IIViMaT

VR Interactive Toolkit for beginners (support : iivimat@univ-lille.fr)

- Documentation -

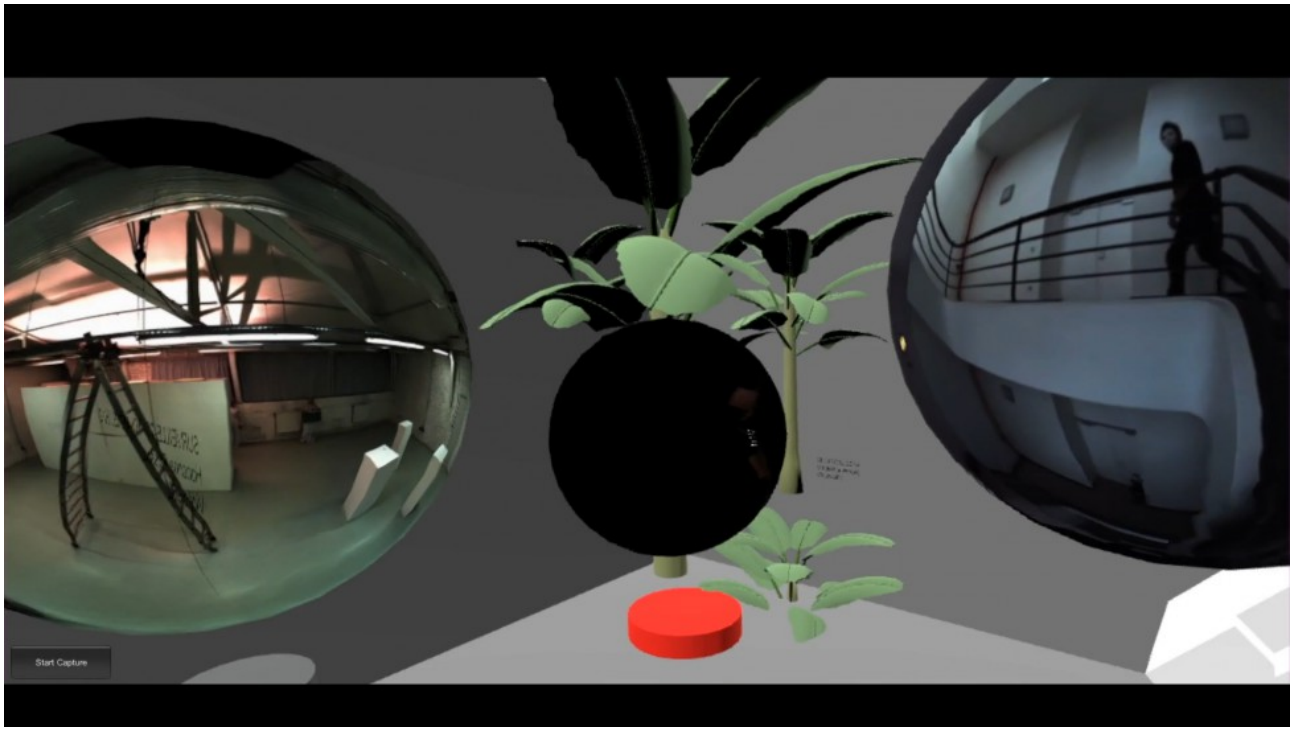
.....What is it ?.....

IIViMaT stands for *Immersive and Interactive Video Making Tool*.

It aims to help people building interactive environment for narrative purposes.

It is a Unity plugin, allowing you to create stories, without worrying about code. You will be able to create interactions thanks to a custom node-based editor (action – reaction model).

It is part of the MAVII Project. You can learn more here : http://www.cristal.univ-lille.fr/oeuvres-et-recherches/2020/03/25/mavii-mediations-audiovisuelles-immersives-et-interactives/



