

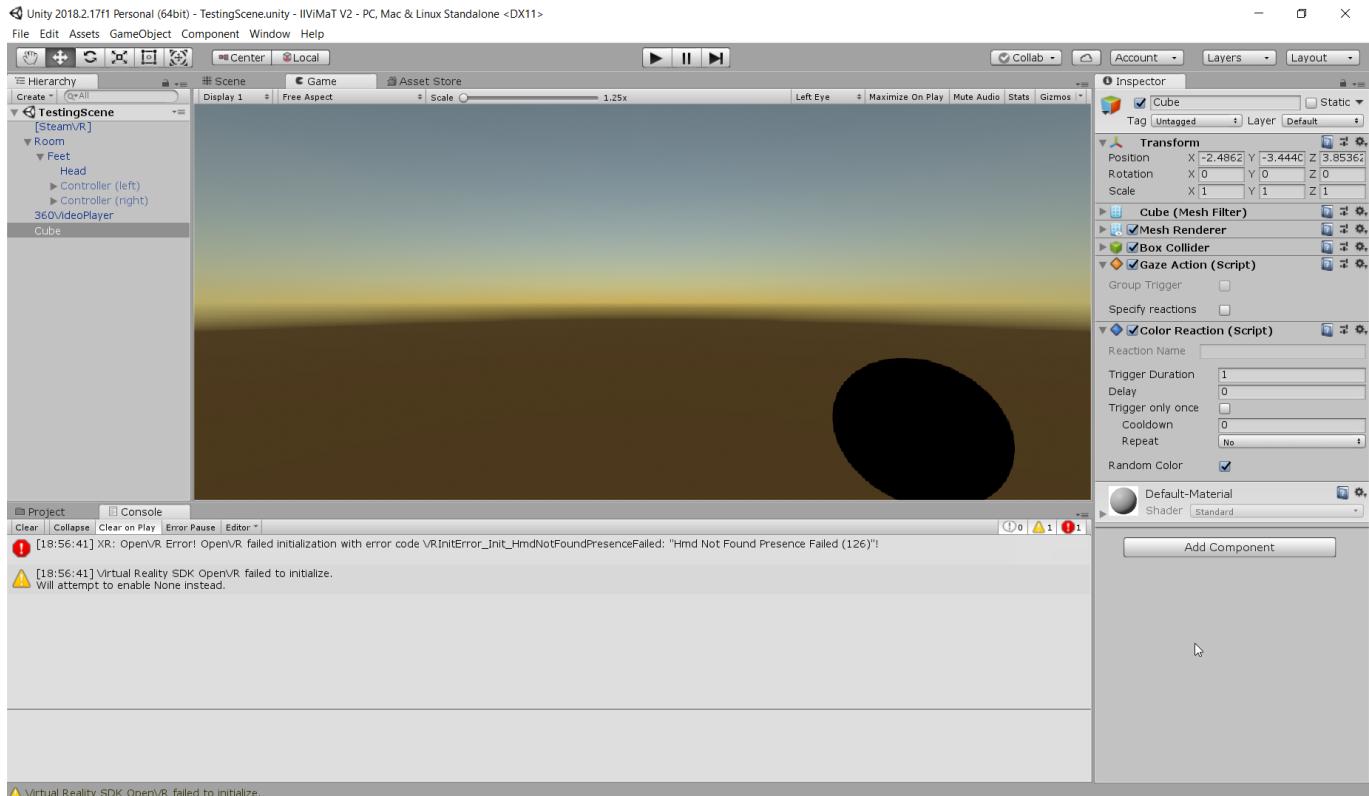
Utiliser IIViMaT avec Unity

Table des matières

1 Interface d'Unity	2
1.1 Hiérarchie de la scène	3
1.2 Détail des composants	3
1.3 Vue de la scène	4
1.4 Explorateur de fichiers	5
2 Utilisation	6
2.1 Introduction	6
2.2 Créer un projet	7
2.3 Glossaire	8
2.3.1 Actor	8
2.3.2 Action	9
2.3.3 Reaction	10
3 Exemple	12
3.1 Lancement et arrêt d'une vidéo 2D par le regard	13
3.2 Modification d'un objet par le regard	14
3.3 Déplacement d'objet automatique	14
3.4 Lancement d'un son en boucle	15
3.5 Activation d'un objet après un nombre d'interactions fixé	16
3.6 Téléportation dans une sphère 360	20
3.7 Lancement d'une vidéo 360 après téléportation	22
3.8 Changement de mode de vue à un certain temps de la vidéo	22
3.9 Lancement et arrêt d'une vidéo 360 par proximité	24
3.10 Rotation d'un objet autour d'un autre	24
3.11 Téléportation dans un espace 3D	26
3.12 Désactivation d'objet	26
3.13 Réactions de groupe	28
3.14 Modifier le volume d'une source audio par propagation	31
4 Conseils	33

1 Interface d'Unity

Pour un tutoriel complet sur Unity vous pouvez vous rendre sur le site officiel :
<https://unity3d.com/fr/learn/beginner-tutorials>



L'interface d'Unity est composée de différentes fenêtres.

L'agencement de ces fenêtres peut être reconfiguré, ne vous inquiétez donc pas si votre interface à l'air différente de celle d'un autre étudiant. Indifféremment de l'agencement, vous devriez pouvoir apercevoir à tout moment :

- La hiérarchie de la scène (ici, à gauche)
- Le détail des composants de l'objet sélectionné (ici, à droite)
- Une vue de la scène (ici, au milieu)
- Un explorateur de fichiers (ici, en bas)

Chaque fenêtre peut comporter plusieurs onglets. Par exemple la vue de la scène comporte un onglet "Scene", un onglet "Game" (ici, sélectionné) et un onglet "Asset Store".

Si jamais vous modifiez l'agencement de votre interface et ne retrouvez plus rien, vous pouvez toujours revenir à un agencement prédéfini. Pour passer à un agencement prédéfini, cliquez sur Window > Layouts, et sélectionnez le layout souhaité (ici, "Default").

1.1 Hiérarchie de la scène

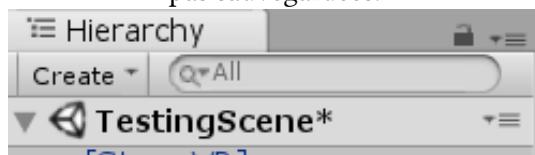
Un objet peut être glissé-déposé sur un autre objet de la hiérarchie pour l'ajouter comme enfant de ce premier objet. Lorsqu'un parent est modifié, l'enfant est modifié de telle sorte que, relativement au parent, l'enfant n'a pas changé.

Par exemple, si le parent se déplace de 1 mètre, l'enfant se déplace également, ainsi l'enfant n'a pas bougé par rapport à son parent, il est toujours à la même distance.

Un parent peut avoir plusieurs enfants, et un enfant peut également être parent d'un objet, qui lui-même peut également être parent, etc...

Pensez à sauvegarder souvent! (Astuce : Utilisez le raccourci clavier CTRL + S)

Lorsque vous avez modifié votre scène, une astérisque apparaît pour vous signaler que des modifications ne sont pas sauvegardées.



1.2 Détail des composants

Cette fenêtre liste les composants de l'objet sélectionné. Le premier composant est le "Transform". C'est ici que vous pouvez changer la position, orientation et proportion de vos objets.

Il y a parfois beaucoup de composants qui ne vous seront pas utiles. Si vous souhaitez remettre les valeurs par défaut d'un composant, vous pouvez cliquer sur la petite roue dentée en haut à droite du composant et sélectionner "Reset". Les acteurs, actions et réactions sont des composants.

Les composants **acteurs** sont démarqués par leur icône violette.



Les composants **actions** sont démarqués par leur icône orange.



Les composants **réactions** sont démarqués par leur icône bleue.



Vous pouvez ajouter un composant par glisser-déposer ou en cliquant sur "Add Component" et en sélectionnant le composant à ajouter (astuce : vous pouvez chercher le composant en tapant son nom).

1.3 Vue de la scène

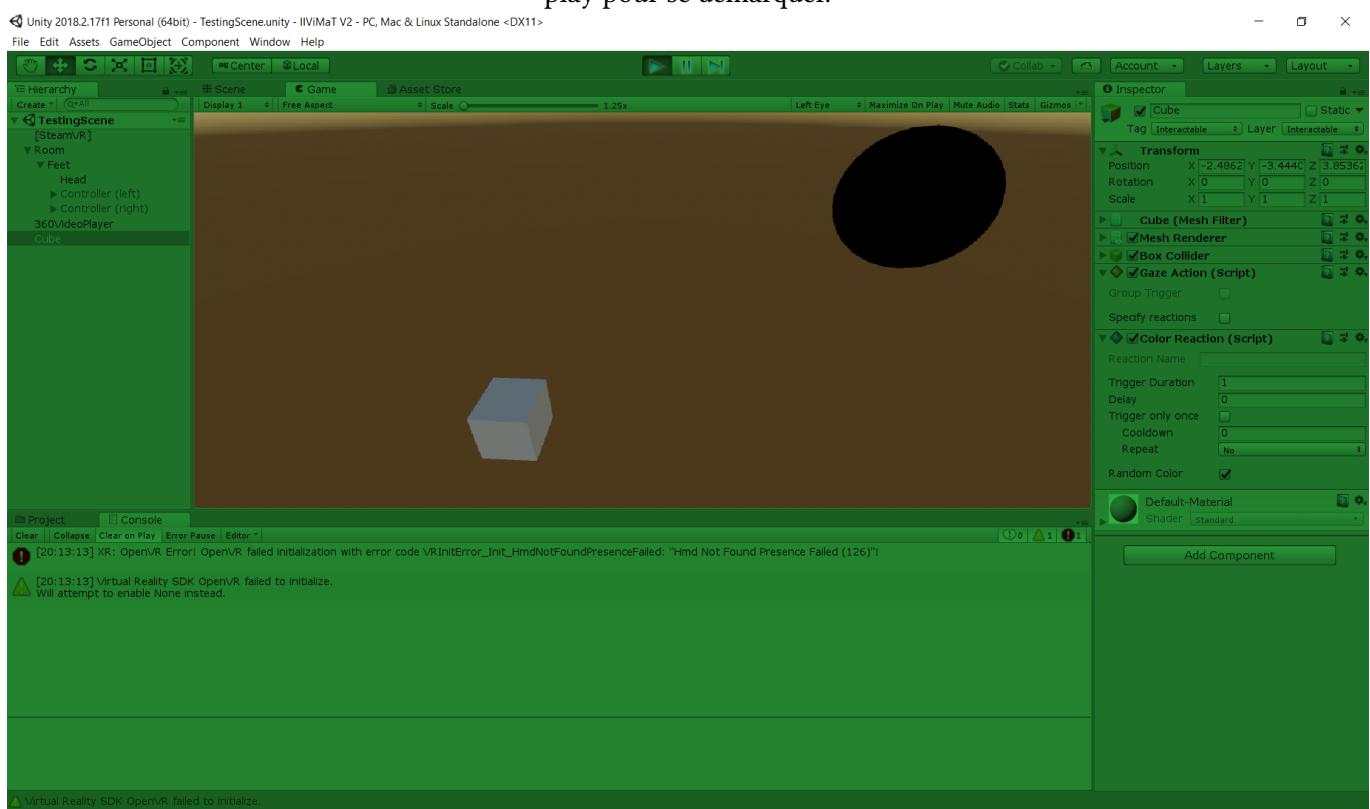
Dans l'onglet "Scene", vous pouvez voir vos objets dans l'espaces 3D. Vous pouvez déplacer votre point de vue en maintenant clique droit appuyé puis en bougeant la souris et en appuyant sur les touches ZQSD.

