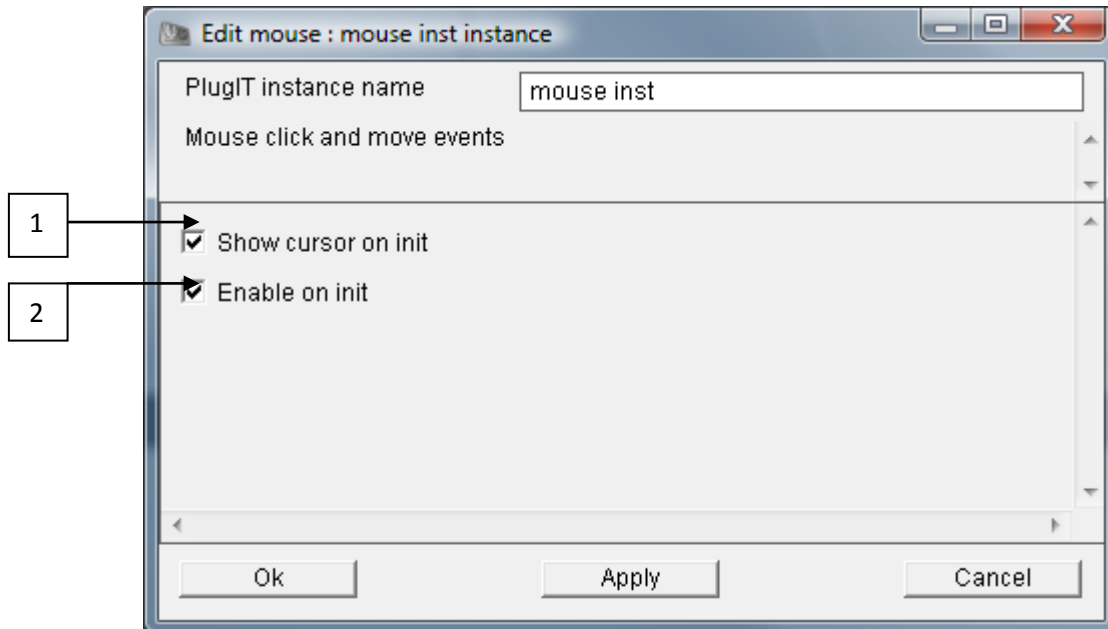
PlugIT : “Wii tools”

Wii tools PlugIT will allow you to configure interactions that will then let you use the wiimote or other wii accessories like nunchuck or balance board as as an input input with your application :



PlugIT : "Mouse"

The Mouse PlugIT lets you define the parameters of control mouse as show or hide cursor, enable or disable the control of the mouse in the application



1° / It defined here if the cursor is visible on init

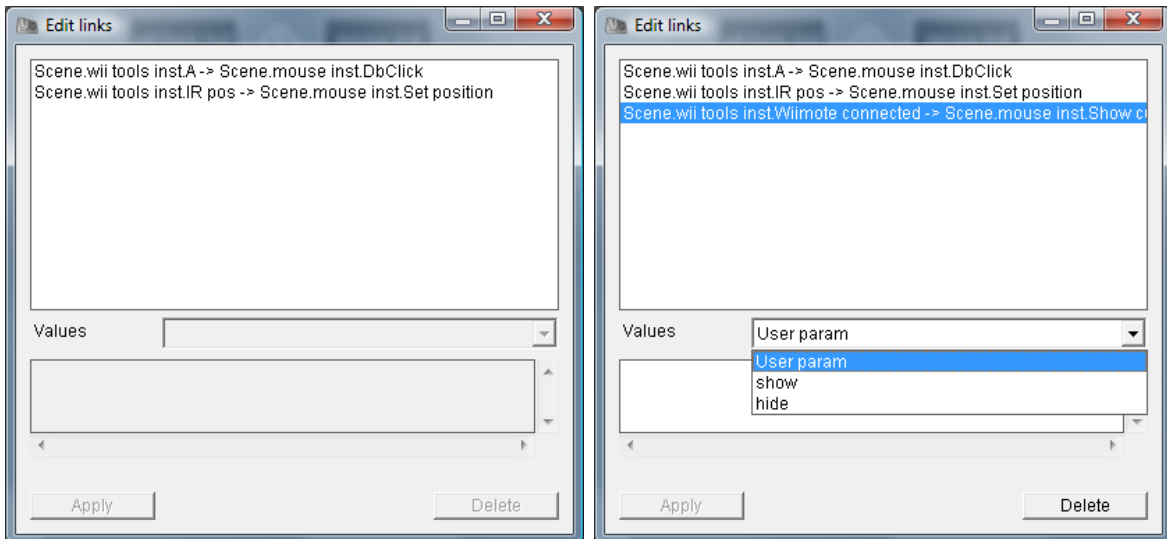
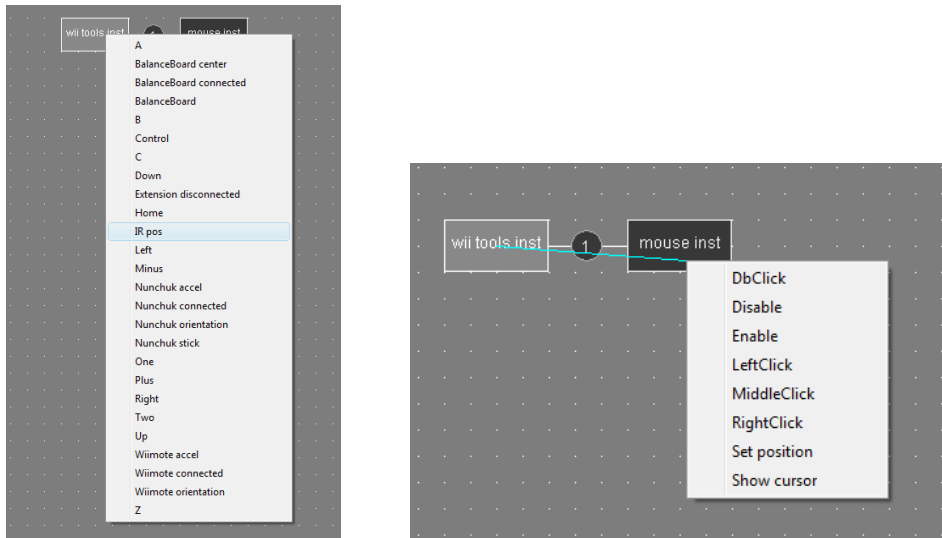
2° / It defined here if the mouse control is active on init

List of available controls on output of the Mouse plugIT :



Example of use of the Mouse plugIT with the wii tools plugIT:

Here we've defined the wiimote as the man-machine interface of the application with a mouse behavior. For each control of the wiimote, you have to create a link from the Mouse plugIT to the wii tools plugIT for example the A key of the wiimote is the double click of a mouse and the position of the Infrared (IR pos) corresponds to the position of the cursor of a mouse.

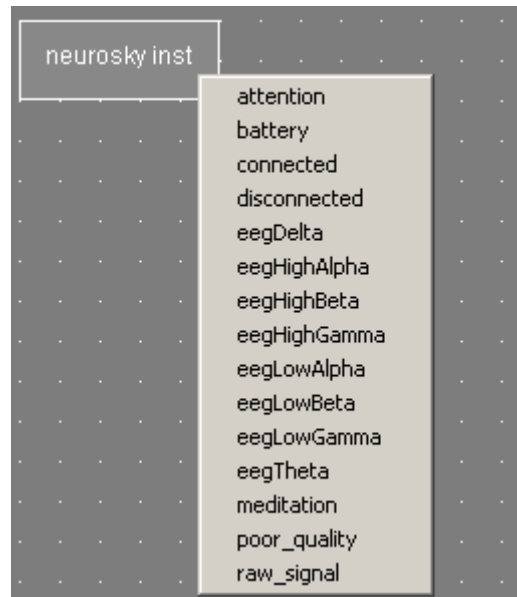


To show the cursor (where the wiimote is connected) is defined so that it displays when editing the settings of the link

PlugIT NeuroSky :

The neurosky's mindset is a BCI (Brain computer interface), it can be used in any applications OpenSpace3D.

More informations : <http://www.neurosky.com/>



Here are the events that can return the plug:

Attention: Focus of the user (0 to 100)

Meditation: meditation of the BCI user (0 to 100)

Battery: Battery level of the mindset

Connected: Send an event when the neurosky is connected

Disconnected: Send an event when the neurosky is disconnected

The EEG : <http://en.wikipedia.org/wiki/Electroencephalography>

EegDelta: Delta frequency of the brain

EegHighAlpha: high alpha frequency of the brain

EegHighBeta: high beta frequency of the brain

EegHighGamma: high gamma frequency of the brain

EegLowAlpha: low alpha frequency of the brain

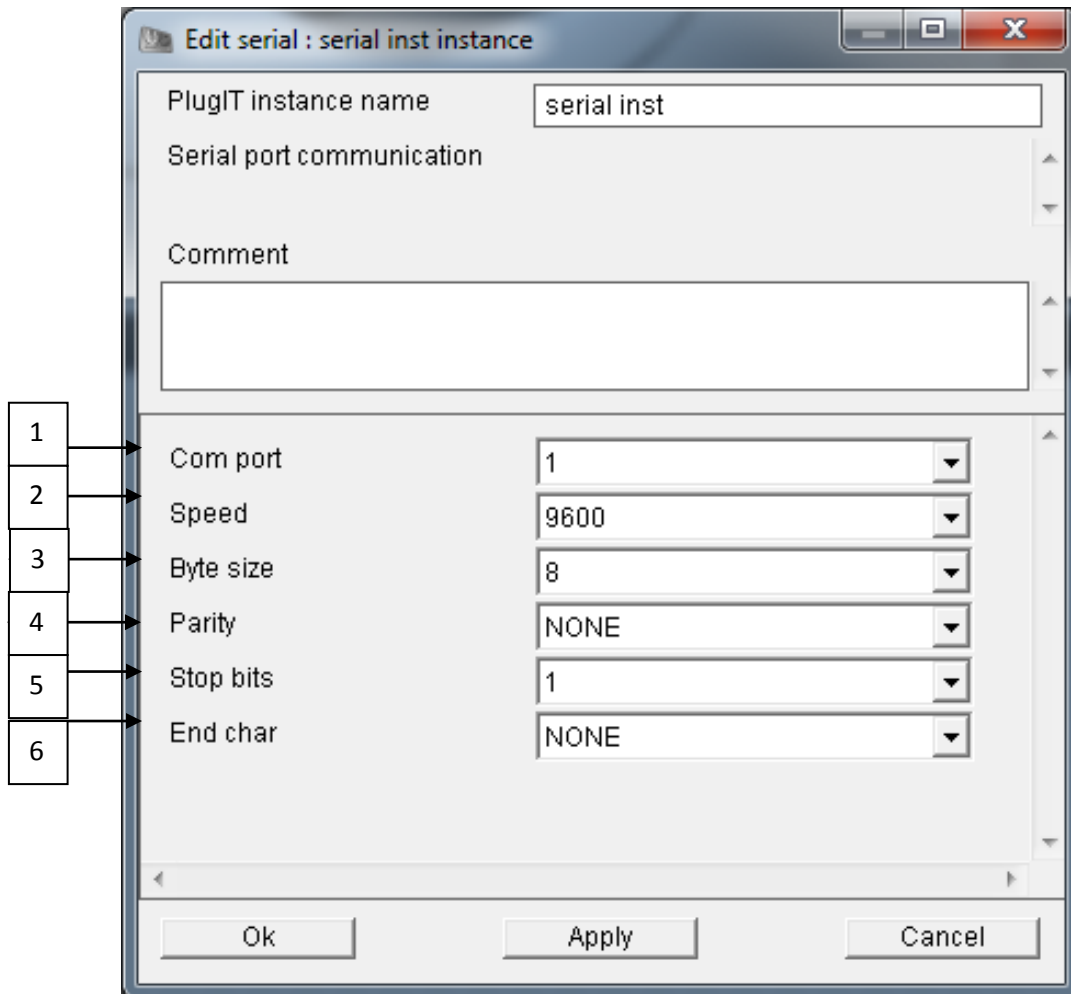
EegLowBeta: low beta frequency of the brain

EegTheta: Theta frequency of the brain

Poor_quality: (0 to 200) Give the signal quality, if value is 0 the mindset is well positioned if the value is 200 the mindset can't receive any signal

Raw_signal : Raw signal from the mindset

PlugIT Serial



The Serial lets you communicate your OpenSpace3D application with a serial communications port

1° / Number of serial port that you connect your device

2° / Speed data transfer

3° / Byte size

4° / Parity control

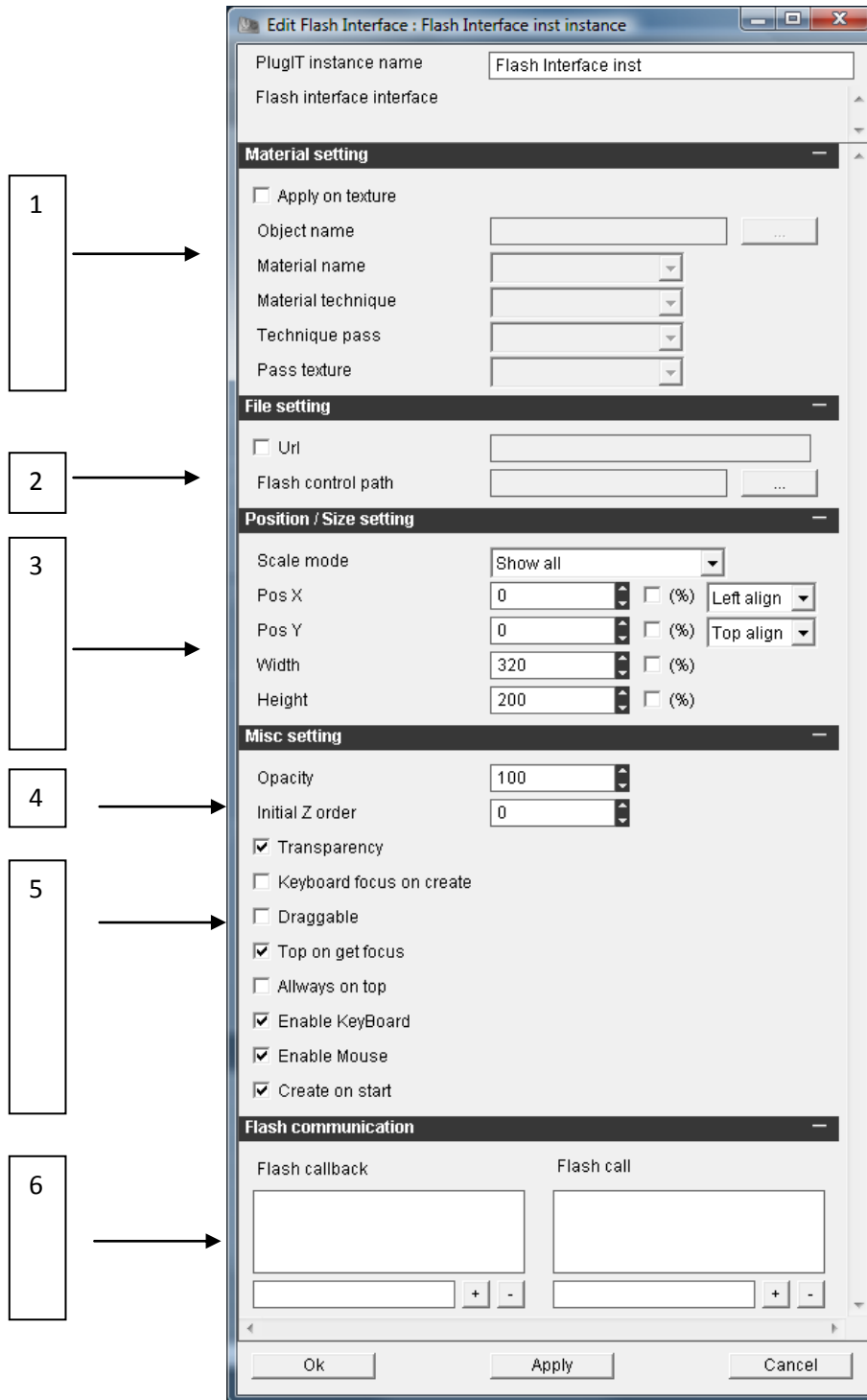
5° / Stop Bits

6° / Character end of message

« Interface » plugIT

PlugIT : « flash Interface »

This useful plugIT gives the possibility to add a flash interface in your application .We could add it on a 3D object or on the viewport directly.





1 °/ If the option « apply on texture » is selected, it means we want our plugIT will be applied on the material of a 3D object. So, you have to define the object name and the material name.

2 °/ To Choose if you want to load a flash control using url or using a local path.

3 °/ Position and scale parameters for the flash control.

4 °/ « z Order » parameter. It will be useful only in the case of 2D interfaces to control the order of display.

5 °/ Parameters to configure options for the display of the flash control.

« Transparency »: Transparency of the flash control

« Keyboard on create »: Give the keyboard focus directly

« Drag gable »: Allow the displacement of the flash control using the mouse right click

« Top on Focus »: Give the possibility for the flash control to pass on all the others.

« Always on Top »: The flash control will be always on top

« Enable Keyboard » gives the possibility to use the keyboard events on flash control

« Enable Mouse » gives the possibility to use the mouse events on flash control

« Create on start »: Creating the flash control at start

6 ° / this part allow the flash control to communicate with your application. Depend of your action script (see example of the chat)

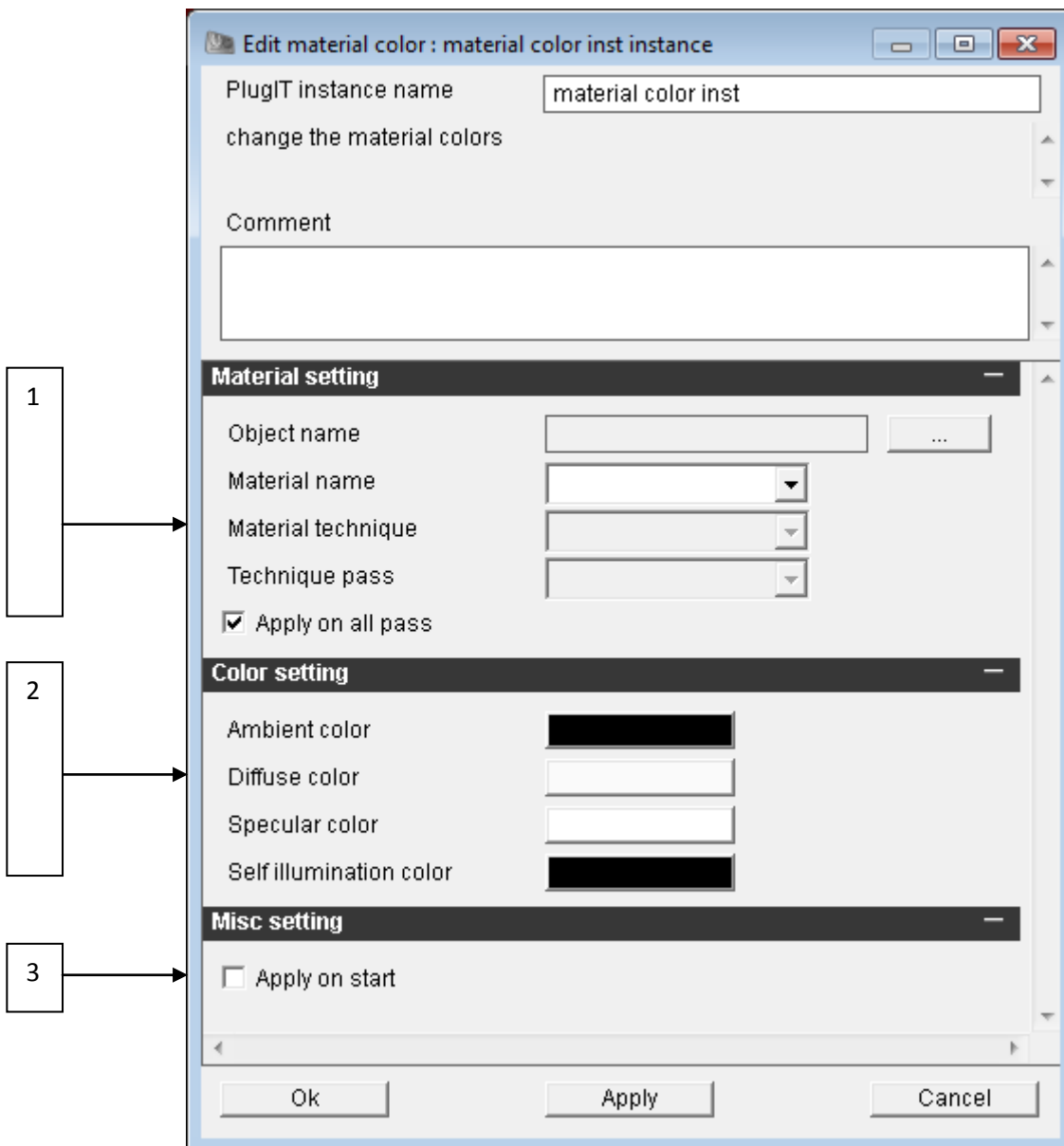
“Flash call-back to get a value from flash to plugIT

« Flash Call » Send value from plugIT to flash.

« Material » plugIT

PlugIT : « Material color »

Material colour PlugIT allows you to modify the colours of a given material.



1 °/Click on « ... » to select the object and the material.

It is also possible to choose the color on a given pass (the application on all passes is enabled by default)

2 °/Ambient colour is the ambient colour which will be applied when the plugIT is activated.

Diffuse colour is the diffuse colour which will be applied when the plugIT is activated.

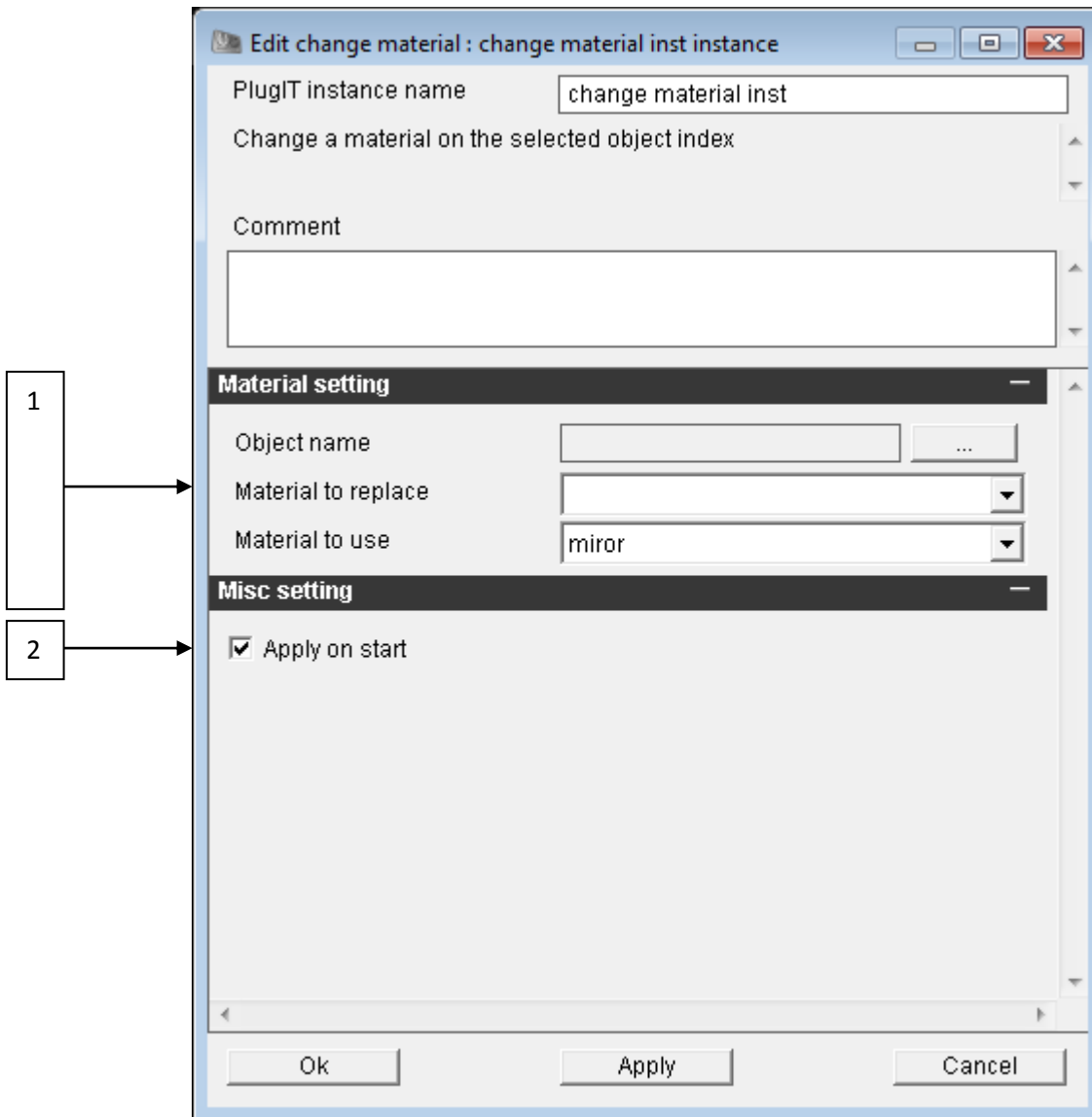
Specular colour is the specular colour which will be applied when the plugIT is activated.

Self illumination colour is the Self illumination colour which will be applied when the plugIT is activated.

3 °/ Enabling change color when you start the application or if unchecked to activate it using an action.

PlugIT change Material

This plugIT allows changing a material on a given object

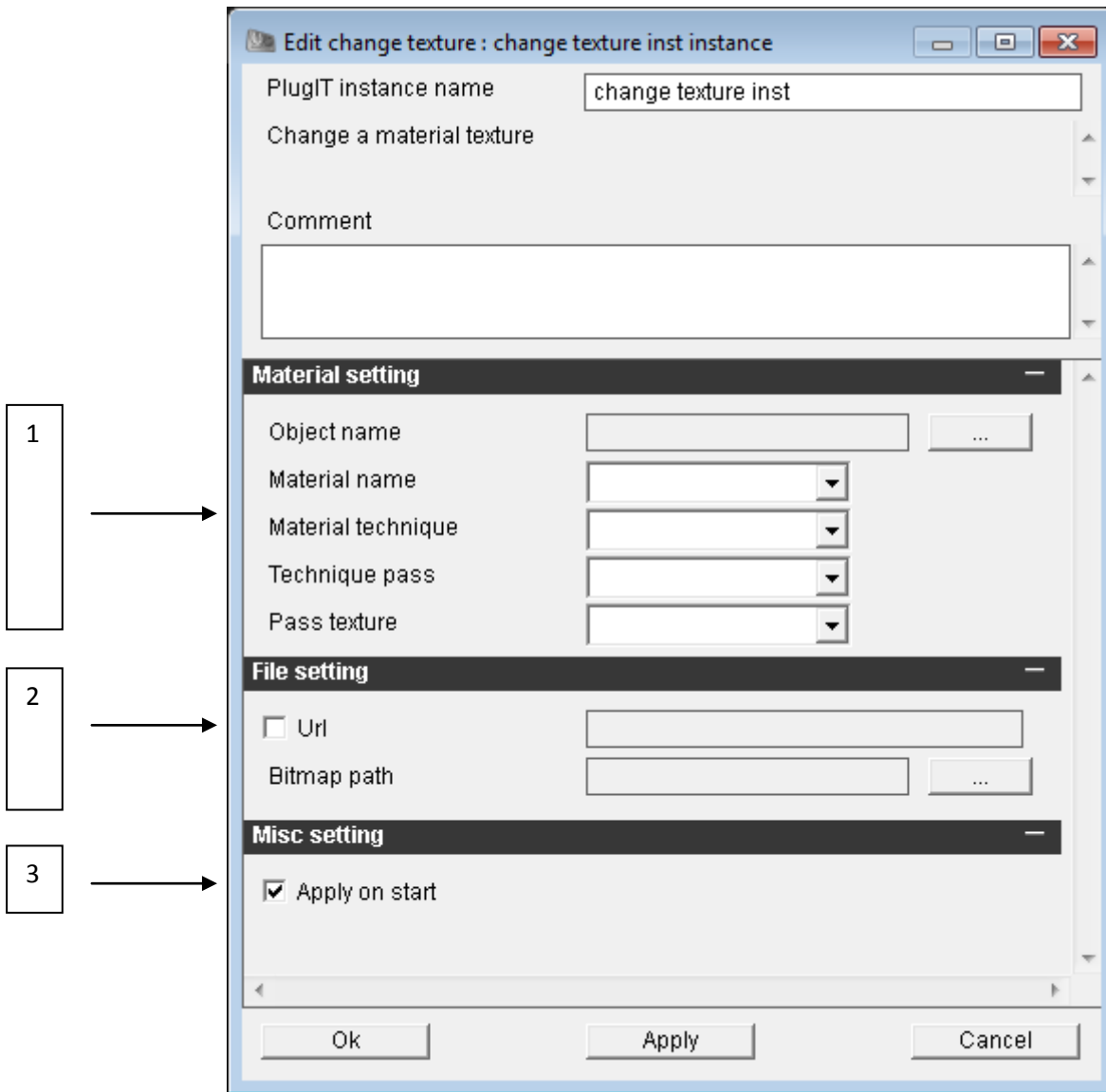


1 ° / the 3D object selected, the current material to change and the material to use

2 °/ We choose here if we want the application of the new texture at launch.

PlugIT : « change texture »

To change a Texture on a material.