ŒUVRES
ET RECHERCHES

Université
de Lille
1 SCIENCES
ET TECHNOLOGIES

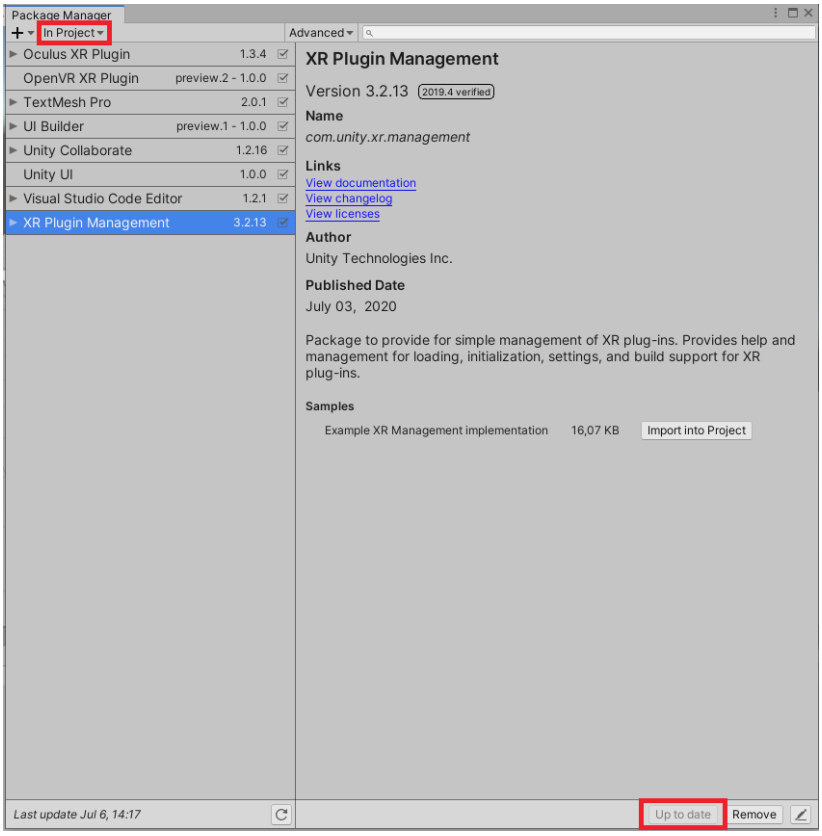
.....How to use it ?.....

SETUP

After you downloaded and imported the plugin through Unity's Asset Store, you are ready to use it.

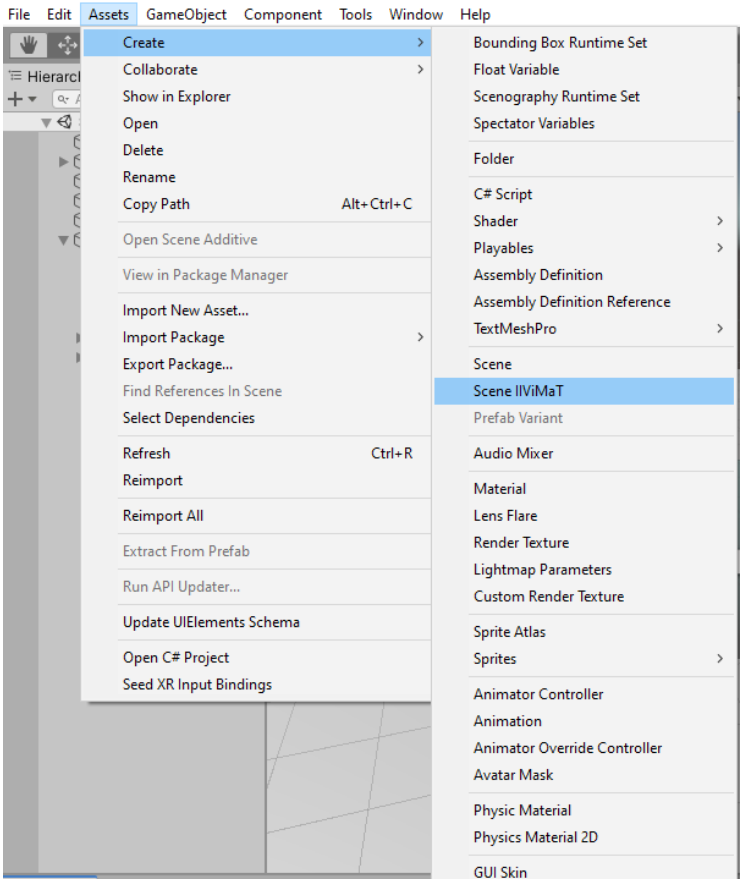
IIViMaT comes with a dependency to Unity's XR Management. This package should automatically be downloaded and up to date. (You can check it in Window > Package Manager, the package should appear in the In Project section. For more information on how to setup Unity's XR Management, please refer to this link :

https://docs.unity3d.com/2019.3/Documentation/Manual/configuring-project-for-xr.html)