Dans l'onglet "Game", vous voyez votre scène du point de vue du spectateur au démarrage de votre expérience. Pour tester les interactions et vérifier le bon déroulement de votre expérience, vous devez passer en mode "Play". Cela se fait en cliquant sur le bouton play au dessus de la fenêtre.

Lorsque vous êtes en mode play, toutes les modifications faites sur votre scène sont temporaires !

Cela est utile pour rapidement tester différents paramètres de vos composants, mais également très décevant lorsqu'on oublie ce détail et perd ses modifications.

Pour ne pas confondre le mode play avec le mode d'édition normal, Unity prend une couleur (ici, verte) en mode play pour se démarquer.

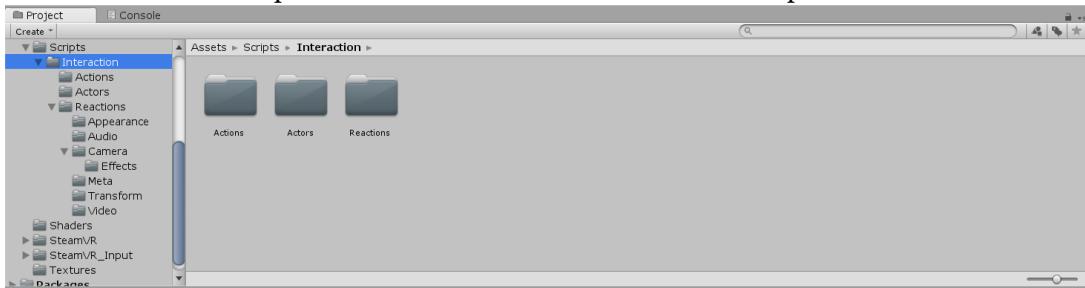


Dans l'onglet "Asset Store" vous pouvez chercher et ajouter des **assets** (des objets 3D, des sons, des images, etc...) à votre projet.

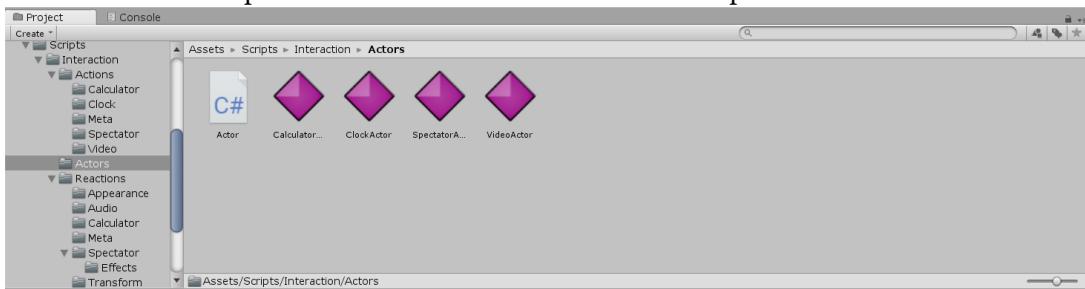
1.4 Explorateur de fichiers

Dans cette fenêtre vous pouvez parcourir vos dossiers pour ajouter des objets ou des composants à votre scène par glisser-déposer.

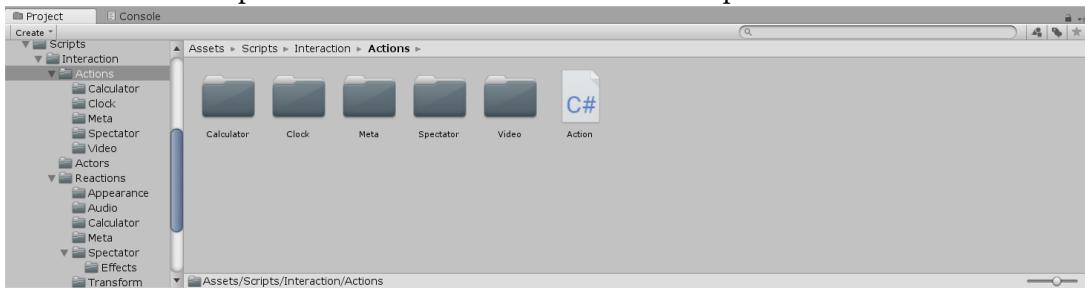
Tous les composants d'IIViMaT sont dans le dossier Scripts/Interaction/.



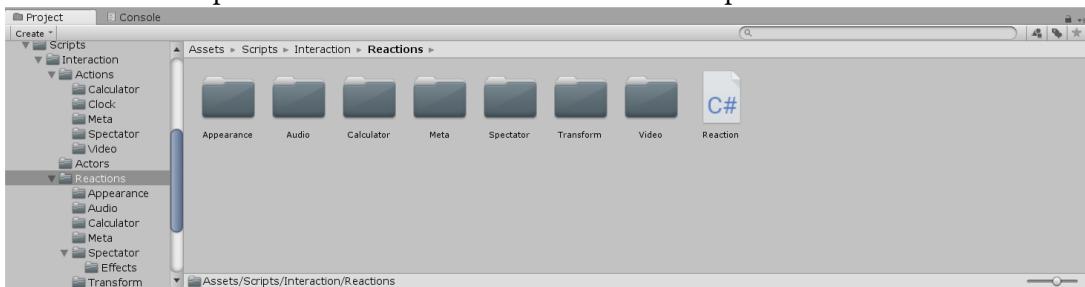
Tous les composants **acteurs** sont dans le dossier Scripts/Interaction/Actors/.



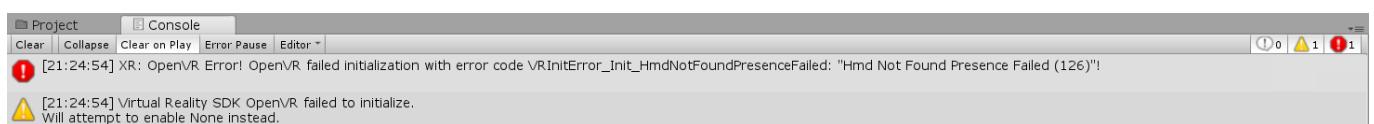
Tous les composants **actions** sont dans le dossier Scripts/Interaction/Actions/.



Tous les composants **réactions** sont dans le dossier Scripts/Interaction/Reactions/.



Dans cette même fenêtre, dans l'onglet "Console", vous verrez potentiellement apparaître des informations, avertissements et erreurs sur votre projet. Ces messages sont utiles pour connaître la cause d'un problème, gardez donc souvent un œil sur la console.



Au lancement du projet vous aurez normalement cette erreur et cet avertissement.

L'erreur signifie que le casque VR n'est pas branché. L'avertissement signifie que la réalité virtuelle n'a pas pu être activée. Ces messages devraient disparaître lorsque vous testez votre projet avec un casque VR.

2 Utilisation

2.1 Introduction

Les **acteurs** provoquent (on parle de "trigger") des **actions**, qui elles-mêmes déclenchent (on parle aussi de "trigger") des **réactions**.

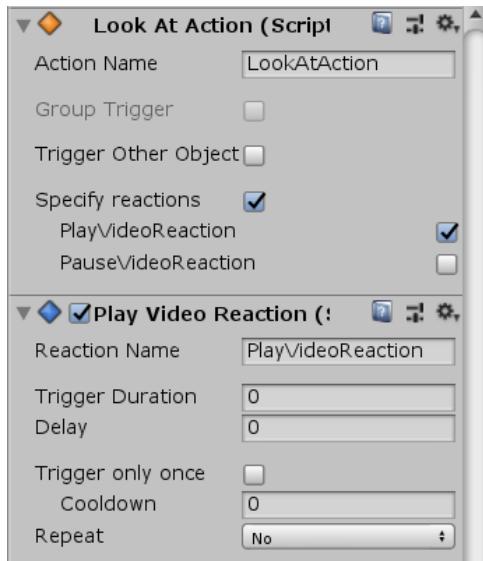
Pour spécifier à quelles interactions un objet réagit, vous devez lui ajouter un composant **action** qui décrit par quel acteur et dans quelles circonstances l'action est provoquée. Pour spécifier comment un objet réagit à une interaction, vous devez lui ajouter un composant **réaction** qui décrit de quelle manière l'objet doit se comporter.

Les réactions ne fonctionnent que si une action vient les déclencher. Les actions ne fonctionnent que si un acteur vient les provoquer.

Plusieurs réactions peuvent être associées à une même action. Plusieurs actions peuvent déclencher les mêmes réactions. Par défaut, une action déclenche toutes les réactions sur le même objet.