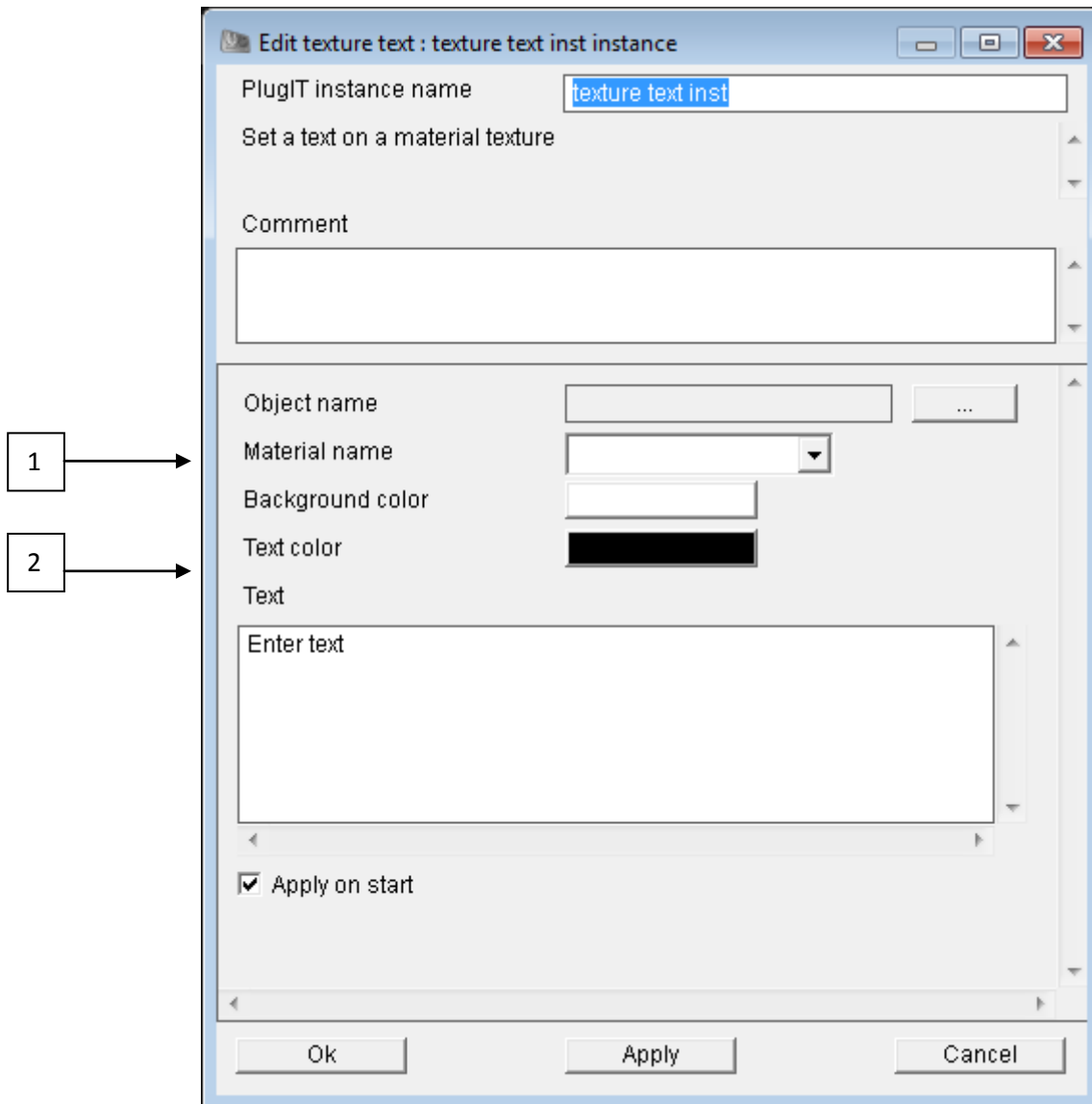
1 °/ the 3D object selected and its material. Provides on which technique, what which and which texture of the material; we should apply the new texture

2 °/ Loading a new texture from an URL or from a local path.

3 °/ We choose here if we want the application of the new texture at launch.

PlugIT: "texture text"

To write text on a texture



1 °/ the 3D object selected and its material.

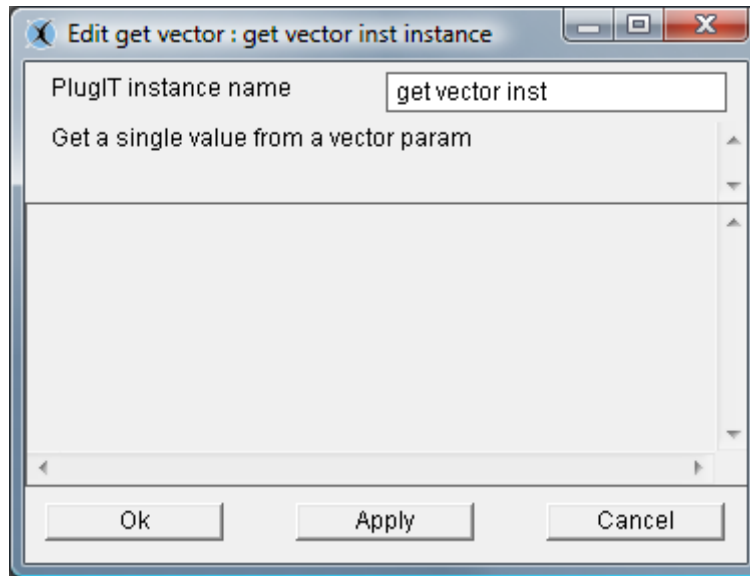
2 °/ Some options concerning the text to be applied:

- Color of the font
- Color of the background
- The text we want to be applied

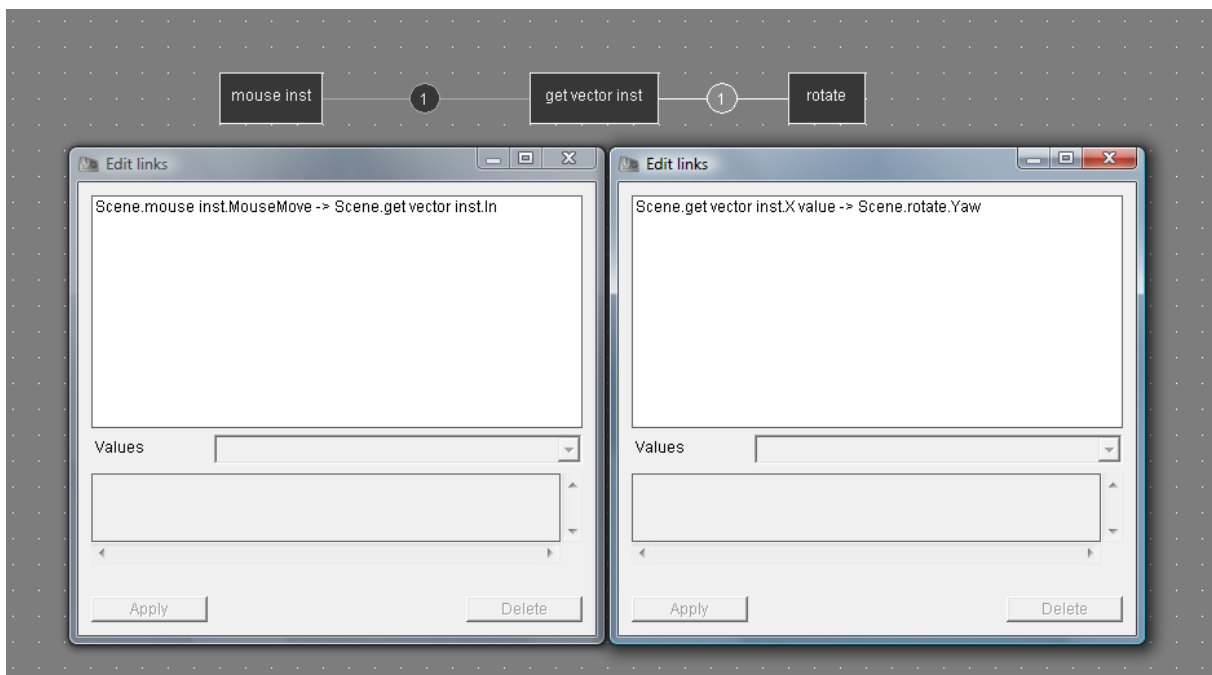
« Maths » plugITs

PlugIT Get vector

This PlugIT to send one of the three value of a vector (x,y or z).

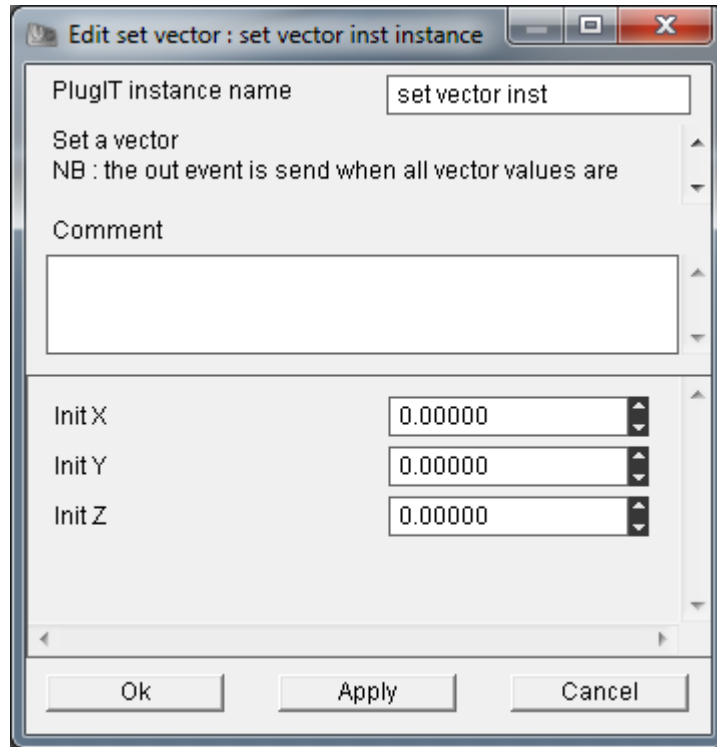


Example :



In this example, the value of X mouse movement is retrieved by the Get Vector plugIT which then sends it to change the value Yaw on the plugIT Rotate

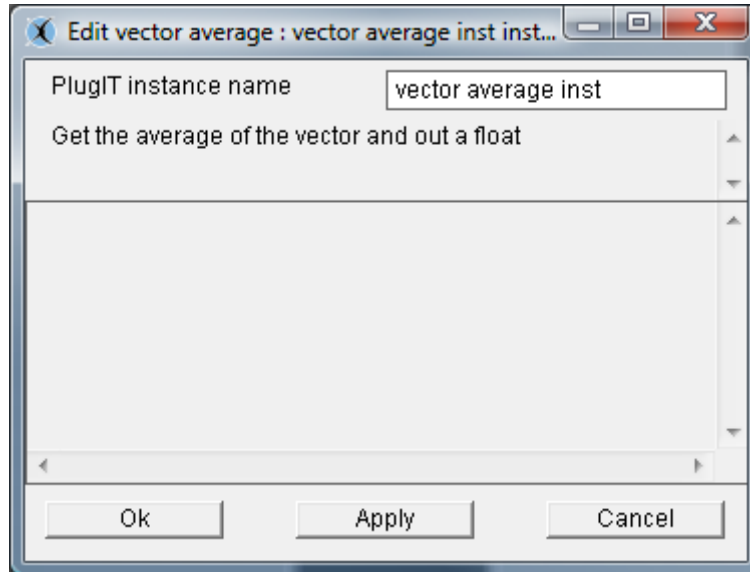
PlugIT Set Vector



The Set Vector PlugIT is used to create or modify the value of a vector according to the value of the input vector and the new value of this vector can then be used as output

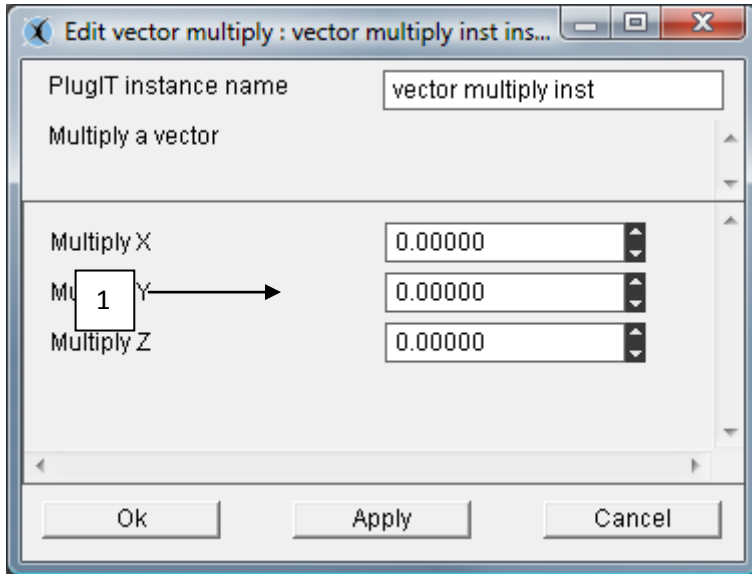
PlugIT: “vector average”

To calculate the average result for a given vector. Output is the value.



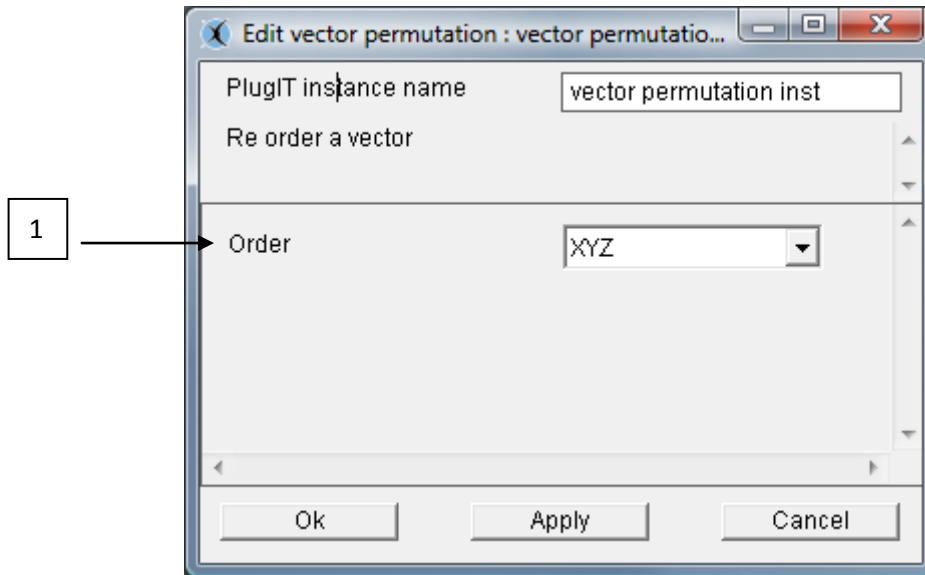
PlugIT vector multiply

To calculate the multiplication of two given vectors. Output is the value.



PlugIT : “vector permutation”

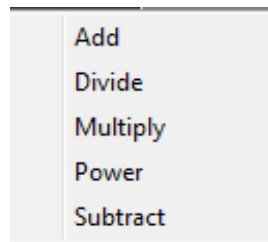
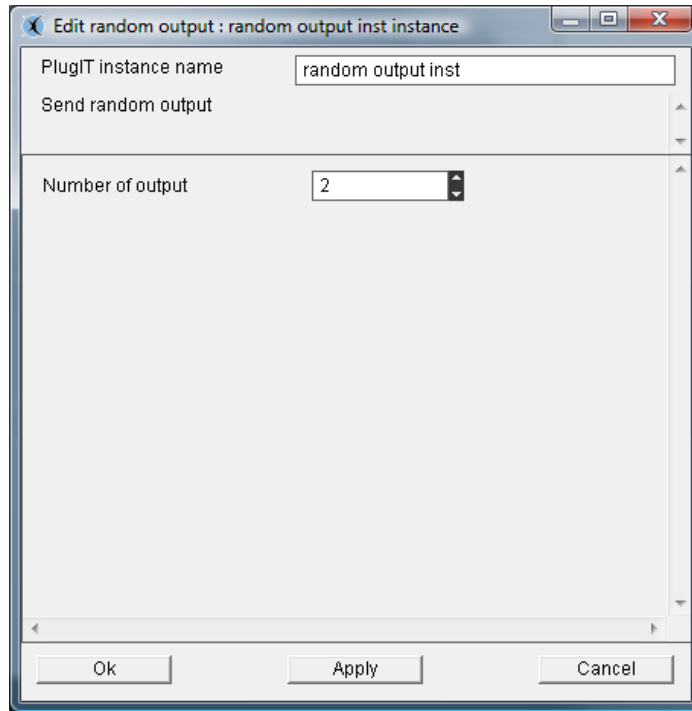
This plugIT allows you to make some transformations on the order of vector’s values depending of the order options.



1 °/ Order Options

PlugIT operator

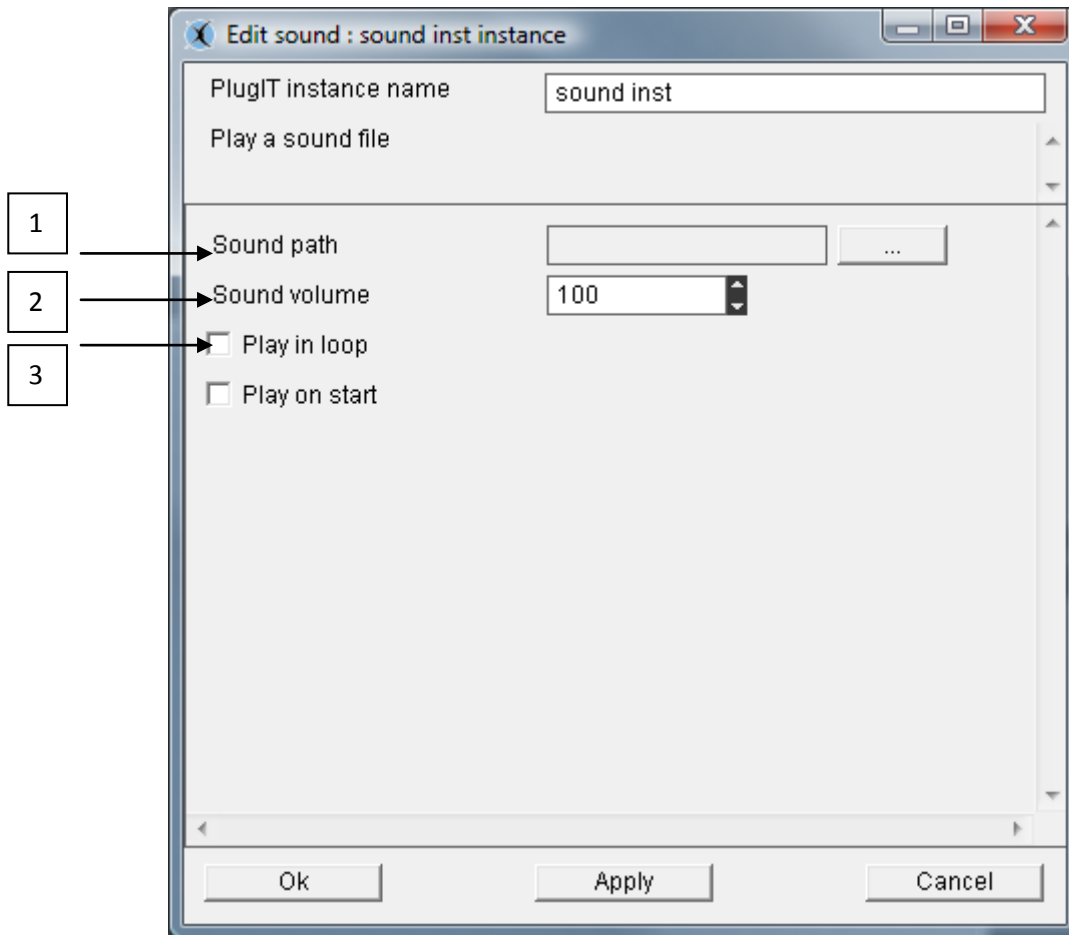
Le PlugIT operator lets you perform a calculation on a value: multiplication, division, subtraction, addition.



« Media » plugIT

PlugIT: "Sound"

This one allows triggering a sound at the starting of the application or during its play.



1 °/Select the sound file that will be played in your Scol partition by clicking the button browse "...". The MP3 and WAV formats are supported.

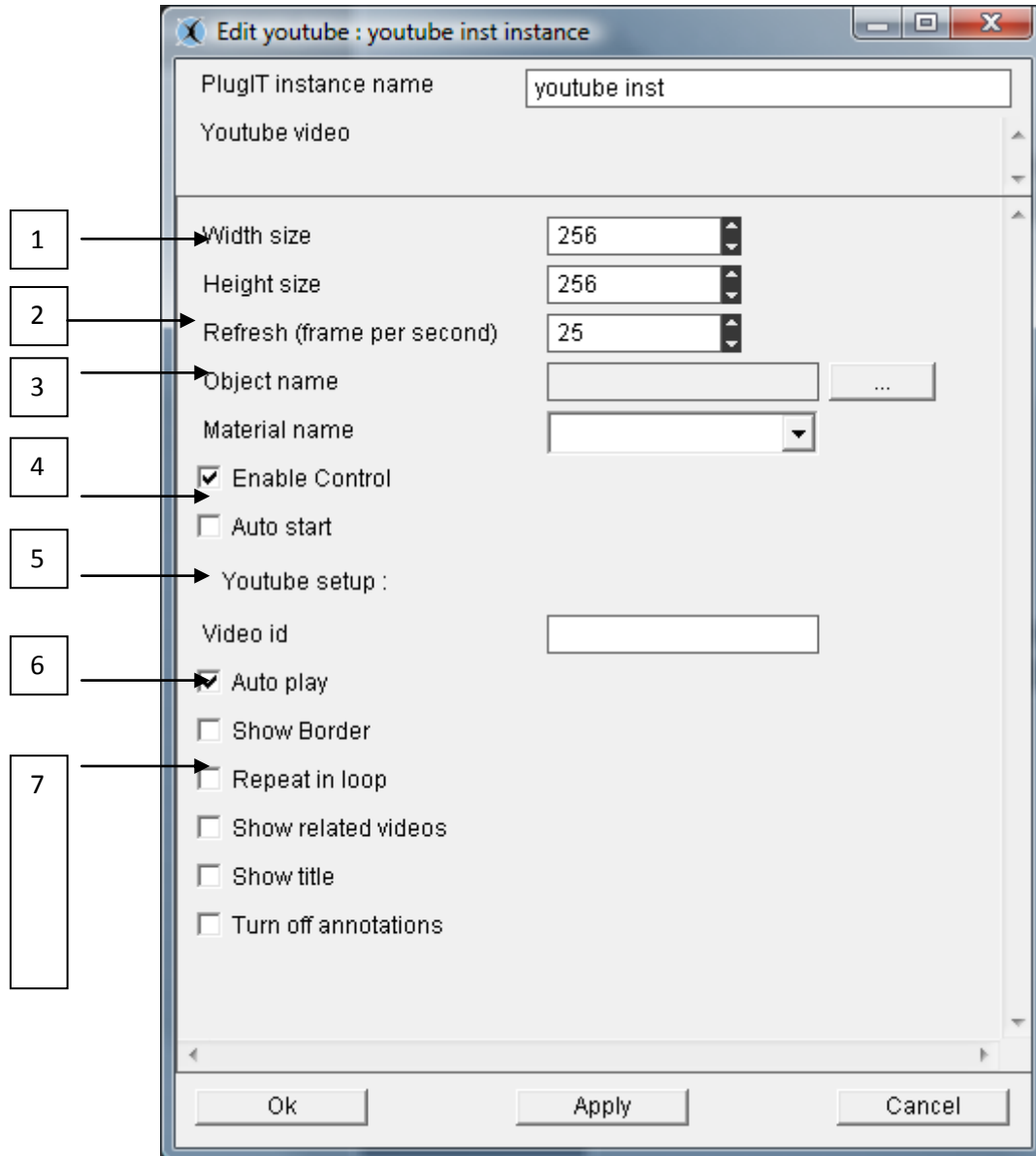
2 °/the "Play in loop" parameter determines whether the sound will be played in loop.

3 °/The "Play on start" parameter is used to define whether the sound will start automatically when you launch the application.

PlugIT: "Flash"

The Flash PlugIT lets you view Flash content on a 3D object.

Following the same method as before to add a function,
 Right click in a blank area: media, select "Flash",



1 / « Width size » and « Height size » to define the size for the video texture applied on the material.

2 / « Refresh » defines the maximum number of frame per second for the display of the flash.

3 / Click « ... » to select an object and a material.

4 / « Enable Control » to activate or deactivate the mouse control on the flash.



5 °/ « Auto start » allows you to run the flash content when the application begins to run.

6 °/ « Video id » YouTube options.

7 °/ « Show border » video rendering option.

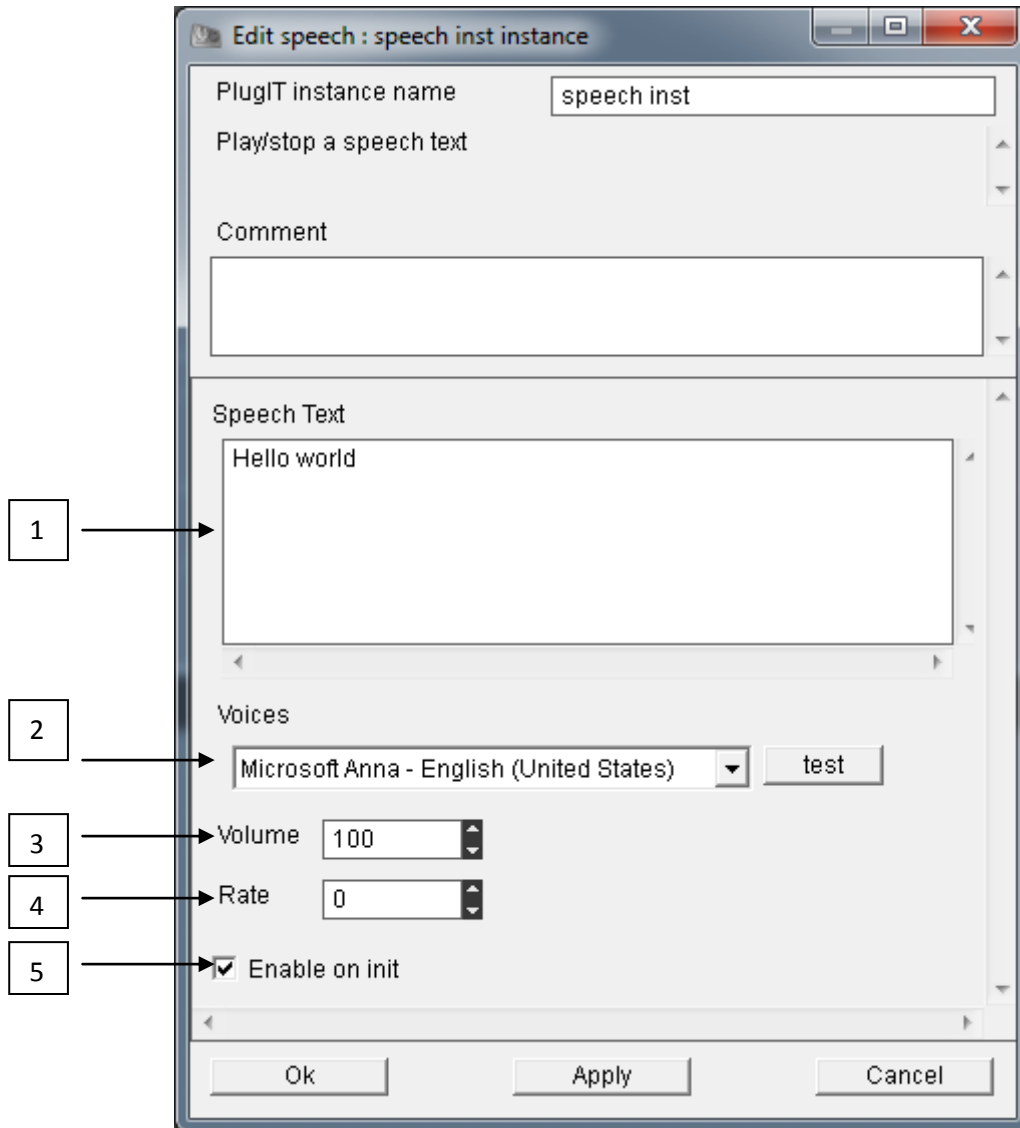
« Repeat in loop » to play in loop

« Show related videos » YouTube options.

« Show title » displays the title of the video.

PlugIT Speech

The Speech PlugIT allows you to use the Windows API Text-to-Speech to generate a voice that will be based on the associated event.



1°/ Box of the text to be read by the speech

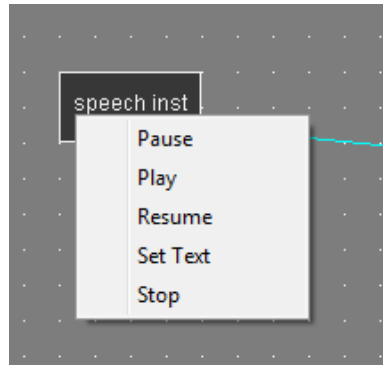
2°/ Selection of voice to use to read the text among those that are installed on the computer

3°/ Volume reading text

4°/ Reading rate or speed at which the text should be read

5°/ Check this box if you want this text to be read at the initialization stage.

The input links of the Speech PlugIT allows you to link an output events to trigger an action on the Speech

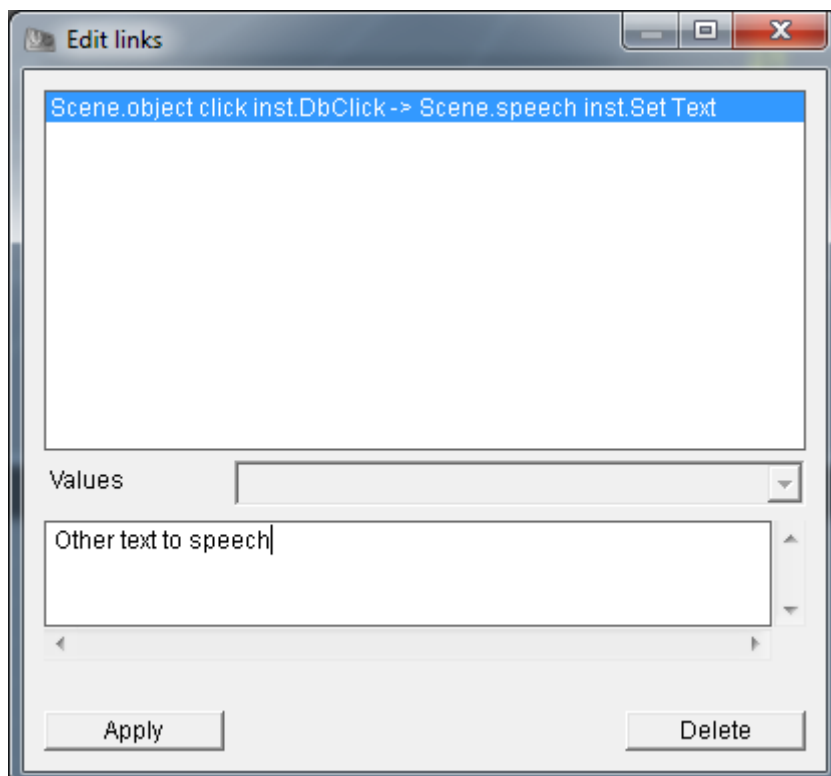


1°/ Pause will pause on reading the text

2°/ Play will start reading the text

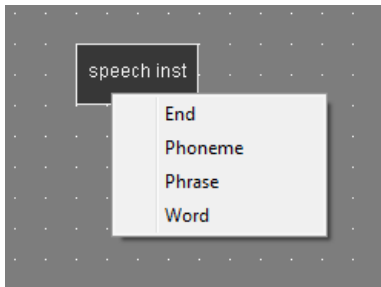
3°/ Resume : resume playback of the text where it was paused

4°/ Set text change the contents of text from another text link parameter "set text"



5°/ Stop to stop reading the text.

