IIViMaT User Guide

Create simple scripted interactions on Unity for Virtual Reality headsets

https://mavii.univ-lille.fr/software

For Unity 2020 version - Official Release - 07/02/2021

1. Context

MAVII is a research-action project conducted since 2017, by a multidisciplinary team, to study the new narrative possibilities offered by Virtual Reality technologies.

Equipped with a VR headset, the spectator is visually and acoustically in the universe of the story but he keeps the possibility to use his body (look, position and posture of the body and use of his hands), limited in the space he has. To put it simply, the spectator / player / user is in a completely new position, with a greater immersion than in the cinema, the possibility to interact as in video games and the ability to look where he wants, to use his body (in position and posture) and his hands to trigger actions.

The objective of MAVII is to delimit the field of possibilities by analyzing existing Virtual Reality productions and by experimenting with students and artists. With these studies, we propose to authors / artists / scriptwriters a software solution allowing to experiment new formats of works with the immersion qualities of the real, the fascination of stories and rich possibilities of interaction: IIViMaT.

The points on which we focus our reflections are :

- How can we use in the story the objects in the room where the spectator is. Can he sit, lie down? etc. What mapping is possible between the virtual universe and the room where the spectator is? We also work on the immersion in large spaces that require large displacements, impossible in a small space. Is it relevant to teleport? Are we going to disorientate the spectator?
- How to use the hands that offer a wide variety of possibilities of interaction with the environment, but also with other humans?

The narrator can revisit the place of the user in interactive narration for VR headsets. We propose in IIViMaT, first primitives for authors to experiment.

2. Introduction

IIViMaT (Immersive and Interactive Video Making Tool) is a software tool designed to facilitate access to virtual reality technologies for artists and other storytellers.

IIViMaT allows a designer to build / realize an interactive and immersive model of his project of installation, narration or scenography of exhibition presenting in a VR headset the main springs of his scenario. It will be possible to create a virtual museum or a scenarized adventure in a very simple way.

This plugin is dedicated to the scripting of interactions. Therefore, before starting a complex project with IIViMaT, it is desirable to have :

- all the 3D models of the scenes where the action will take place
- all the audiovisual contents (films, 360° sequences, photos, sounds)
- the animations
- a first description of the story's development, according to the spectator's actions

Based on these objects, the spectator will be able to interact in the story according to different modalities: his gaze, his position, his posture and his hands.

Through a graphical interface, IIViMaT proposes the creation of simple interactions between the user and his virtual environment.

IIViMaT is a graphical interface composed of "nodes"; boxes corresponding to "objects", "actions" and "reactions" that the designer links together. Their links allow them to generate interactions between the spectator and the space of the scene, the 3D objects or the media which are in the "environment".

3. Summary

Context 2

Introduction 3

Summary 4

Glossary 6

Installation of the IIViMaT package 7

Via the .package file 7

Use of the Oculus Integration 8

Use of Bézier Path Creator 8

Presentation of the IIViMaT interface 9

The project tree 9

The IIViMaT scene hierarchy 10

The IIViMaT Scene 10

The Oculus IIViMaT scene 10

Creation of a new IIViMaT scene & the IIViMaT graph 11

IIViMaT graphs 13

Actors 15

Actions 16

Head-related actions 16

Body-related actions 16

Global actions 16

Sequence actions 17

Hand-related actions 17

How the actions work 18

Head-related actions 19

Body-related actions 21

Global actions 23

Sequence actions 24

Hand-related actions 25
Reactions 29
Appearance reactions 29
Reactions on media players 29
Reactions on Transform 29
General reactions 30
How the reactions work 31
Appearance reactions 32
Reactions on media players 35
Reactions en Transform 39
Global reactions 42
Operators [Experimental] 44
Unary operators 44
Binary operators 44
How the operators work 45
Unary operators 46
Binary operators 47
Paths 48
Settings 49
Export the project to an Oculus Quest 1 / 2 executable 50
Examples of graphs 52
FAQ 55
Summary of IIViMaT objects 57
Summary of hand postures 60

4. Glossary

- **Action** : User's act that can be linked to actors and reactions
- **Actor** : Object in the environment that can be linked to actions
- **Graph** : Set of IIViMaT boxes forming the interactions
- **Head** : Camera
- **Operator** : Boolean calculations returning the result as output
- **Reaction** : Effect that can be linked to an action
- **Signal** : Trigger launched by an action to the reactions in order to launch them
- **Spectator** : User of the application
- **Target** : Element on which the reaction acts
- **Telepad** : Visual marker of teleportation with the hands

The terms related to Unity are in *italics* in this document.

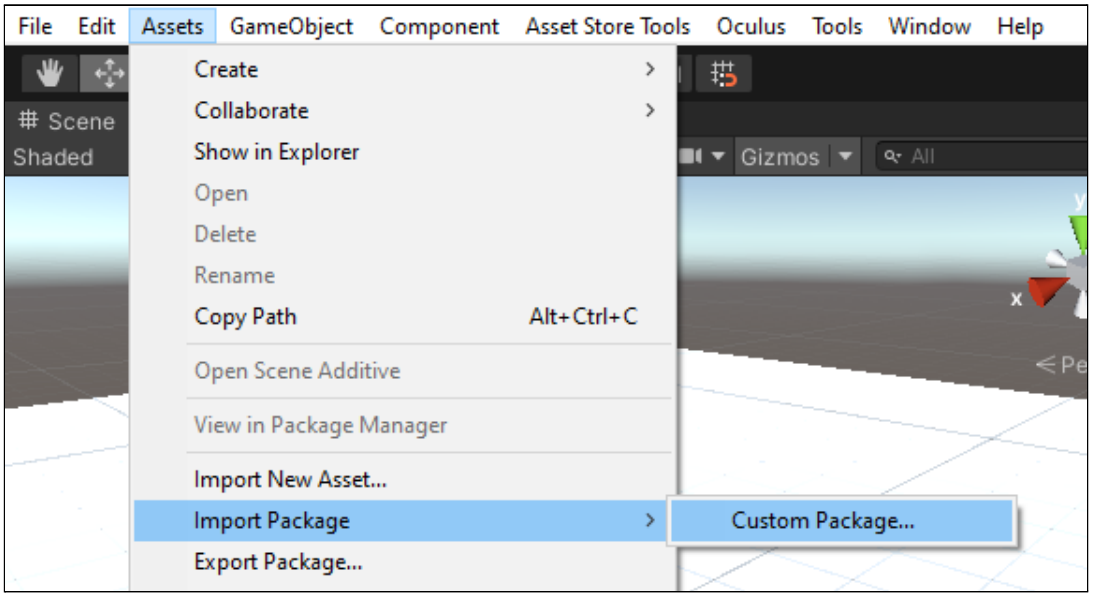
5. Installation of the IIViMaT package

IIViMat must be used with the Oculus Integration and Bézier Path Creator present on the Asset Store. Without them, you will find several errors in the code.

5.1. Via the .package file

**** For Windows ****

- Click on "Assets" → "Import Package" → "Custom Package..."

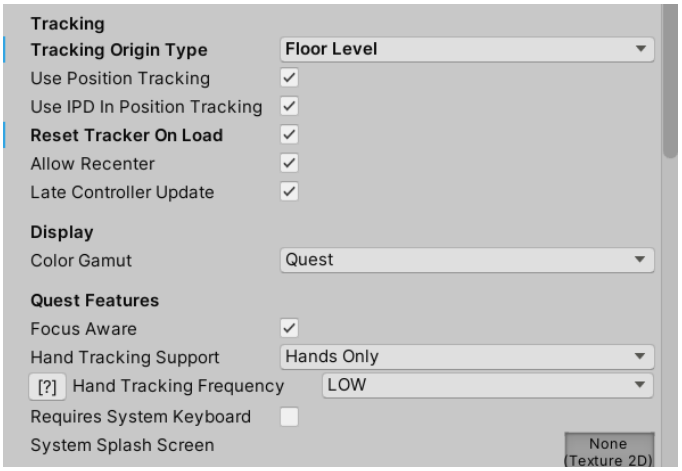


- Select the .package file

- Check that all files are checked by selecting "All" and import by clicking on "Import".