Vous pouvez spécifier quelles réactions une action déclenche en cochant la case **Specify reactions** et en cochant les réactions souhaitées. Une action peut également déclencher des réactions sur un autre objet. Pour cela, il faut cocher la case **Trigger Other Object**.

Vous pouvez combiner ces deux options pour déclencher des réactions spécifiques d'un objet spécifique.



2.2 Créer un projet

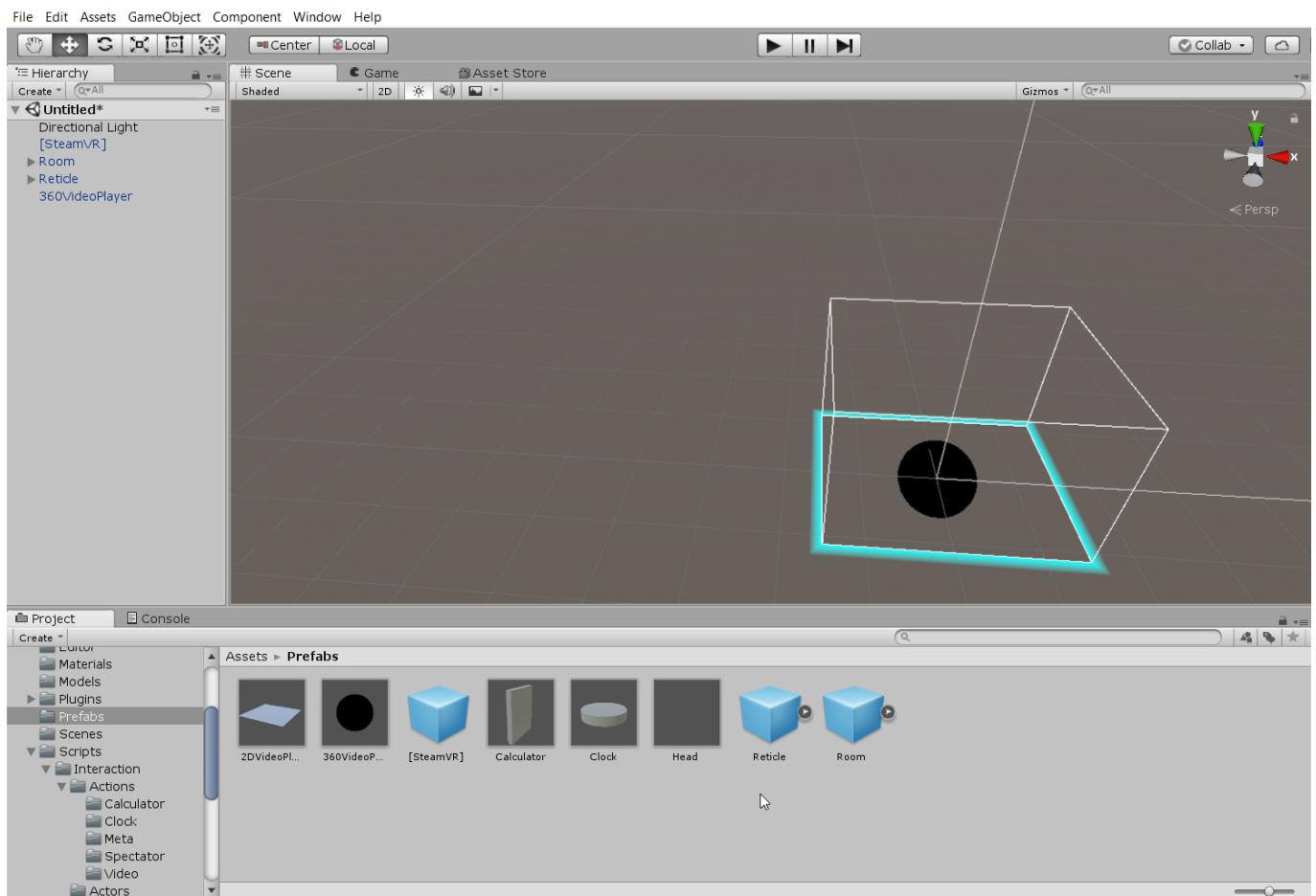
Pour créer un nouveau projet, cliquez sur le menu File > New Scene.

Supprimez l'objet nommé **Main Camera** car vous allez le remplacer par un objet équivalent ensuite. Celui-ci représente le point de vue du spectateur (le point de vue de la caméra dans un film traditionnel). Ouvrez ensuite le dossier **Prefabs** dans la fenêtre explorateur de fichiers.

Glissez-déposez le préfab [**SteamVR**] et **Room** pour avoir un projet vide qui fonctionne en réalité virtuelle.

Pour ajouter une nouvelle sphère 360, glissez-déposez le préfab **360VideoPlayer**. De même, pour une vidéo 2D, utilisez le préfab **2DVideoPlayer**

Pour avoir un indicateur du centre de l'écran lorsque vous testez votre projet, glissez-déposez le préfab **Reticle**.



2.3 Glossaire

2.3.1 Actor

Vous trouverez les acteurs dans le dossier Assets/Scripts/Interaction/Actors/.

SpectatorActor : Cet acteur doit être présent sur la tête du spectateur. Il provoque des actions liées au comportement du spectateur (Regard, proximité, ...). Il n'y a qu'un seul SpectatorActor par projet car il n'y a qu'un spectateur par projet.

VideoActor : Cet acteur doit être présent sur chaque objet vidéo. Il provoque des actions liées au déroulement de la vidéo (Début, fin, temps, ...).

ClockActor : Cet acteur peut être présent sur n'importe quel objet. Il provoque des actions liées au temps écoulé depuis le début de l'expérience. Il ne doit y avoir qu'un seul ClockActor par projet car il n'y a qu'une seule ligne du temps et vous avez plusieurs ClockActor vous risquez donc de déclencher les mêmes actions plusieurs fois.

CalculatorActor : Cet acteur peut être présent sur n'importe quel objet ayant un composant Calculator. Il provoque des actions liées à un calcul sur la valeur du composant Calculator.

2.3.2 Action

Vous trouverez les actions dans le dossier Assets/Scripts/Interaction/Actions/. Il y a ensuite 5 sous-dossiers permettant de trouver plus facilement une action.

Spectator

Ces actions sont provoquées par l'acteur spectateur.

ProximityAction : Exécute les réactions voulues lorsque l'acteur est dans une zone proche d'un objet.

EnterAction : Exécute les réactions voulues lorsque l'acteur rentre dans une zone proche d'un objet.

ExitAction : Exécute les réactions voulues lorsque l'acteur sort d'une zone proche d'un objet.

GazeAction : Exécute les réactions voulues lorsque l'acteur est en train de regarder un certain objet.

LookAtAction : Exécute les réactions voulues lorsque l'acteur pose son regard sur un certain objet.

LookAwayAction : Exécute les réactions voulues lorsque l'acteur retire son regard d'un certain objet.

Video

Ces actions sont provoquées par les acteurs vidéos.

PlayVideoAction : Exécute les réactions voulues lorsque la vidéo est démarrée.

PauseVideoAction : Exécute les réactions voulues lorsque la vidéo est mise en pause.

VideoTimeAction : Exécute les réactions voulues lorsque la vidéo atteint un certain temps.

EndVideoAction : Exécute les réactions voulues lorsque la vidéo se termine (y compris si elle boucle).

Clock

Ces actions sont provoquées par l'acteur horloge.

ClockTimeAction : Exécute les réactions voulues lorsque le temps écoulé depuis le début de l'expérience atteint un certain temps.

Calculator

Ces actions sont provoquées par les acteurs calculatrices.

CalculatorEqualsToAction : Exécute les réactions voulues lorsque la valeur de la calculatrice (Calculator) associée est égale à la valeur souhaitée.

CalculatorMultipleOfAction : Exécute les réactions voulues lorsque la valeur de la calculatrice (Calculator) associée est un multiple de la valeur souhaitée.

Meta

Ces actions permettent une manipulation plus poussée de vos projets.

PropagatedAction : Exécute les réactions voulues lorsqu'une réaction propage son déclenchement.

2.3.3 Reaction

Vous trouverez les réactions dans le dossier Assets/Scripts/Interaction/Reactions/. Il y a ensuite 7 sous-dossiers permettant de trouver plus facilement une réaction.

Transform

Ces réactions permettent de modifier le composant Transform de l'objet.

PositionTransformReaction : Permet de modifier la position.

OrientationTransformReaction : Permet de modifier l'orientation.

ScaleTransformReaction : Permet de modifier la proportion, c'est à dire la taille relative. Cela revient à multiplier la taille par un nombre.

SizeTransformReaction : Permet de modifier la taille. Cela revient à additionner un nombre à la taille.

RotationTransformReaction : Permet de tourner autour d'un point ou objet selon l'axe souhaité.

Appearance

Ces réactions permettent de changer l'apparence de l'objet.

ColorReaction : Permet de modifier la couleur.

TransparencyReaction : Permet de modifier la transparence.

VisibilityReaction : Permet de rendre visible ou invisible.

Audio

Ces réactions permettent de manipuler les effets sonores.

PlayAudioReaction : Permet de jouer un son.

PauseAudioReaction : Permet de mettre en pause la lecture d'un son. La lecture peut ensuite être reprise au même point.

StopAudioReaction : Permet d'arrêter la lecture d'un son. Le son est remis à zéro.

ChangeVolumeAudioReaction : Permet de modifier le volume du son.

Video

Ces réactions permettent de manipuler les vidéos 2D et 360.

PlayVideoReaction : Permet de jouer une vidéo à partir du moment où elle s'est arrêtée.

PauseVideoReaction : Permet de mettre en pause la lecture d'une vidéo. La lecture peut ensuite être reprise au même point avec une PlayVideoReaction.

StartVideoReaction : Permet de jouer une vidéo à partir du début ou du moment souhaité.

StopVideoReaction : Permet d'arrêter la lecture d'une vidéo. La vidéo est remise à zéro.

Spectator

Ces réactions affectent le spectateur.

ChangeViewModeReaction : Permet de changer de mode de vue. En **Fixed view**, le spectateur est coincé dans une vidéo et la vidéo est immobile. La vidéo ne bouge pas et le spectateur non plus. En **Follow view**, le spectateur est coincé dans une vidéo mais la vidéo est mobile. La vidéo se déplace avec le spectateur. En **Free view**, le spectateur peut se déplacer librement. Il rentre et sort des vidéos comme il le souhaite.