The output links of the Speech PlugIT allows you to link events of this plugIT to trigger action to another PlugIT.



example such a text like : Hello I am

1 / End recognizes the end of the reading text

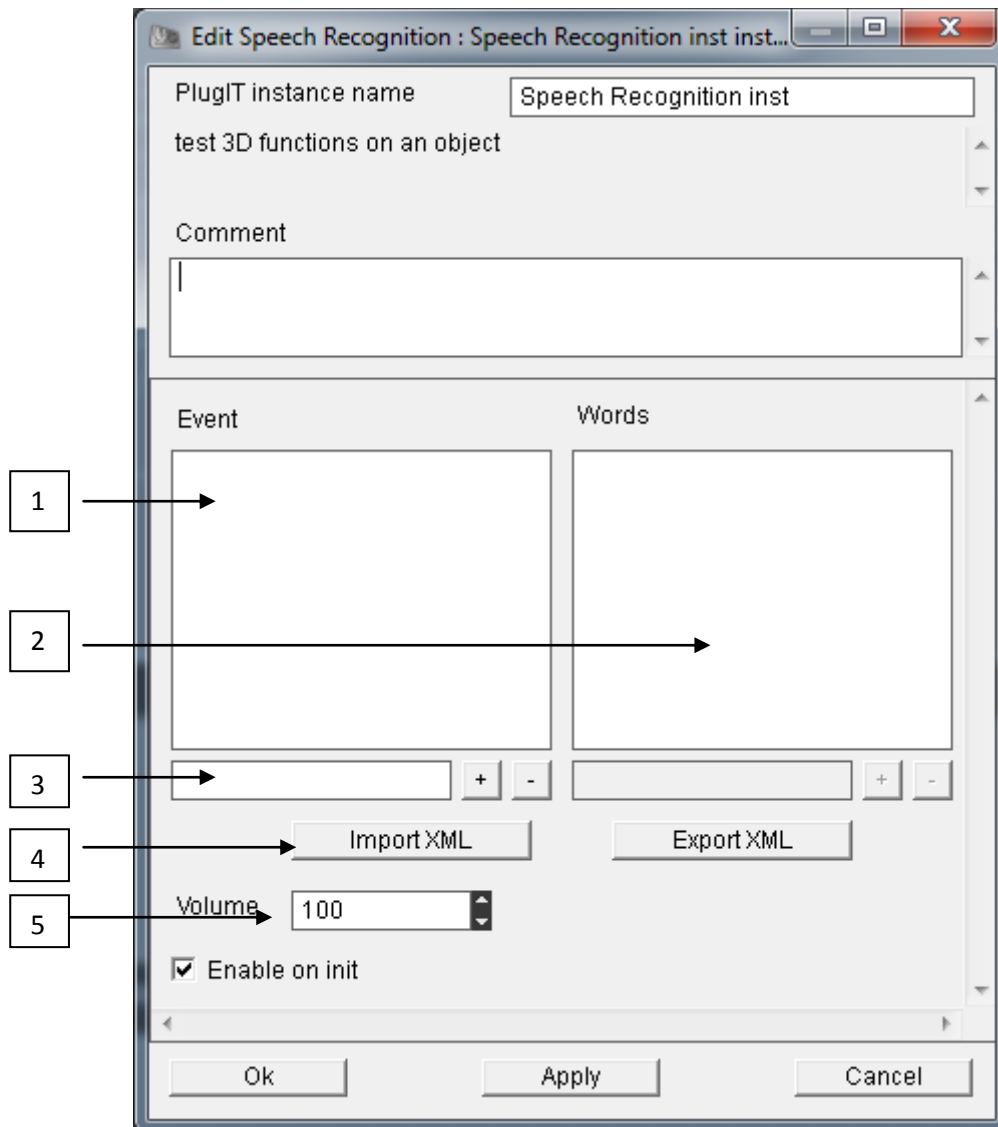
2 / Phoneme recognize a syllable as "good", "day", "I", "am", "the"

3 / phrase "hello I am"

4 / Word "hello"

PlugIT Speech Recognition

The Speech Recognition PlugIT allows you to use speech recognition to interact with the application to trigger such an action on a plugIT when a word is detected



1°/ Event are events of the PlugIT

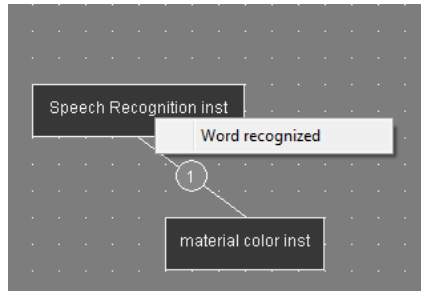
2°/ Words correspond to recognized words to be associated with events

3°/ new events or words to recognize

4°/ Import/Export .xml file

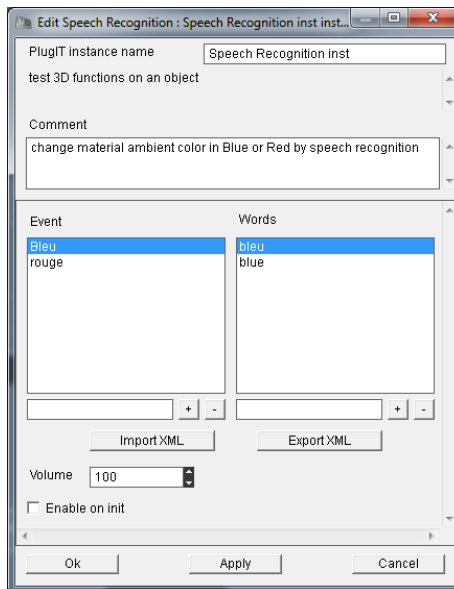
5°/ Volume

All events created in the plugIT Speech recognition is then available at the output of the PlugIT



Example of integration of Speech Recognition PlugIT to change the ambient color of a material where the recognized word corresponds to a color in the events of PlugIT:

The Blue and Red Events have been created in the interface of Speech Recognition PlugIT to each of them the English and French words to be recognized were associated with the respective events



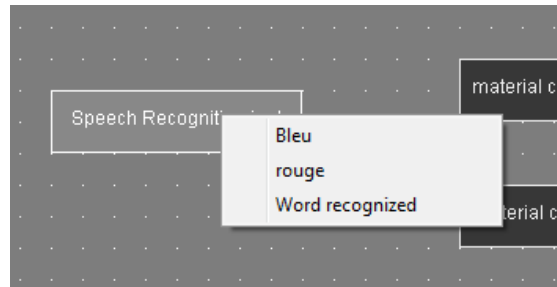
Here is the structure of the xml file that can be exported:

```

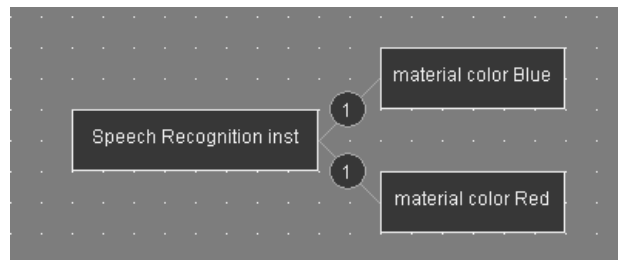
- <wordlist>
- <event value="Bleu">
  <word value="bleu" />
  <word value="blue" />
  </event>
- <event value="rouge">
  <word value="red" />
  <word value="rouge" />
  </event>
</wordlist>

```

Events available at the output of Speech Recognition PlugIT are:



The PlugITs to change the color of the ambient material have been added and linked to the events of the Speech recognition PlugIT :

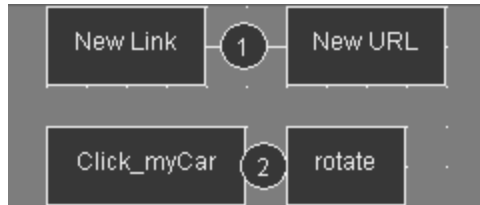


Now when the user decide for example the word "blue" the ambient material will be changed to blue

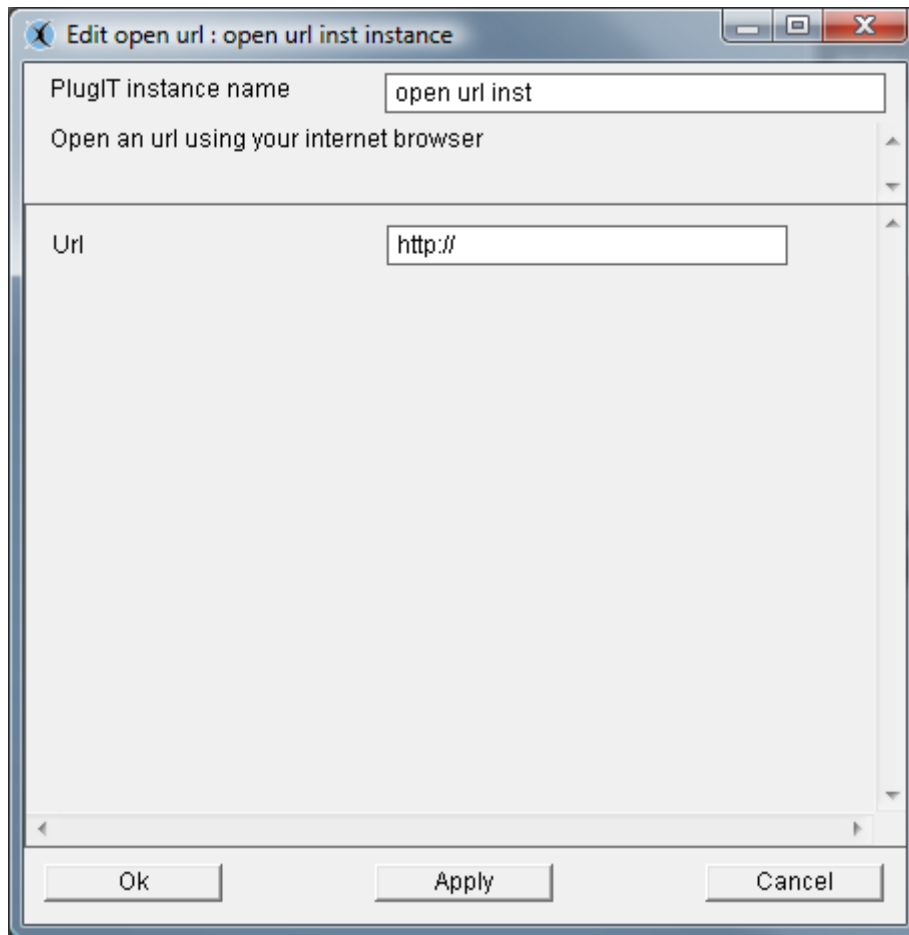
« Misc » plugITs

PlugIT : « Open Url »

The Open Url PlugIT allows to open an url (web page)



Double-click the function (new URL) to edit the settings.



The « Url » parameter allows you to enter the url that will open.

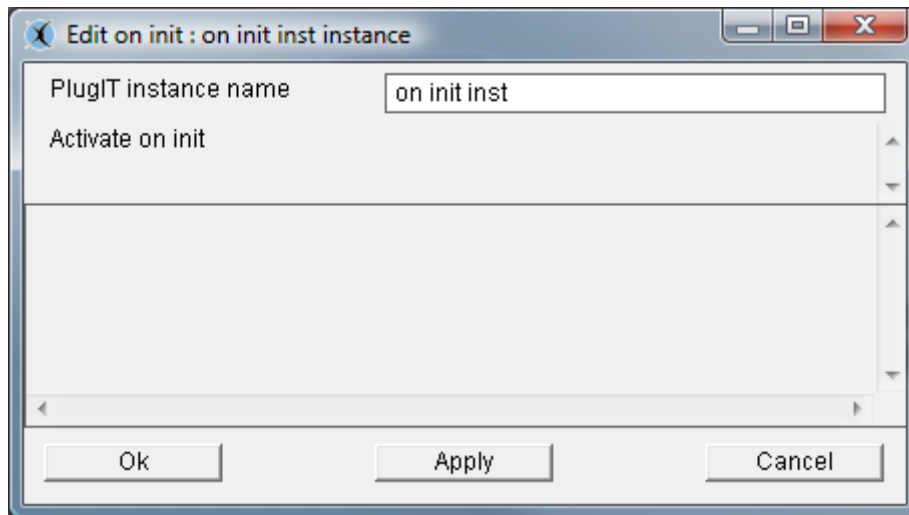
PlugIT: "On init"

It is used to trigger an event when the entire scene is set.

Following the same method as before, add an On Init PlugIT, and then edit the instance.

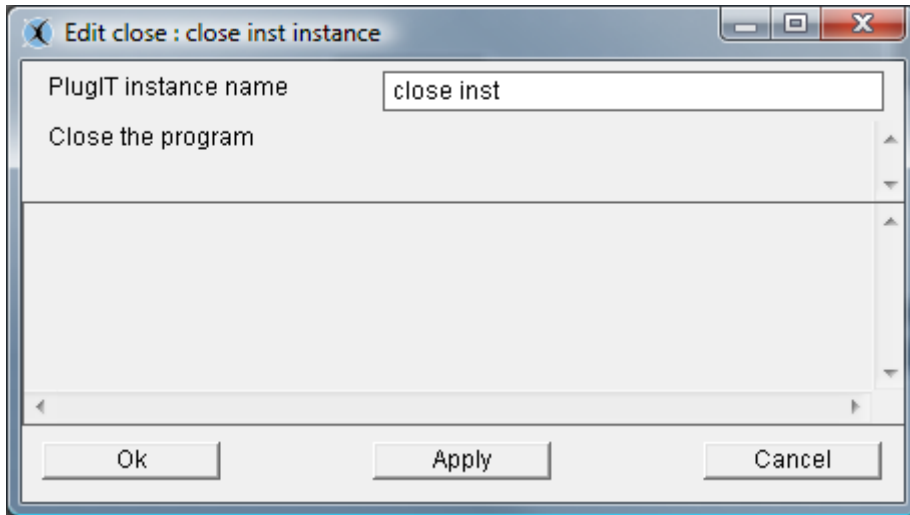
This function has no parameters to edit.

It will be used with most other PlugIT to perform an action when launching the application.



PlugIT: “Close”

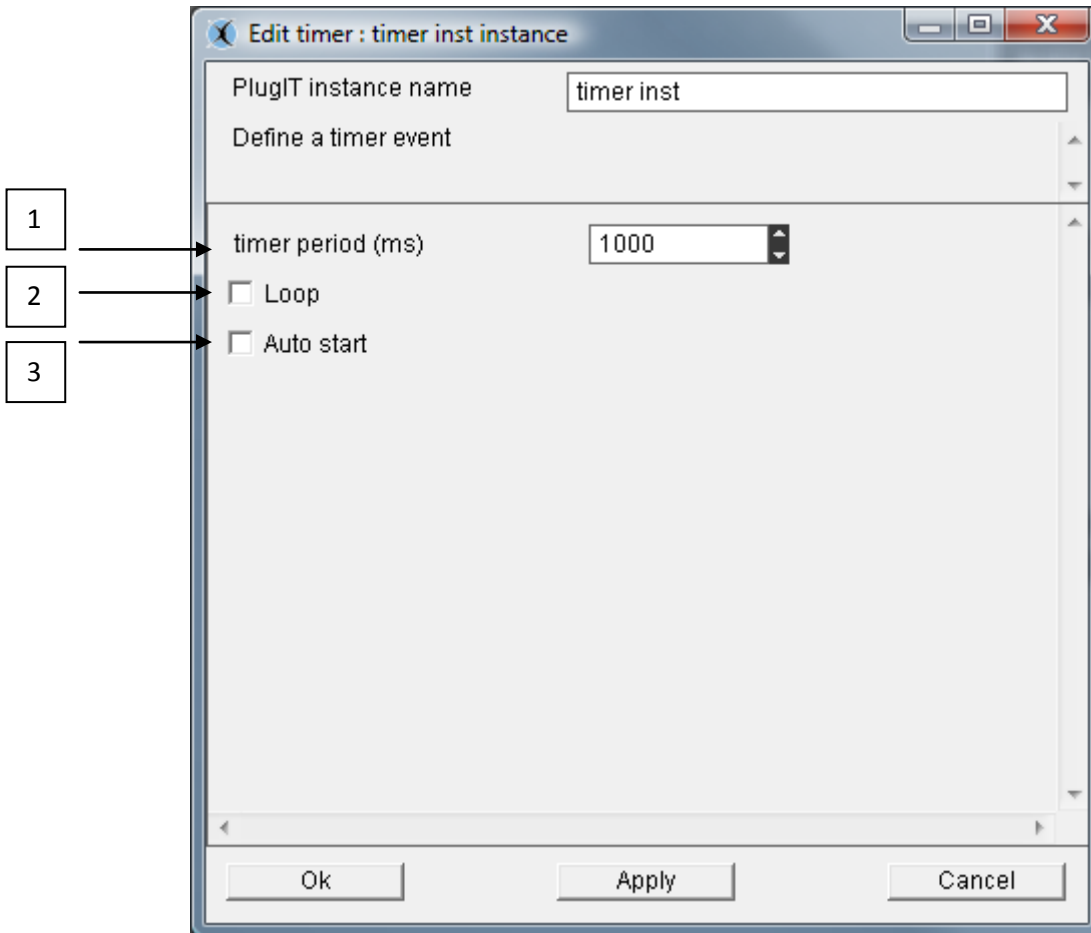
The Close plugIT allows closing the application.



Following the same method as before, add a Close plugIT, and then edit the instance.
This function has no parameters to edit.

PlugIT: "Timer"

It allows you to trigger an event at the end of a period.



1 °/ the "timer period" parameter is used to define the duration in milliseconds of the period before the start of the event.

2 °/ The "Loop" parameter sets whether the event will be triggered only once after the period or at regular intervals after this same period.

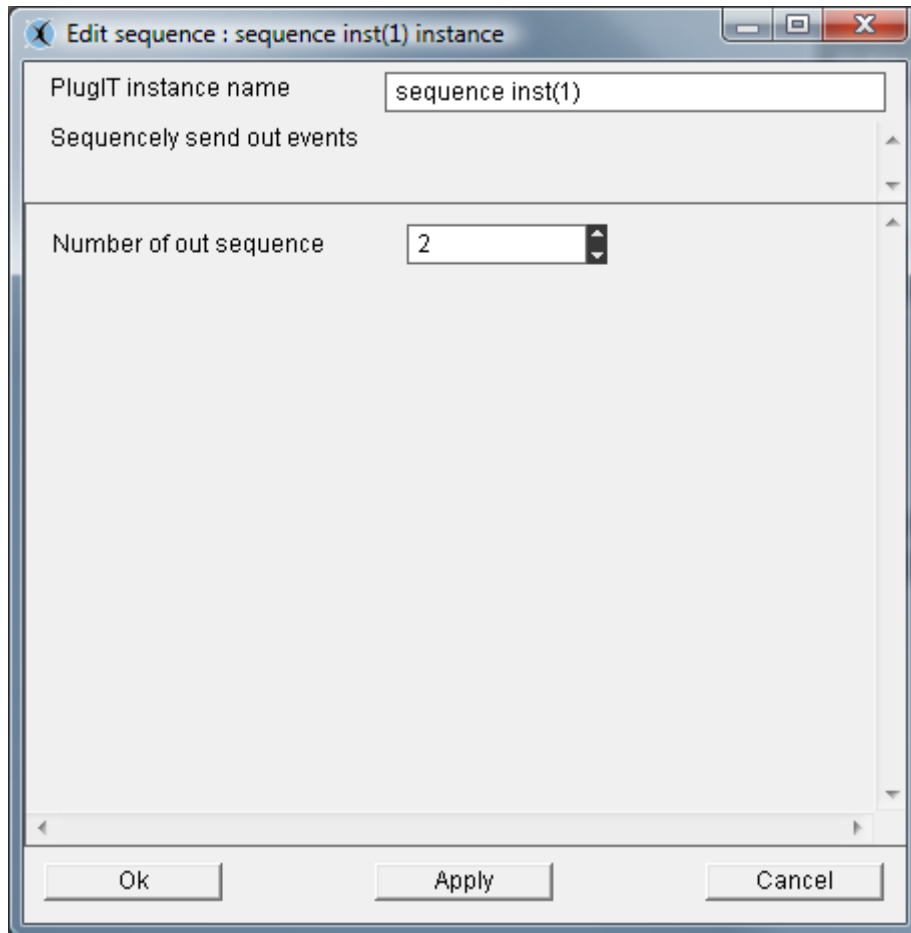
3 °/ The "Auto start" parameter is used to determine if the timer will be started automatically at application startup.

PlugIT: “Sequence”

It allows triggering two events as alternative.

Following the same method as before, add a Sequence PlugIT.

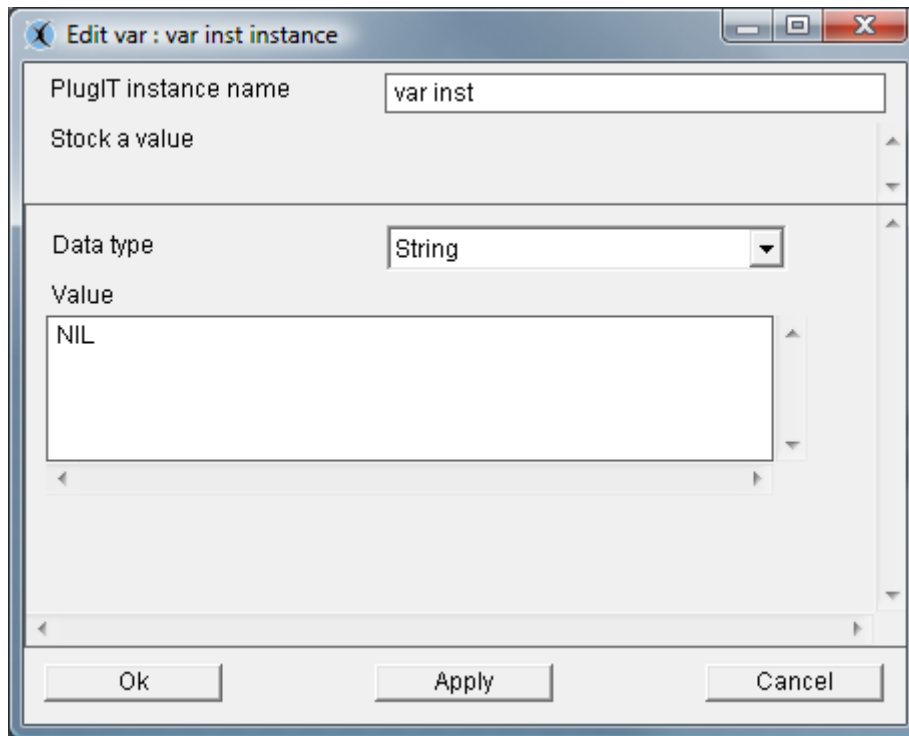
This function allows you to edit the number of sequences that can be connected to another plugit



For example, this function can be used to trigger on a single click via the Object Link function alternately play or stop an animation or video.

PlugIT: "Var"

This PlugIT allows you to define a variable with a value.

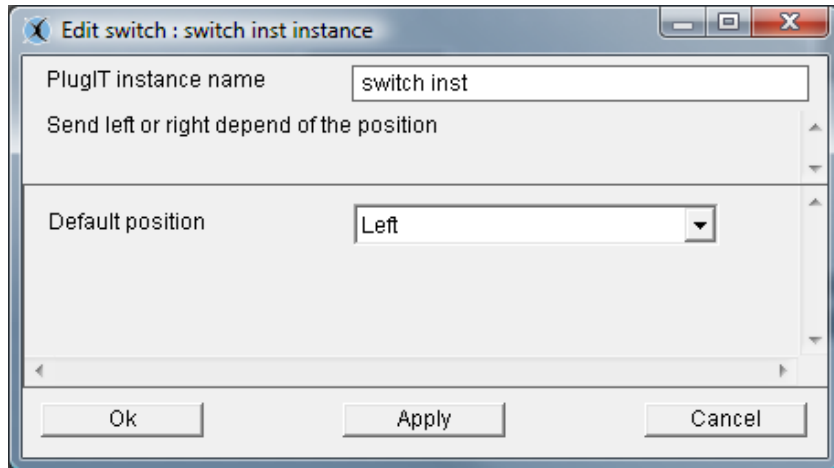


« Data type » type of the variable.

« Value » value at init.

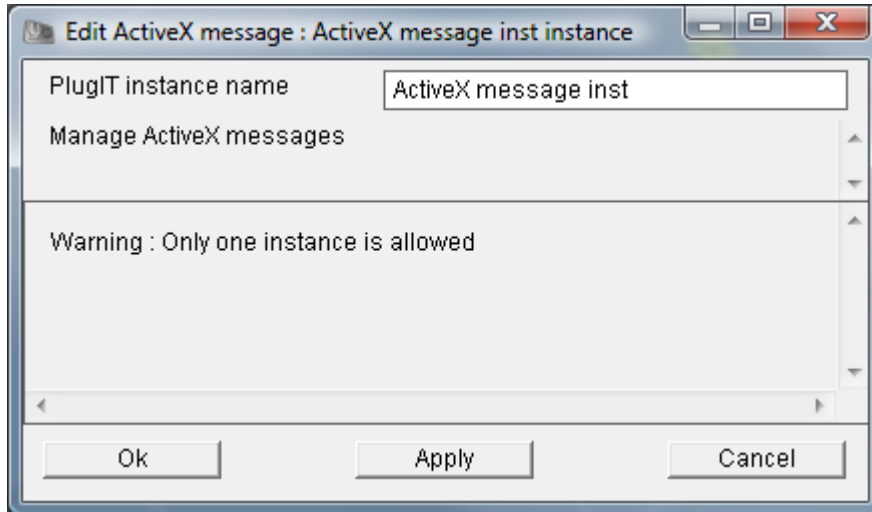
PlugIT: "Switch"

This PlugIT allows you to define two outputs and choose which output to use at a given moment.



The "Default Position" Left or Right parameter allows you to define the position of the switch at the initialization of the plugIT.