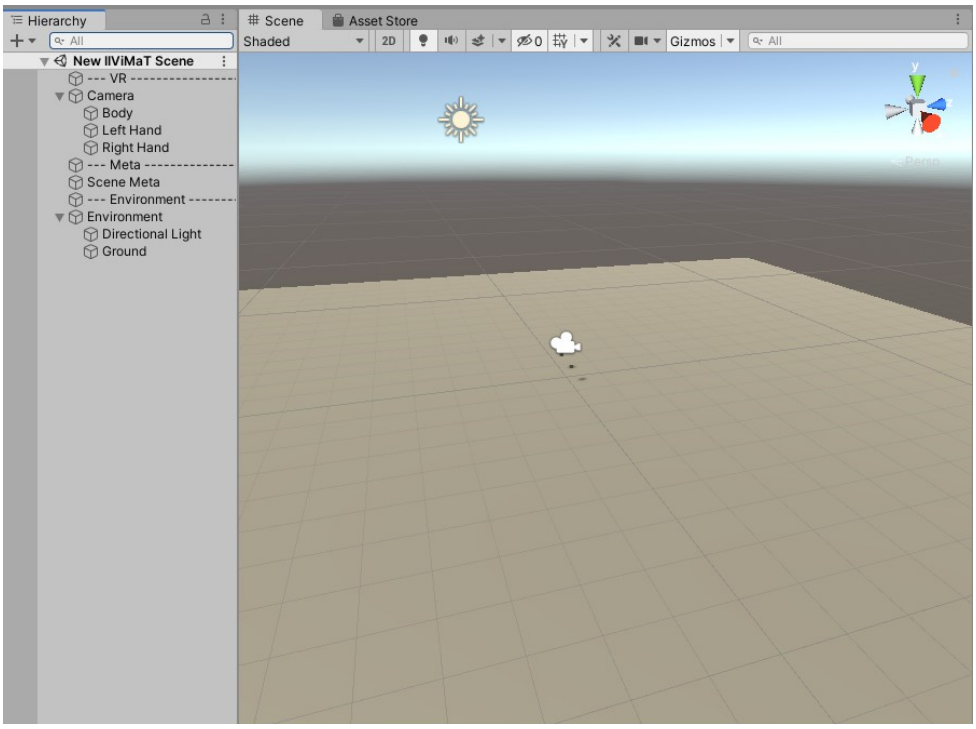


First thing to know is that IIViMaT comes with a custom scene you have to use to make the environment interactive.

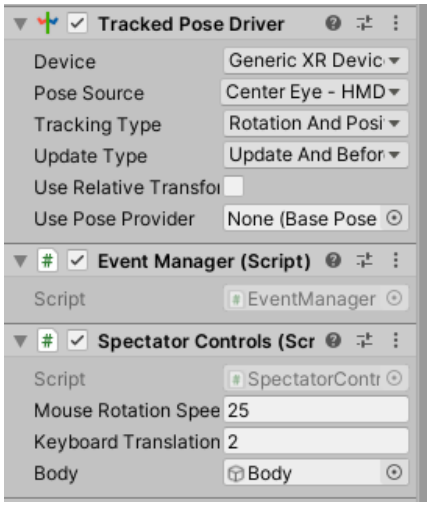
You can create a new IIViMaT scene by right-clicking in the Project View > Create > Scene IIViMaT or directly in the Assets in the top toolbar > Create > Scene IIViMaT.



Let's review what is in this scene (Top-down in hierarchy) :



- The --- VR --- (and others --- Something ---) is just an empty GameObject which is used to make the hierarchy clearer.
- The Camera GameObject represents the headset you will wear (Tracked Pose Driver Component – Device should be set to Generic XR Device ; Pose Source should be set to Center Eye ; Tracking Type should be set to Rotation and Position). If you run this scene without having connected any head mounted device, you will be able to walk around your scene with classic first person controls (Spectator Controls Component – You can modify mouse speed and body movement speed). The Event Manager Component is used to trigger some actions (We will see this later in this documentation).