5.2. Use of the Oculus Integration

- In order to make IIViMaT work, it is **necessary to use** the Oculus Integration.
- Modify these different parameters of the OVRCameraRig:
    - Check that the "Tracking Origin Type" is set to "Floor Level".
    - Check "Reset Tracker On Load" to always start your VR experience in the direction the camera is set in the scene.
    - Select "Hands Only" in the "Hand Tracking Support" to force the use of hands and refuse the controllers.
    - Increase the “Hand Tracking Frequency” to improve the quality of hand recognition



5.3. Use of Bézier Path Creator

- In order to make IIViMaT work, it is **necessary to use** the Bézier Path Creator :

https://assetstore.unity.com/packages/tools/utilities/b-zier-path-creator-136082

6. Presentation of the IIViMaT interface

6.1. The project tree

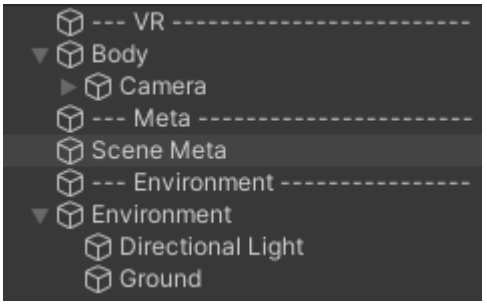
The IIViMaT package is composed of several folders, we have :

- **Editor** : Contains the files allowing to modify the editor (automatic addition of the scripts and creation of the graph)
- **IIViMaT** : main files
  - **Documentation** : user documentation
  - **Scenes** : Contains the demo scenes and the Templates
  - **Script** : Scripts allowing the operation of IIViMaT
- **Oculus** : Files allowing the support of the Oculus headset
- **Resources** : Resources
  - **IIViMaT_Resources :** Resources used by IIViMaT

6.2. The IIViMaT scene hierarchy

We will present here the objects present in the default scenes of IIViMaT. It is necessary to use one of these two scenes in order to use the objects of the hierarchy and make them interactive. We provide two Template scenes to create an IIViMaT scene :

6.2.1. The IIViMaT Scene

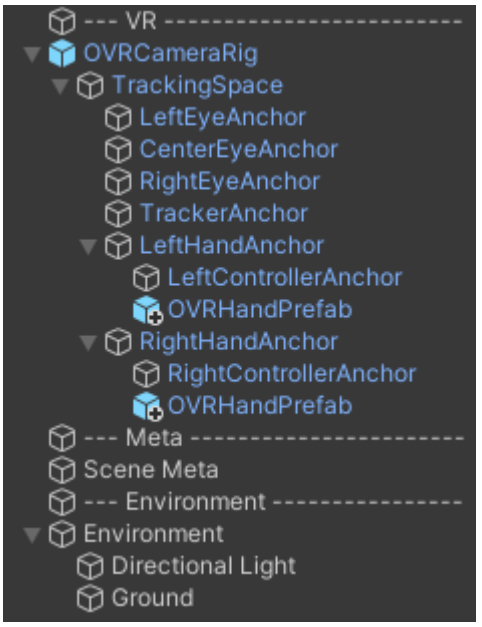


The IIViMaT scene is the default Template scene. It allows to use all the actions / reactions of IIViMaT except those requiring the hands :

We find in the hierarchy :

- **Body** : Represents the body in space
  - **Camera** : Represents the user's head
- **Scene Meta** : Contains the scripts allowing IIViMaT to work properly
- **Environment** : Contains all the objects present in the scene ; the objects must be children of this object to be used as an actor.

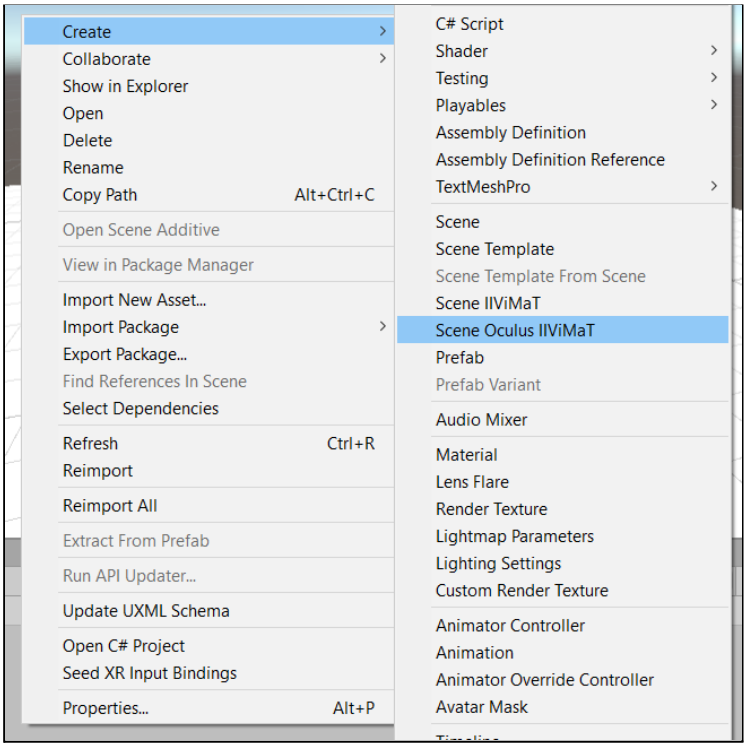
6.2.2. The Oculus IIViMaT scene



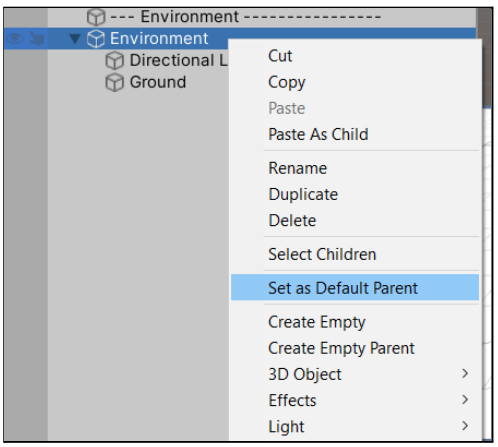
The IIViMaT scene is the second Template scene. It allows use of all the actions and reactions of IIViMaT with the recognition of hand postures.

7. Creation of a new IIViMaT scene & the IIViMaT graph

To create a new scene, “Right click in your folder > Create > Scene IIViMaT” or "Right click in your folder > Create > Scene Oculus IIViMaT”.



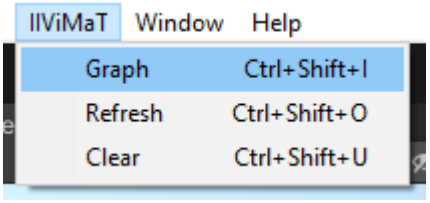
In order for different *GameObjects* to be recognized for use in IIViMaT, they must be children of Environment. To be sure you don't make a mistake, Unity offers the feature to define a default Parent when an object is created in the hierarchy. Any object created will then be a child of the environment. Environment then appears in bold.



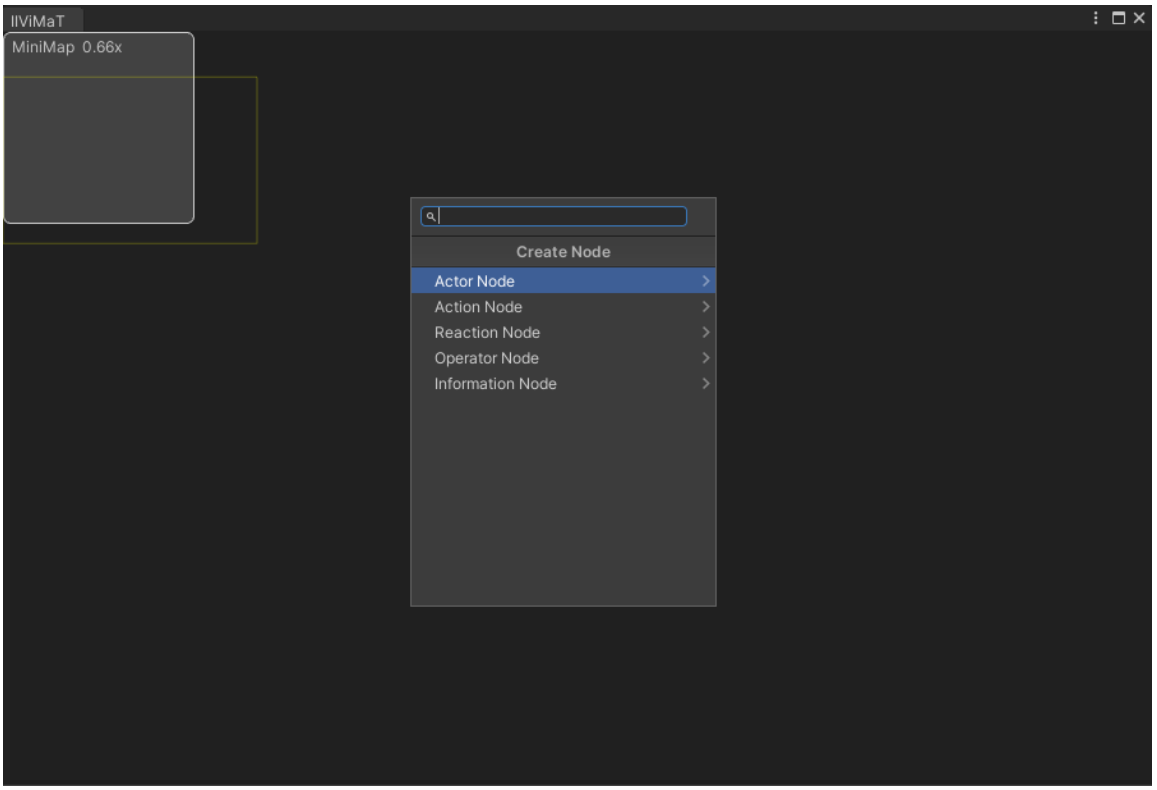
To open the graph, click on "IIViMaT > Graph" (Ctrl + Shift + I). The "Clear" button (Ctrl + Shift + U) allows you to delete all the IIViMaT objects and to start from a new graph.

The “Refresh” button allows you to clean up the IIViMaT interactions if you ever see interactions being triggered that do not appear in the graph.