PlugIT ActiveXmessage



Here is a sample code php page running with the ActiveXmessage PlugIT:

```
<?php
include "conf.inc.php";

?>

<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<link rel="stylesheet" href="style.css" type="text/css">

<script language="JavaScript">
<!--
function Load()
{
<?php
if($WEBINTEGRATED == 1)
echo
"document.scol.LaunchMachine("\\$browser$%5fload+%22locked%2fstduser%2epkg%22%0amain+%22" .
$SCOL_SITE_URL_OLD . "%22+ffffff+NIL CSDMRWK 1000000\",1,0);";
else
echo "document.scol.LaunchMachine("\\$browser$%5fload+%22locked%2flink%2epkg%22%0amain+%22" .
$SCOL_SITE_URL_OLD . "%22%0a CSDMRWK 1000000\",1,0);";
?>
}
//-->
</script>

</head>
```



```

<body onLoad="Load()">
<div align="center">
<OBJECT classid="clsid:7A96FF35-4937-11D1-8F2C-00609779BDA3"
  codebase="<?echo $WIN_32_PLUGIN_URL?>"
  width="<?php if($WEBINTEGRATED == 1) echo $WIDTH; else echo '0'?>"
  height="<?php if($WEBINTEGRATED == 1) echo $HEIGHT; else echo '0'?>"
  id="scol">
  <PARAM name="ForceInstall" value="no" />
  <PARAM name="ScolVersionNeeded" value="<?php echo $MINIMUMSCOLVERSION;?>" />
  <EMBED
  TYPE="application/x-scol"
  width="<?php if($WEBINTEGRATED == 1) echo $WIDTH; else echo '0'?>"
  height="<?php if($WEBINTEGRATED == 1) echo $HEIGHT; else echo '0'?>"
  name="scol"
  ForceInstall="no"
  ScolVersionNeeded="<?php echo $MINIMUMSCOLVERSION;?>"
  onScolEnd="ScolEnd"
  onScolMessage="MessageFromScol"
  PLUGINSURL="<?php echo $WIN_32_PLUGIN_URL;?>"
  PLUGINSPAGE="<?php echo $WIN_32_PLUGIN_URL;?>"
  </EMBED>
</OBJECT>

<script>
function ScolEnd()
{
  //Your code here
  document.the_form.messval.value="Scol closed";
}

function MessageFromScol(msg)
{
  document.the_form.messval.value=msg;
  //Your code here
}

function setCarColor(nb)
{
  document.scol.SendScolMessage('AXMessage "color'+nb+'");
}
</script>

<!-- CALLBACK FOR IE !-->
<SCRIPT FOR=scol EVENT=onScolMessage(msg)>
  MessageFromScol(msg);
</SCRIPT>
<!-- CALLBACK FOR IE !-->
<SCRIPT FOR=scol EVENT=onScolEnd>
  ScolEnd();
</SCRIPT>

<form name="the_form">
<INPUT TYPE="text" NAME="messval" VALUE="" SIZE="25">

```



```
<input type=button value="Color1" onclick="setCarColor('1')">
<input type=button value="Color2" onclick="setCarColor('2')">
<input type=button value="Color3" onclick="setCarColor('3')">
<input type=button value="Color4" onclick="setCarColor('4')">
</form>

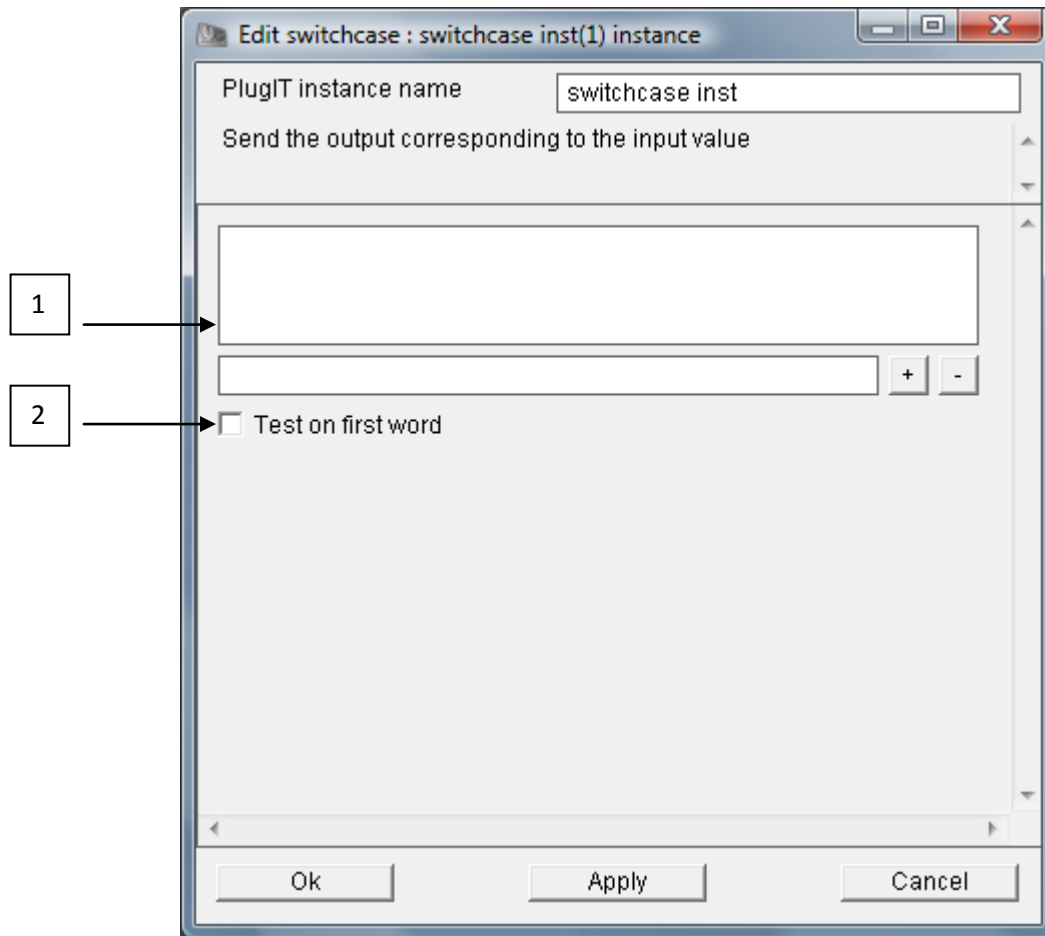
</div>
<p align="center"></p>

</body>

</html>
```

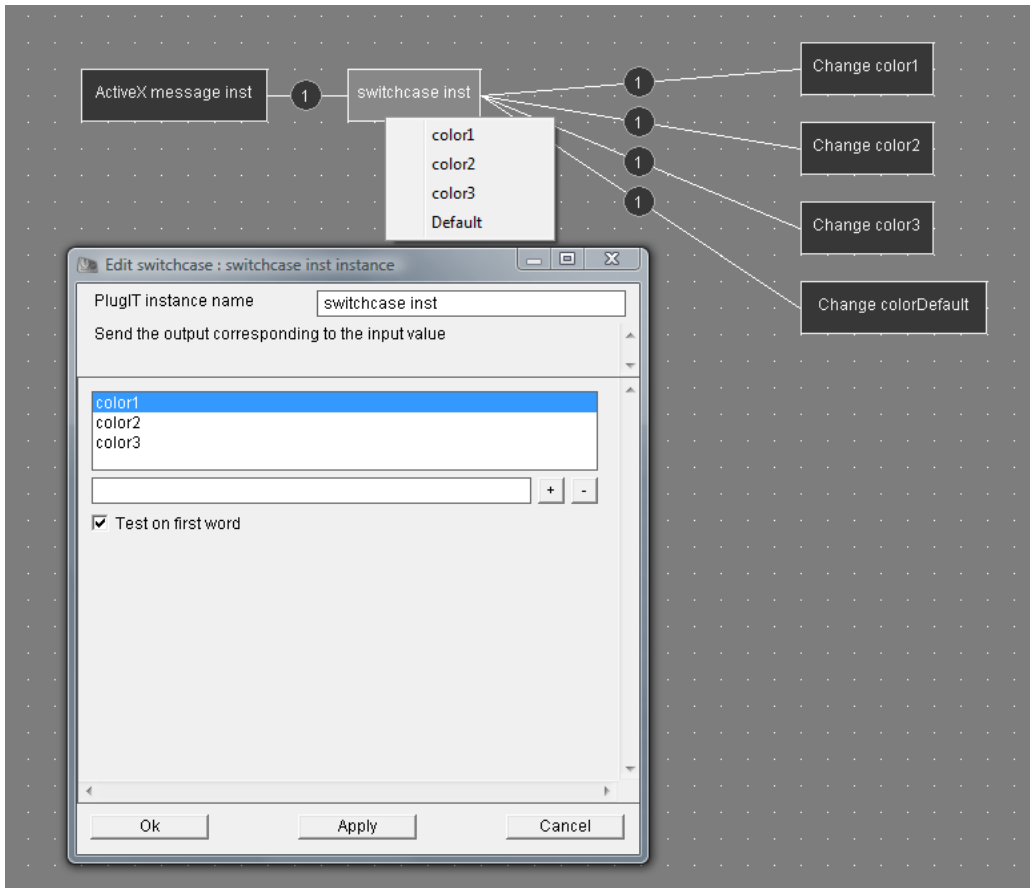
PlugIT « Switchcase »

The Switchcase plugIT can retrieve an input parameter and then execute the event corresponding to the value of a test on a word and assign an action based on the value



1° / Parameters List

2° / activate the test on the first word of the parameter only

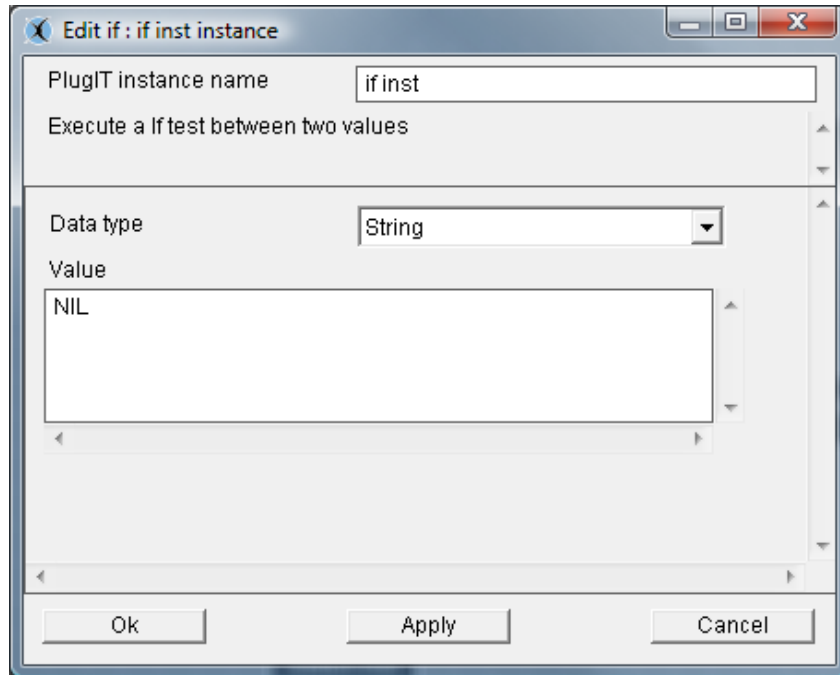


In this example, the ActiveXmessage plugIT sends color settings to the Switchcase plugIT, when the the Switchcase plugIT recognizes any of the terms that were defined in its schedule on the first word addressed to him by the ActiveX, so it triggers the event associated with it, otherwise it triggers the default event..

Ex : When the message ActiveX contains the word "color1" in its parameters, the Switchcase plugIT triggers the color change defines by the value color1 (Change color1).

PlugIT: "If"

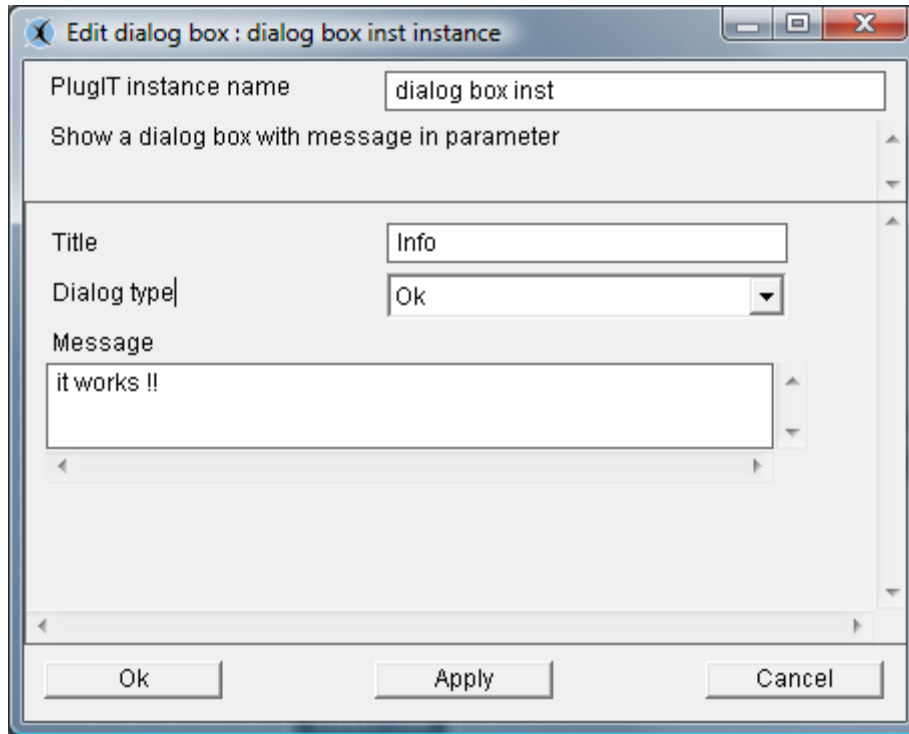
This plugIT allows to test the values defined for example with the Var PlugIT and to send back an event based on the result.



The "Data Type" parameter is the type of variable you want.
 The "Value" parameter is the value of the variable to compare.

PlugIT: "Dialog box "

The Dialog box PlugIT allows creating a dialog box.



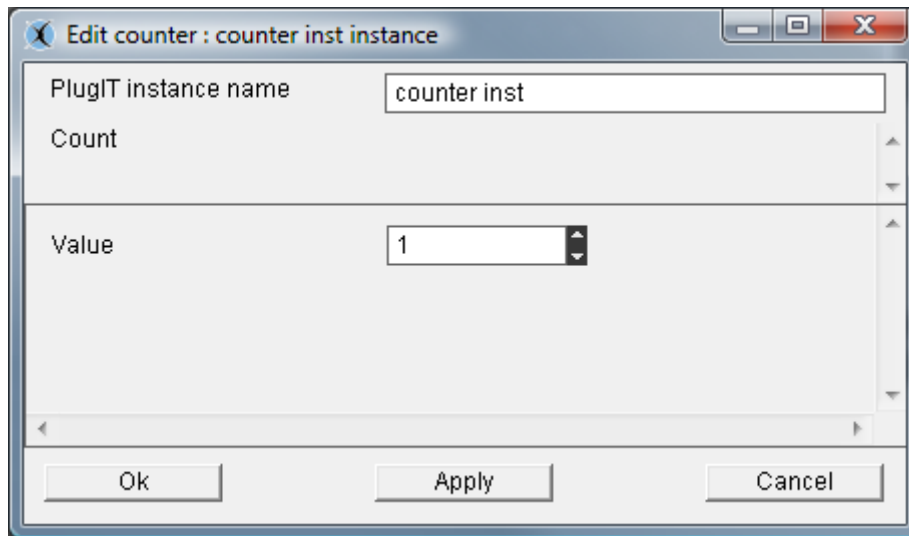
The "Title" parameter is the title of the dialog box.

The "Dialog type" parameter is the type of dialog you want, with the simple " OK "button or with two possible answers" OK / Cancel" or "Yes / No ".

The "Message" parameter is the default message displayed in the dialog box.

PlugIT: “Counter”

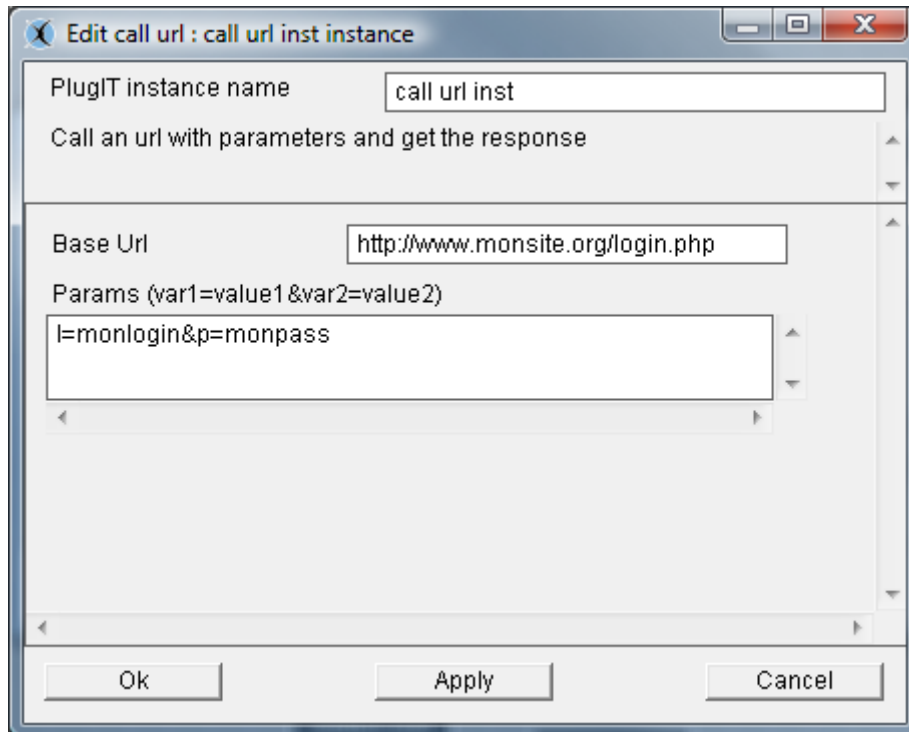
The Counter PlugIT allows creating a variable and triggering an event when the account is reached.



The « Value » parameter is the initial value of the counter.

PlugIT: "Call Url "

It allows sending an http request and retrieving the result, this plugIT can be used for example to query a php website.

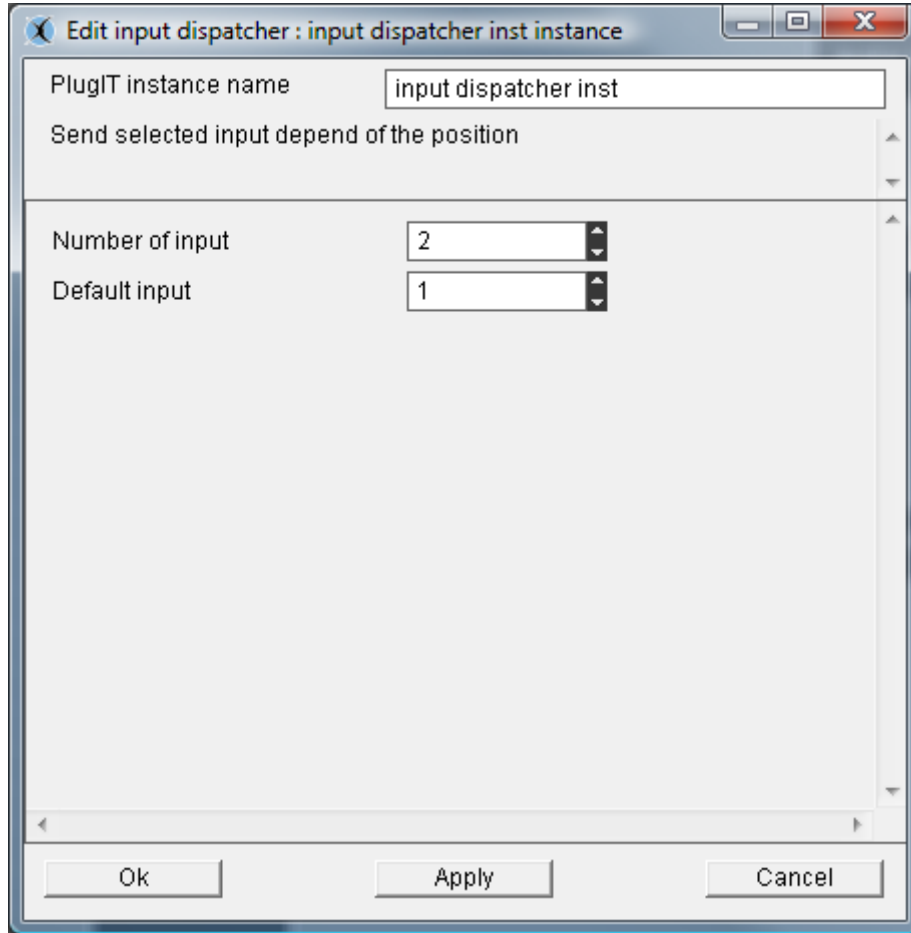


The "Base Url" parameter is the url of the request.

The "Settings" parameter allows you to specify the parameters to send following the url.

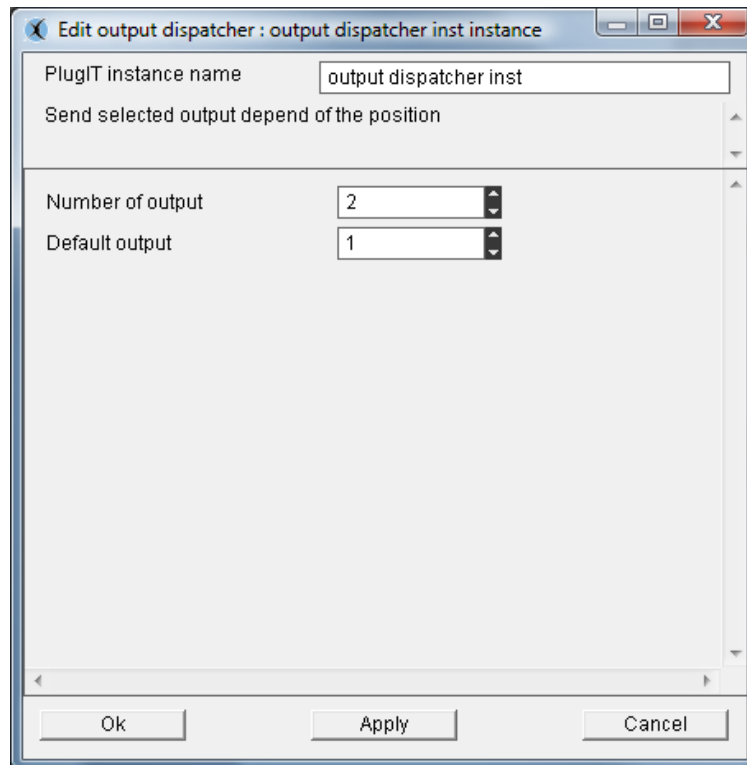
plugIT input Dispatcher

This logic plugIT allows to distribute different entries depending on the position



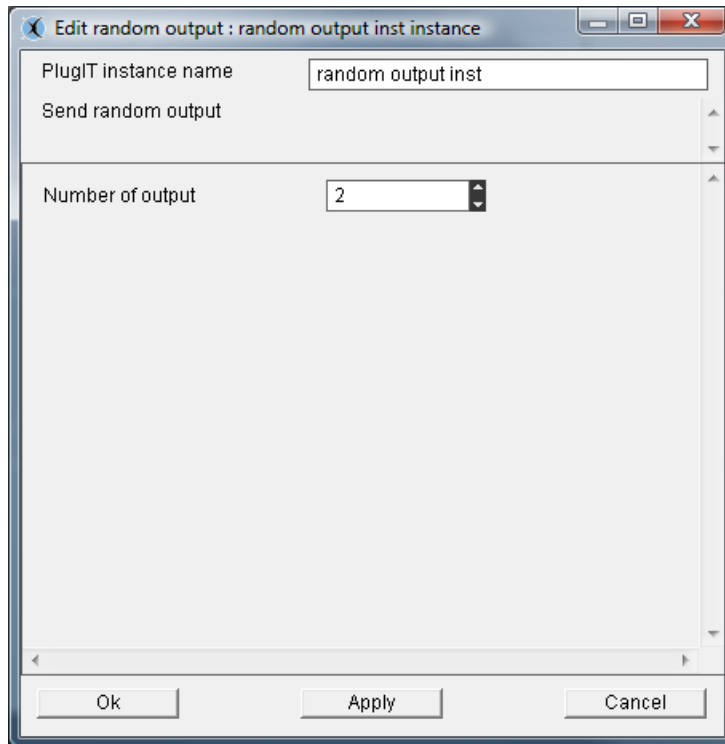
PlugIT output Dispatcher

This logic plugIT allows to distribute an output depending on the position.



PlugIT Random Output :

The plugIT random output allows you to define a number of output that will run randomly

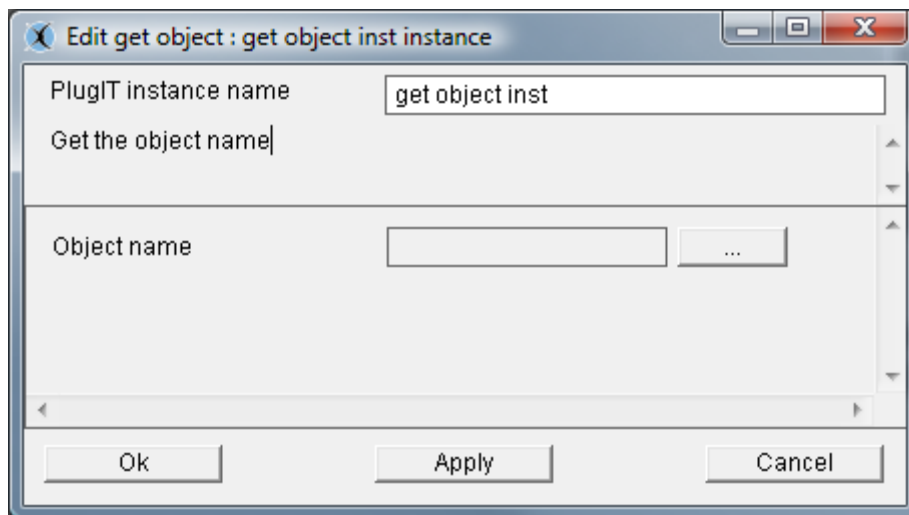


« Object » plugIT

PlugIT : « Get Object »

The Get Object PlugIT allows retrieving the name of an object and then returning it as a parameter. It is only useful when used with other plugIT, for example with the Object Link plugIT.

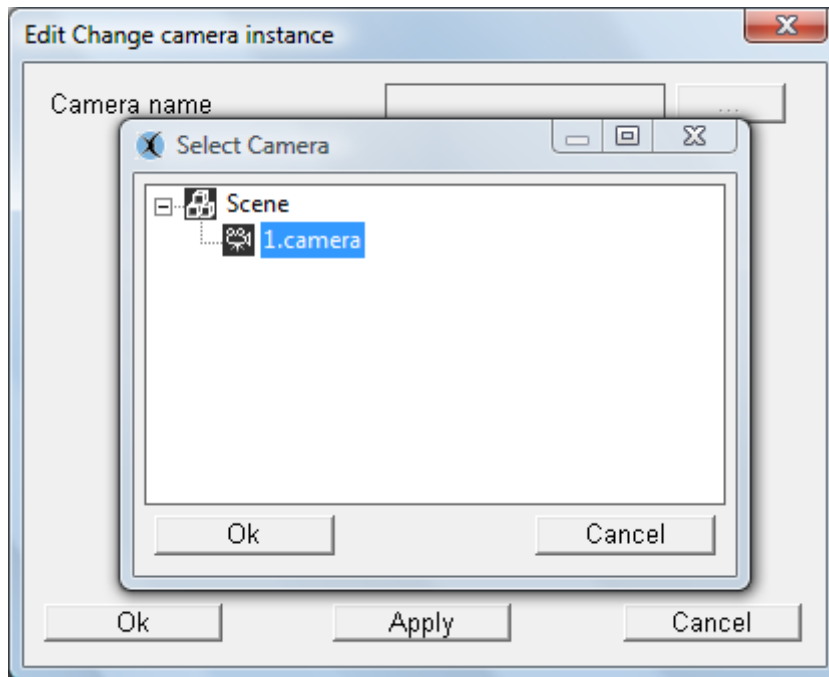
Following the same method as before, add a Get Object plugIT, and then edit the instance.



Select the item on which you want to act via the browse button.

PlugIT : “Set Active camera”

PlugIT « Set Active camera » allows you to change the used camera.



Select the camera.

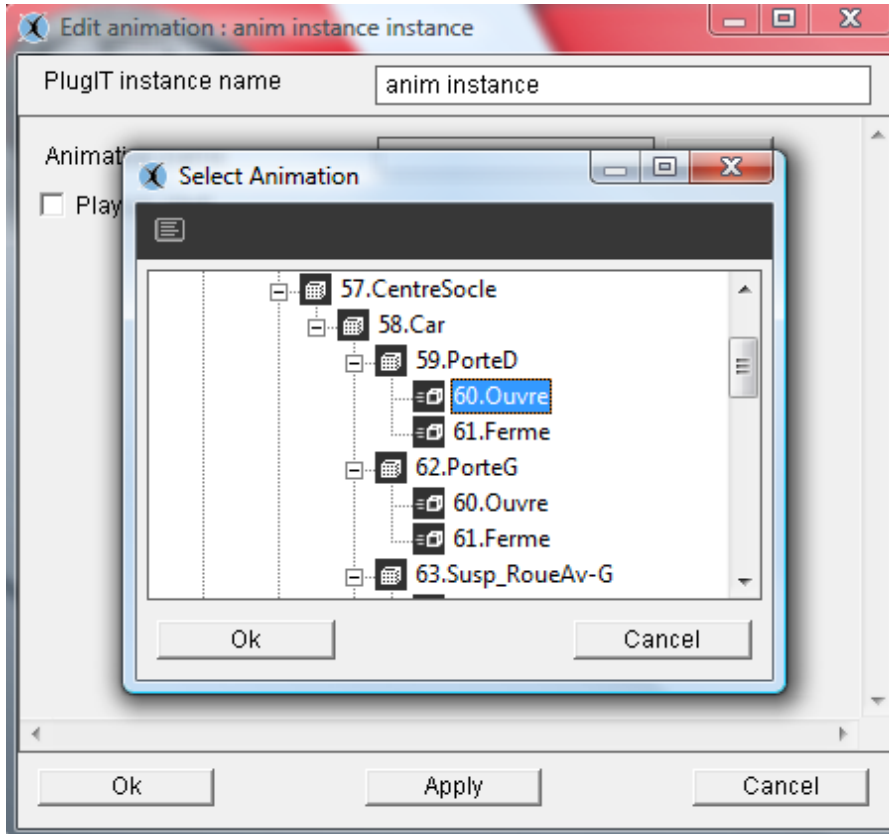
It is possible, to activate or retrieve the camera by using link edition.

PlugIT : "Animation"

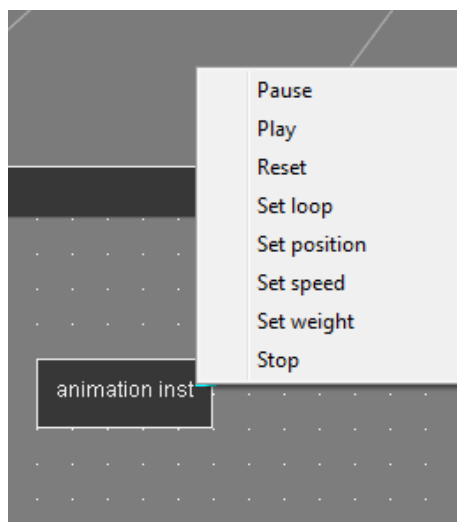
Animation PlugIT allows you to read or stop an animation.

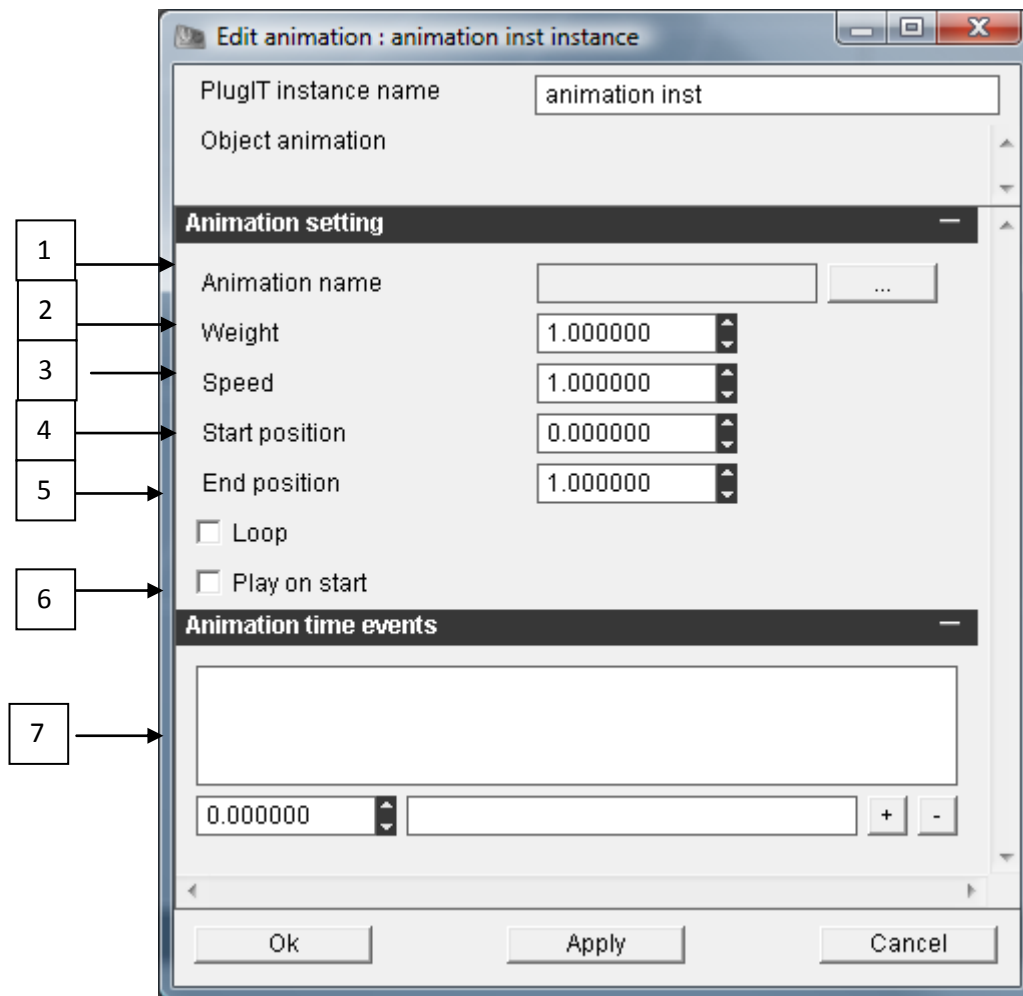
To control an animation using the animation function it is advisable to disable the animation in the scene tree.