TeleportationReaction : Permet de téléporter le spectateur à un certain endroit.

BlackAndWhiteCameraEffectReaction : Permet d'afficher un effet visuel global de noir et blanc.

FadeCameraEffectReaction : Permet d'afficher un effet visuel global de fondu en noir (entrée et sortie).

Meta

Ces réactions permettent des manipulations plus poussées de vos projets.

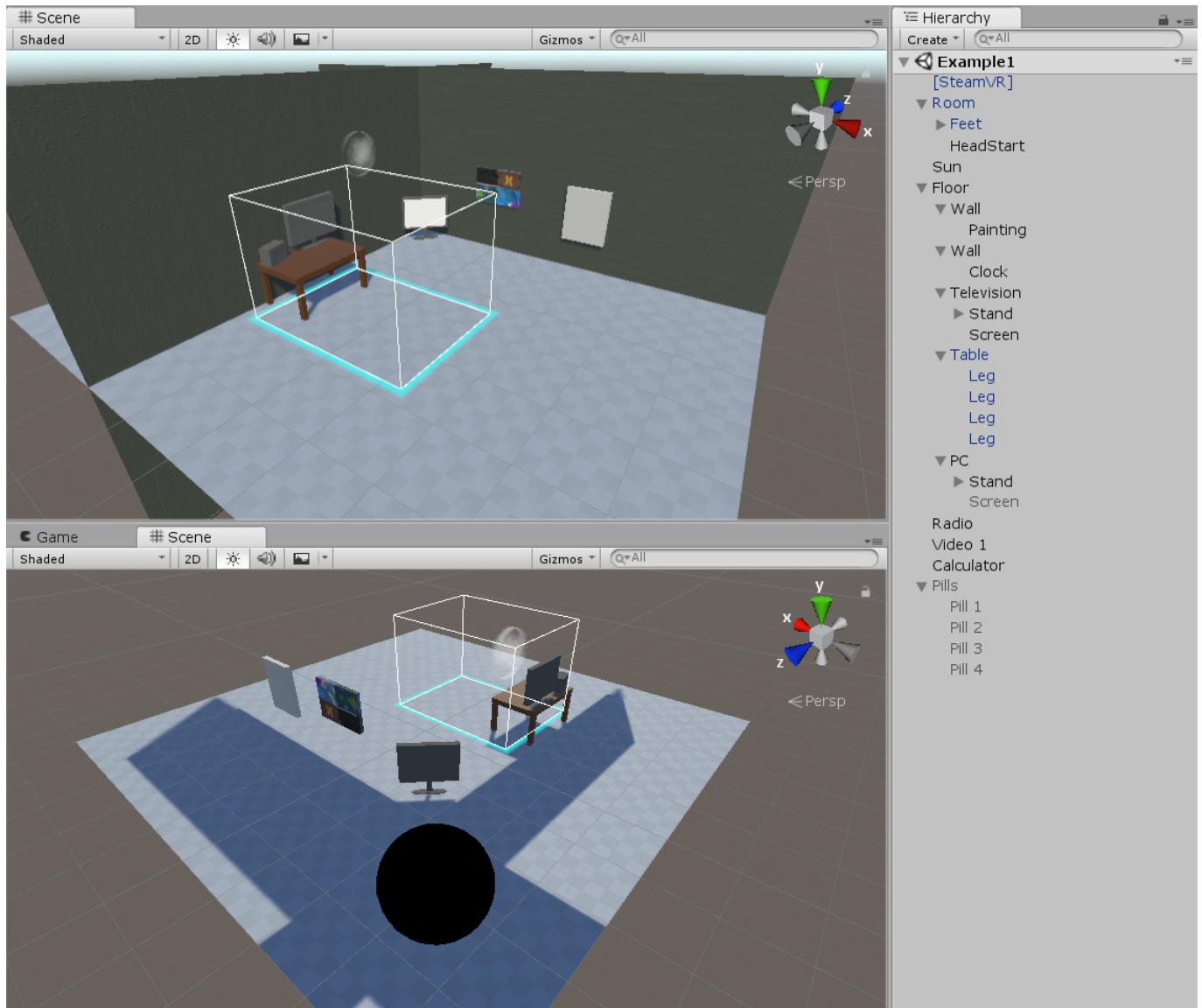
ActivationReaction : Permet d'activer ou de désactiver un objet. Un objet désactivé est non seulement invisible, mais il ne peut également pas interagir.

PropagateReaction : Permet de provoquer des **actions propagées** (PropagatedAction) sur d'autres objets. Cela permet de propager l'interaction à d'autres objets.

3 Exemple

Cet exemple de projet met en situation la majeure partie des acteurs, actions et réactions disponibles. Il met également en évidence certains des problèmes qui apparaissent lorsqu'un projet devient suffisamment complexe et donne des exemples de solutions. Il serait possible de réaliser ce même projet de plein de manières différentes, le but est ici d'illustrer l'utilité des différents composants.

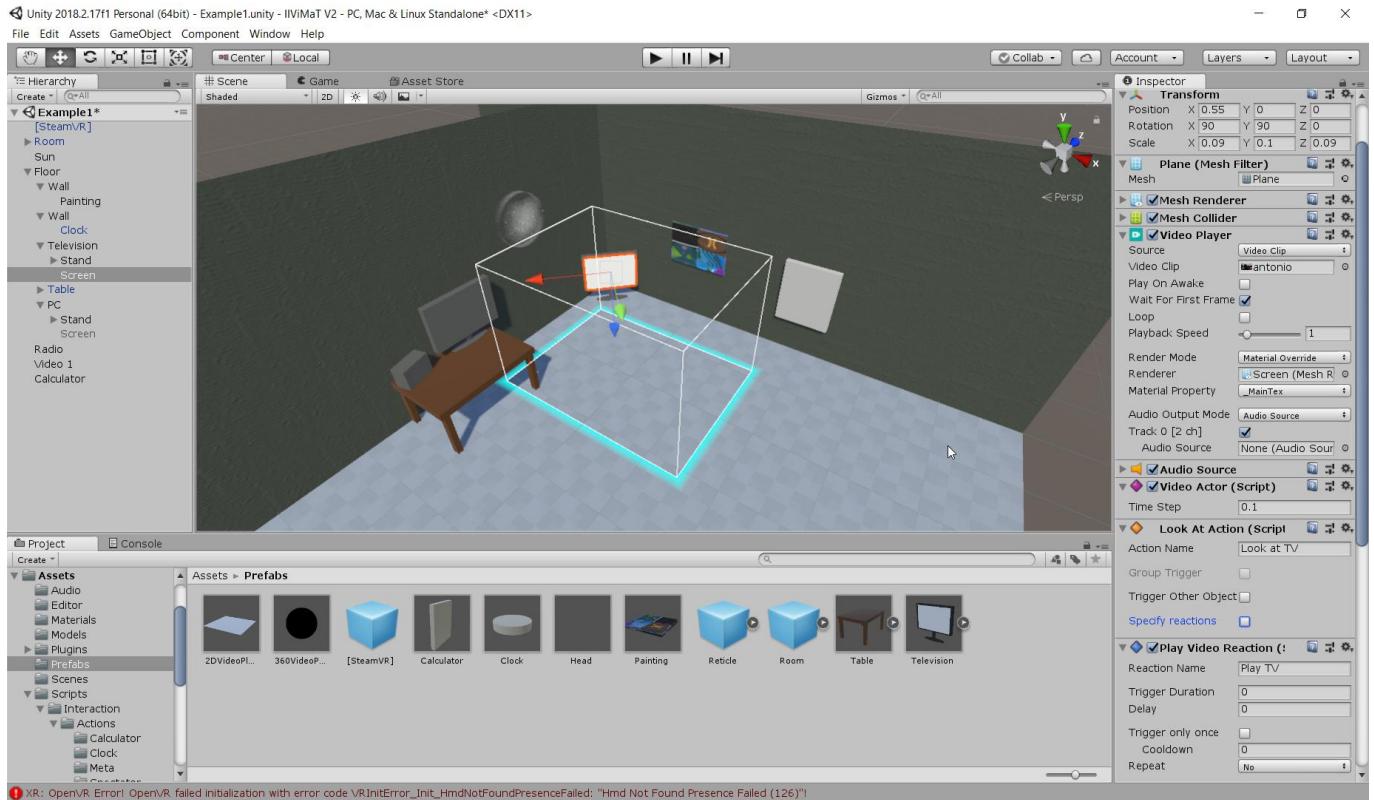
Voici l'espace 3D de notre projet et la hiérarchie de ses objets :



Vous pouvez tester cette expérience en ouvrant la scène Example1.

3.1 Lancement et arrêt d'une vidéo 2D par le regard

On souhaite d'abord que lorsque le spectateur regarde l'écran de télévision, celui-ci joue une vidéo. Pour cela on place sur notre écran une **LookAtAction** et une **PlayVideoReaction**.

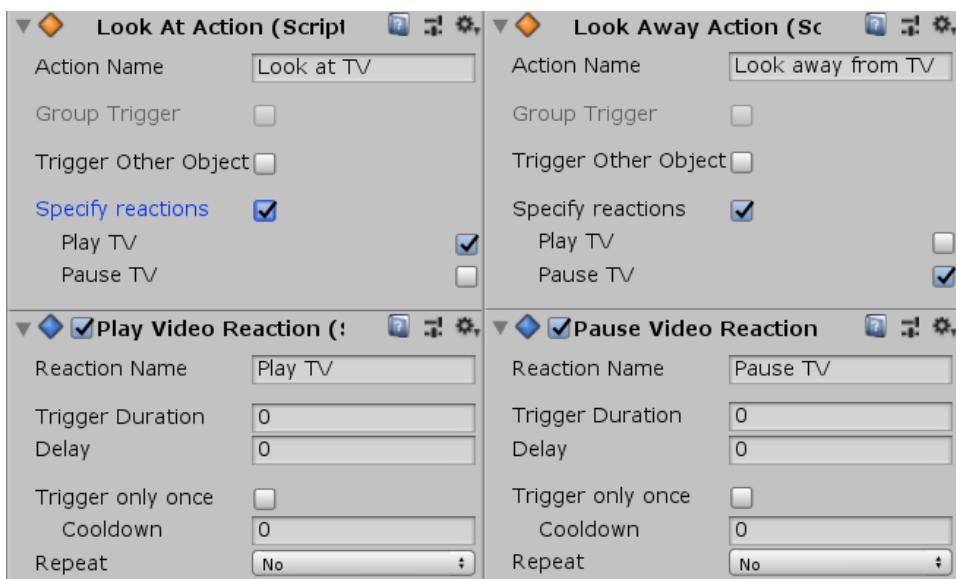


N'oublions pas de sélectionner la vidéo souhaitée dans le composant **Video Player**, paramètre **Video Clip**, et de décocher la case **Play On Awake**, pour que la vidéo ne démarre pas au lancement de l'expérience.