The "Clear" button deletes all the IIViMaT objects linked to this scene and allows to restart it from a new IIViMaT model.

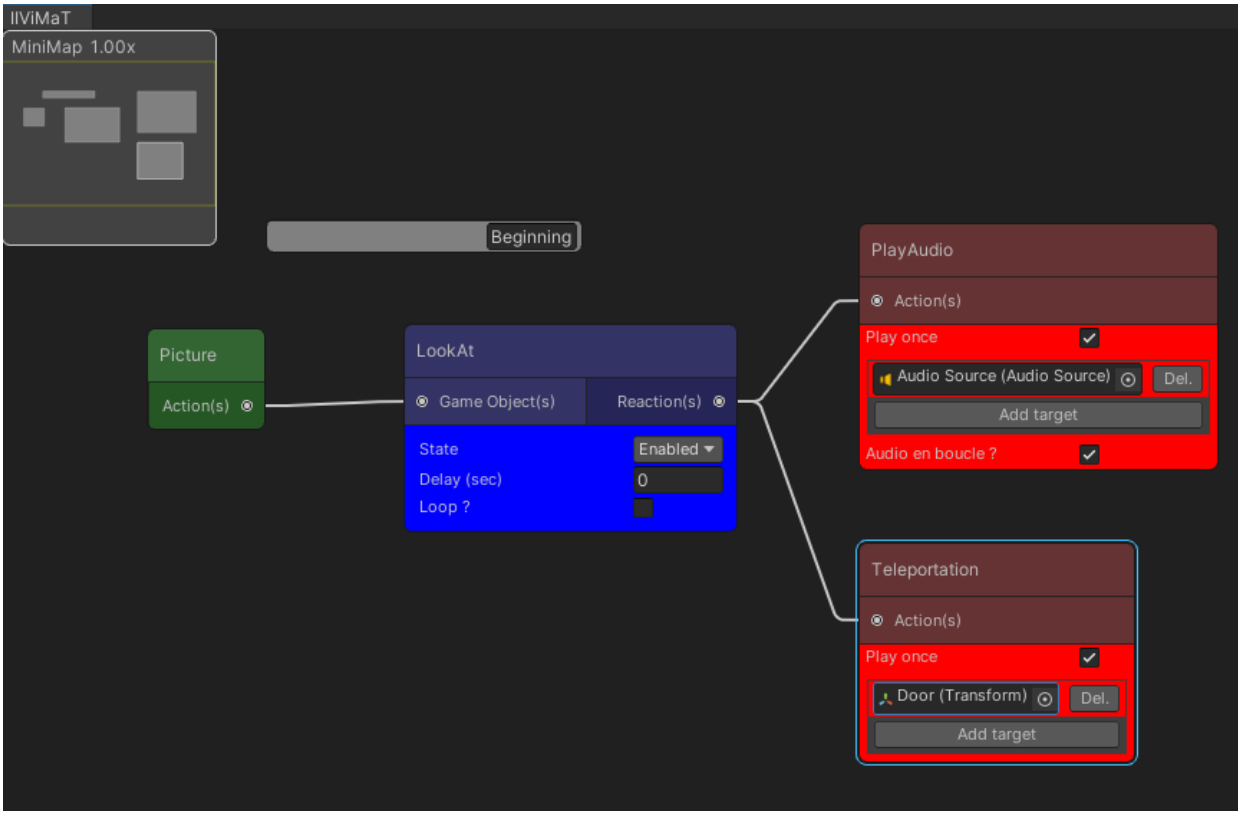


If you are in an IIViMaT scene, the graph window opens. To add boxes to the graph, right click in the window and choose "Create Node". You can also press the spacebar.



8. IIViMaT graphs

IIViMaT revolves around three types of objects: actors, actions and reactions.



The actors will trigger the actions associated with them. The latter will themselves trigger the related reactions. On the previous example, we can read this graph in the following way:

"When I look at the picture, I play the audio track in a loop and teleport the user to the position of the door object."

A reaction is only triggered if an action activates it. Concerning actions, there are two types: classical actions and global actions. A classical action requires at least one actor in input to trigger it. Global actions do not require an actor as input and are triggered in a different way.

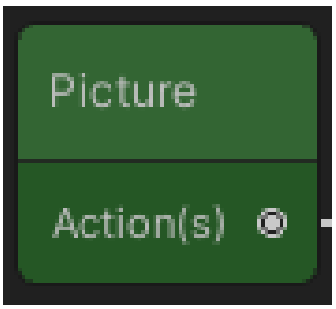
IIViMat has other types of objects that are a supplement to the operation of IIViMaT :

- Comment nodes allow you to find your way around the graph by putting annotations on it.
- IIViMaT operators are the representations of logical operators in

IIViMaT graphical format. They allow combining the result of several actions and operators and to trigger the output objects of the operator if the condition is respected.

- The Settings node allows you to configure certain aspects of IIViMaT to make the experience more flexible.

9. Actors



Actors are the targets of the actions. For the moment, the only actors are the *GameObjects*. For some actions, it is necessary that these actors have either a *VideoPlayer* or an *AudioSource* as a *Component*.

For a *GameObject* to be eligible as an actor, it must be in the Environment in the scene hierarchy. All the objects present in the Environment are automatically assigned scripts allowing them to become an actor. An actor can also be the target of a reaction.

For Prefabs, it is necessary that the original Prefab does not have the IIViMaT Scripts "LocalActions", "UniqueID" and "LocalCoroutineHandler". When an instance of the Prefab is added to the scene, the scripts will be assigned to it as a basic GameObject.

10. Actions

Actions are events mainly created by the user. They are linked to actors in input. When the event described by the action is detected on one of the input actors, the action will send a signal to the output reactions. Some actions may not have input actors or output reactions.

10.1. Head-related actions

The Head-related actions are linked to the orientation and the position of the head in the scene. There are :

- **Proximity** : Thrown when the head is close to the actor
- **Exit** : Thrown when the head leaves the proximity of the actor
- **Look At** : Thrown when the look passes over the actor
- **Look Away** : Thrown when the look leaves the actor

10.2. Body-related actions

The body-related actions correspond to the posture of a user in relation to an actor. There are four actions for the user's posture :

- **Stand Up** : Thrown when the user is standing on an actor
- **Crouch** : Thrown when the user is crouching on an actor
- **Sit** : Launched when the user is sitting on an actor
- **Lie Down** : Thrown when the user is lying on an actor

Posture recognition is based on the user's height when the application is launched. He must be standing when the application is launched.

10.3. Global actions

Global actions are actions that do not require actors as input. Hand-related actions are also global actions. We have :

- **At Start** : Launched after a given time since the beginning of the scene

10.4. Sequence actions

Sequence actions are used to create sequences in response to some sources. We have:

- **On Video End** : Launched when the video on the actor ends
- **On Audio End** : Launched when the audio present on the actor ends

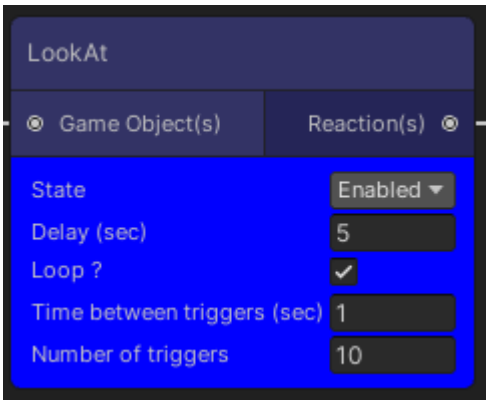
10.5. Hand-related actions

Hand-related actions are related to the user's hands. We find here the posture of the hands or their position in space. We have:

- **Hand Gesture Right** : Launched when the posture selected in the action is recognized on the right hand.
- **Hand Gesture Left** : Launched when the posture selected in the action is recognized on the left hand.
- **Proximity Hand** : Launched when one of the hands is close to the actor
- **Exit Hand** : Thrown when one of the hands leaves the proximity of the actor
- **Point Out Right** : Thrown when the right hand points to the actor with the index finger
- **Point Out Left** : Thrown when the left hand points to the actor with the index finger
- **Hand Teleport** : Thrown when the right hand does the teleportation posture and stops it.

10.6. How the actions work

We will detail here how each action works by defining the parameters or conditions of activation.



The actions have a state: *Enabled* or *Disabled*. If the action is *Enabled*, it will send its signal to the output reactions if the action is recognized. In case the action is *Disabled*, even if the condition of the action is triggered, the signal will not be sent. The state of the actions can be changed via the "ActivationAction" reaction. The color of the box darkens when the action is disabled.

As soon as the condition of the action is resolved, the signal is directly transmitted to the reactions. It is possible to delay the sending of this signal by modifying the action's Delay. The action will transmit its signal after X seconds of delay.

They can also be played in a loop. The action will trigger the reactions as long as the condition of the action is true. When the Loop box is checked, two fields appear: the first one is the time between each sending of the signal to the outputs of the action, the second value represents the number of times this sending will be done. If this second field is 0, the action will be done indefinitely.

The above action can be summarized as:

"The action is active. The action will wait 5 seconds before being launched after its detection. Every 1 second, 10 times, the action will send the signal to the reactions."