Following the same method as above, add an animation PlugIT and then edit the instance.



Select the animation using the browse button.





1 / name of the animation

2 / weight of animation

3 / speed of animation: You can speed up or slow down an animation from its original speed and if the value is negative, the animation will be played backwards

4 / sets the value on which to start the animation

5 / sets the value that should stop the animation

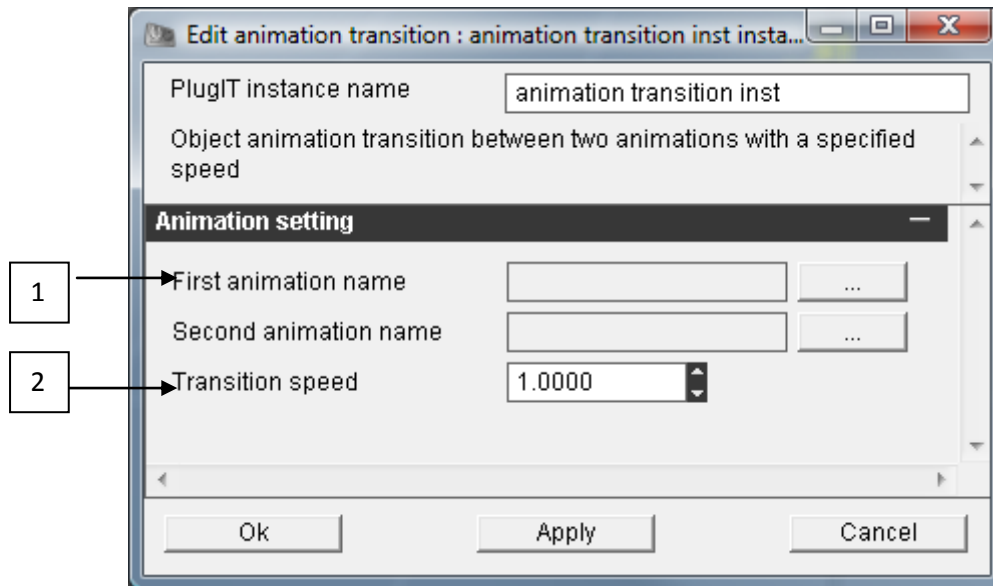
6 / The parameter "loop" defines whether the animation should be played repeatedly. The parameter

"Play on start" sets whether the animation will be played automatically launch the application

7 / Allows you to define events according to a given time in the animation

PlugIT : Animation Transition

Allows to make transitions between 2 animations.

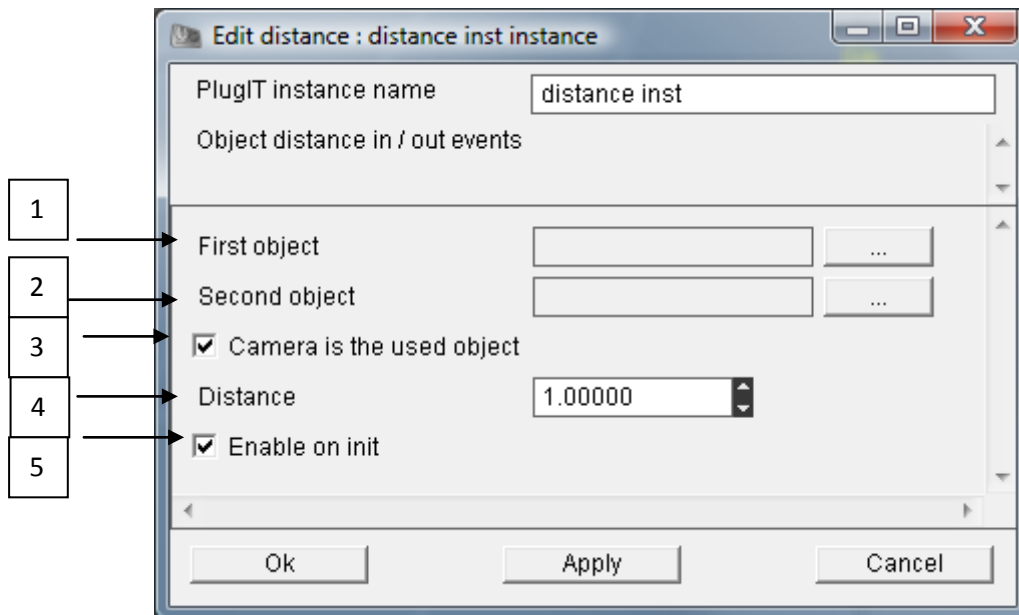
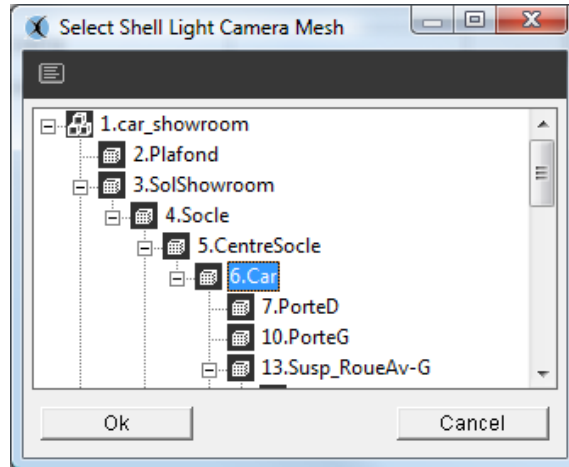


1°/ selection of animation 1 and 2

2°/Speed of transition

PlugIT : "Distance"

The Distance PlugIT is used to trigger an event when one enters or leaves an area around an object. Following the same method as before, add a Distance PlugIT, and then edit the instance.

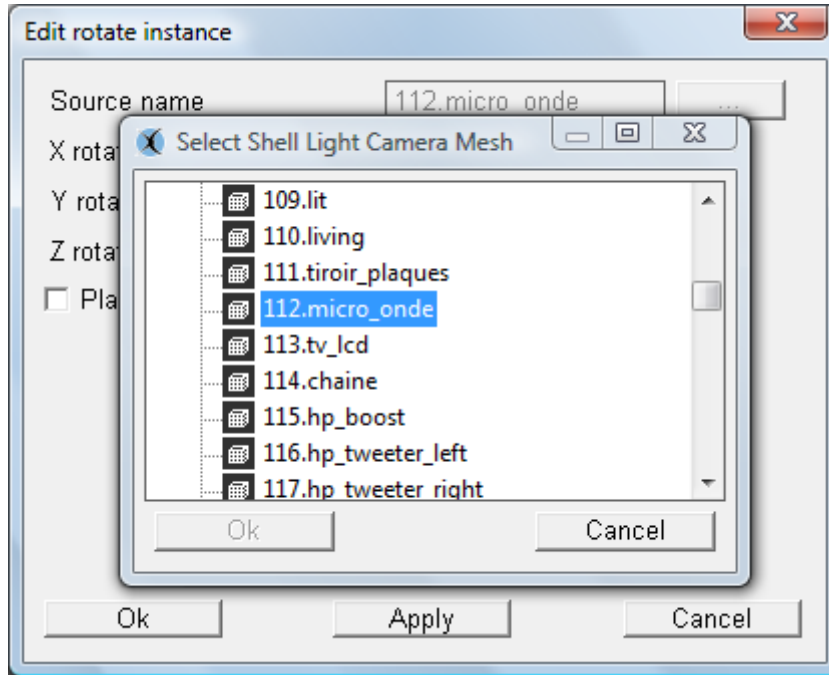


- 1 / Select the object that will serve as a source area via the browse button.
- 2 / Select the object that will serve as a delineation of the area via the browse button.
- 3 / Select the camera is above the current object reference
- 4 / The parameter "distance defines the radius of the area around the source object.
- 5 / Enable on Startup

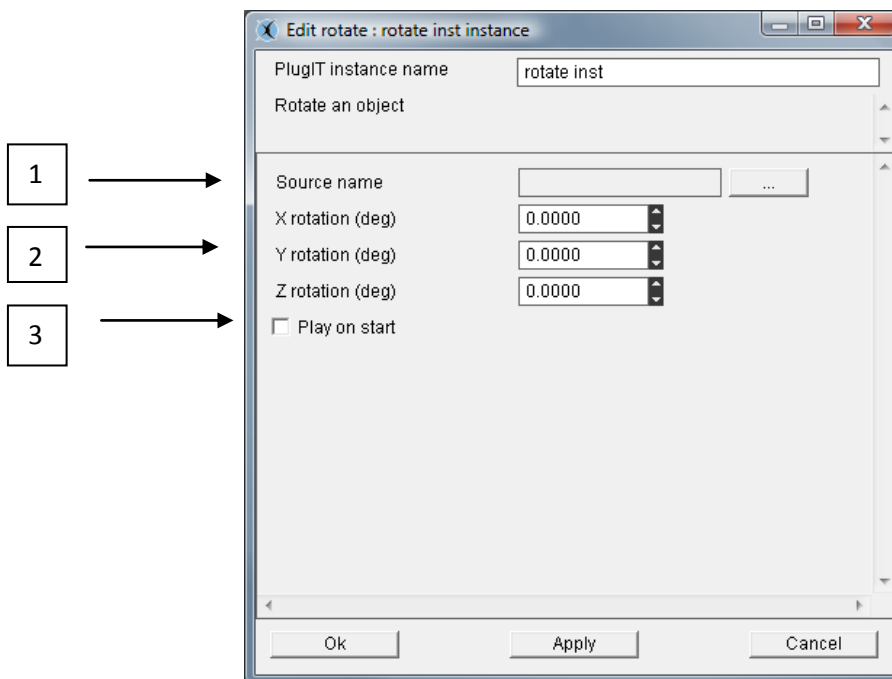
PlugIT: "Rotate"

Rotate PlugIT allows applying a rotation on an object or a node.

Following the same method as before, add a PlugIT Rotate, and then edit the instance.



Select the object you want to run via the browse button.

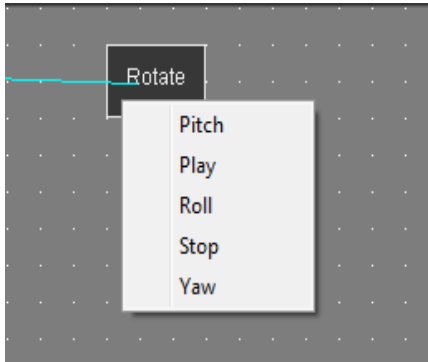


1°/ Select source Object

2°/The parameters "rotation" defines the angle of rotation applied by axis and rendered image in degrees.

3°/The "Play on start" parameter is used to define whether the rotation will start automatically when you launch the application.

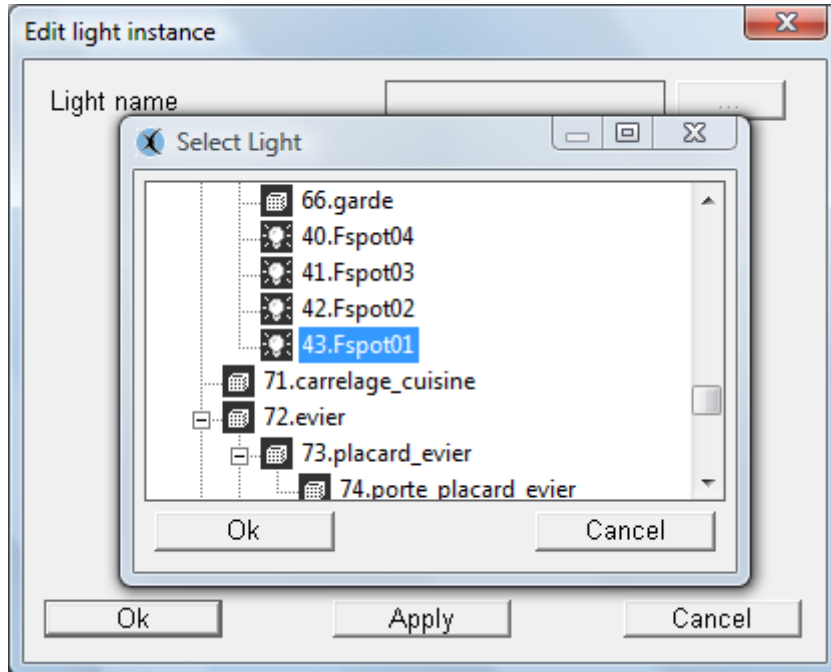
It is possible to choose one of this types of rotation : "Yaw, pitch or roll"



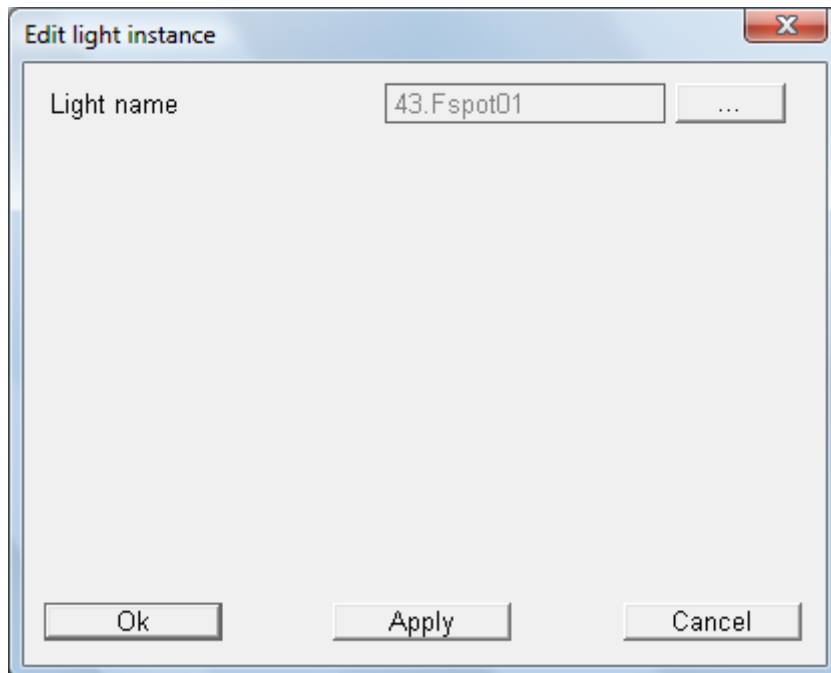
PlugIT: "Light"

Light PlugIT allows activating or deactivating a light dynamically.

Following the same method as before, add a light PlugIT, then edit the instance.



Select the light on which you want to act via the browse button.

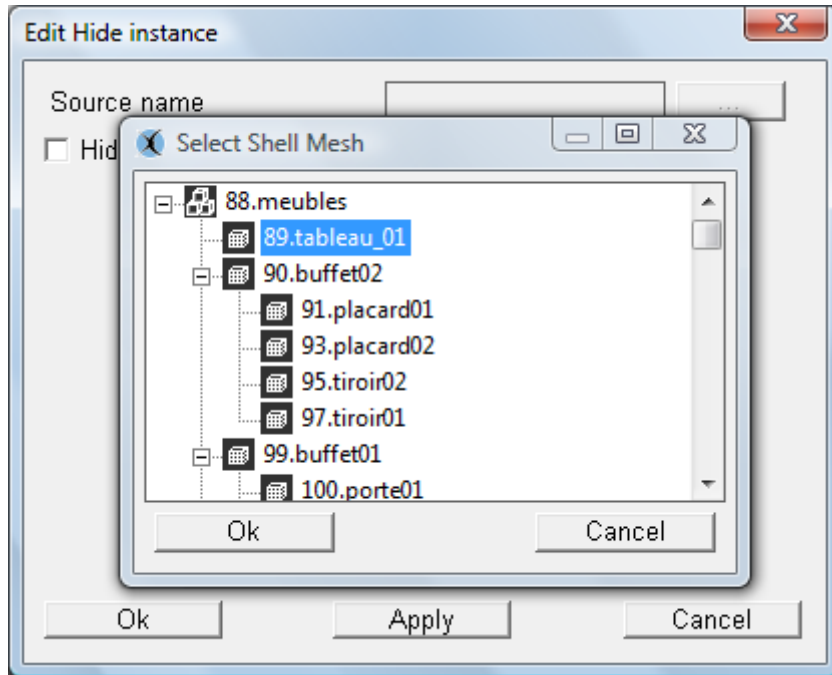


Light function has no specific parameter; you can activate or deactivate the light via the action "On" and "Off" using the links.

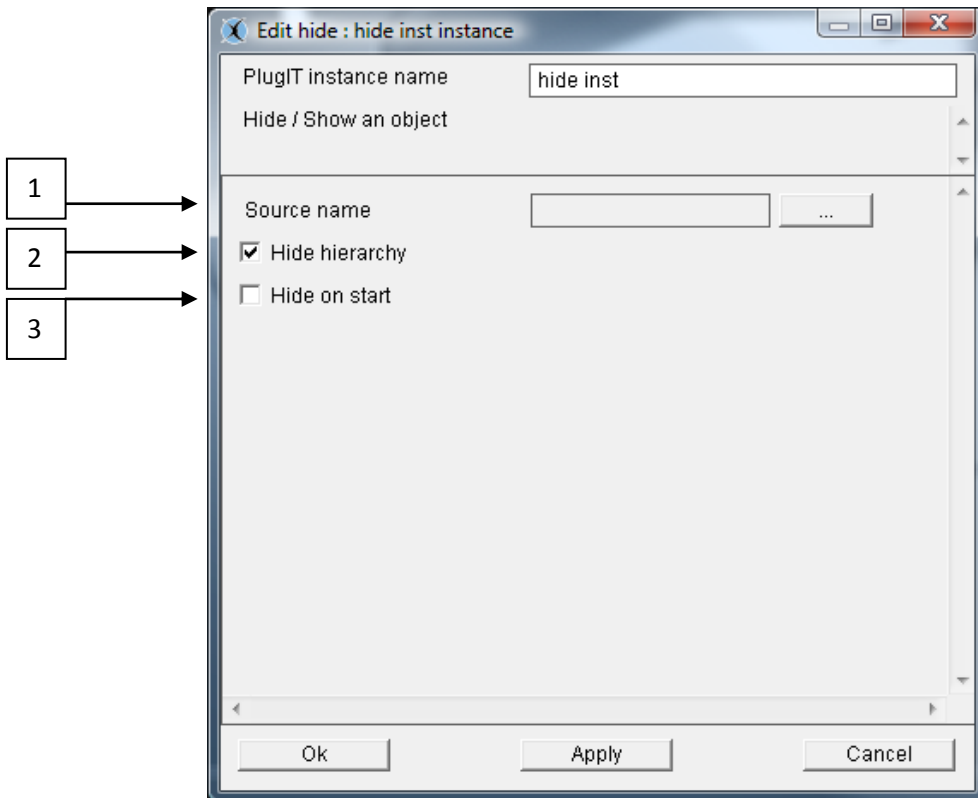
PlugIT: "Hide"

Hide PlugIT allows you to hide or show an element in the 3d.

Following the same method as before, add a PlugIT Hide, then edit the instance.



Select the item you want to act on via the browse button.





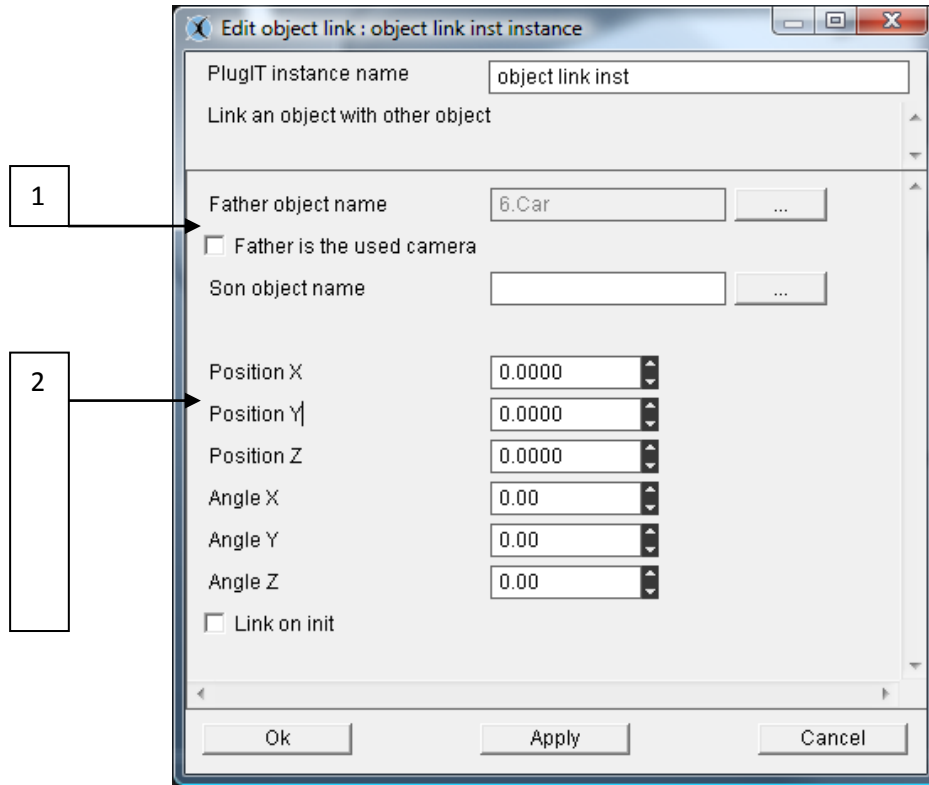
1°/ Select source object

2°/ The « Hide hierarchy » parameter is used to define whether the sons of the object will be hidden.

3°/ The "Hide on start" parameter is used to define whether the element will be hidden automatically when you launch the application.

PlugIT: "Object link "

The Object link PlugIT allows to link one object to another object or the active camera.



1°/ Select father object to which your other object will be linked via the browse button or click the "Father is the used camera" if you want to link an object to the camera.

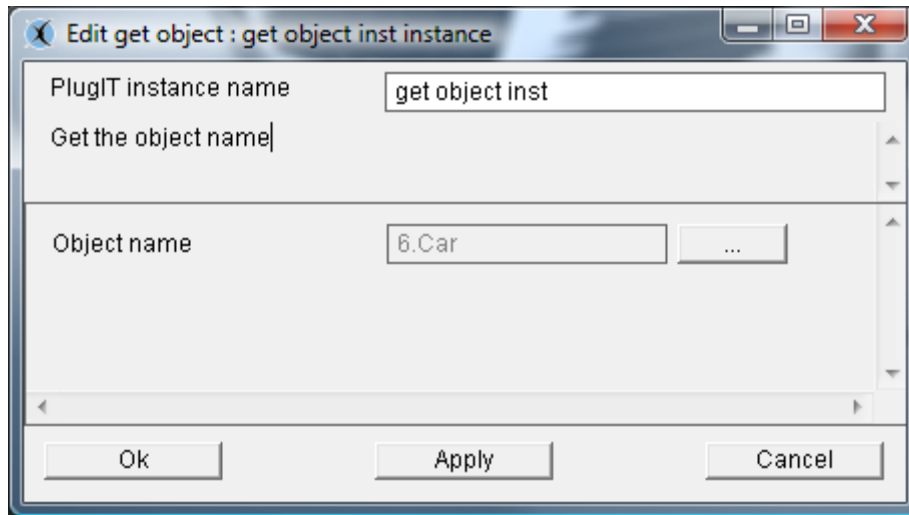
2°/The "position" parameter is used to define the position relative to the parent object. The "angle" parameter determines the orientation relative to the parent object.

To use this plugIT you will need to use a plugIT Get Object or Get Camera and link:
 get object.Object -> object link.Link
 or set the name of the object as a parameter of the link.

PlugIT Get Object

PlugIT Get Object allows to retrieve the name of an object and then return parameter. It is only useful when used with another plugIT, for example with the Object Link plugIT.

Following the same method as above, add a plug Get Object, and then edit the proceedings.

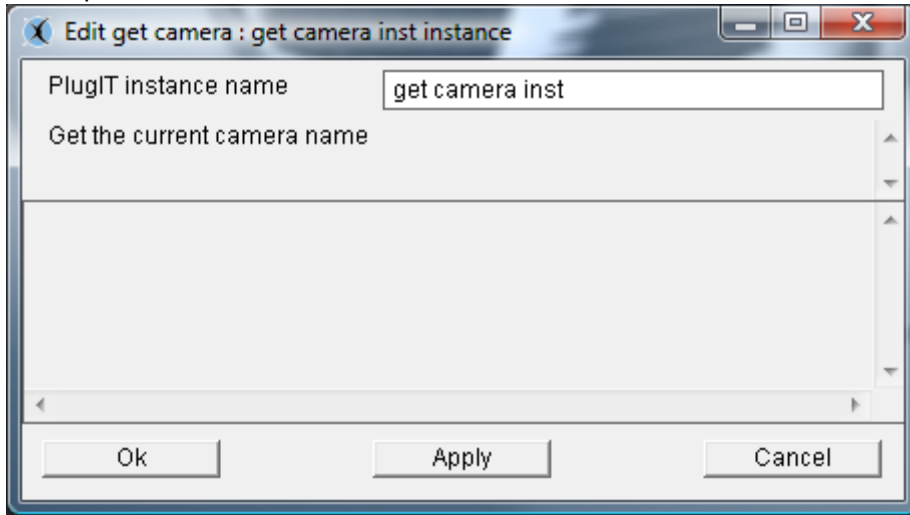


Select the item you want to act on via the browse button

PlugIT: “Get Camera “

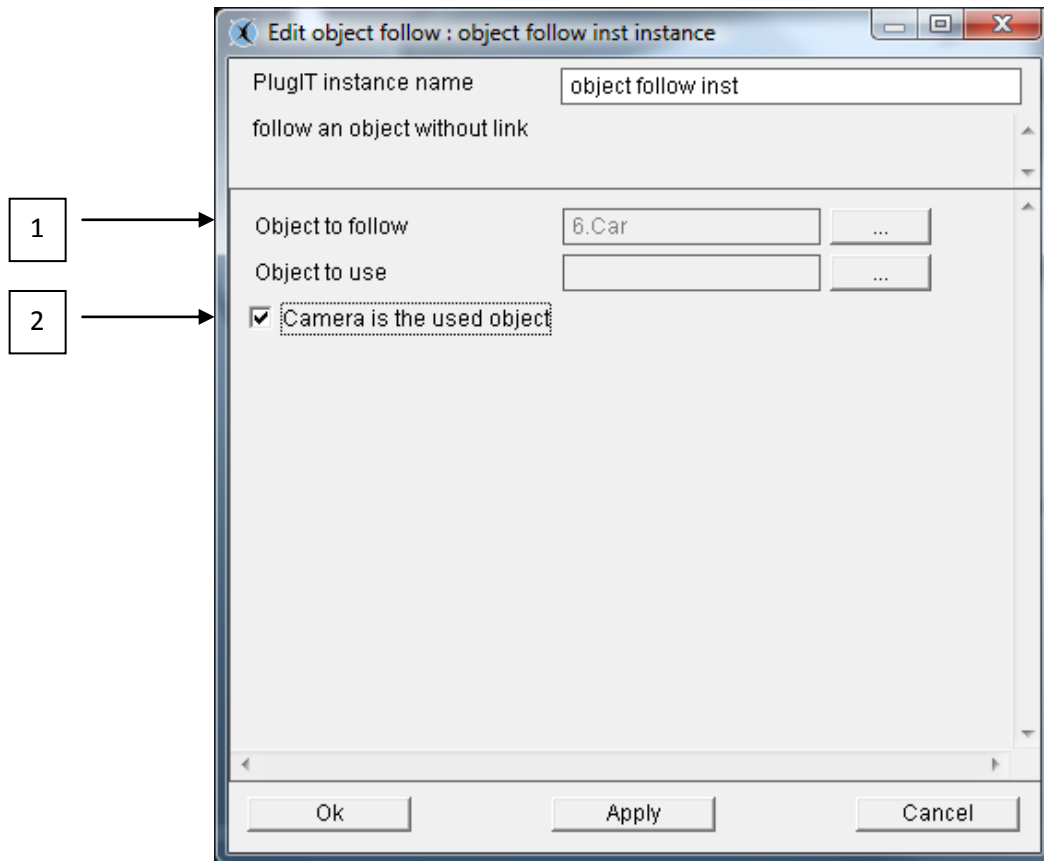
The Get Camera PlugIT allows retrieving the current camera and returning it as a parameter. It is only useful when used with other plugIT, for example with the plugIT Object Link.

This function has no parameters to edit.