On souhaite ensuite que la vidéo se mette en pause quand le spectateur regarde ailleurs, et qu'elle reprenne lorsqu'on la regarde de nouveau. Il faut alors ajouter une **LookAwayAction** et une **PauseVideoReaction**.

Il ne faut pas oublier, maintenant qu'il y a 2 actions, de spécifier quelle action va avec quelle réaction. Si on ne spécifie rien les 2 actions déclenchaient chacune les 2 réactions.

On décoche également les cases **Trigger Only Once** pour que ces réactions soient déclenchées à chaque fois que les actions sont faites, et non pas seulement la première fois.



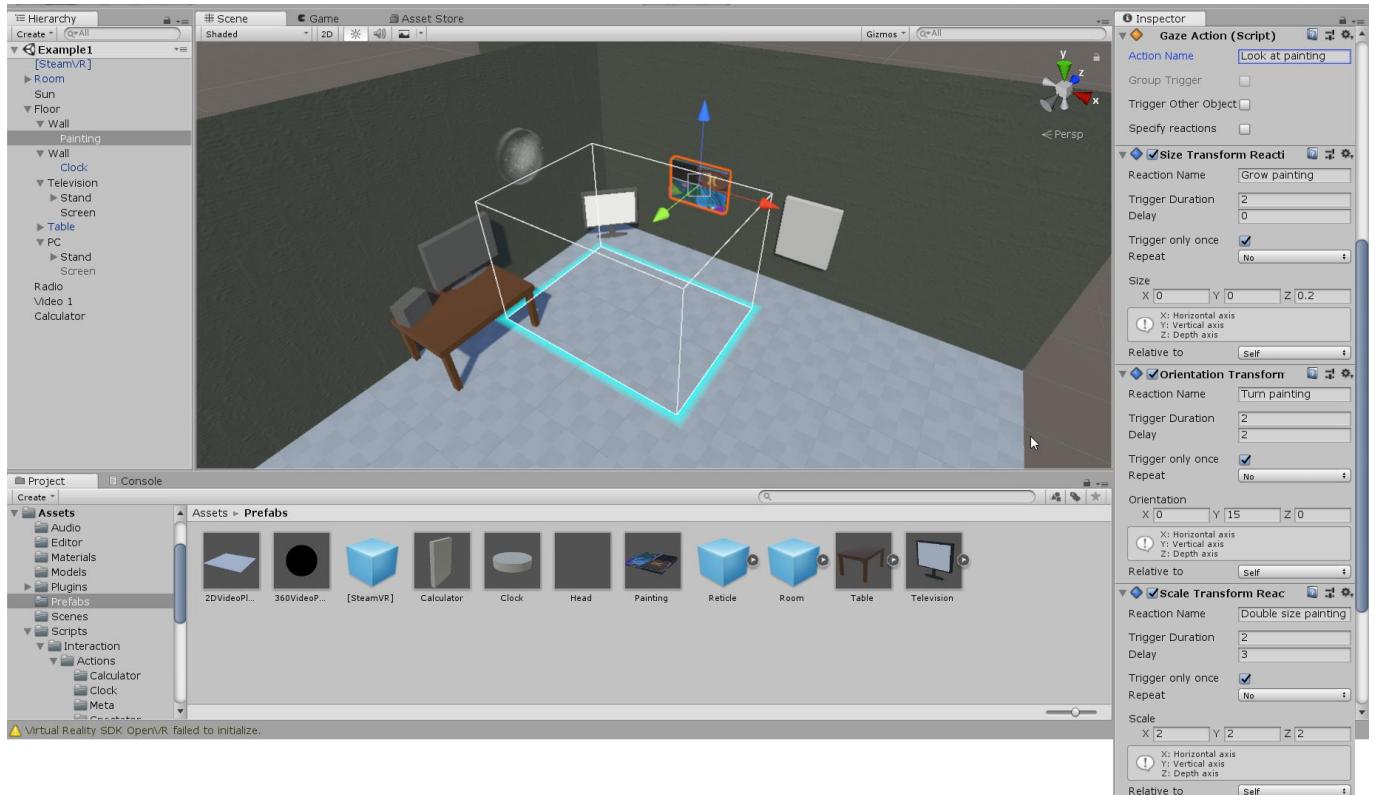
3.2 Modification d'un objet par le regard

On souhaite maintenant que lorsqu'on regarde le tableau pendant 2 secondes, celui-ci grandisse, puis 2 secondes plus tard qu'il s'incline, puis 1 seconde après ça qu'il double de taille.

Pour cela il faut lui ajouter une **GazeAction** et 3 réactions.

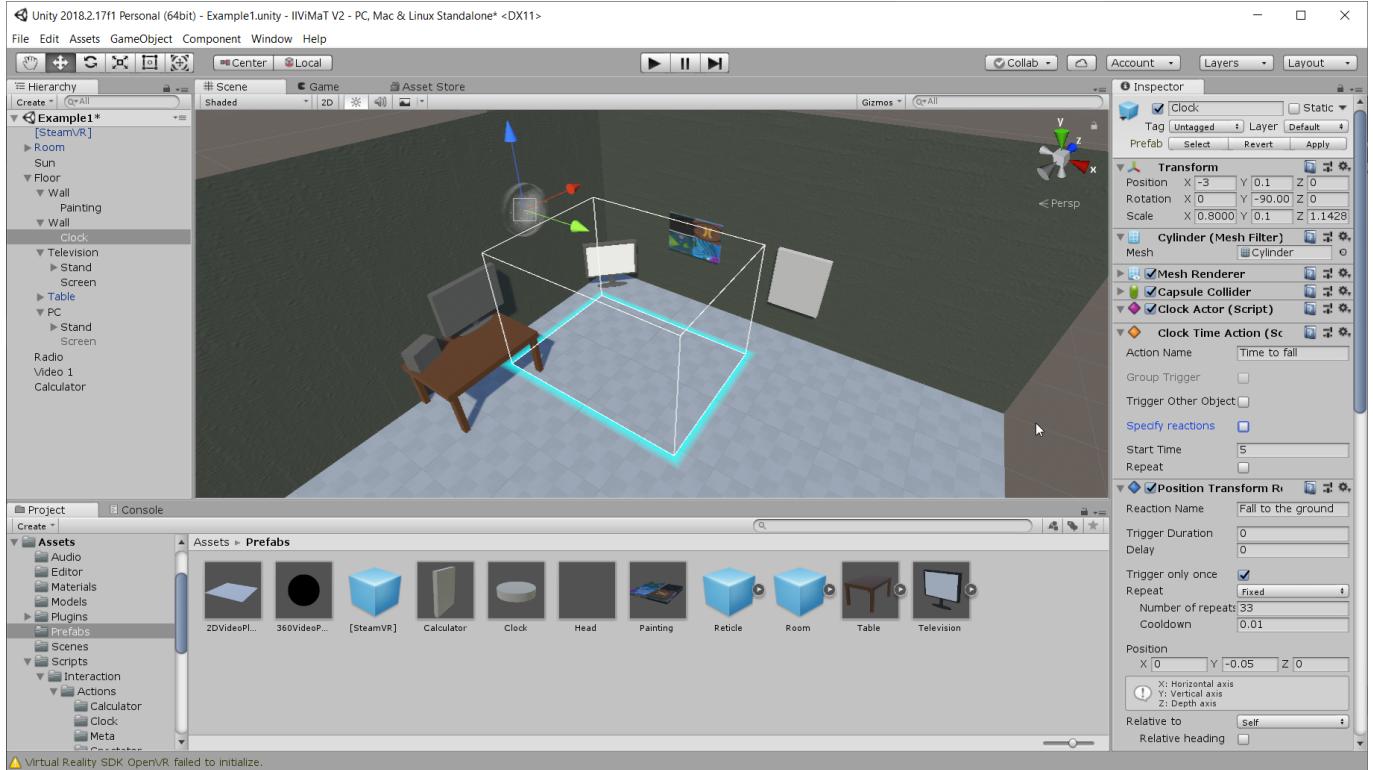
Une **SizeTransformReaction** pour qu'il grandisse, une **OrientationTransformReaction** pour qu'il tourne et une **ScaleTransformReaction** pour qu'il double de taille.

On spécifie les paramètres **Trigger Duration** pour que les réactions ne se déclenchent qu'après 2 secondes ininterrompues où l'action est provoquée. On spécifie les paramètres **Delay** pour que les réactions se fassent successivement et non simultanément.