10.6.1. Head-related actions

Proximity	
Description	Action launched when the spectator's head enters one of the areas around one of the actors in input.
Additional parameter(s)	-----
Configuration	When an actor is linked to this action, he gets a child *ProximityCollider*. When the head enters the capsule represented by this *Collider* (green capsule), the action is triggered. It is possible to modify the position and scale of the capsule by modifying the *Component Transform*.
Other	It is possible to change the shape of the *Collider*. In the *ProximityCollider*, delete the *Component Capsule Collider* and add a new *Component Collider* with the desired shape. **Be careful**, the **Is Trigger** ❑ box must be checked. The area will no longer be represented by the gray sphere but by the green wire shape.

Exit	
Description	Action launched when the spectator's head comes out of one of the areas around one of the actors in input.
Additional parameter(s)	-----
Configuration	When an actor is linked to this action, he gets a child *ProximityCollider*. When the head enters the capsule represented by this *Collider* (green capsule), the action is triggered. It is possible to modify the position and scale of the capsule by modifying the *Component Transform*.
Other	It is possible to change the shape of the *Collider*. In the ProximityCollider, delete the *Component Capsule Collider* and add a new *Component Collider* with the desired shape. **Be careful**, the **Is Trigger** ❑ box must be checked. The area will no longer be represented by the gray sphere but by the green wire shape.

Look At	
Description	Action launched when the user looks at one of the actors in input.
Additional parameter(s)	-----
Configuration	A ray is launched in a straight line in the direction of the user's gaze. If this ray hits a *GameObject* **with a *Collider*** and this *GameObject* is the actor of the action, the action is triggered.
Other	It is possible for rays to ignore some objects by placing them in *Layer 2: Ignore Raycast*. The ray will pass through. This option is available in the inspector under the name of *GameObject*.

Look Away	
Description	Action launched when the user looks away from one of the actors in input .It is necessary to have looked at the actor before to launch it. It is not launched by default at the beginning of the application if the user does not look at the actor.
Additional parameter(s)	-----
Configuration	A ray is launched in a straight line in the direction of the user's gaze. If this ray hits a *GameObject* **with a *Collider*** and this *GameObject* is the actor of the action, when the ray leaves the actor and points to the void or another *GameObject*, the action is triggered.
Other	It is possible for rays to ignore some objects by placing them in *Layer 2: Ignore Raycast*. The ray will pass through. This option is available in the inspector under the name of *GameObject*.

10.6.2. Body-related actions

Stand Up	
Description	Action launched when the user is standing on an actor.
Additional parameter(s)	-----
Configuration	A ray is launched vertically from the user's head towards the ground. If this ray hits a *GameObject* **with a *Collider*** and this *GameObject* is the actor of the action, when the user is standing on it, the action is triggered.
Other	The action is launched when the user's head is between 90% and 115% of the registered user's height at launch. When the application is launched, the user must **remain standing** for one second in order to save his height. It is currently not possible to calibrate the user's height in any other way or to start an application in another posture.

Crouch	
Description	Action launched when the user is crouching on an actor.
Additional parameter(s)	-----
Configuration	A ray is launched vertically from the user's head towards the ground. If this ray hits a *GameObject* **with a *Collider*** and this *GameObject* is the actor of the action, when the user is crouching over it, the action is triggered.
Other	The action is launched when the user's head is between 56% and 90% of the registered user's height at launch. When the application is launched, the user must **remain standing** for one second in order to save his height. It is currently not possible to calibrate the user's height in any other way or to start an application in another posture.

Sit	
Description	Action launched when the user is sitting on an actor.
Additional parameter(s)	-----
Configuration	A ray is launched vertically from the user's head towards the ground. If this ray hits a *GameObject* with a *Collider* and this *GameObject* is the actor of the action, when the user is sitting on it, the action is triggered.
Other	The action is launched when the user's head is between 33% and 56% of the registered user's height at launch. When the application is launched, the user must **remain standing** for one second in order to save his height. It is currently not possible to calibrate the user's height in any other way or to start an application in another posture.

Lie down	
Description	Action launched when the user is lying on an actor.
Additional parameter(s)	-----
Configuration	A ray is launched vertically from the user's head towards the ground. If this ray hits a *GameObject* with a *Collider* and this *GameObject* is the actor of the action, when the user lies on it, the action is triggered.
Other	The action is launched when the user's head is between 0% and 33% of the registered user's height at launch. When the application is launched, the user must **remain standing** for one second in order to save his height. It is currently not possible to calibrate the user's height in any other way or to start an application in another posture.

10.6.3. Global actions

At Start	
Description	Action launched after a certain time since the beginning of the scene.
Additional parameter(s)	**Time before start** : Time in seconds since the beginning of the scene before triggering the action
Configuration	When the scene is launched, a chronometer is started. When the time has reached the one entered in parameter, the action is triggered.
Other	-----

10.6.4. Sequence actions

On Video End	
Description	Action launched when the video of one of the actors with a *VideoPlayer Component* has ended or a Loop has occurred.
Additional parameter(s)	-----
Configuration	It is **necessary** that the input actors each have only one *VideoPlayer Component*.
Other	-----

On Audio End	
Description	Action launched when the audio of one of the actors with an *AudioSource Component* has ended or a Loop has occurred.
Additional parameter(s)	-----
Configuration	It is **necessary** that the input actors each have only one *AudioSource Component*.
Other	-----

10.6.5. Hand-related actions

Hand Gesture Right	
Description	Action launched when the user produces with his right hand one of the pre-registered postures chosen in parameter.
Additional parameter(s)	**Hand gesture** : Choice of the posture to be recognized (see the postures section)
Configuration	The use of this action requires to be on an Oculus IIViMaT scene.
Other	-----

Hand Gesture Left	
Description	Action launched when the user produces with his left hand one of the pre-registered postures chosen in parameter.
Additional parameter(s)	**Hand gesture** : Choice of the posture to be recognized (see the postures section)
Configuration	The use of this action requires to be on an Oculus IIViMaT scene.
Other	-----

Proximity Hand	
Description	Action launched when one of the spectator's hands enters one of the areas around one of the actors in input.
Additional parameter(s)	-----
Configuration	When an actor is linked to this action, he gets a child *ProximityCollider*. When one of the hands enters the sphere represented by this *Collider* (grey sphere), the action is triggered. It is possible to modify the position and scale of the sphere by modifying the *Component Transform*.
Other	It is possible to modify the shape of the *Collider*. In the *ProximityCollider*, delete the *Component SphereCollider* and add a new *Component Collider* with the desired shape. **Please note that the Is Trigger ❑ box must be checked**. The area will no longer be represented by the gray sphere but by the green wire sphere.

Exit Hand	
Description	Action launched when one of the spectator's hands comes out of one of the areas around one of the actors in input.
Additional parameter(s)	-----
Configuration	When an actor is linked to this action, he gets a child *ProximityCollider*. When one of the hands leaves the sphere represented by this *Collider* (grey sphere), the action is triggered. It is possible to change the position and scale of the sphere by modifying the *Component Transform*.
Other	It is possible to modify the shape of the *Collider*. In the *ProximityCollider*, delete the *Component SphereCollider* and add a new Component *Collider* with the desired shape. **Please note that the Is Trigger ❑ box must be checked**. The area will no longer be represented by the gray sphere but by the green wire sphere.

Point Out Right	
Description	Action launched when the spectator's right hand points to the actor with the index finger
Additional parameter(s)	-----
Configuration	A ray is launched in a straight line in the direction of the index finger of the user's hand that is in the pointing posture. If this ray hits a *GameObject* **with a *Collider*** and this *GameObject* is the actor of the action, the action is triggered.
Other	It is possible for rays to ignore some objects by placing them in *Layer 2: Ignore Raycast*. This option is available in the inspector under the name of the *GameObject*.

Point Out Left	
Description	Action launched when the spectator's left hand points to the actor with the index finger
Additional parameter(s)	-----
Configuration	A ray is launched in a straight line in the direction of the index finger of the user's hand that is in the pointing posture. If this ray hits a *GameObject* **with a *Collider*** and this *GameObject* is the actor of the action, the action is triggered.
Other	It is possible for rays to ignore some objects by placing them in *Layer 2: Ignore Raycast*. This option is available in the inspector under the name of the *GameObject*.

Hand Teleport	
Description	Action to teleport to the location targeted by the hand
Additional parameter(s)	-----
Configuration	The input actors are the *GameObjects* that the user can step on.
Other	-----

11. Reactions

Reactions allow effects to be applied in response to actions. They have a list of targets on which these modifications are applied.