PlugIT: “objectFollow”

PlugIT allows an object to follow another one. Indeed they are linked by hierarchy

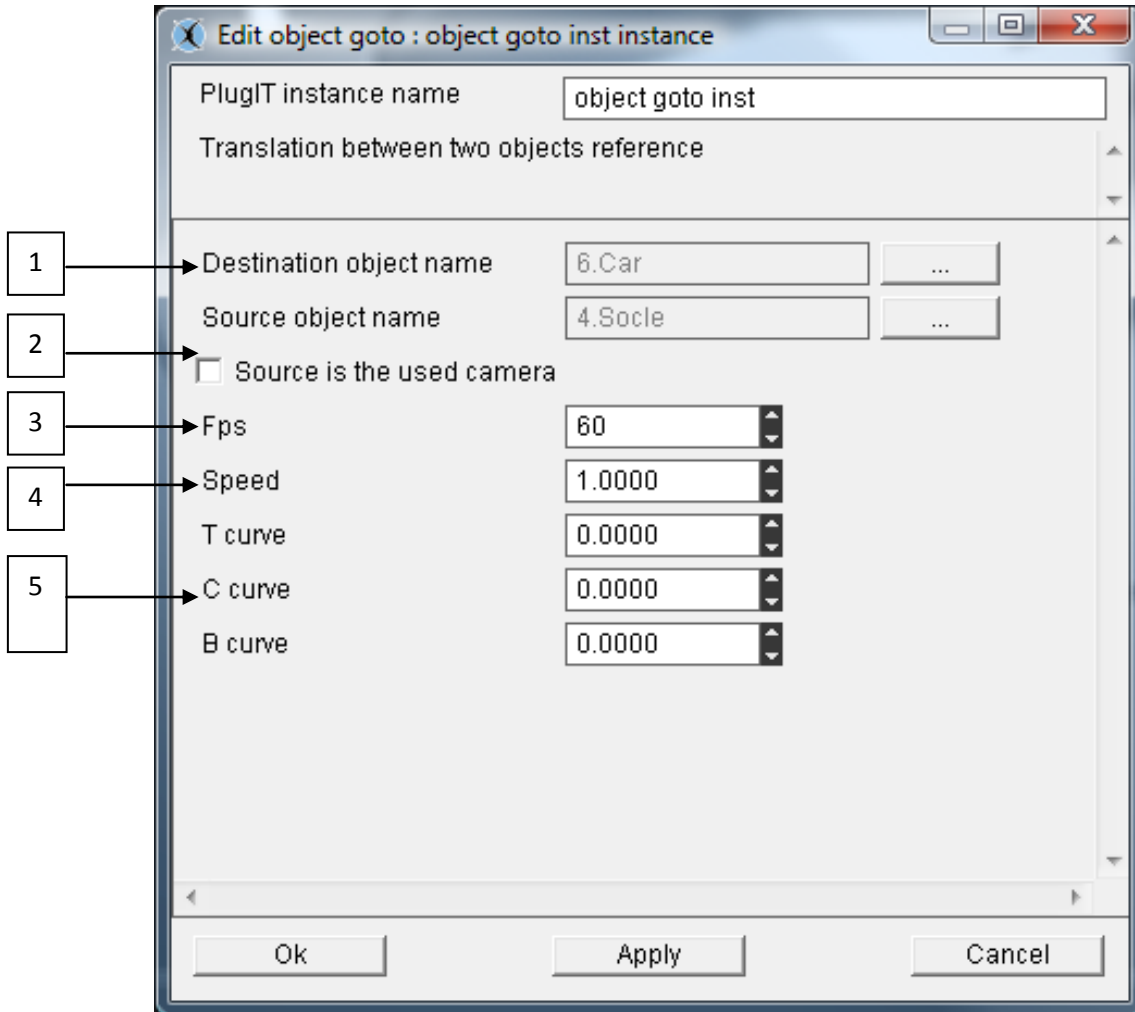


1 °/ Target Object

2 °/ If « camera is the used object » the current object is the camera

PlugIT: “objectGoto”

PlugIT gives the possibility to send an object at a given position automatically.



1 °/ the selected Object

2 °/ Destination position by using an other object or a vector or « source is the used camera »

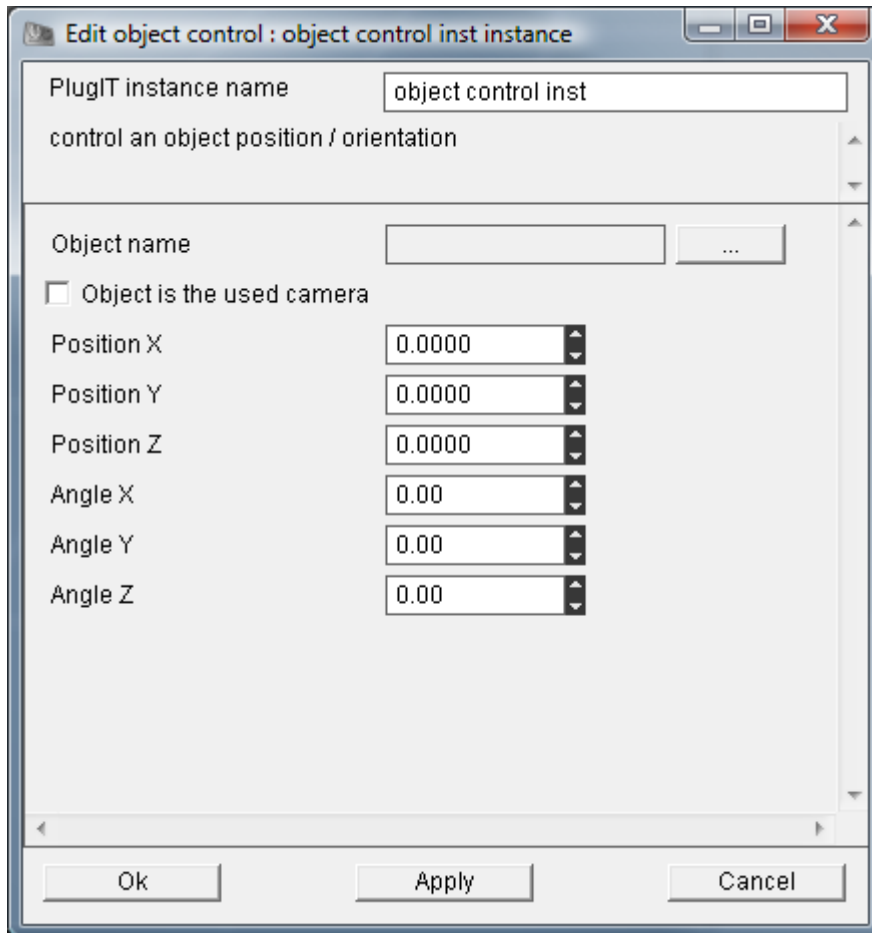
3 °/ Frame per second of the calculated animation

4 °/ Speed of animation

5 °/ TCB parameters for the calculation of positions by using Bezier interpolation.

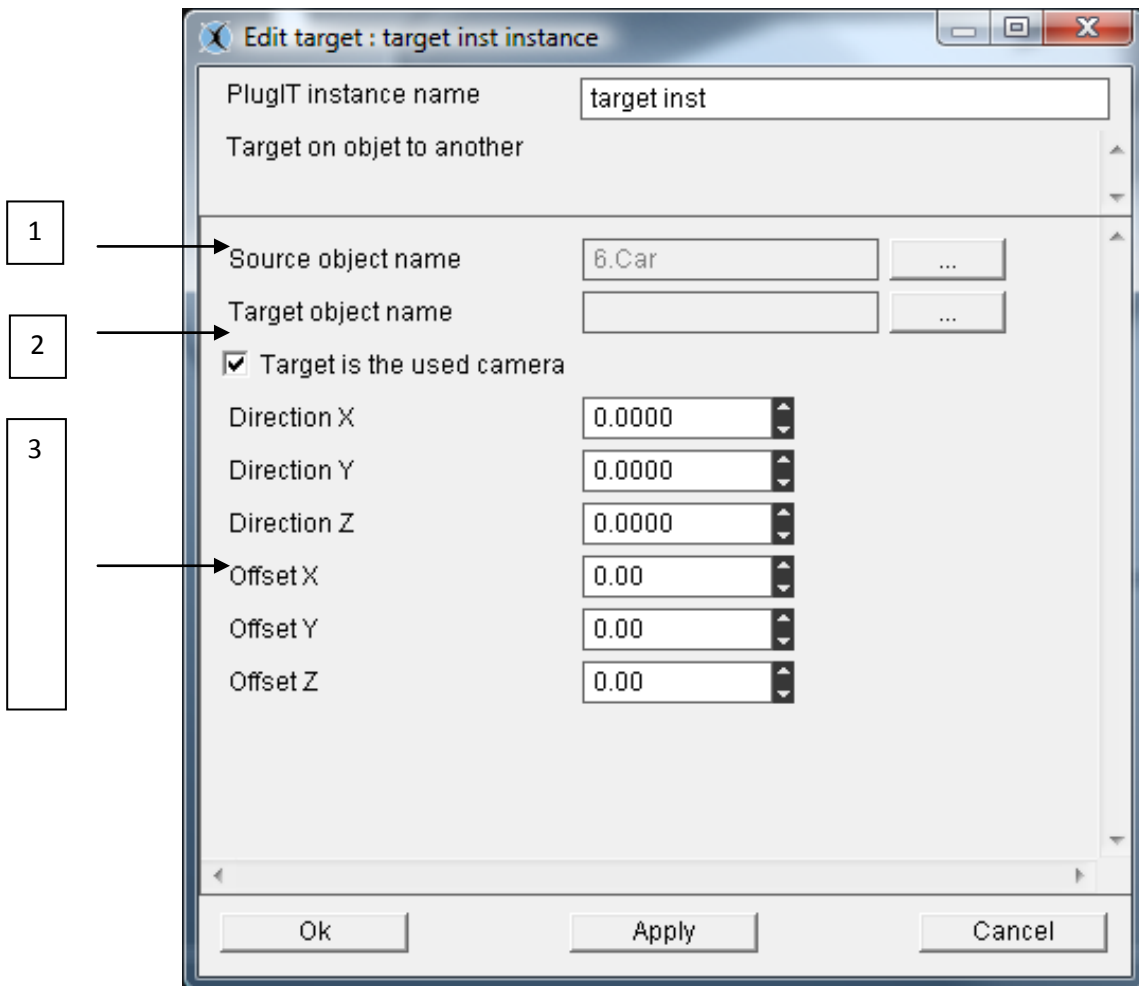
PlugIT : "Object Control"

The plugIT « Object control » allows you to control position and orientation for an object, this object can also be the used camera



PlugIT: "target"

PlugIT to fix the orientation of an object.



1 °/ Selected Object

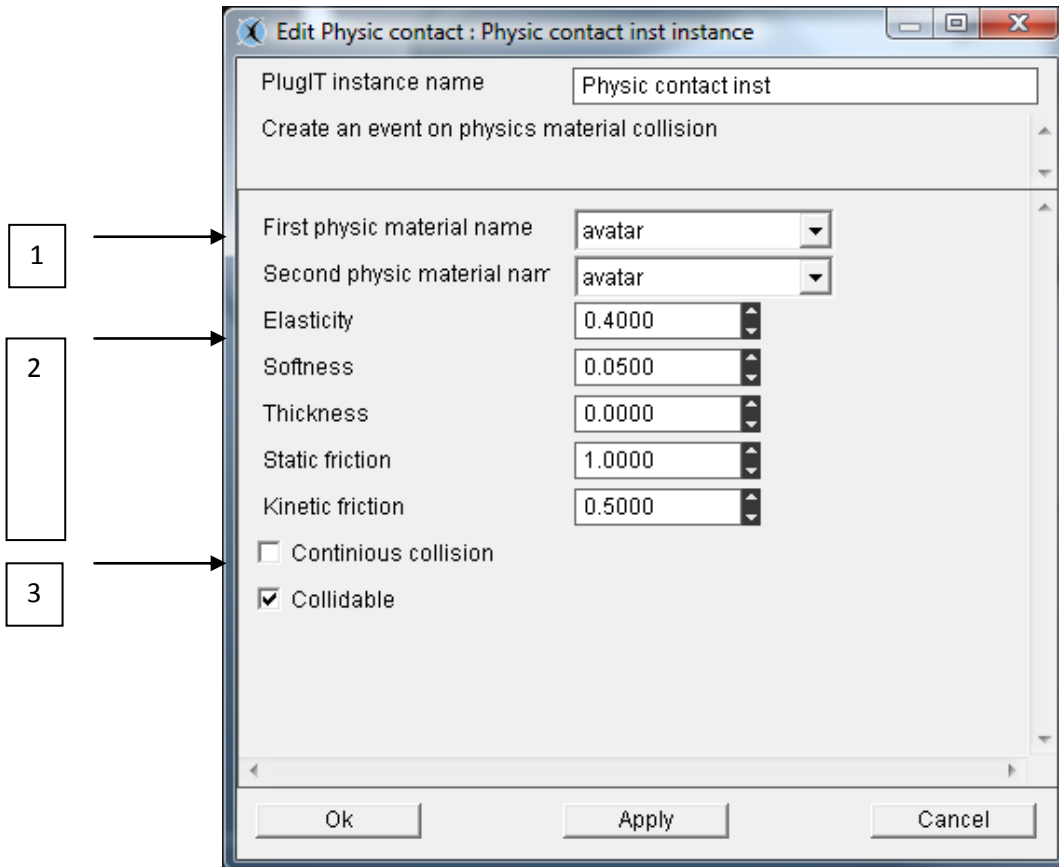
2 °/ Target object or « target is the used camera »

3 °/ Vector orientation and position for the object

« Physics» plugITs

PlugIT: “Physic Contact”

This PlugIT allows when physics are activated a contact between two physic materials (see 24 Advanced Use and definitions)



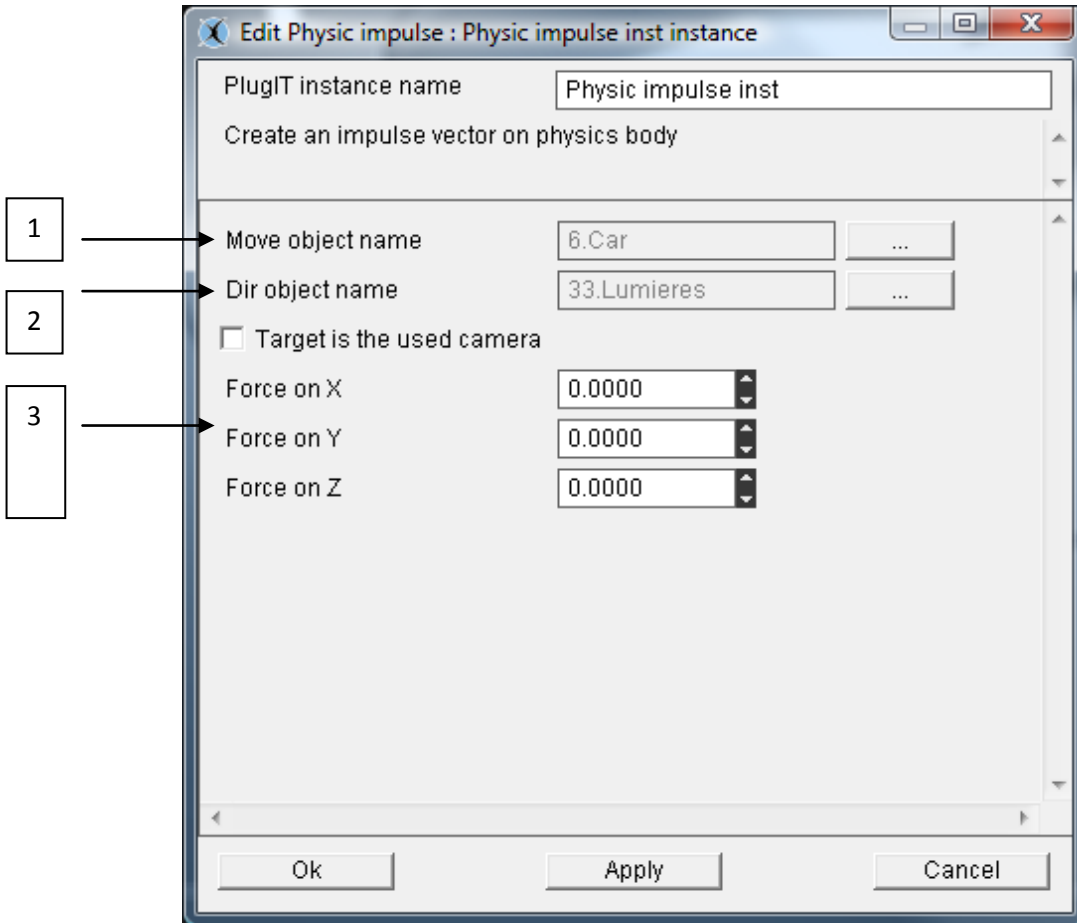
1 °/ we inform here the contact between two materials by example: between the materials by default applied to the body and the material corresponding to the avatar (movement of camera)

2 °/ We so parameterize the physical reactions when these two materials have a collision: (elasticity, softness, thickness, static friction, kinetic friction)

3 °/ Options to set continuous collision (More precise but greedier) and « collidable » to specify that a collision could happen between two materials

PlugIT : « PhysicImpulse »

This plugIT allows simply to add an impulse (cf : 24) on a body



1 °/ we inform here the body of the object source to which the impulse will be applied

2 °/ we inform here the direction by the object or by the current camera

3 °/ we could parameterize an impulse defined by a vector Force.

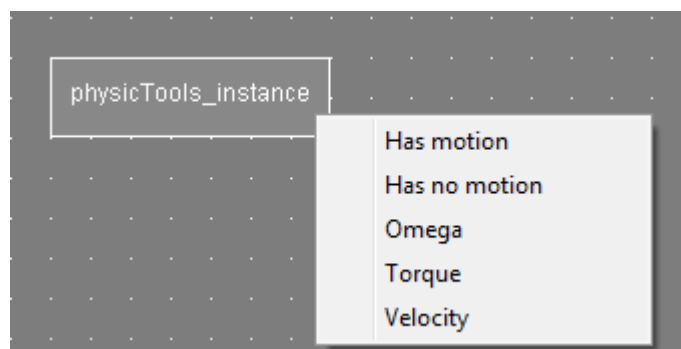
PlugIT: « *physicTools* »

This complete plugIT allows having access to a complete toolbox to parameterize the physical reactions on an object.



We inform object's name

Next:

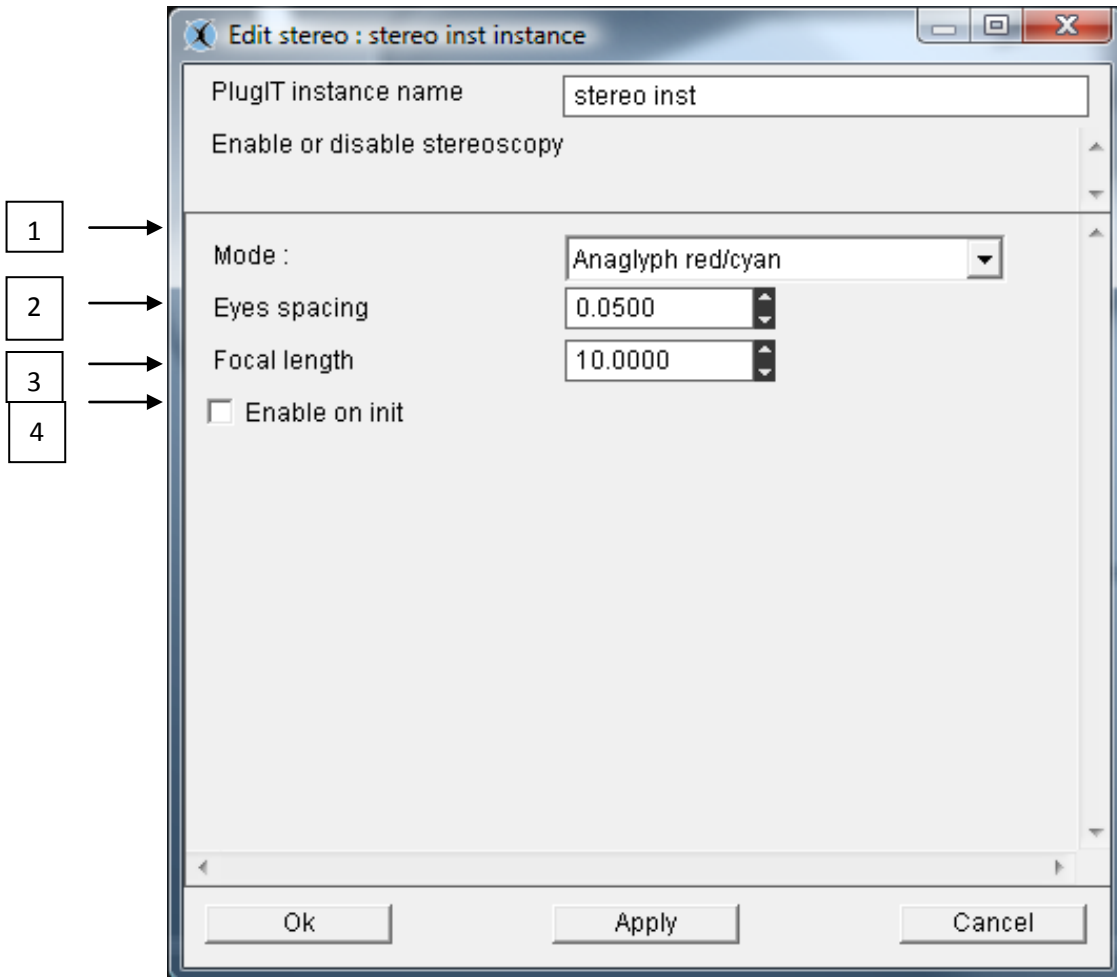


We can parameterize the body of the object to give it value of Omega, Torque and of velocity (cf: 24. Annex physical engine Newton). And getting is "motion" and "no motion" state.

PlugIT “Rendering”

PlugIT: « Stereo »

This plugIT allows activating the stereoscopic mode.



1°/ Activation of the stereo mode in the launch of the application. Different types :

- Anaglyph red/cyan
- Anaglyph yellow/blue
- Interlaced Horizontal
- Interlaced Vertical
- Interlaced with checkerboard

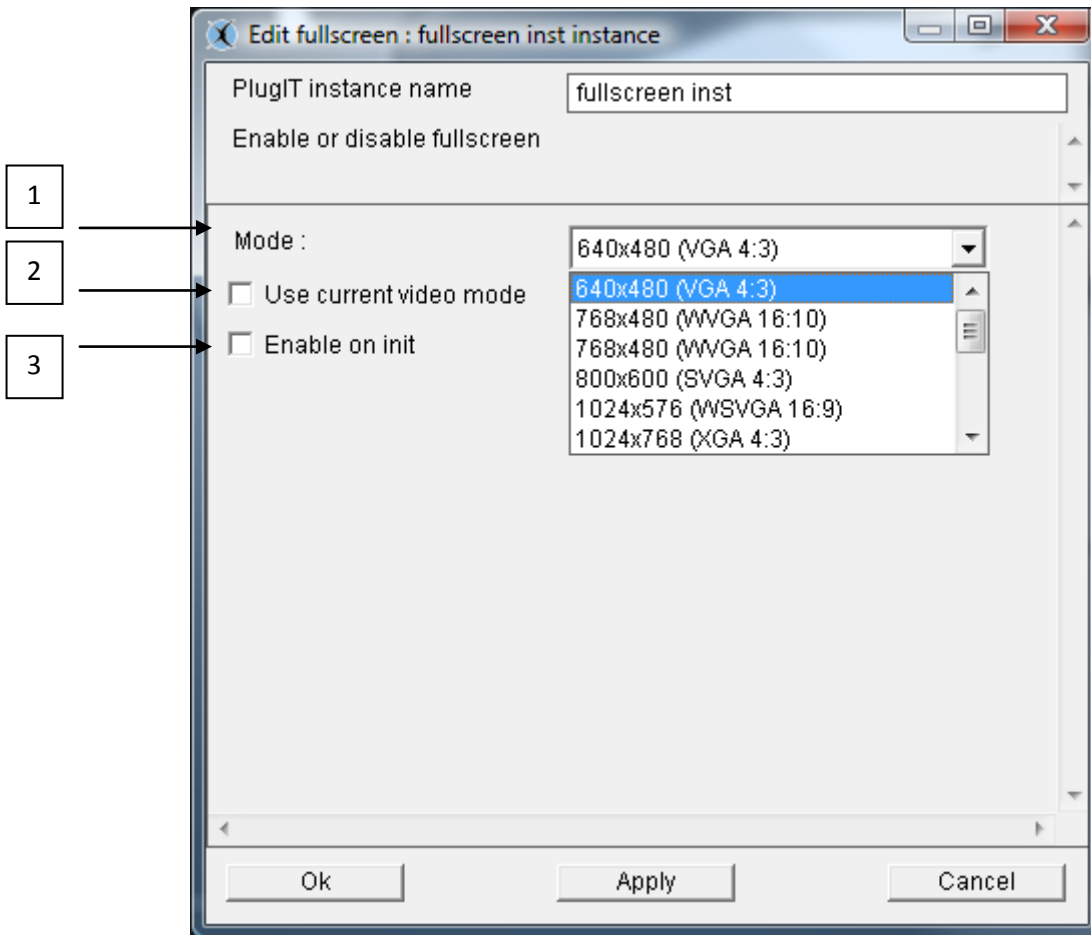
2°/ Definition of the distance between eyes

3°/ Focal value

4°/ Activate at launch.

PlugIT: « fullscreen »

This plugIT allows you to run your application in fullscreen mode..



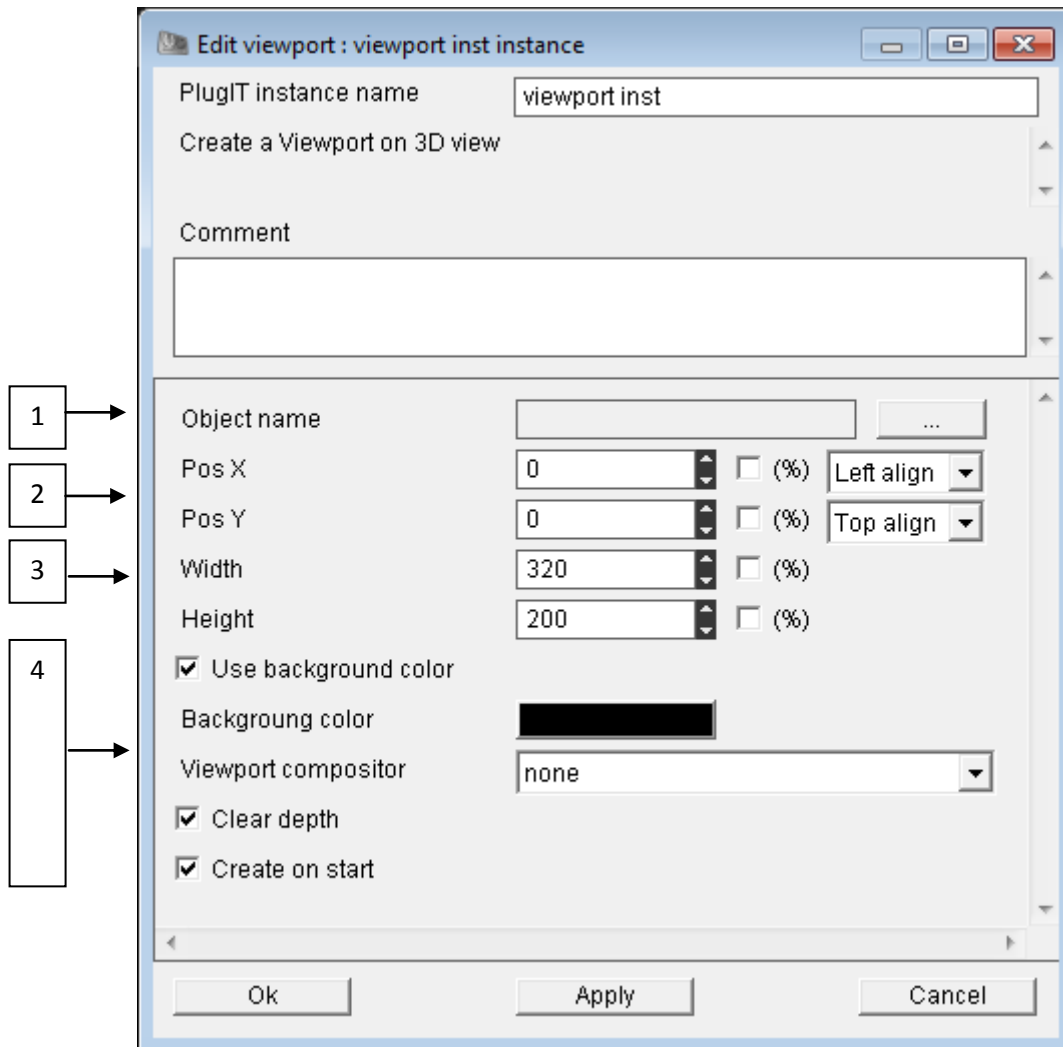
1°/ Select the fullscreen mode and resolution

2°/ Select the current video option

3°/ Activate at launch

PlugIT Viewport

PlugIT viewport allows creating a viewport, it applies on a camera.



1°/ Choose the camera for rendering

2°/ The viewport position on the 3D main view

3°/ Size for the created viewport

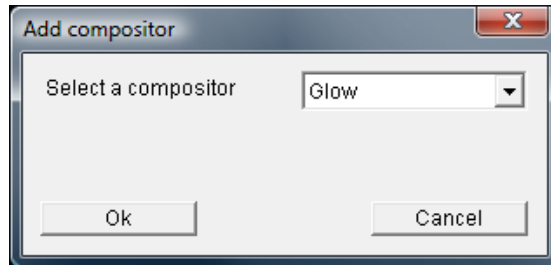
4°/ « use background color »: Allows you to use a color background on the viewport

« Clear depth »: Allows you to choose the use of the depth buffer

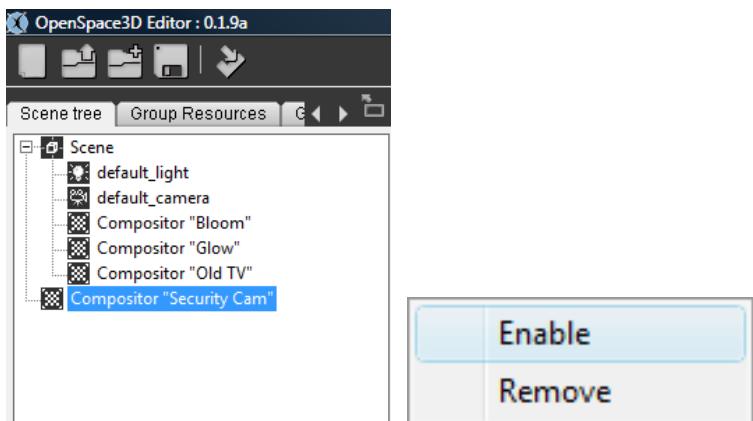
« Clear on Start » : Create the viewport at start

PlugIT Compositor

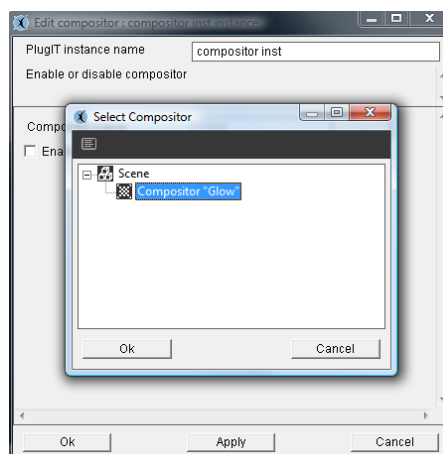
PlugIT Compositor allows to apply a compositor at the scene to change the record. Adding a compositor: Make first a right click on the Scene tree and then "Add Resources" Browse your PartitionLocalUsr to select all the resources of the compositor: program - script - Textures Now you can "Add compositor" which will offer the compositor you just add the resources



Validate: Your compositor Glow was added to the scene and activate it by right-clicking:



Do not activate if you want to activate the compositor with a plugIT). In Scene Group, add the rendering plugIT: Compositor And select the compositor




Check "Enable on Init" (do not check if you want to activate the compositor with another plugIT).