3.3 Déplacement d'objet automatique

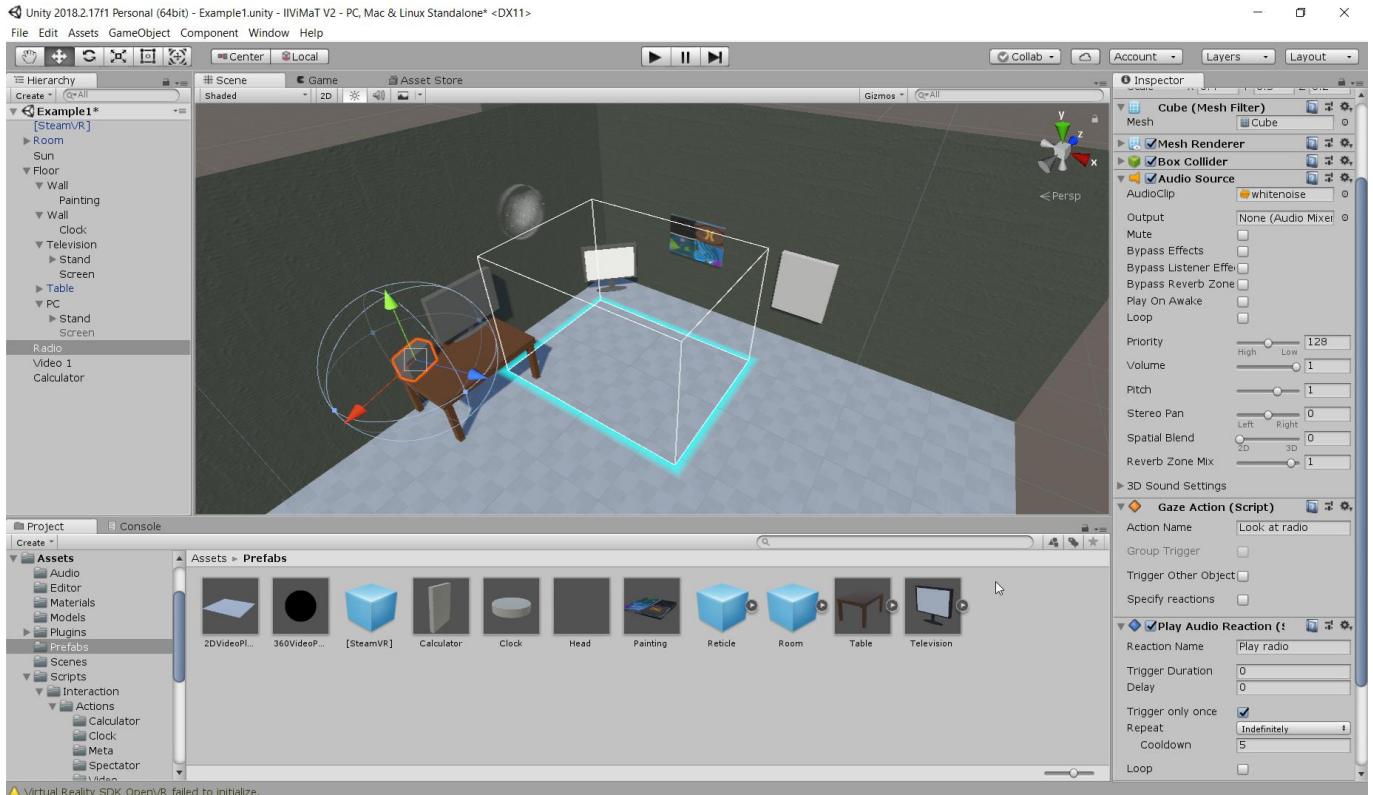
On souhaite maintenant que l'horloge accrochée au mur tombe au sol 5 secondes après le début de l'expérience, peu importe les actions du spectateur. On utilise alors une **ClockTimeAction** et une **PositionTransformReaction**. Pour animer la chute au sol, on fait descendre l'horloge (Y négatif) d'une petite distance (0.05 mètre) 33 fois, avec 0.01 seconde entre chaque mouvement. Il faut bien veiller à sélectionner une position relative à soit-même (**Relative to Self**), car la nouvelle position est 0.05 mètres en dessous de la précédente.



3.4 Lancement d'un son en boucle

On souhaite maintenant que lorsque l'utilisateur s'approche de la radio, celle-ci joue un son, qui se répétera indéfiniment toutes les 5 secondes. Pour cela on ajoute une **GazeAction** et une **PlayAudioReaction**. On sélectionne une répétition infinie avec un temps de 5 secondes entre chaque répétition.

On décoche bien la case **Play On Awake** dans le composant **Audio Source** pour que le son ne joue pas automatiquement au lancement de l'expérience.

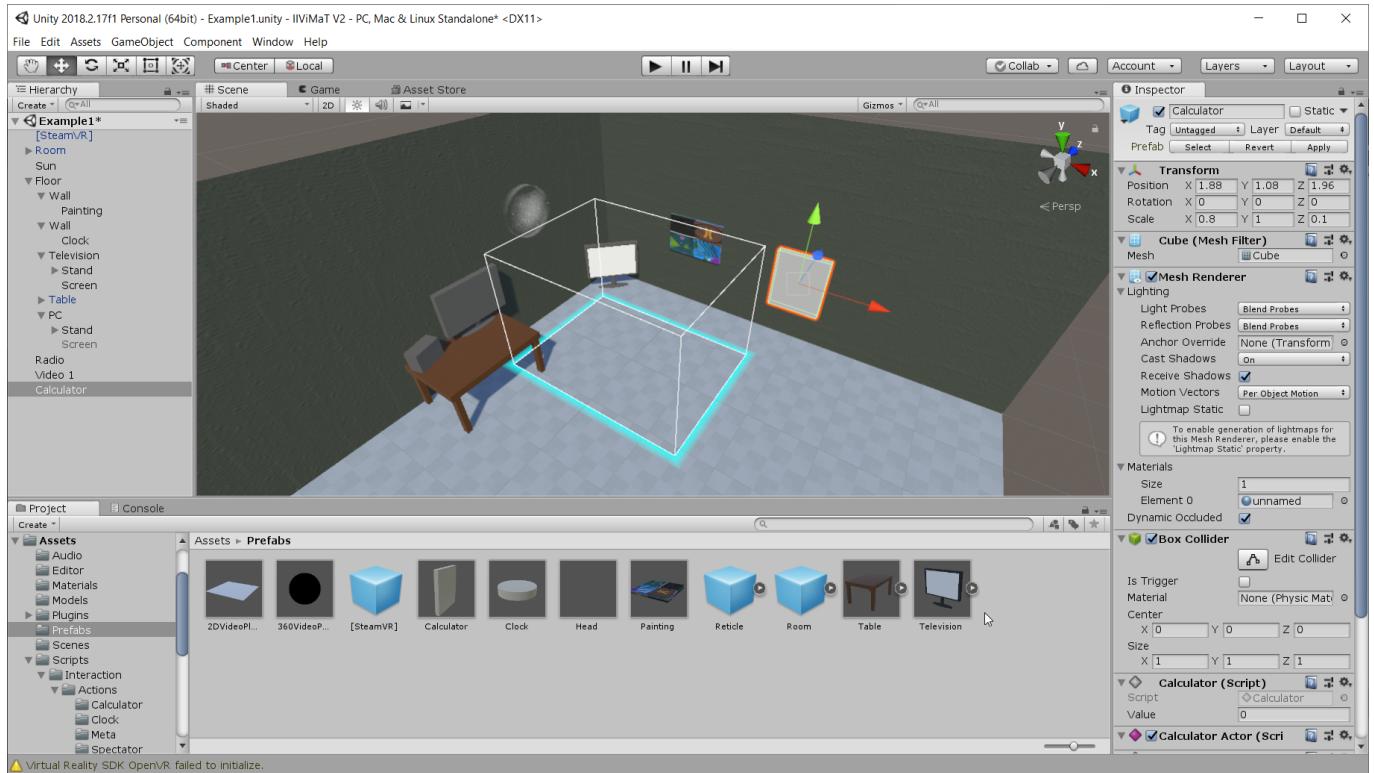


3.5 Activation d'un objet après un nombre d'interactions fixé

On souhaite maintenant allumer l'écran de l'ordinateur, mais uniquement une fois que le spectateur a exploré tous les objets dans la pièce. Si on allumait l'écran suite à une action sur un des objets de la pièce, le spectateur n'aurait alors qu'à interagir avec cet objet en particulier pour afficher l'écran. Si on allumait l'écran suite aux actions sur chacun des objets, le spectateur n'aurait alors qu'à interagir avec un seul objet quelconque.

Le seul moyen de savoir si le spectateur a bel et bien interactué avec tous les objets, c'est de compter le nombre d'objet avec lesquels il a interactué.

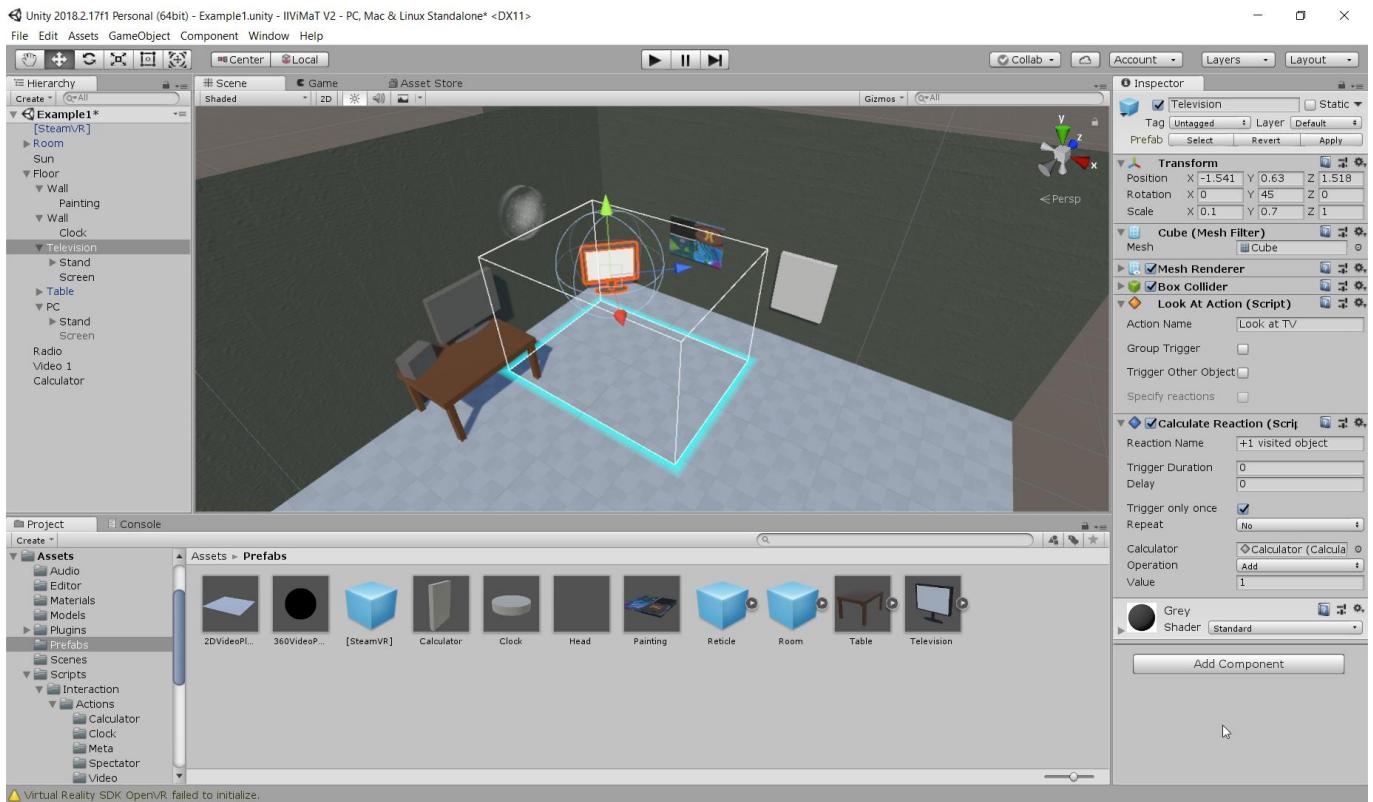
Pour cela, il faut ajouter un composant calculatrice (**Calculator**) à un objet de notre projet. Ici, elle est placée sur une toile de peinture neuve accrochée au mur. On ajoute également un composant **CalculatorActor** pour pouvoir provoquer des actions liées à la valeur de notre calculatrice.