11.1. Appearance reactions

The appearance reactions change the visual aspect of a target. They include :

- **Color Change** : Allows you to change the color of a *Renderer*
- **Texture Change** : Allows you to change the texture of a *Material*
- **Shader Change** : Allows you to change the shader of a *Material*
- **Transparency** : Change the transparency of a *Renderer*
- **Visibility** : Change the visibility of a *Renderer*
- **Activation** : Allows to activate or deactivate a *GameObject*
- **Skybox Change** : Allows to change the skybox of the scene

11.2. Reactions on media players

The reactions on media players allow you to play videos or sounds as well as to change the scene. There are :

- **Play Video**: Allows to launch a *VideoPlayer*
- **Pause Video** : Allows you to pause a *VideoPlayer*
- **Stop Video** : Allows you to stop a *VideoPlayer*
- **Play Audio** : Allows you to launch an *AudioSource*
- **Pause Audio** : Allows you to pause an *AudioSource*
- **Stop Audio** : Allows you to stop an *AudioSource*
- **Change Volume Audio** : Allows to change the volume of an *Audio Source*
- **Play Video 360** : Allows to launch a Video 360
- **Pause Video 360** : Allows to pause a Video 360
- **Stop Video 360** : Allows to stop a Video 360

11.3. Reactions on Transform

The reactions on Transform are reactions that change the position, orientation or scale of a target. We have :

- **Orientation Transform** : Allows to change the orientation
- **Position Transform** : Allows to change the position

- **Rotation Transform** : Allows to change the orientation
- **Scale Transform** : Allows you to change the scale of an object
- **Teleportation** : Allows to change the position of the user
- **Stare At** : Allows you to orient the front axis of an object towards a position

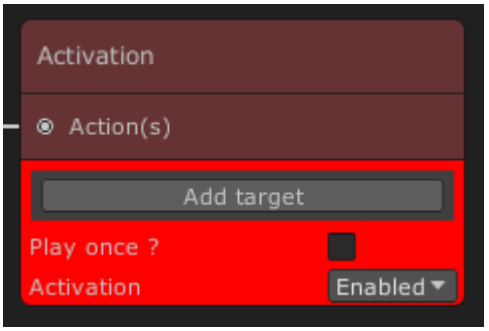
11.4. General reactions

General reactions are reactions that do not fit into any of the above categories. We have:

- **Launch Script** : Allows you to launch a script that implements a specific interface
- **Activation Action** : Change the state of an action (Enabled or Disabled)
- **Load Next Scene** : Allows to launch another scene

11.5. How the reactions work

We will detail here the functioning of each reaction by defining the parameters and their effects.



The vast majority of reactions define the targets on which they intervene by adding targets. These targets can be added with the Add Target button. The selectable targets can only be of the type on which the reaction acts.

All reactions have the ability to be run only once. When the *Play once* ❑ box is checked, the reaction will only be launched once even if one of the input actions sends it a signal.

11.5.1. Appearance reactions

Color Change	
Description	Reaction to change the color of a *Renderer*.
Additional parameter(s)	**Random color** : Apply a random color **Specific color** : Apply a specific color
Configuration	If the Random color parameter is disabled, the color chosen in the Specific color parameter is applied to the targets.
Other	-----

Texture Change	
Description	Reaction to apply a texture to a *Renderer*.
Additional parameter(s)	**New Texture** *:* New texture to apply to the *Renderer*
Configuration	Applies the texture passed in parameter New Texture to the targets.
Other	-----

Shader Change	
Description	Reaction to change the Shader of a *Renderer*.
Additional parameter(s)	**New Shader** : New shader to apply to the *Renderer*
Configuration	Applies the shader passed in parameter New Shader to the targets.
Other	Shaders are used to add visual effects to materials. There are many available online.

Transparency	
Description	Reaction to change the mode and transparency of a *Renderer*.
Additional parameter(s)	**Transparency** : Factor of transparency **Transparency mode** : Type of transparency
Configuration	The transparency factor allows you to vary the level of transparency (0 = opaque, 1 = transparent) The transparency mode allows you to play with the factors of each component (Real = BlendMode.One, Virtual = BlendMode.ScrAlpha (see unity documentation))
Other	-----

Visibility	
Description	Reaction to change the visibility of a target (without disabling it).
Additional parameter(s)	**Visibility mode** : Choose the visibility mode
Configuration	Explanation of the visibility modes : • **Enabled** : Enables the target's *Renderer* • **Disabled** : Disable the *Renderer* of the target • **Toggle** : Activate the *Renderer* of the target *GameObject* if it is disabled and vice versa
Other	-----

Skybox Change	
Description	Reaction to change the skybox of the scene
Additional parameter(s)	**New Skybox** : Réaction pour changer la skybox de la scène
Configuration	-----
Other	-----

Activation	
Description	Reaction to activate or deactivate a *GameObject*.
Additional parameter(s)	**Activation** : Choose the activation mode
Configuration	Explanation of the activation modes : • **Enabled** : Activates the target *GameObject* • **Disabled** : Disable the target *GameObject* • **Toggle** : Activate the target *GameObject* if it is disabled and vice versa
Other	-----

11.5.2. Reactions on media players

Play Video	
Description	Reaction to launch the *VideoPlayer* on a target
Additional parameter(s)	**Play in loop** : Choice to play the *VideoPlayer* in a loop
Configuration	-----
Other	Make sure that the *VideoPlayer*'s **Play On Awake** ❑ option is not checked.

Pause Video	
Description	Reaction to pause the *VideoPlayer* on a target
Additional parameter(s)	-----
Configuration	-----
Other	-----

Stop Video	
Description	Reaction to stop the *VideoPlayer* on a target
Additional parameter(s)	-----
Configuration	-----
Other	-----

Play Audio	
Description	Reaction to launch the *AudioSource* present on a target.
Additional parameter(s)	**Play in loop** : Choice to play the *AudioSource* in a loop
Configuration	-----
Other	Make sure that the *AudioSource*'s **Play On Awake** ❑ option is not checked.

Pause Audio	
Description	Reaction to pause the *AudioSource* present on a target.
Additional parameter(s)	-----
Configuration	-----
Other	-----

Stop Audio	
Description	Reaction to stop the *AudioSource* present on a target.
Additional parameter(s)	-----
Configuration	-----
Other	-----

Change Volume Audio	
Description	Reaction allows you to modify the volume of an *Audio Source*
Additional parameter(s)	**Volume variation (-/+)** : Value to add to the volume (can be negative)
Configuration	-----
Other	The volume of an *Audio Source* is between 0 and 1.

Play Video 360	
Description	Reaction to launch a 360° video sphere
Additional parameter(s)	**Play in loop** : Choice to play the *VideoPlayer* in loop **Fixed to the head** : Choice to fix the sphere on the spectator's head **Back to initial position** : Choice to bring the sphere back to its original position at the end of the video
Configuration	In order to use this reaction, it is necessary to use the prefabs provided by IIViMaT. "Right click on the Scene hierarchy > Video > Sphere 360".
Other	Make sure that the *VideoPlayer*'s Play On Awake ❑ option is not checked.

Pause Video 360	
Description	Reaction to pause a 360° video sphere
Additional parameter(s)	**Stop following the head** : Choice to leave the sphere where it is in space **Back to initial position** : Choice to bring the sphere back to its original position
Configuration	-----
Other	-----

Stop Video 360	
Description	Reaction to stop a 360° video sphere
Additional parameter(s)	**Stop following the head** : Choice to leave the sphere where it is in space **Back to initial position** : Choice to bring the sphere back to its original position
Configuration	-----
Other	-----

11.5.3. Reactions on Transform

Orientation Transform	
Description	Reaction allows to change the orientation of a target by making it turn by an angle around the 3 axes (first the z, x and y axis).
Additional parameter(s)	**Transform Values** : Values (in degrees) for each rotation axis **Relative To** : Reference around which to rotate **Reference Object** : Reference object to rotate around if the "Relative To" option is Object
Configuration	Explanation of the different reference objects: • **World** : Rotates the object along the axes of the scene • **Self** : Rotate the object around its point of origin • **Object** : Rotate the object by recovering the orientation (Rotation) of the object and by adding the angles • **Head** : Rotate the object by recovering the orientation (Rotation) of the head and by adding the angles
Other	-----

Position Transform	
Description	Reaction allows to change the position of a target.
Additional parameter(s)	**Vector position** : Values of the position vector **Relative To** : Reference from which to make the position modification **Reference Object** : Reference object to rotate around if "Relative To" is Object
Configuration	Explanation of the different reference objects : • **World** : Places the target in space at the coordinates of the position vector • **Self** : Place the target at its own coordinates added with the position vector in the local coordinate system • **Object** : Places the target at the object's position added with the position vector in the local coordinate system • **Head** : Places the target at the coordinates of the head added with the position vector in the local coordinate system
Other	-----

Rotation Transform	
Description	Reaction allows the target to rotate around an axis passing through a point by an angle.
Additional parameter(s)	**Rotate around axis** : Value of the axis (world reference) to rotate around **Angle** : Value of the angle (in degrees) **Relative To** : Reference of the origin point of the axes **Reference Object** : Reference point of the axes if the option "Relative To" is Object
Configuration	Explanation of the different reference objects : • **World** : The origin of the axis is the point (0,0,0) world • **Self** : The origin of the axis is the origin of the target itself • **Object** : The origin of the axis is the origin of the object passed in reference • **Head** : The origin of the axis is the origin of the head
Other	-----

Scale Transform	
Description	Reaction to change the scale of a target.
Additional parameter(s)	**Scale multiplier factor** : Multiplier factor values **Relative To** : Reference of the scale **Reference Object** : Reference of the scale if the option "Relative To" is Object
Configuration	Explanation of the different reference objects: • **World** : The scale of the target is set to the values of the factors in the world reference • **Self** : The scale of the target is multiplied by the multiplier factor • **Object** : The scale of the target is defined by the value of the object passed in parameter multiplied by the multiplier factor
Other	-----

Teleportation	
Description	Reaction to teleport the user to a position in the scene.
Additional parameter(s)	**On-site** : Check if the whole experience is to be done on site
Configuration	The target of the teleportation must be where the user's feet will be.
Other	The On-site option is important for an onsite experience because if the teleportation takes place when the user does not have their head above their feet (e.g. bending over), a lag will be created.