Debug et Log

During the development of an application under OpenSpace3D, it could be very useful to enable debug mode to know step by step the progress of our features. So, this can help to determine a dysfunction in the application.

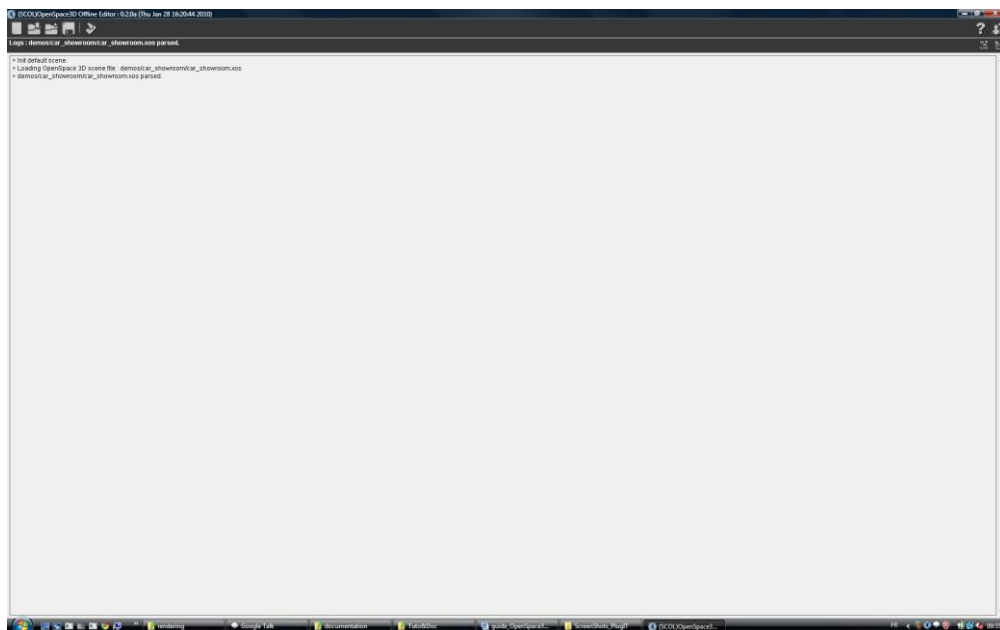
For that purpose, there are tools which are being situated below on the right of the editor and corresponding to 2 icons:

Window Log: 

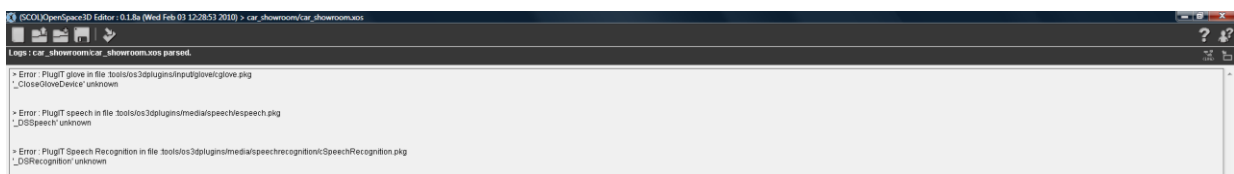
Activation of link debug: 

Windows log

This window shows all the events relative to your development (loading of an object, recordings and an export of your scene)



As if, there are loaded in the OpenSpace3D editor, plugITs could not be initialized. In that particular case, the log window will inform you.



It is possible to reload plugITs without restart OpenSpace3D by right clicking on the plugITs Edition window and choose “refresh plugITs”.

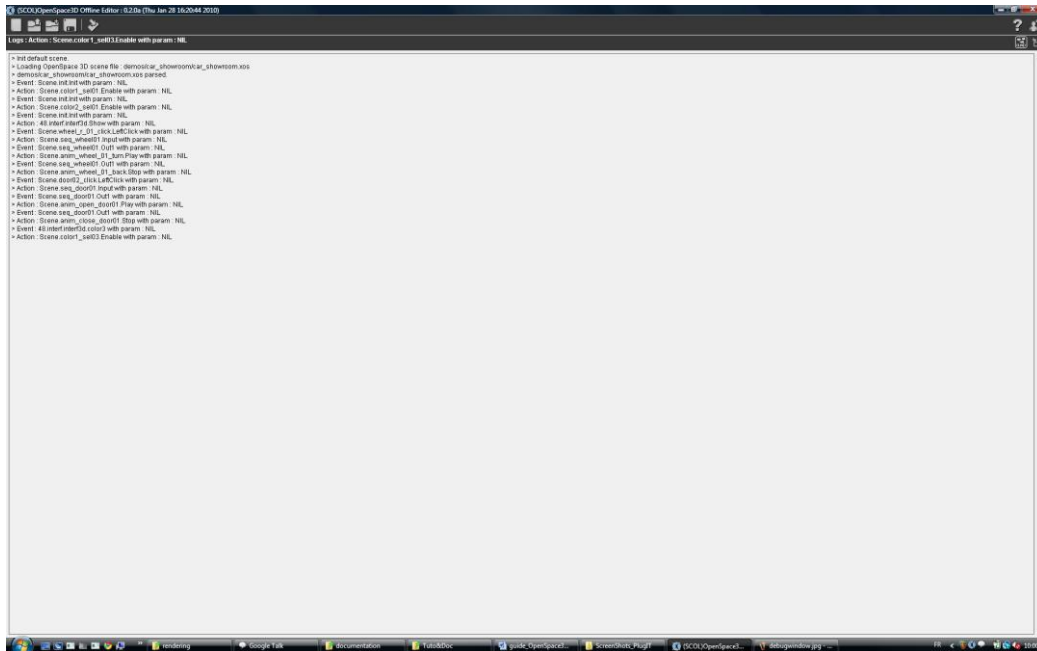
The activation of the debug for the follow-up of the links

By marking this option, it allows you to follow all the actions / events at the level of your application.

This information will allow following in real time the development of your application.

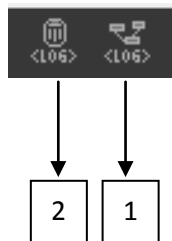
Be careful, these «debug “mode could slow down the execution of plugITs. It is very useful to allow you to determine the cause of a dysfunction during your scene creation.

Informations will be display in the window of Log:



1° / activate/deactivate the logs

2 ° / clear the log window



Advanced use and Definitions

Graphic Resources (3D)

In 3D, a graphic Resource is an element containing data to render a scene or object.

These resources can be:

Mesh: This is the format of the 3D model written in the form of polygons or points that define the visual envelope

Material: This is the set of properties concerning the visual aspect of an object (reaction to light, color, appearance)

The photorealism of a 3D object is enabled through the use of texture

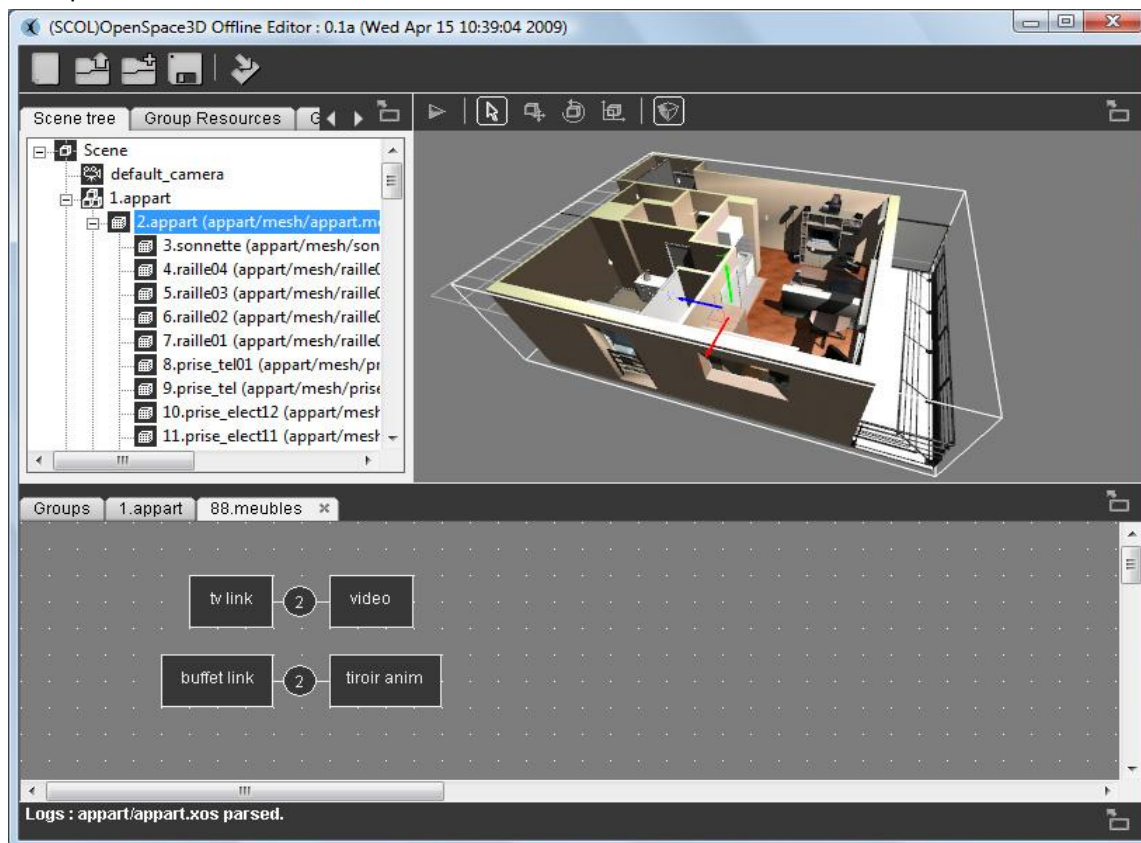
Texture: This is a set of 2D pixels that can be applied on a surface or a 3D volume. In short, it can be seen as deformable plastic wallpaper applied in 3D by specifying the geometry processing operations for each pixel of paper applied on the 3D element. The pixel thus handled in 3D is called texel.

A texture can be defined analytically by an algorithm (procedural texture) or consist of an array of pixels (a bitmap image for example)

- Resource Groups

In the SCOL 3D engine, a resource group is a set of graphic resources that allows integration in a group of objects scene3

Example:

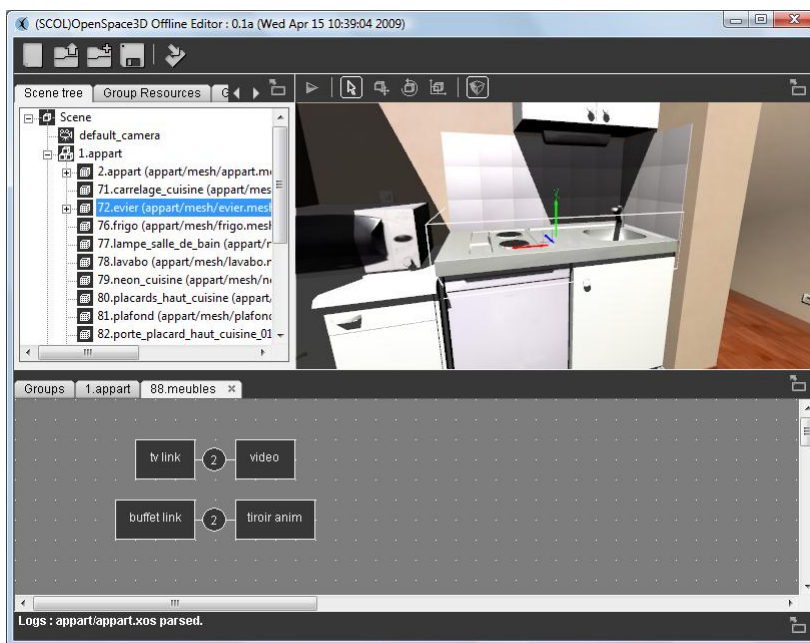
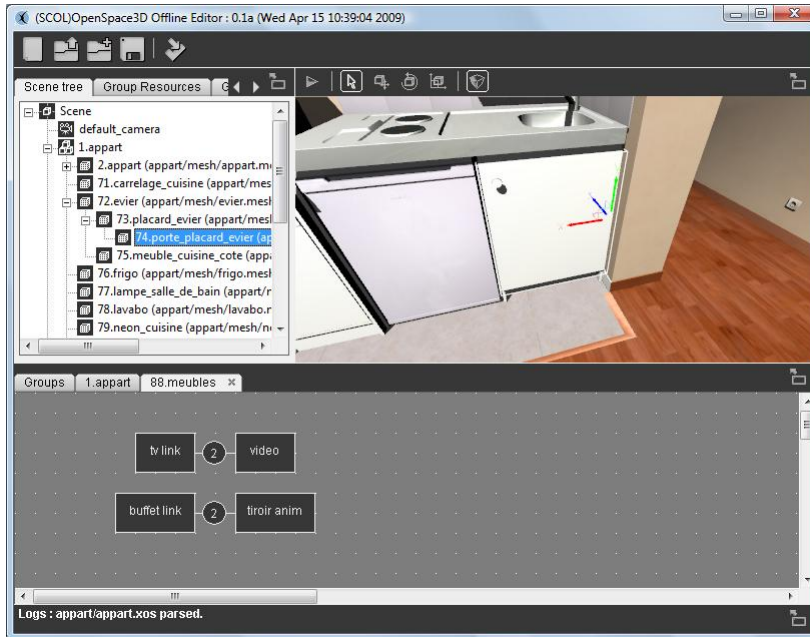


Here 1. Appart is a resource group containing all the graphics for the display of the apartment

- Groups of meshes

Set of meshes forming an object

Example:



In the images, 72.sink consists of mesh: 73.door_closet_sink, 74.door_closet_sink and 75.furniture_kitchene_cote.

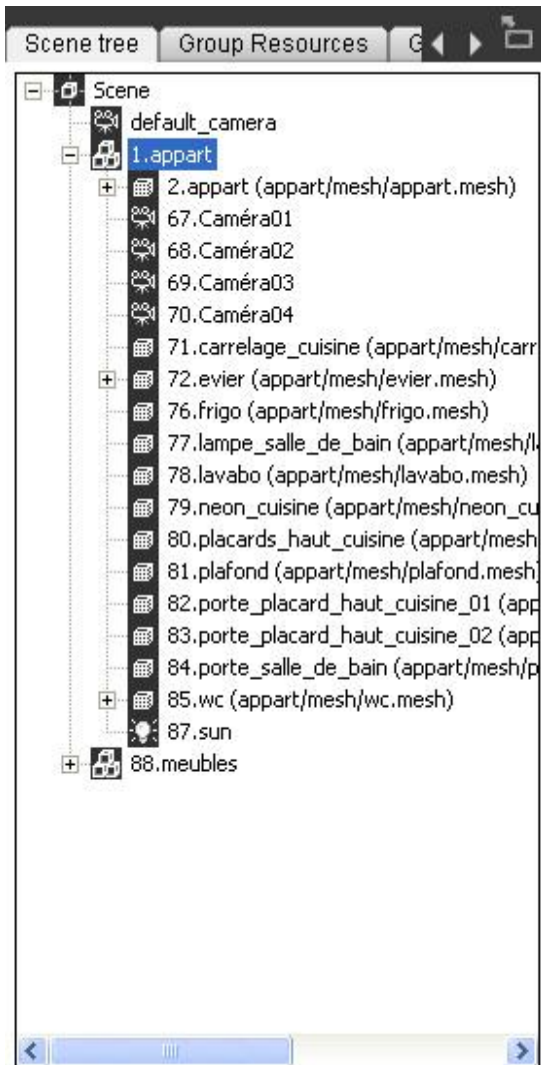
- Scene Tree:

A scene tree is a data structure commonly used in modern applications and computer games. The graph scene is a structure that organizes the logic and spatial representation of the graph scene.

A scene tree is a collection of nodes of a graph or structure tree. A node has many children, but only one parent.

If a transformation is applied to the parent then it affects the whole hierarchy

Example:



- Scene Node

A scene node is an element of the hierarchy of a scene tree it may be a group, object, light, a camera or the scene itself which is the main node of the tree.

- A dummy


It is a blank scene node ie a marker in 3D on which it is sometimes useful to place the other nodes in order to monitor dynamically the scene graph

Group Resources

The interface of this tool is the following :



Here, is selected the group :

 1.appart In the Scene Tree

This group is the representation of a set of graphics that compose it

In the "Resources Group" tab you will find all the resources that can then be manipulated.



You will find the resources on the textures used by the group 1.appart.

Group Meshes

 2.appart (appart/mesh/appart.mesh)



Here is selected the group :

  1.appart In the Scene Tree

This group is the representation of a set of graphics that compose it

In the "Resources Group" tab you will find all the resources that can then be manipulated.

You will find the resources on the textures used by the group 1.appart.

Resources Directories



As explained previously, the 3D engine has to know all the paths relative to the SCOL partition to load the various resources needed to display in the 3D scene the different objects composing the group.

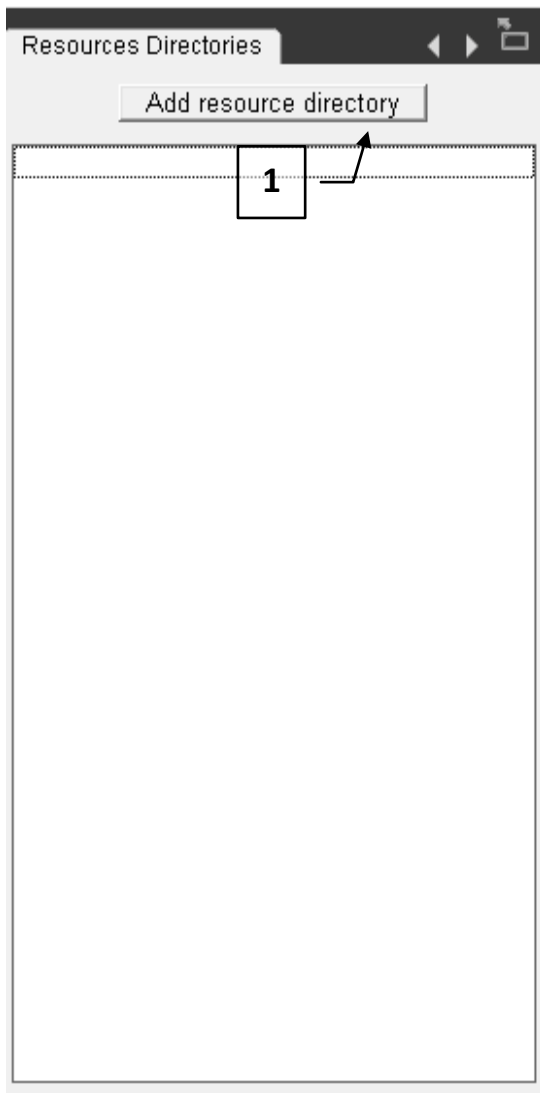
Import an Ogre resource (shaders, texture, material, mesh)

It may occur in the construction of the 3D scene to try to import an object directly without going through an Ogre scene XML.

To do so, you must know that Ogre Max allows you to export . Mesh files and material resources (. Material)

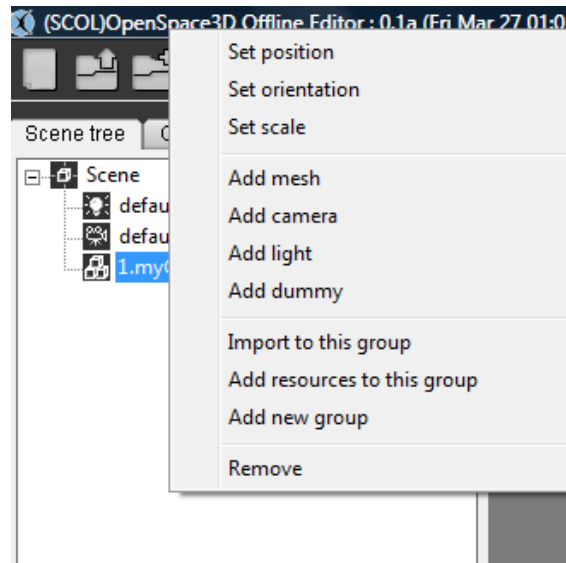
To export these data in the 3D scene or in a group use the menu by right clicking in the scene tree on the scene or the group in which you want to add the object. How to add a resource consists of several steps that must be followed for the proper conduct of the import of the object.

1 / The Ogre 3D engine needs to know where the resources to be loaded should go

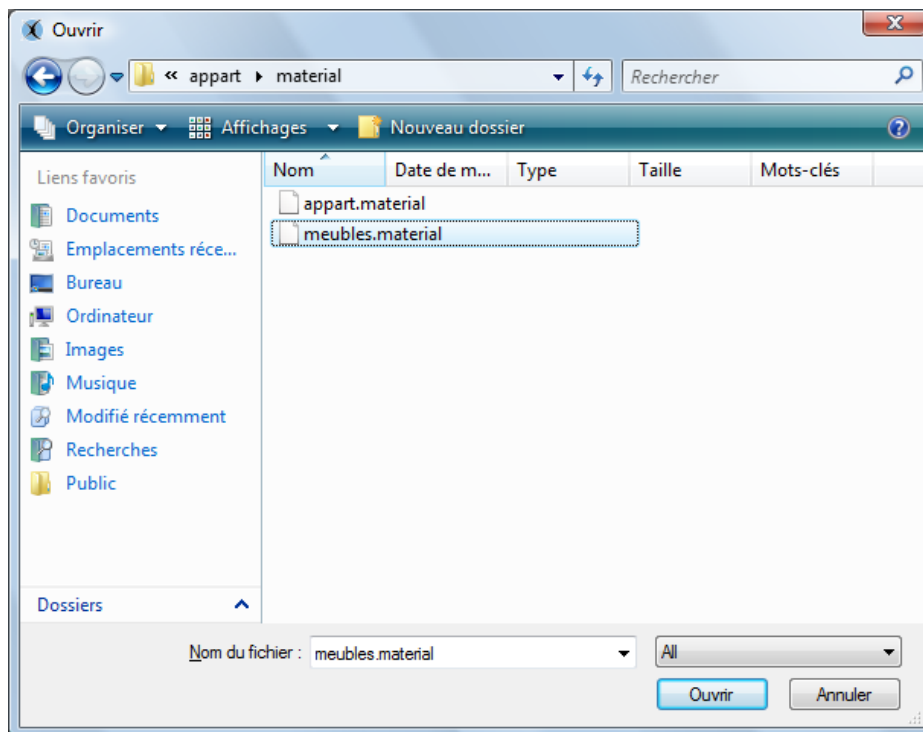


Thus, in the tab, Resources Directories it is necessary to add ALL paths (relative to the SCOL partition) where the different graphic resources will be found (. Program. Material, textures ...)

2 / The next step is to add graphic resources (.program and. Material)

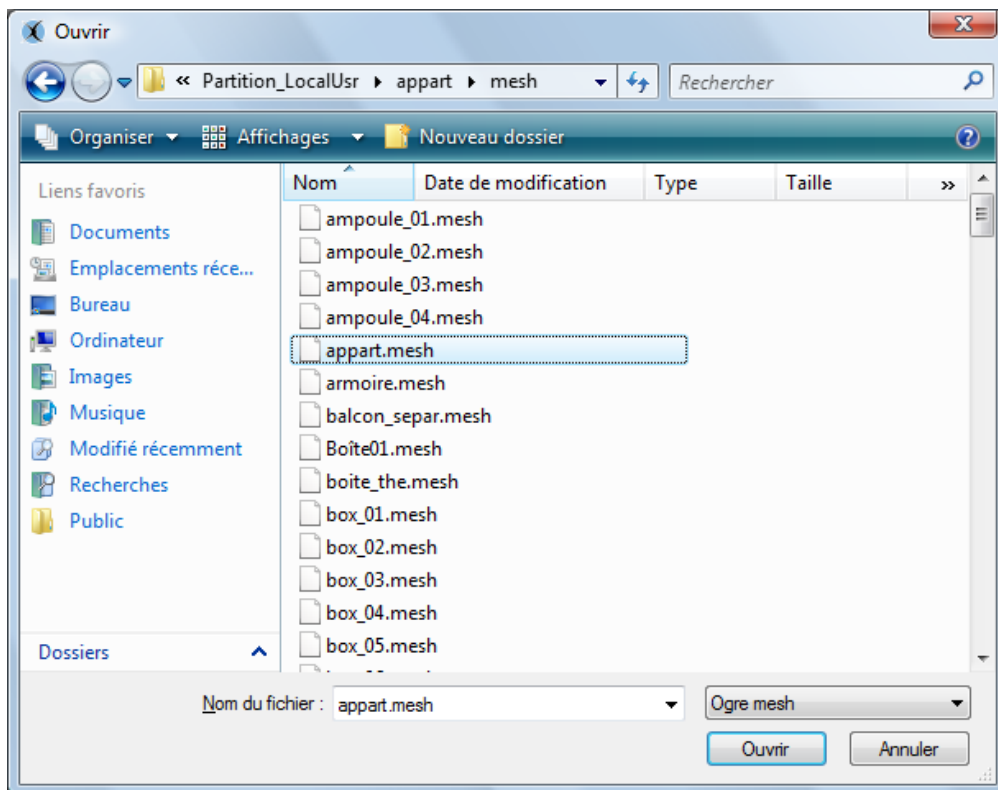
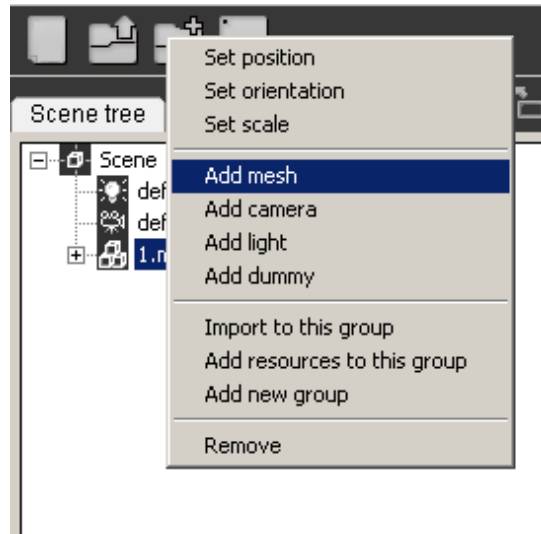


This tool will open the dialog box:



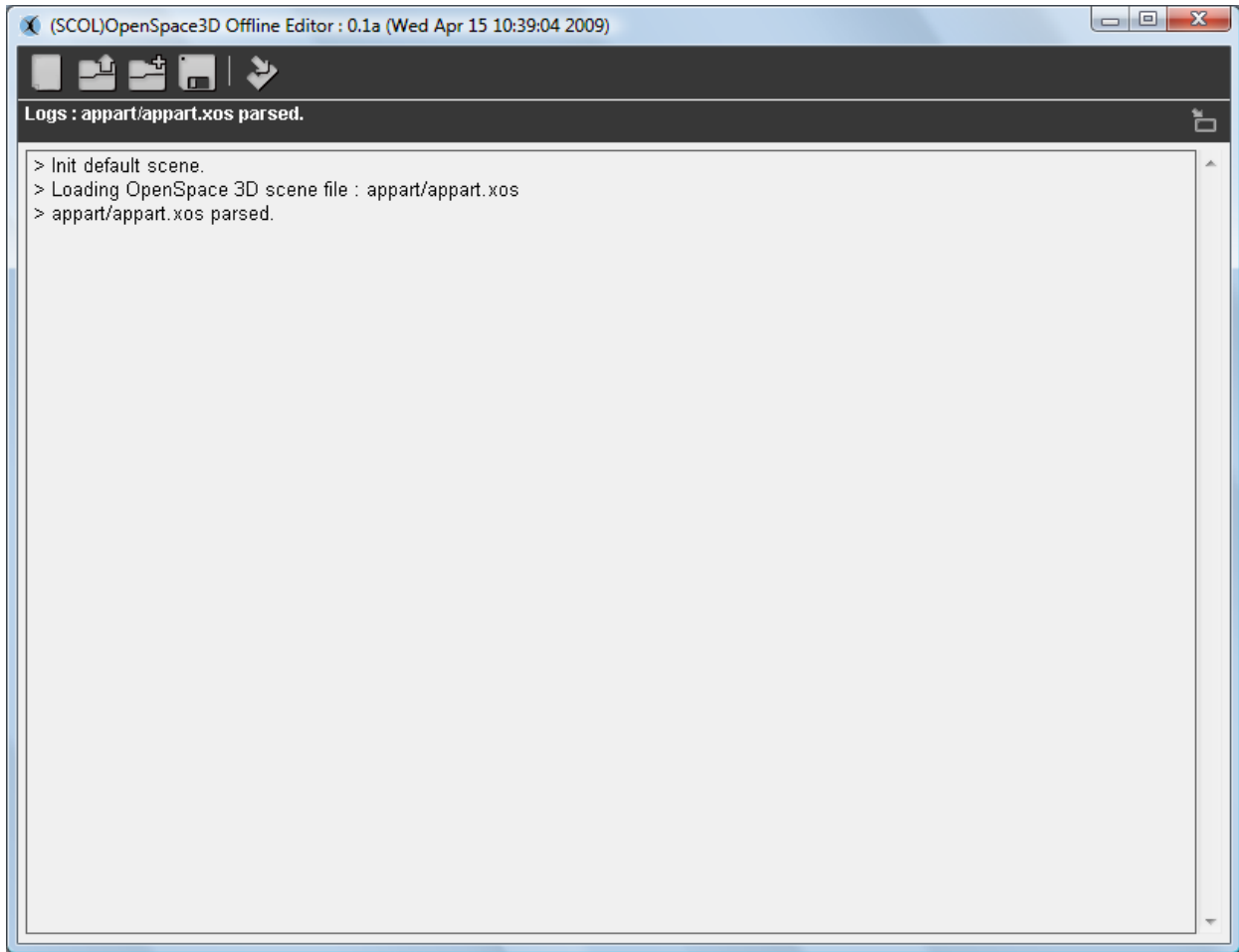
This is where we need to go to load the resources (. Material. Program)

3 The last step is to simply add the Mesh file corresponding to the object you want to load into the scene or group



The graphic resources have been previously loaded with steps 1 and 2, the 3D object will be displayed in the center of the scene with all its material data, program and textures.