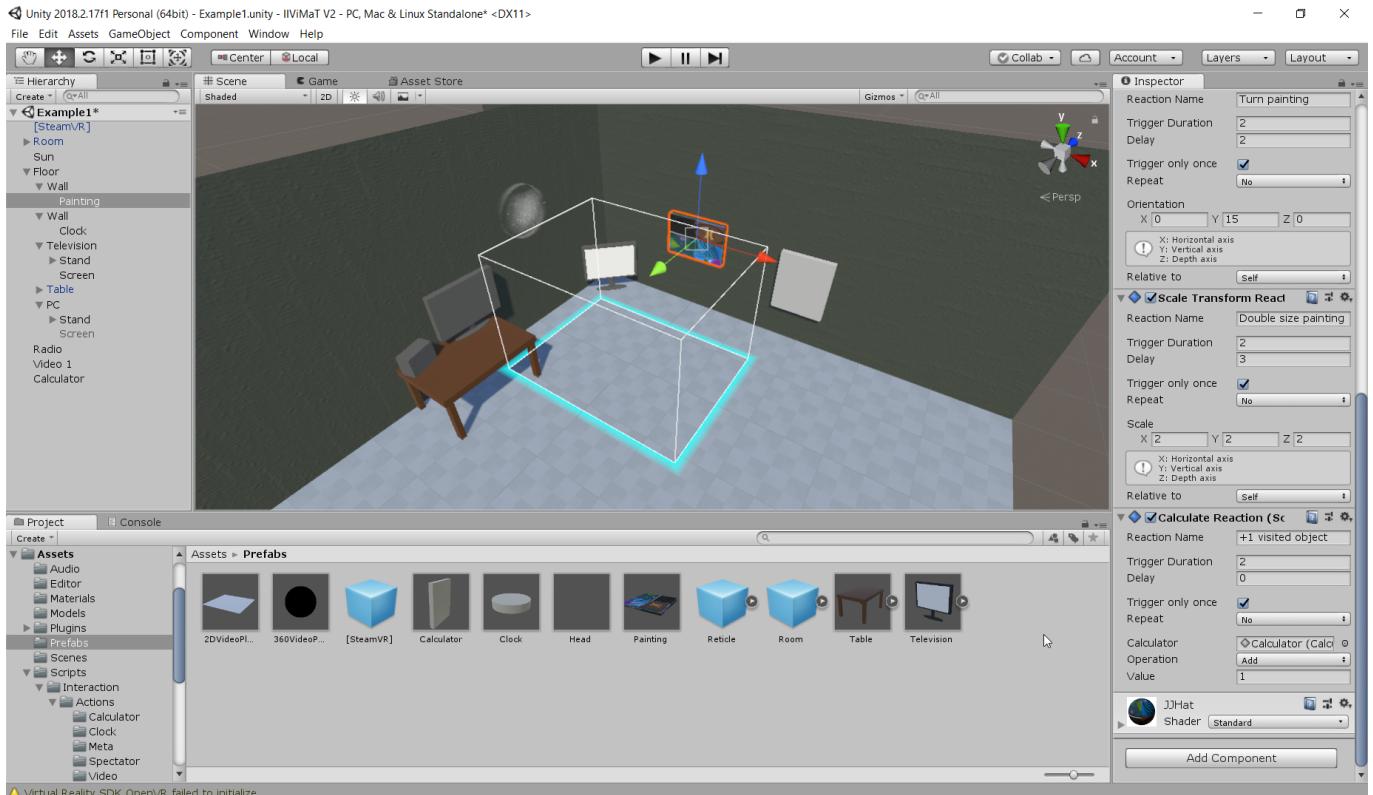
On met la valeur de départ de notre calculatrice à 0. On veut donc que chaque objet ajoute 1 à cette valeur lorsque le spectateur interactue.

On va alors placer des **CalculateReaction** sur chacun des objets. On spécifie à chaque fois que l'on souhaite ajouter 1 à la valeur de notre calculatrice.

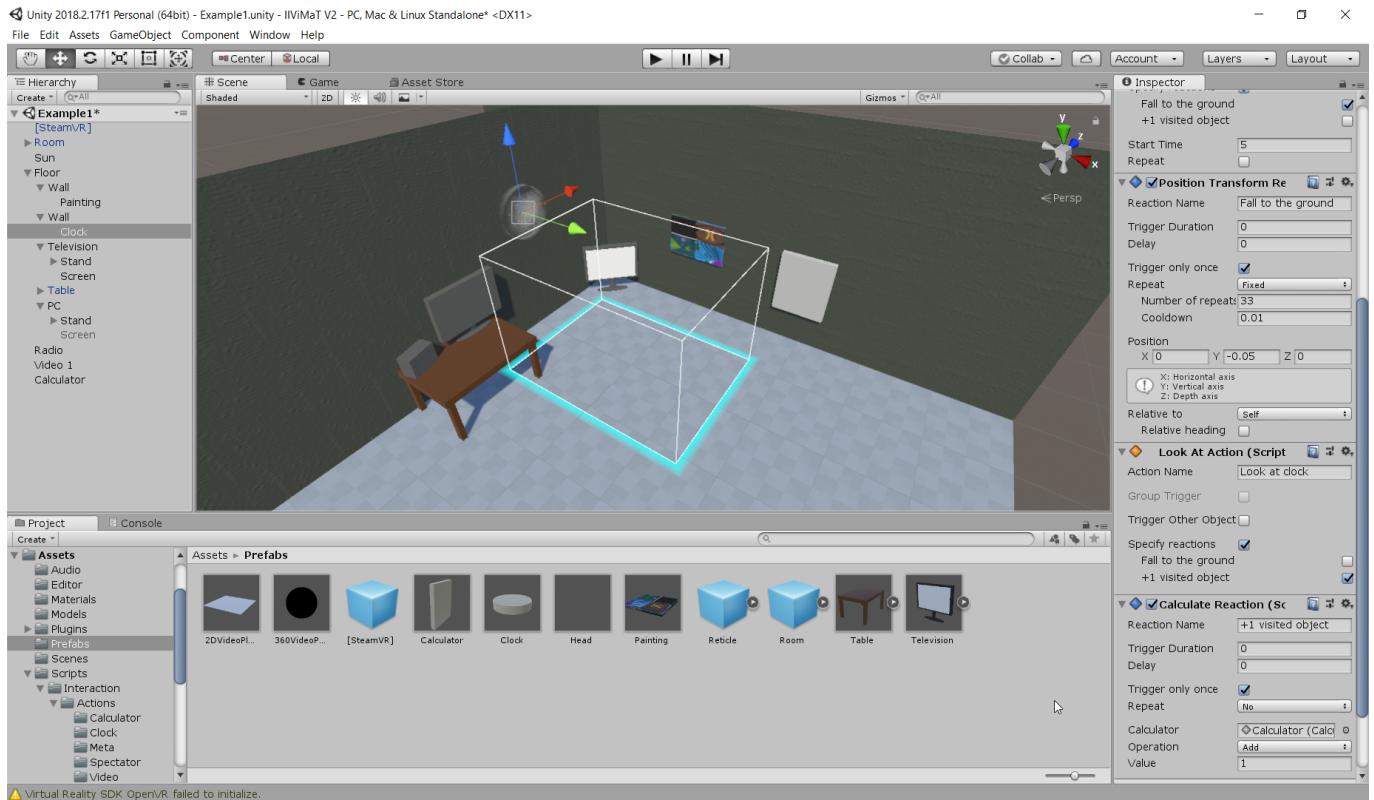
Sur l'objet télévision :