Stare At	
Description	Reaction to orient a local axis of an object towards a position.
Additional parameter(s)	**Relative To** : Reference to stare at **Point** : Point in space to stare at **Reference Object** : Object to stare at if the option "Relative To" is Object **Axis Staring At** : Local axis of the object to be oriented to the position (Z axis (forward) by default)
Configuration	Explanation of the different gaze references : • **Point** : the target will stare at a point in space • **Object** : the target will stare at an object in space • **Head** : the target will stare at the user Explanation of the different axes: • **Forward** : Z Axis • **Backward** : - Z Axis • **Up** : Y Axis • **Down** : - Y Axis • **Right** : X Axis • **Left** : - X Axis
Other	-----

11.5.4. Global reactions

Launch Script	
Description	Reaction allowing to launch any C# code. This allows the opening of reactions to any script present on a target *GameObject*. It allows launch behaviors not present on IIViMaT.
Additional parameter(s)	-----
Configuration	The script must implement the IReactionScript interface and define the "void ReactionToScript()" method. When the reaction is activated, the set of ReactionToScript() methods present on all scripts of all targets is called.
Other	It is possible that a *GameObject* is several scripts implementing the interface. Each of the scripts will then be called. No certainty is given about the order of the calls.

Activation Action	
Description	Reaction allows us to change the state of an action.
Additional parameter(s)	**State** : State that the target will take when the reaction is called
Configuration	Just enter the name of the action as it is in the graph (ex : LookAt_6)
Other	-----

Load Next Scene	
Description	Reaction to launch a scene.
Additional parameter(s)	-----
Configuration	Just enter the name present on the Scene object.
Other	It is advised not to put spaces in the name of the scene and to replace them by ' _ '. To be able to change the scene, it must be in the build.

12. Operators [Experimental]

IIViMaT logical operators are objects that trigger output objects only if the condition of the operator is true. They allow combinations of actions and operators to trigger reactions. It is also possible to put operators in input of other operators to make combinations of actions and operators more complex.

12.1. Unary operators

Unary operators are operators that take only one IIViMaT object as input but can be linked to several objects as output. We have :

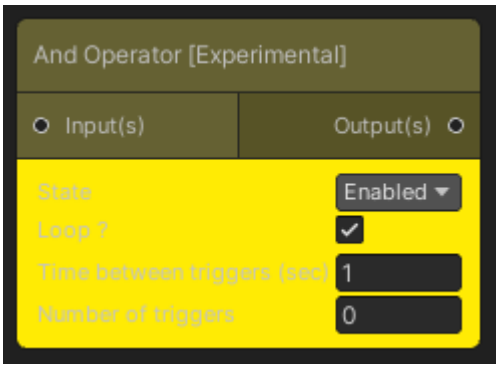
- **Not** : Is triggered when the input action or operator is not activated

12.2. Binary operators

Binary operators are operators that take several IIViMaT objects as input and output. We have :

- **And** : Is triggered when all actions and operators in input are active
- **Or** : Is triggered when one of the actions or one of the operators in input is active

12.3. How the operators work



IIViMaT logical operators are objects that allow the creation of more complex scenarios.

The operators have a state : *Enabled* or *Disabled*. If the operator is *Enabled*, it will send its signal to the output reactions or operators. In case the operator is *Disabled*, even if the condition of the operator is triggered, the signal will not be sent. The state of the operators can be changed via the reaction "ActivationAction". The color of the box darkens when the operator is disabled.

As soon as the condition of the operator is resolved, the signal is directly transmitted to the reactions or operators in output.

They also have the possibility to be played in a loop. The operator will trigger the reactions or operators as long as the condition of the operator is true. When the Loop box is checked, two fields appear: the first one is the time between each sending of the signal to the outputs of the action, the second value represents the number of times this sending will be done. If this second field is 0, the triggering will be done indefinitely. If among the inputs of the operator there is an action or an operator in loop, their loops are ignored and it is the operator's loop that is used.

12.3.1. Unary operators

Not	
Description	Is triggered when the input action or operator is not activated.
Additional parameter(s)	-----
Configuration	-----
Other	-----

12.3.2. Binary operators

And	
Description	Is triggered when all actions and operators in input are active.
Additional parameter(s)	-----
Configuration	-----
Other	-----

Or	
Description	Is triggered when one of the actions or one of the operators in input is active.
Additional parameter(s)	-----
Configuration	-----
Other	-----

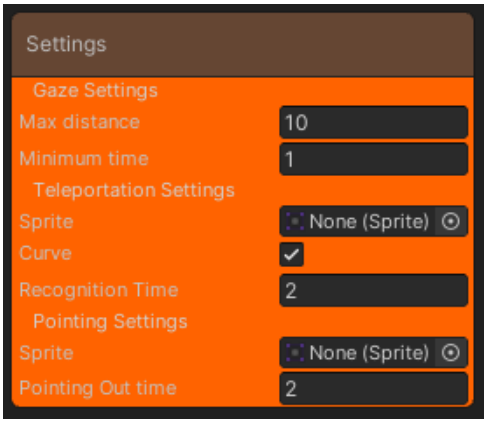
13. Paths

The paths use the Asset created by Sebastian Lague. To configure or modify these paths, we invite you to read the documentation in the Asset.

14. Settings

In order to make IIViMaT more configurable, several parameters can be customised via the Settings box.

This box can only be present once per graph. One Settings box is needed per scene ; this allows each scene to be set independently.

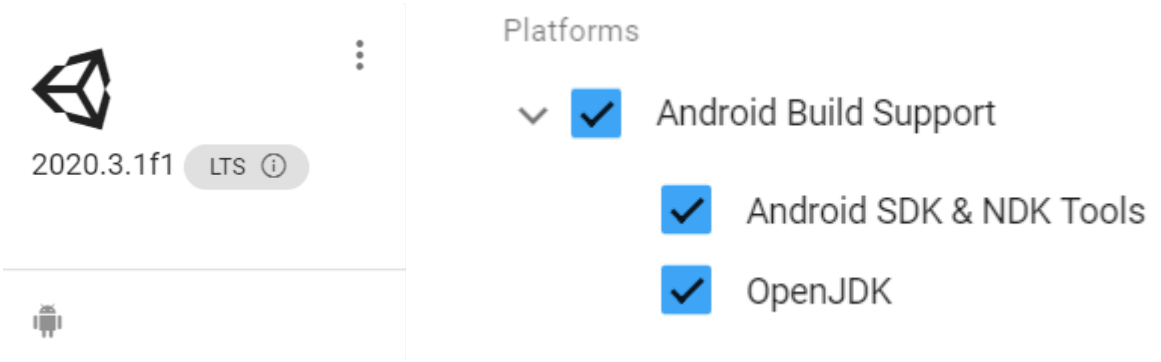


Settings	
Gaze Settings	
Max distance	Maximum look distance. Beyond that, the look At and look Away actions are not be triggered. 0 = look to infinity
Minimum time	Minimum time to look at an actor to launch the look at actions. 0 = instant reaction
Teleportation Settings	
Sprite	Sprite displayed at the target location for teleportation.
Curve	Causes a line to appear showing the teleportation curve.
Pointing Settings	
Sprite	Sprite displayed at the location targeted by finger pointing.
Pointing Out time	Minimum time to point at an actor to initiate Pointing Out actions. 0 = instant reaction

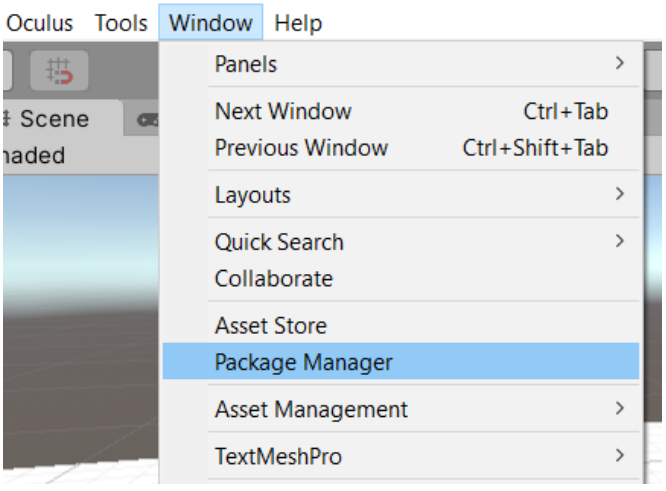
15. Export the project to an Oculus Quest 1 / 2 executable

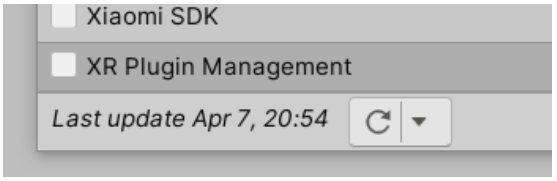
In order to export your Unity project to an Oculus executable, you need to follow a few configuration steps.