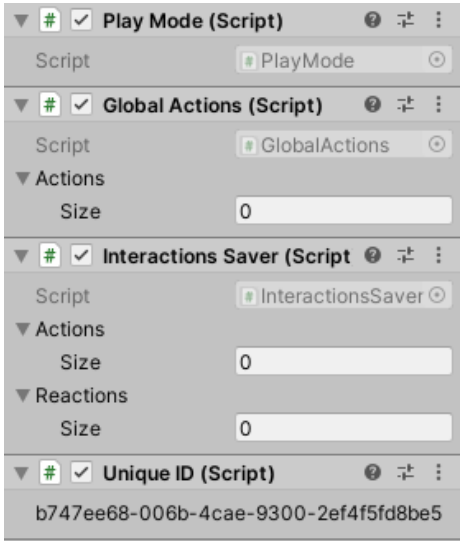


- The Body GameObject represents the spectator physical body. It will be used in play mode to check things like spectator positions, whether or not the spectator is close to something or on a chair. You don't have to change the Body.
- The Left/Right Hand GameObjects represent, well, the hands of the spectator. The Tracked Pose Driver Component's Device should be set to Generic XR Controller ; 's Pose Source should be set to Left/Right Controller.
- The Scene Meta GameObject is used by IIViMaT to retrieve and store data tied to the current scene. You don't have to change this neither.

  Play Mode Component hides feedbacks (which we will see later) when entering play mode.

  Global Actions Component stores all the actions not tied to specific GameObject in the scene.

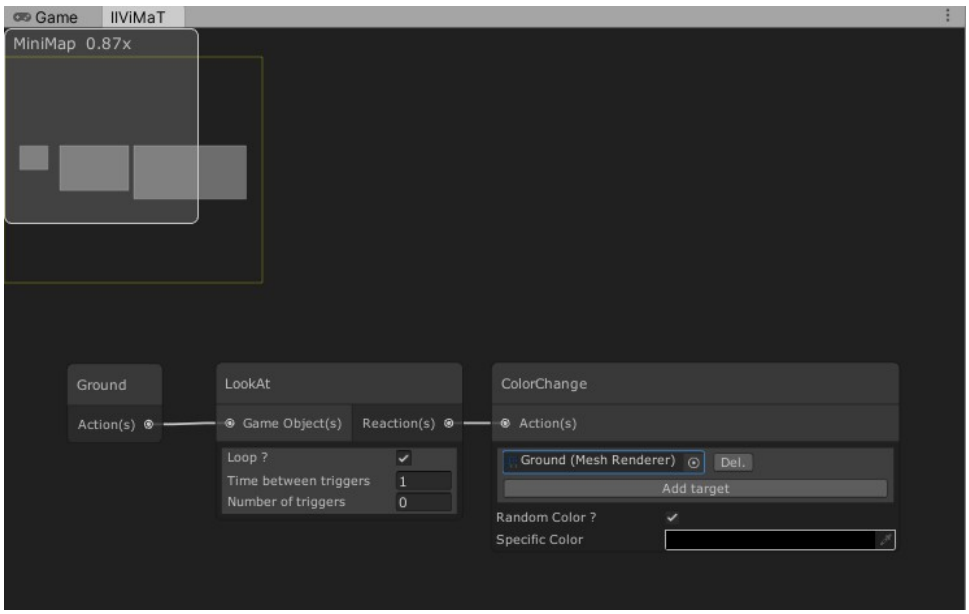
  Interactions Saver Component stores and saves all the actions and reactions of the current scene.