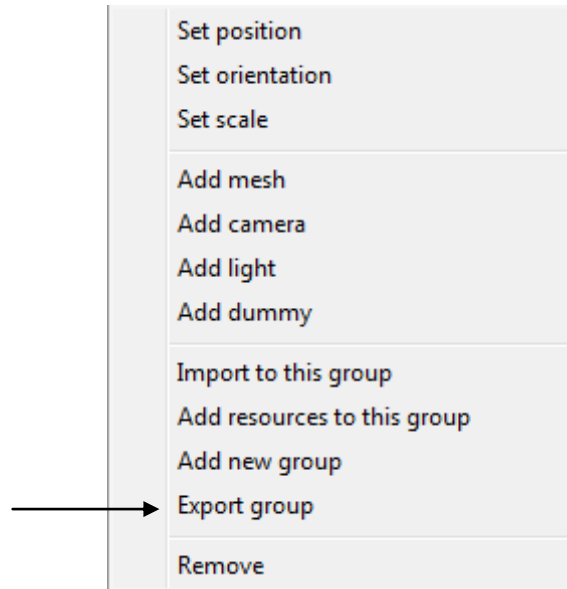
NB: It is possible by displaying the log tab to see if any error has been issued



Note 2: The tabs " Group Resource " and "mesh resources" in the hierarchy and 3D information area are used to verify if our resources have been loaded into the desired group.

Group Export

This feature is important when we wish to develop a consequent project bringing in several persons. So, it is possible to export a group that is all the 3D objects and the resources the constituent but also all the plugIT and the features of this group.



So, from another scene we could import the group directly via the addition of the .xos corresponding to this group. (See import Scene)

Definitions and advanced use of the parameters

- Light

In 3D, a light is used to illuminate the scene.

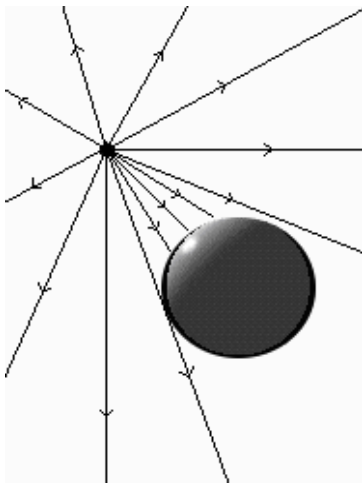
A single and ambient light and is defined in the scene (this is a global luminous intensity)

Then other types of lights are available and will behave differently on the rendering of the scene:

- The omnidirectional or point lights:

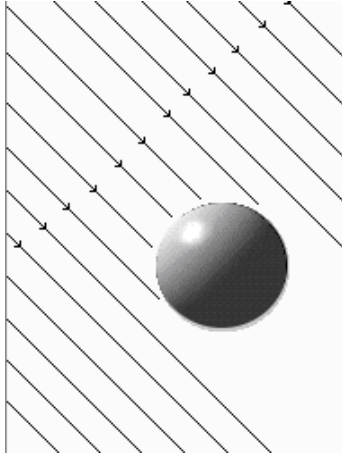
These lights are represented by a point that allows the light intensity to diffuse in all directions.

Their representation in the physical world may be for example the bulb.



- Directional Lights:

These lights are represented by a vector defining direction, a directional light is often equated with the sun because it corresponds to a light source at infinity whose intensity comes with a given direction.

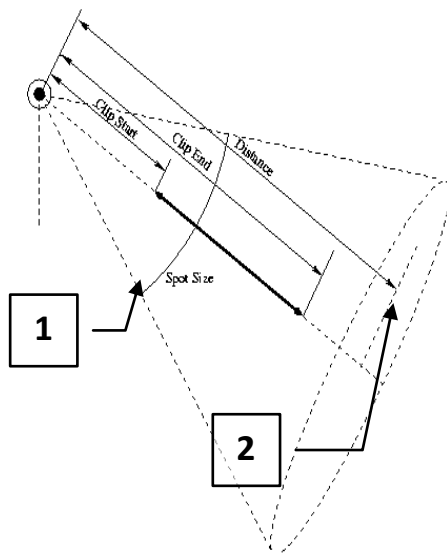


- Spot Lights:

As their name suggests these lights can be equated with bulbs "Spot."

Several parameters characterize them:

- 1 Their opening or spot size that corresponds to an angle of diffusion (the type of omnidirectional light is a special case of spots with an aperture of 360 °)
- 2 The far clip or Range is the distance of illumination of the light. This value determines the distance from which the light no longer illuminate.



Attenuation

Attenuation of light in 3D is the way the light intensity will decrease, depending on the distance.

This data is defined with the following formula:

$$attenuation = \frac{1}{k_c + k_l \times d + k_q \times d^2}$$

With d: distance to light

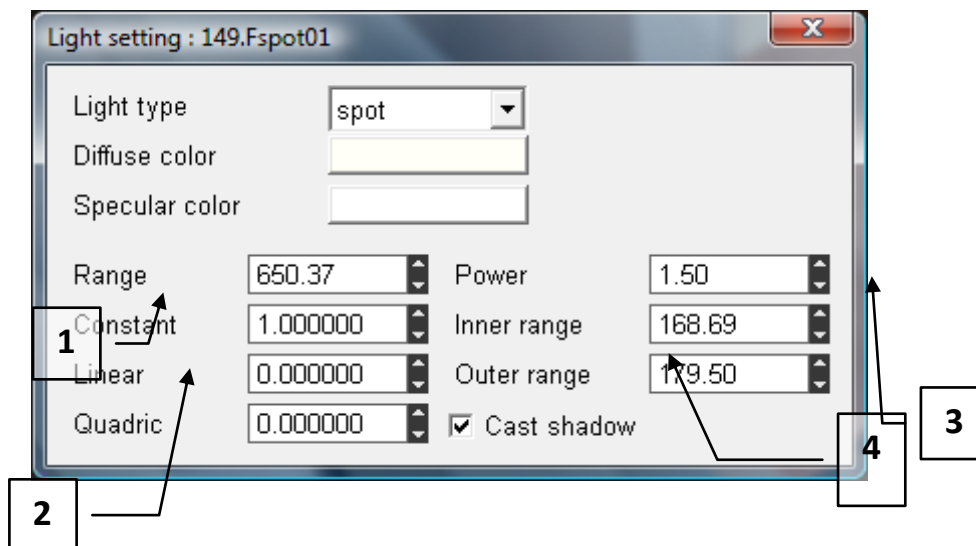
kc: constant attenuation coefficient

kl: linear attenuation coefficient

kq: quadratic attenuation Coefficient

Therefore, following the variation of the distance, the coefficients will be more or less influential (except for kc which is an attenuation constant that defines therefore the power of light intensity)

Link with the Edit Setting lights in OS3DEditor:



1: Distance to which the light illuminates

2: These are the coefficients of attenuation of light

3: This variable is active only in the particular case or HDR is enabled on the scene

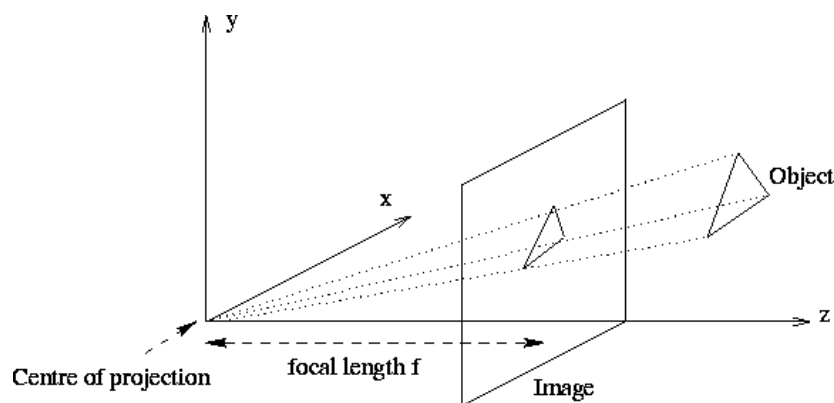
4: Inner and Outer Range define the aperture angle of the spots

- The Cameras

In 3D, camera is a virtual representation of a real camera with parameters identical to a physical camera.

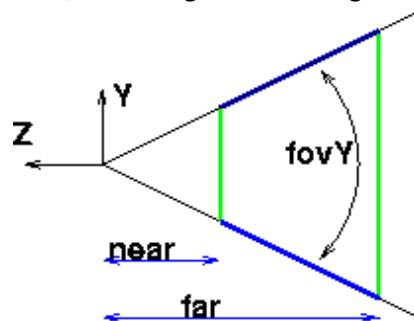
Thus, a 3D camera is the eye of the user allowing the observation and the projection of the 3D world on a 2D image representing the observed image.

The usual representation is called the "pin hole camera model



Focal length: The focal length of an object is defined by measuring the power of convergence, it is related to the optical system because it defines the power of convergence or divergence of the lens of the camera model.

Thus, the change in focal length causes a deformation of the 3D environment.



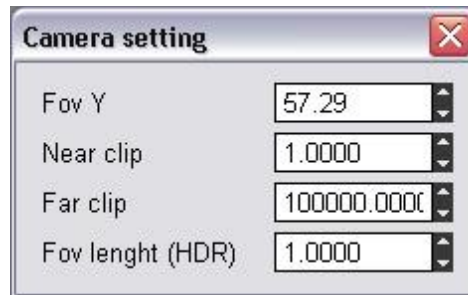
Near Clip and Far Clip: These two parameters correspond to the area where the camera will calculate the rendering (They define the frustrum)

If an item is beyond the far clip distance then it will not be made in the image.

Similarly, if an object is before the near clip distance it will not be delivered on the final image.

Fov Y: For Field Of View, which defines the angle of the camera as shown above.

Link with cameras Edit Setting in the OS3DEditor:



In the edit setting of cameras are the parameters for establishing the model of camera used.

- Shadows

Shadows and shading represent a particularly active area in 3D. Shade is at the heart of realism to a 3D scene. Shadows result from an interaction between light and objects.

Depending on the environment and the type of light you use one or the other projection techniques of shadows.

The principal methods of shadow projection are:
Stencil Shadow Method and Texture Shadow Method

- Stencil Shadow Method:

In the overall functioning of stencils Shadow volumes, three actors:

Light

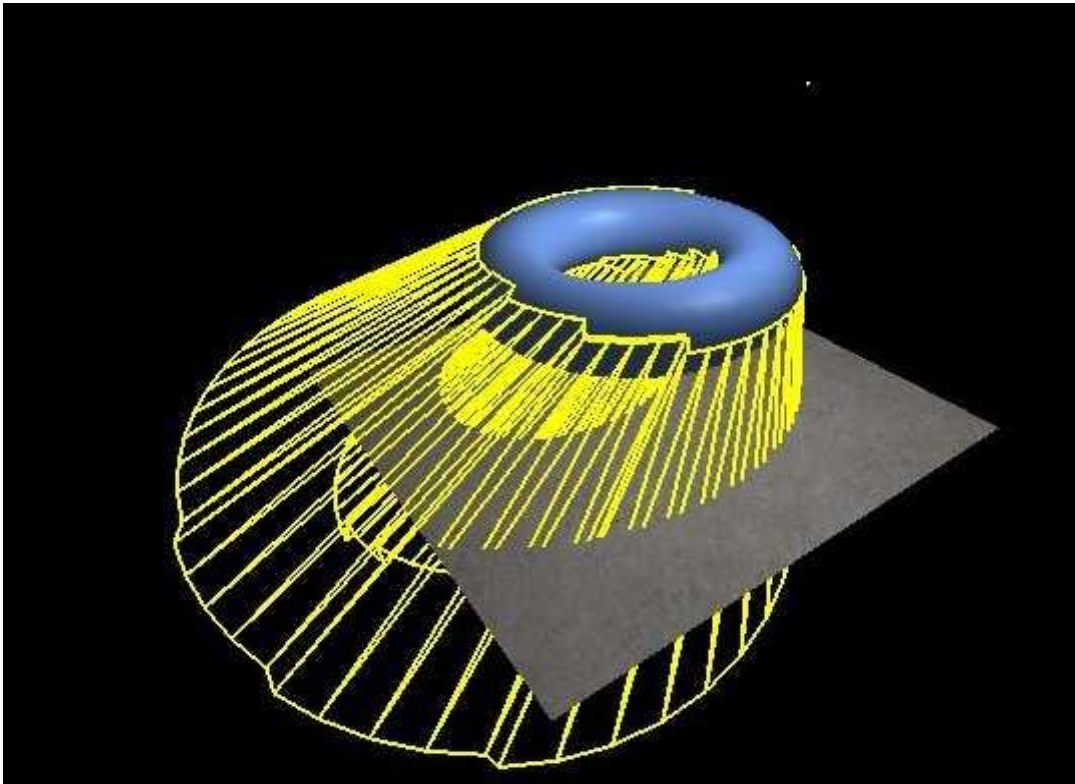
The Shadow-caster

The Shadow-receiver

The light is usually a light source or type positional OMNI.

The Shadow-caster is the object that will block light and create the shadow itself. The Shadow-receiver is an object that "receives" the shadow, or more precisely is the object (whole or part only) that is deprived of light.

The shadows are so-called volume shadows simply because the Shadow-caster will create a shadow volume in which the objects will not receive light.





– Texture Shadow Method

```
textureModulative  
textureAdditive  
textureAdditiveIntegrated  
textureModulativeIntegrated
```

The principle is quite simple and natural. Indeed what makes a shadow on an object?

The answer is obvious there is another object (or itself) to prevent light from reaching the shaded part. In other words, there is a shadow over an area if there is another object closest to the light and which is on the path of the radius that should illuminate our area.

The principle is simple and very logical. Using Pixel shader we will see if there is an object that is on the path of the ray of light and if so the area is shaded.

The basic algorithm will be done in two stages.

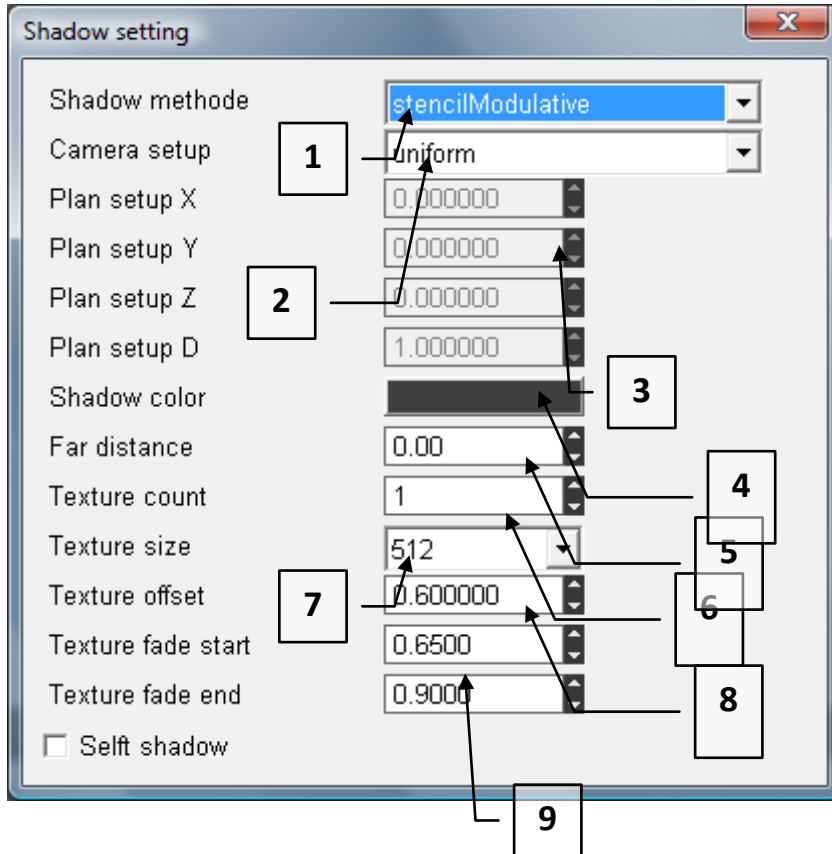
The first is to render the scene in a texture (render target) by taking as camera the light. If you have several lights, it has to be done for each of them. For this rendering, we will store as output, only what we need to know, ie the depth.

In the case of an omnidirectional light, all this is complicated, it must pass through a cube map for the depth of everything that is illuminated by light.

The second step is to display the scene normally and deal with the shadow using pixel shader.

The pixel shader needs the coordinates of the point in the light mark. Then we have all the informations to know whether the current pixel is shaded or not. Indeed, if the depth found in the depth texture with the coordinates of the current pixel is smaller than (in the light mark) that of the current pixel then the pixel is shaded.

Link with the Edit Setting of the shadows on the scenes in the OS3DEditor:



1 / To select the type of technique of the method of shadow (stencil or texture)

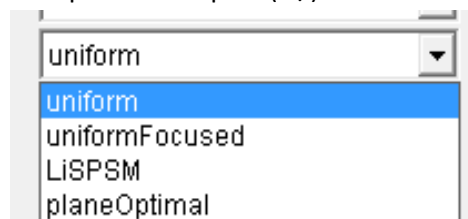
Additive and modulation allow you to choose how shadows will mix with the rest of the 3D rendering

Additive will be added pixels

By modulating the pixels are multiplied

2 / Option in the case of textures Shadow, to optimize the calculation of shadows and a greater efficiency of the calculation of shadow

If the choice is done with an optimization in "optimal plan" then it is necessary to set the parameters of the optimization plan (3 /)



4 / Lets just set the color of shadows cast

5 / To set a distance from which the objects will no more project shadows (Remote Camera / Object). that optimizes the scenes in making no calculation of projection beyond a given distance.



6 / Sets the number of texture shadows to be used. One shadow is often defined by light in the scene.

If we have 3 lights in the scene so we set the texture count to 3

However, it is recommended to optimize the scene not to set a texture count too large (all the lights are not required to project shadows ...)

7 / Set the resolution of the texture shadows ie the resolution quality of the shadows.

8 / Defines an offset on the rendering (an offset is a shift of the texture shadow)

9 / These parameters will define the attenuation of the shadow on the edges from fade Start to fade End where projection will stop.

Note: Depending on the type of light in the scene the shadows will have a better or worse rendering. For example, stencils Shadow work very well with directional types of lights while shadow textures works better with spot or omnidirectional types of lights.

Note 2: It is impossible to define multiple methods of shadows for a scene.

SkyBoxes and scenes environnements

The scene files loaded in openspace 3d, exported via Ogre Max are files containing XML tags defining the objects, groups of objects but also the environment variables.

These environments define for example the type of method of shadows in the scene or the Fog settings, background color ...

Anything that refers to the scene directly.

Thus, the skies are also variables.

There are 3 Ogre types of Skies:

- The skyboxes: For this kind of Sky, the sky is represented as a box encompassing the scene
- The SkyDome: For this kind of Sky, the sky is represented by a dome
- The Skyplan: For this type of sky, the sky is represented by a plan located above the 3d scene.



Physics Engine

- Newton Physics Engine :

The physics engine used in OpenSpace3D is Newton.



A description of it is available on wikipedia :

http://fr.wikipedia.org/wiki/Newton_Game_Dynamics

- A partial integration

The integration of the newton physics engine in OpenSpace3D is partially realized. Some functionalities will be available in a next version of OpenSpace3D.

However, OpenSpace3D is able to manage basic physics to create realistic effects in developed applications.

- Definitions :

World: It is the representation of the physic world associated with a 3d scene (it has a size and basic options like the gravitational constant)

Body: It is the physic shape applied on a 3D Object to optimize collision calculation

Shape: It is a particular shape automaticly claculate on the base of the 3D mesh information

Collision Tree: Exact collision shape applied on static object (no Strength) but with collision detection (ex: House Walls, floor...)

Architecture model: Level of calculation precision. It will be used by the physic Engine for the simulation

Solver model: Level of calculation precision for the result of the different strengths applied on an object

Solver model: Level of precision for the friction model.

Frame Rate: Speed of the physic calculation.

Physic material: Physic materials applied on 3D Object. Ex : Wood, iron...

Angular Damping: Definition of the position for the center of mass of the 3D Object.

Contact: When two objects (associated with physic material) have a collision.

N.B: The reaction values (elasticity, softness, friction...) when two bodies collide is the result of the collision of two physic material.

Elasticity: Value for the elasticity between two physic materials

Softness: Value corresponding to the depreciation between two physic materials

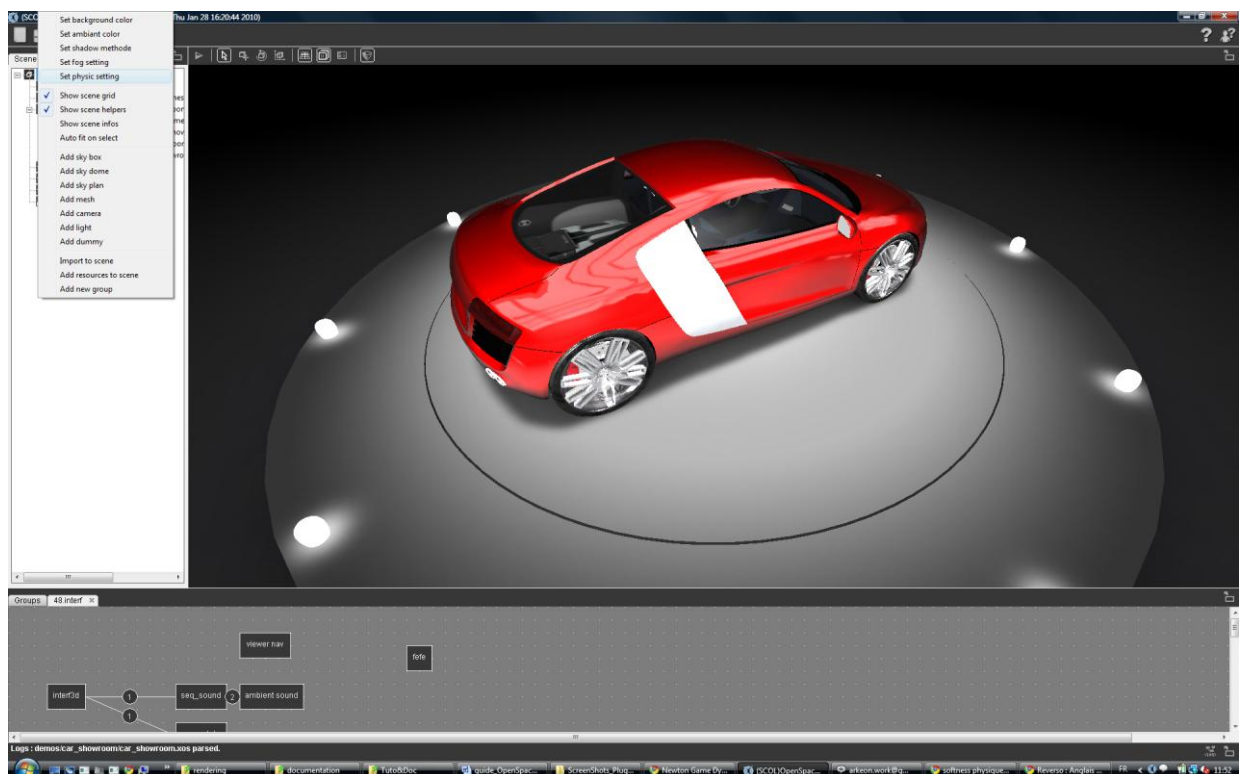
Thickness: Value corresponding to the thickness of the physic contact

Static friction : Value for the static friction

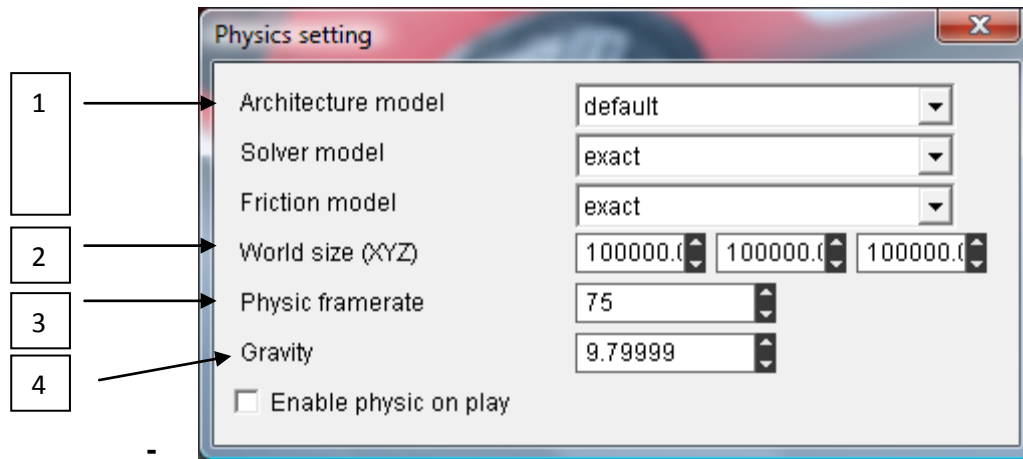
Kinetic friction: Value for the kinetic friction; It means a coefficient depending of the movement of the 3D Object during the physic contact.

- Physic parameters on OpenSpace3D

Scene Level :



Using a right click (In the scen Tree) you could have the General configuration of the physic options for the scene



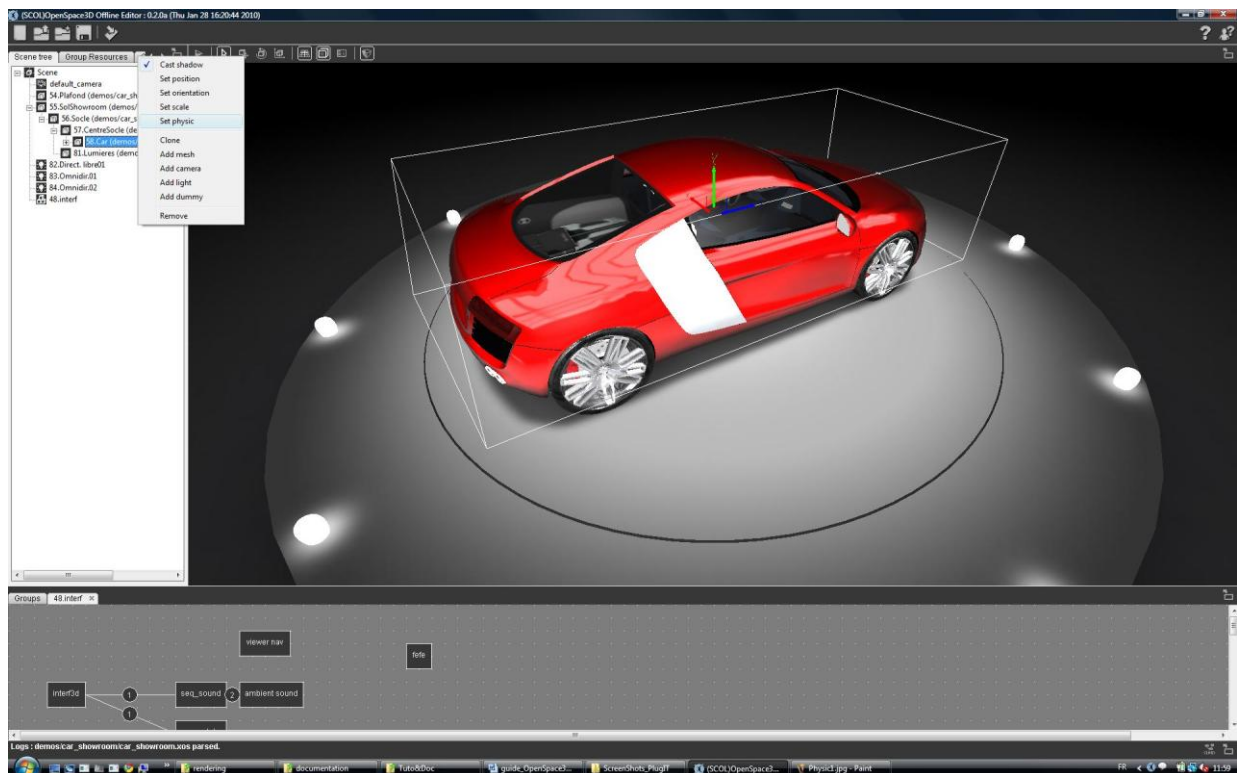
1 °/ Calculation options of the physic Engine

2 °/ Size of the physic world

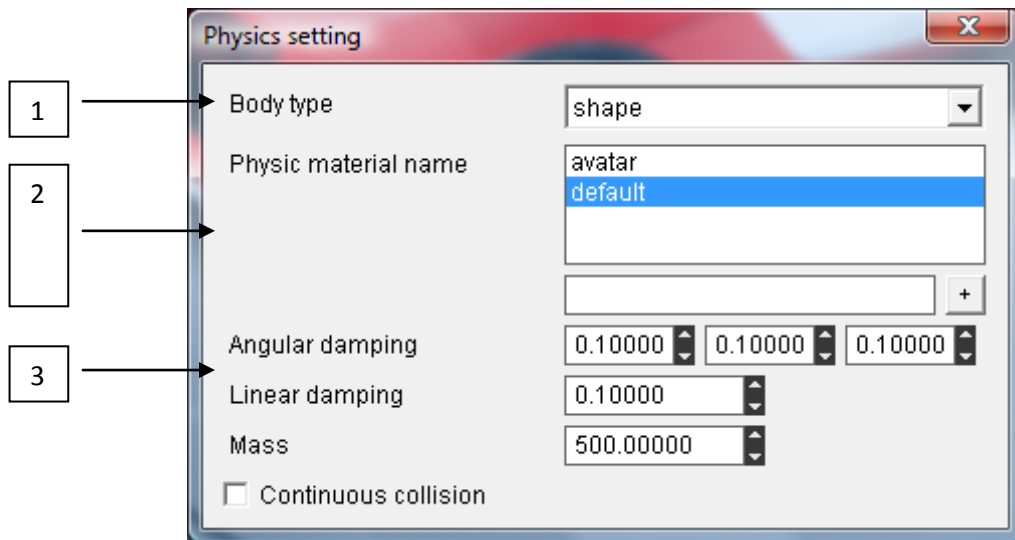
3 °/ Update value for the physic world

4 °/ Value of the gravitation (On earth : 9.81)

Object Level :



Using a right click (In the scen Tree) you could have an access on the 3D Object Physics options.



1 °/ Body type

2 °/ Management of the physic materials : You choose a material already presents in the scene or ad dit by a new name

3 °/ Initial value for local informations on the body applied on the 3D Object.

Contact and Support

Don't hesitate to contact us for any information (dysfunctions, proposition of add-ons, formations...) using the form at:

<http://www.openspace3d.com/support/>

The forum where OpenSpace3D developers could help you quickly:

SCOL technology forum: <http://www.scolring.org/>