First, you need to have downloaded the Android Build Support for your version of Unity. To find out if you have it, just check that you have the Oculus logo under your version of Unity. If you don't see it, click the " ⋮ " of your version and select " Add Modules". Check the Android Build Support, making sure that the submenus are also checked. Install everything.

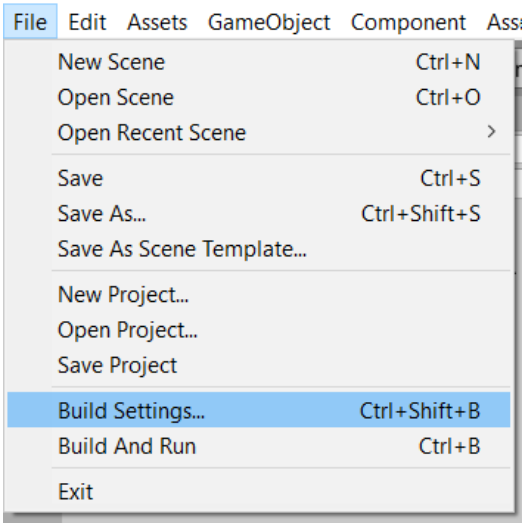


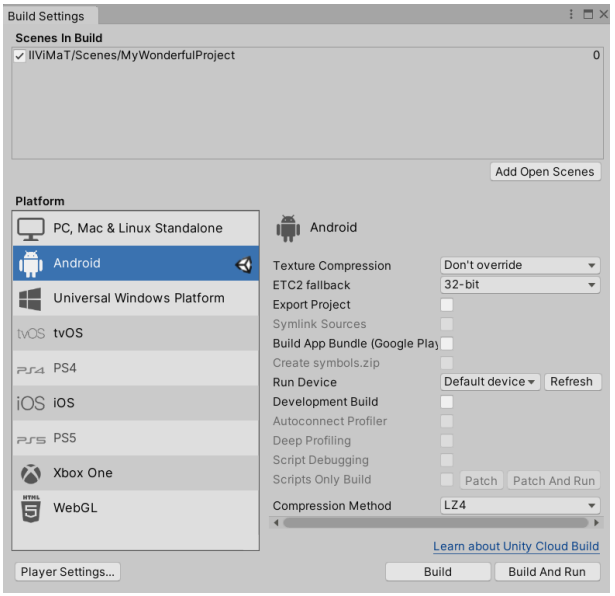
We continue by installing the XR Plugin Management. Go to "Windows > Package Manager". Look for the "XR Plugin Management" in the list and install it.



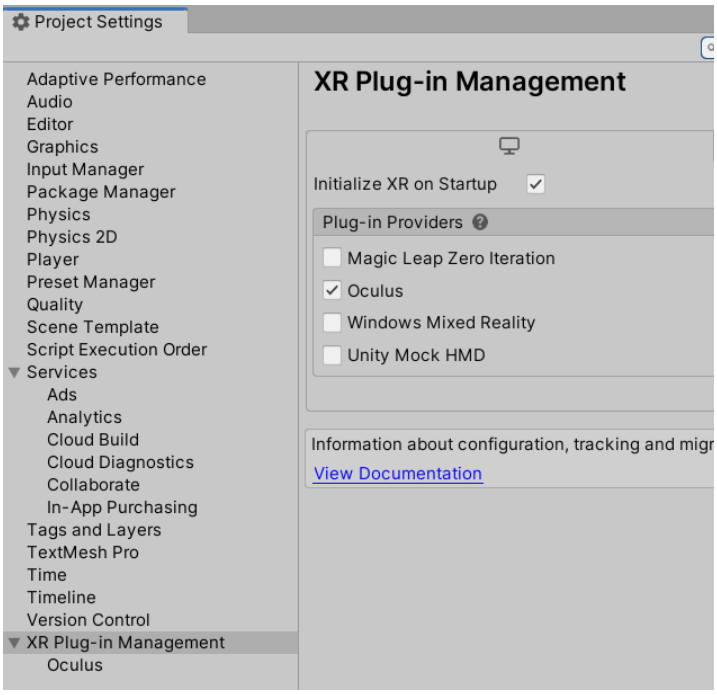


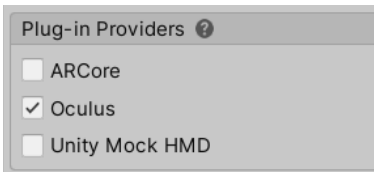
Select "Edit > Build Settings...". In the top window, select the set of scenes that will be in your executable (Add the opened scene or drag them). In the menu below, select the Android menu and click on "Switch Plateform".





We are going to activate the Oculus Plugins through the XR Plugin Management. Go to "Assets > Project Settings..." . Selecting the XR Plugin Management, check the Oculus boxes in the Windows and Android menu.





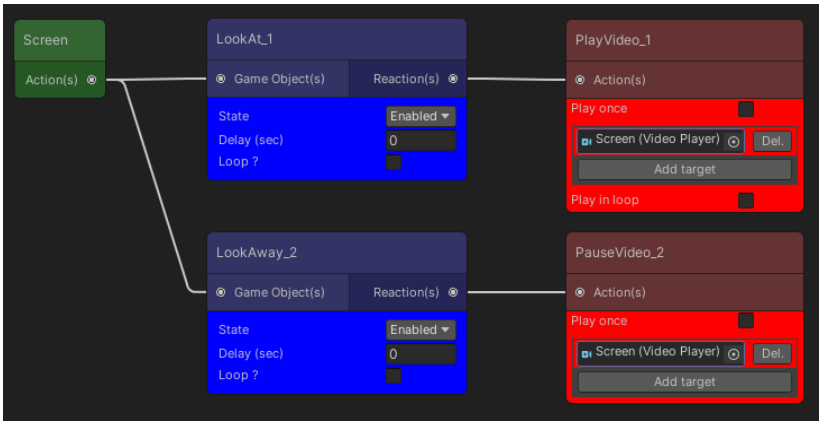
When the export is finished, click on Build and wait for the generation of your APK. You can now install your executable on your headset using software like SideQuest or adbLink.

16. Examples of graphs

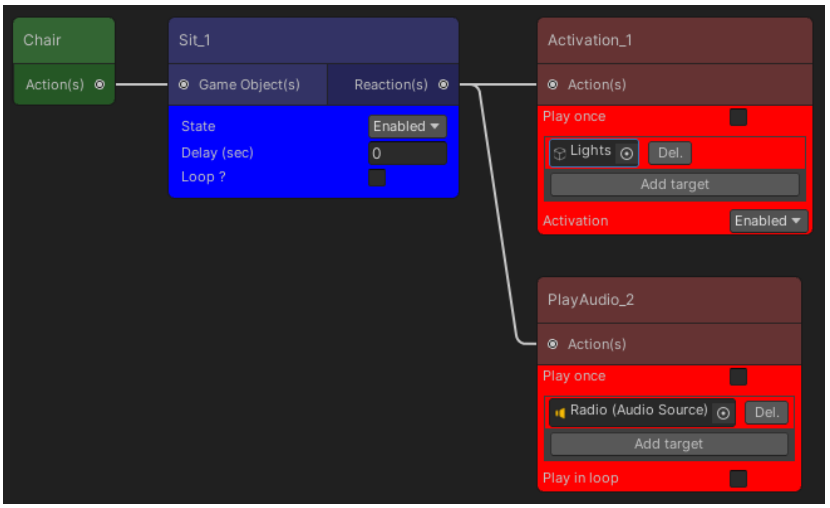
Here are some examples of graphs to make simple interactions. You can also have a look at the demo scenes in the plugin and see how they were created.

"When the user looks at the screen, we play the video on the screen.

When he looks away from the screen, we pause the video on the screen."



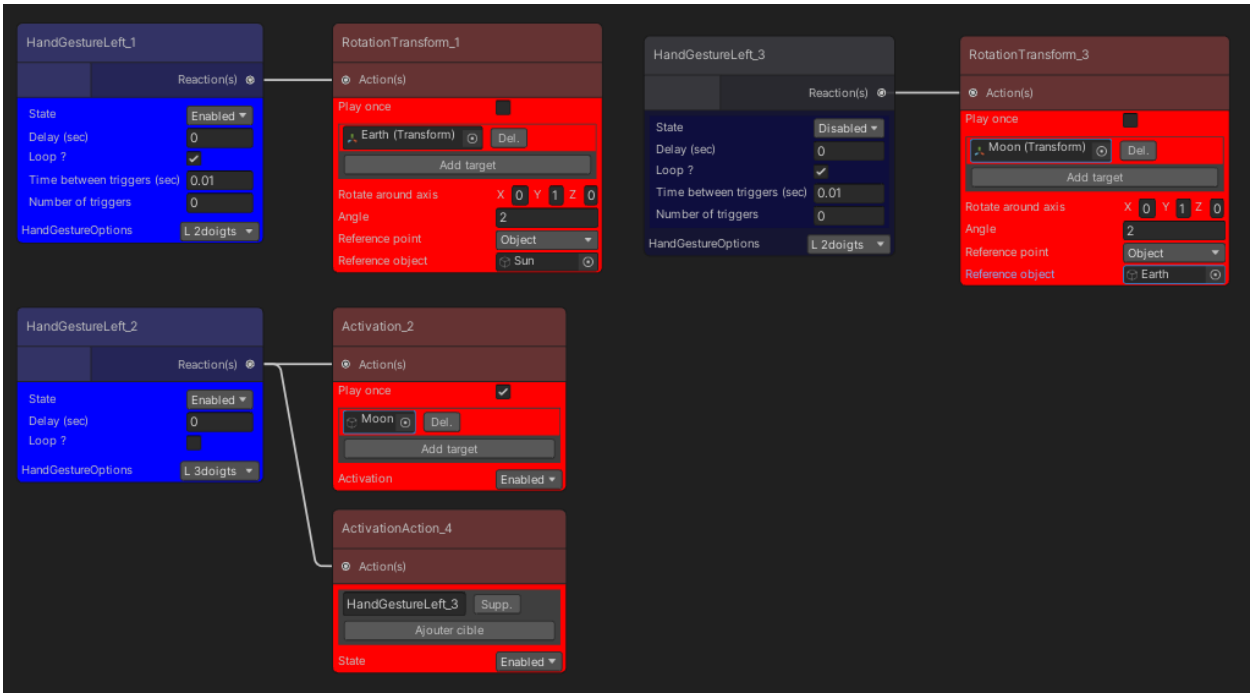
"When the user sits on the chair, we activate the lights and play the audio source on the radio ."