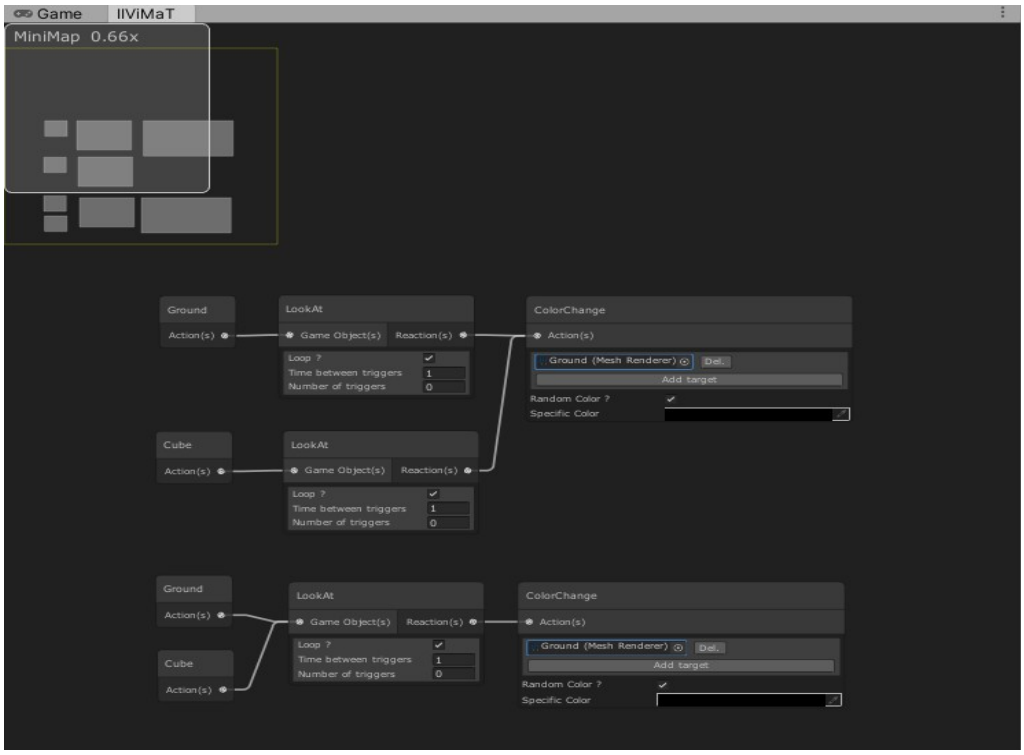
  Unique ID Component, which we will see again later on (on all GameObjects child of Environment), is used by IIViMaT to retrieve the interactions model of the scene.
- Finally, the Environment GameObject has one purpose : Making all of his children interactable. You will be able to edit interactions through a node-based editor, which we will now discuss on.

NODE-BASED INTERACTIONS EDITOR

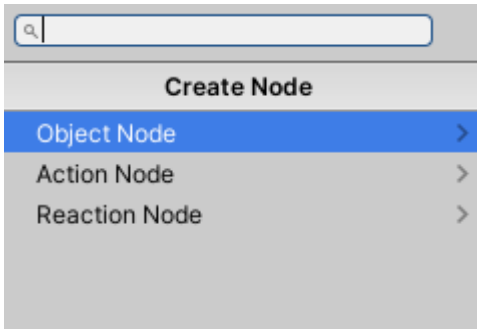
The node-based interactions editor can be open either by going to the top bar Tools > IIViMaT or by pressing Shift+Ctrl+i.



In this example above, three nodes are created. This can be read as « When the user *looks at* the *ground*, a *color change* is applied (onto all the targets) ». What we can learn from this is that there are three types of nodes : Object Node, Action Node and Reaction Node. Every links can be read as « When *Action occurs* onto *Object*, trigger *Reaction* ». You can connect how many nodes you want to the compatible ports. Another example below showing you two ways to do the same thing. Those two paths mean « When the user *looks at* the *ground* OR the *cube*, apply *color change* ».

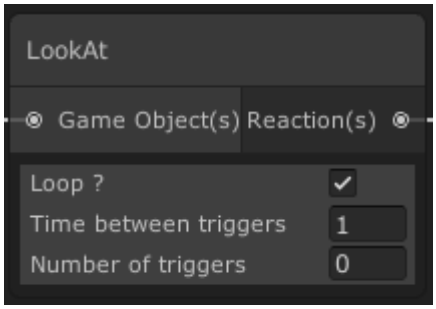


You can create a new node by right-clicking in the IIViMaT window and selecting Create Node. This window will appear. Let's review those three types of node :

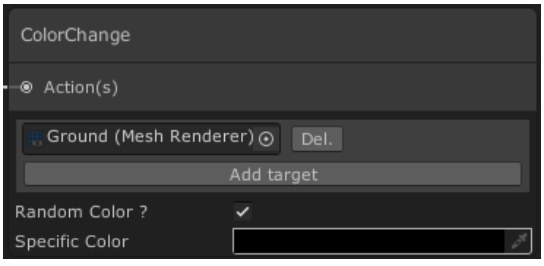


- First, we have the **Object Node**. The Object Node represents a GameObject in your scene you want to interact with, as a *source* of an interaction. What you want to know here is the concept of *source* and *target*. If you want to make a GameObject react in a certain way when an reaction is trigerred, you will have to specify this GameObject as a *target* in the Reaction Node. In another hand, the Object Node specifies the *source* GameObject the actions you linked it to will watch for a positive result. That's why the Object Node has Action(s) as an output. Note that in order for a GameObject to appear as an interactable element (ie. In the Object Node list), you will have to make it child of the Environment GameObject. For example, by default, when creating a new IIViMaT scene, two GameObjects are children of Environment (Directional Light & Ground) and so will appear in this list.

- Then comes the **Action Node**. The Action Node specifies an action the experience will wait for the user to do (Generally, see full details in the tab below). This type of node can be connected to a single or more Object Nodes (or also zero – those are called Global Actions, which we will cover right after) as an input, and one or more Reaction Nodes as an output. There are two types of Action Nodes : Local & Global ones. Local ones have to be connected to one or more Object Node ; not Global ones (Note how all local actions refer to a GameObject in the next section). Every Action Node has at least three parameters : Loop, Time between triggers and Number of triggers. The last two will appear only if loop is enabled. The loop parameter can be read as « Shall this action be raised *as long as* the user does it ? ». If you choose to make this action loops, then two more parameters are available. Time between triggers is pretty straight forward. It is expressed in seconds. The last one, number of triggers, specifies how many times this action shall loop. Note that 0 is infinitely.



- Finally, the **Reaction Node**. The Reaction Node specifies a reaction that affects something in the scene. It will be raised when one, OR more, of the actions it is linked to is triggered. All Reaction Nodes have a list of targets, which you can modify as you want. Those targets represent Unity.Object in the scene, corresponding to the type the reaction affects. For example, the ColorChange reaction will affect Mesh Renderer.



ACTIONS & REACTIONS DETAIL

Action	Type	Concept	Specific Parameters
Project Clock	Global	Will be triggered X seconds after the game / experimentation has begun.	Time (in seconds) before action is trigerred.
Exit	Local	Will be triggered when the user gets away from a GameObject.	No parameters in node, but visual feedback appears as a child of source GameObject in scene (which you can modify the range).
Look At	Local	Will be triggered when the user looks at a GameObject.	
Look Away	Local	Will be triggered when the user looks away from a GameObject.	
Proximity	Local	Will be triggered when the user closes gap to a GameObject.	Same as Exit.

Reaction	Type	Concept	Specific Parameters
Activation	GameObject	Enables / Disables a GameObject.	Activation Enum choice between enabled and disabled.
Color Change	Mesh Renderer	Changes the color of a Mesh Renderer	Random Color ? Shall the color change be random ? Specific Color If no, specify a color.
Transparence	Renderer	Changes the transparency of a Renderer.	Transparency Between 0 and 1 (0 = invisible, 1 fully opaque). Transparency Mode Enum choice between real and virtual.
Visibility	Renderer	Changes the visibility of a Renderer.	Visibility Enum choice between toggle (changes to opposite when raised), hidden and visible.
Load Next Scene	String	Loads another scene by name (String). Please make sure to include the target scene to build (Build	

		Settings).	
Pause Audio	Audio Source	Pauses audio.	
Pause Video	Video Player	Pauses video.	
Play Audio	Audio Source	Plays audio.	Loop ? Shall this audio loop ?
Play Video	Video Player	Plays video.	Loop ? Shall this video loop ?
Stop Audio	Audio Source	Stops audio.	
Stop Video	Video Player	Stops video.	
Orientation	Transform	Changes orientation of a Transform (Where it looks at).	Transform values Changes to apply Relative to Enum choice between World, Self (What you will want most of the time), Object (Given in last parameter) and Head (of user). Reference Object
Position	Transform	Changes position of a Transform.	Same as Orientation + Relative Heading ? Whether or not the target should move in local space axes or world space axes.
Rotation	Transform	Changes rotation of a Transform.	Same as Orientation + Angle Angle in degree the target should rotate around axes.
Scale	Transform	Changes scale of a Transform (Multiply).	Same as Orientation
Size	Transform	Changes size of a Transform (Add).	Same as Orientation

.....Missing functionnalities & known issues.....

- No support yet for AND condition between actions.
- No support yet for delay before action triggers.
- No support yet for reaction sequencing (Allowing the user to trigger some actions after a specific reaction has been raised – You can workaround it by creating a new IIViMaT scene)
- No support yet for animations management through Reaction Nodes.
- Global Actions still display GameObject(s) as input whereas it should not.