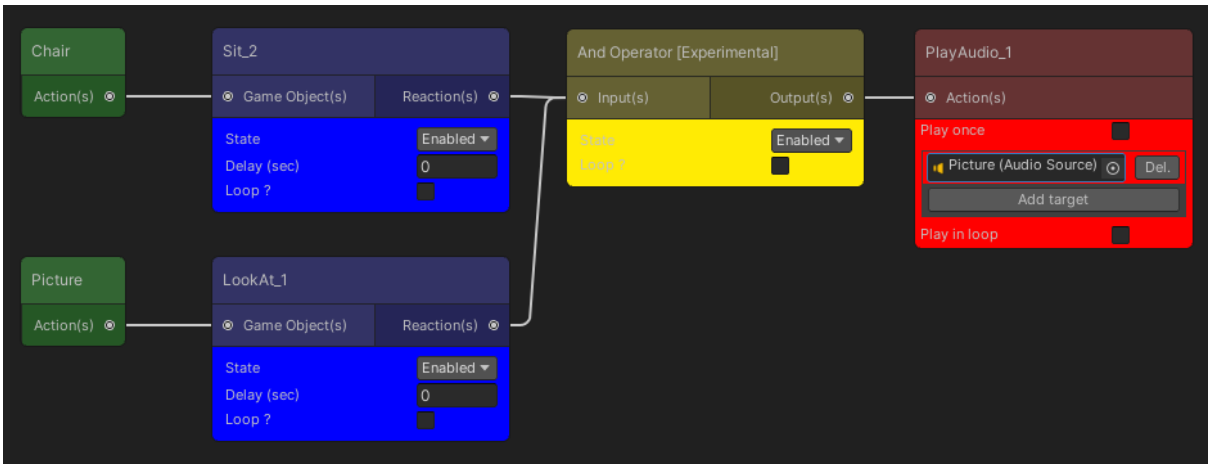
"When the user does the "L2doigts" posture, the Earth rotates around the sun.

When the user does the "L3doigts" posture, the Moon is activated. The action named "HandGestureLeft_3" changes to the Enabled state.

When the user does the "L2doigts" posture, the Earth rotates around the sun and the Moon rotates around the Earth. "

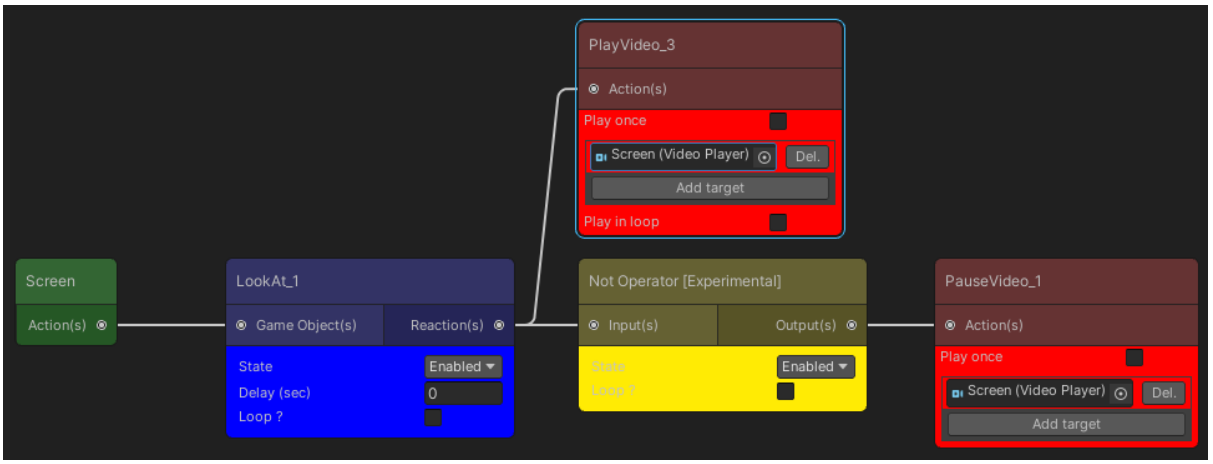


"When the user sits on the chair and looks at the board, we play the audio source on the picture."



"When the user looks at the screen, we play the video on the screen.

When the user doesn't look at the screen, we pause the video on the screen."



17. FAQ

- **Q : The error message indicates: "Current scene is not an IIViMaT scene", what should I do ?**
- **A** : In order to use the IIViMaT plugin, it is necessary to be in an IIViMaT or Oculus IIViMaT scene. For that, see the section Creating a new IIViMaT scene.

- **Q : When a reaction starts, the targets are not affected, what should I do ?**
- **A** : Check that the objects that should be affected by the reaction are in the targets. An actor is not the target of a reaction. Also check that the targets have the elements on which the reactions intervene (*AudioSource*, *VideoPlayer*, *Renderer*, etc.).

- **Q : What should I do when the "Look At" action does not trigger when I look at my Game Object ?**
- **A** : In order for the "Look At" action to detect an object, it is necessary that the object has a *Collider*. Add one by clicking on the "*GameObject* > Add Component > *Collider* (of your preference)".

- **Q : When I do a posture on an actor, the posture is not recognized. What should I do ?**
- **A** : The posture uses the same principle as for the "Look At", so the actor must have a *Collider* in order to detect it. In addition, the postures are calculated according to the size of the user when the application is launched. It is necessary to start the application standing.

- **Q : When I do a posture with a "HandGesture" action, the posture is not recognized. What should I do ?**
- **A** : To recognize a hand gesture, check that your project is an Oculus

---

IIViMaT scene. The posture must be identical to those listed in the section Summary of hand postures.

- **Q : I can't change the scene in the demo. What should I do ?**
- **A** : Check that the five demo scenes are in the project build.

- **Q : When I put actions on a Prefab, they don't start, what should I do ?**
- **A** : In order for the Prefabs to be used, the original Prefab must not have the "LocalActions", "UniqueGUID" and "LocalCouroutineHandler" components. When instantiating the Prebab, the scripts will be added automatically and it will become an overload of the Prebab instance.

- **Q : Some actions and reactions are launched when they don't exist anymore in the graph, how to clean this up?**
- **A** : It happens that some actions and reactions remain while the objects in the graph no longer exist. In order to fix this, you just have to click on the "Refresh" button of the IIViMaT menu. The IIViMaT objects that are not in the saved model of the graph will be deleted.

18. Summary of IIViMaT objects

Actors		
GameObject	GameObject	
	Prefab	
	GameObject with VideoPlayer	
	GameObject with AudioSource	
Actions		
General	At Start	
Sequence	On Video End	
	On Audio End	
Spectator	Head Relative	Proximity
		Exit
		Look At
		Look Away
	Body Posture	Stand Up
		Crouch
		Sit
		Lie Down
	Hand Gestures	Hand Gesture Right
		Hand Gesture Left
		Proximity Hand
		Exit Hand
		Point Out Right
		Point Out Left
		Hand Teleport

Reactions		
Appearance	Color Change	
	Texture Change	
	Shader Change	
	Transparency	
	Visibility	
	Activation	
	Skybox Change	
Media Player	**Video**	Play Video
		Pause Video
		Stop Video
	Audio	Play Audio
		Pause Audio
		Stop Audio
		Change Volume Audio
	Video 360	Play Video 360
		Pause Video 360
		Stop Video 360
Transform	**Path**	Follow Path
		Pause Follow Path
		Stop Follow Path
	Rotation Transform	
	Scale Transform	
	Teleportation	
	Stare At	
General	Launch Script	
	Activation Action	

	Load Next Scene

Operators	
Unary	Not
Binary	And
	Or

19. Summary of hand postures

	Left hand	Right hand
Fist		
Flat		
Pistol		
Thumb Up		
Two Fingers		
Three Fingers		

Four Fingers		
Action's posture		
Point Out		
Hand Teleportation