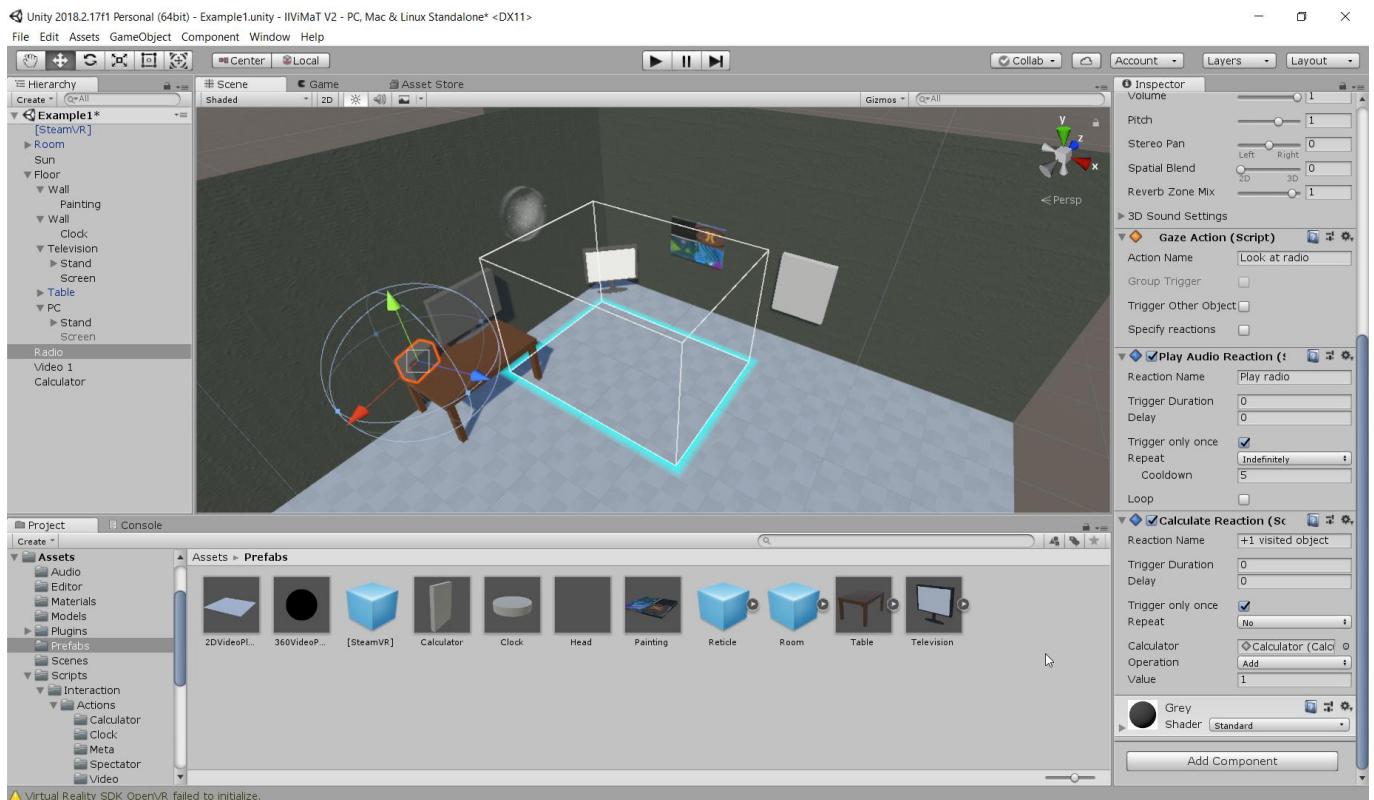
Sur le tableau :



Sur l'horloge, lorsqu'on la regarde (Il faut donc ajouter une **LookAtAction** et spécifier quelle action déclenche quelle réaction)

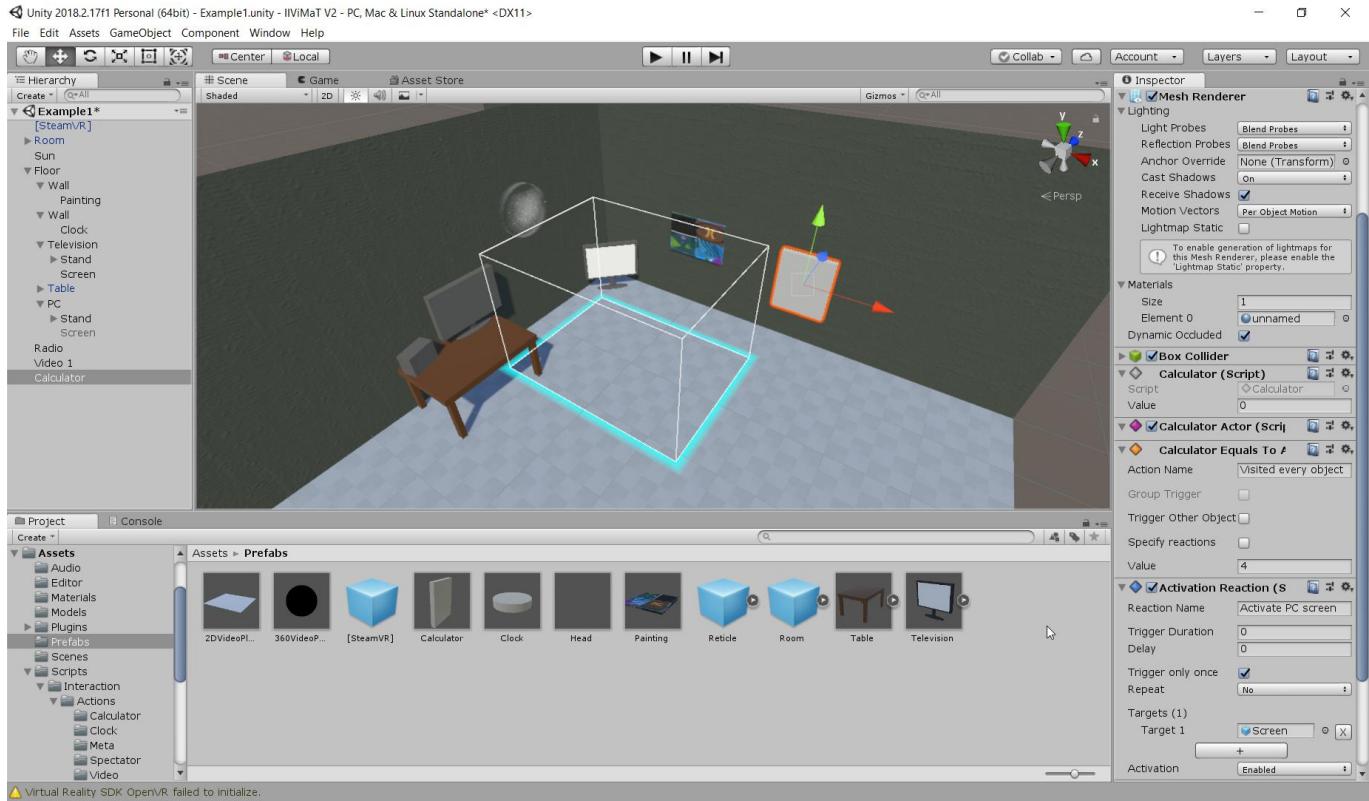


Et enfin sur la radio :

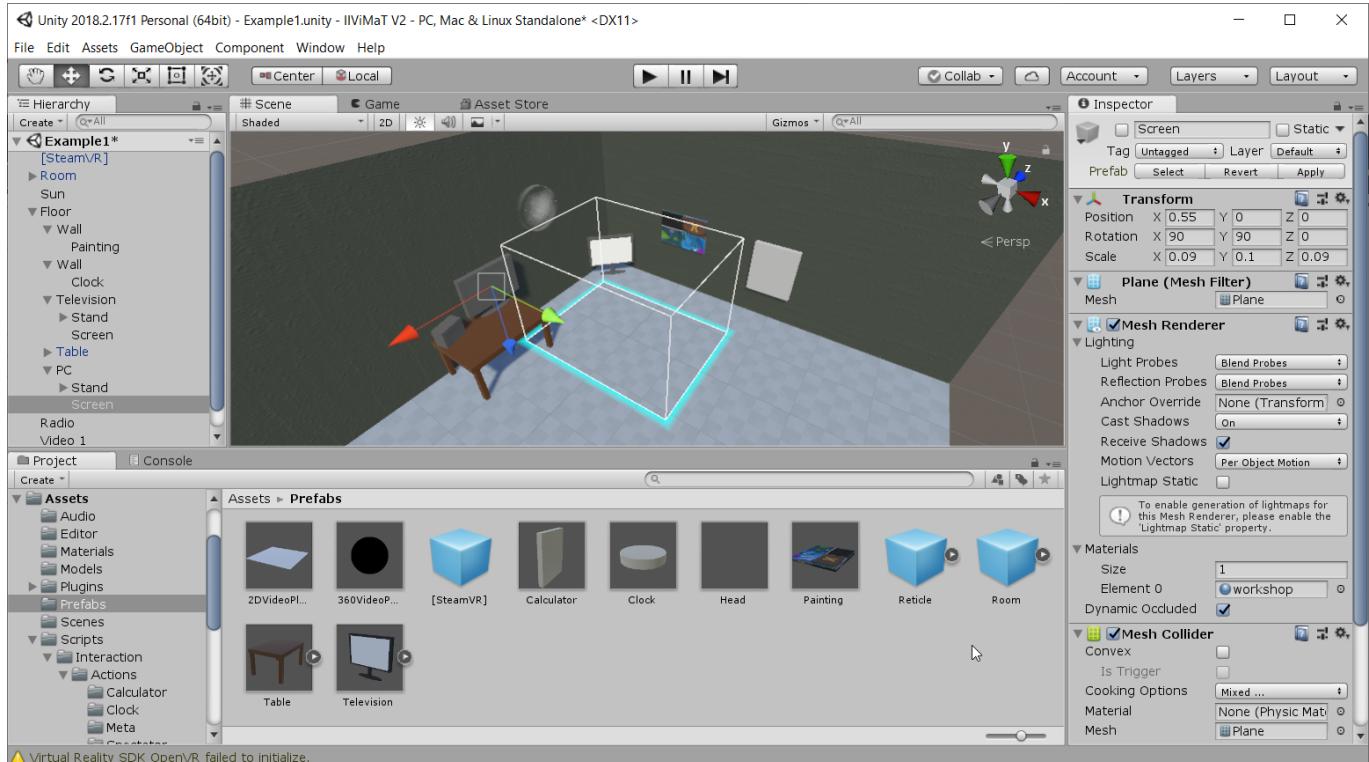


Une fois que le spectateur a interactué avec tous les objets, la calculatrice devrait donc avoir la valeur 4. Il faut maintenant indiquer quoi faire lorsque la calculatrice atteint cette valeur.

Pour rappel, on veut allumer l'écran lorsque le spectateur a visité toute la pièce. On ajoute donc sur notre objet ayant le composant calculatrice une **CalculatorEqualsToAction** et une **ActivationReaction**. On précise qu'on souhaite activer l'écran d'ordinateur.



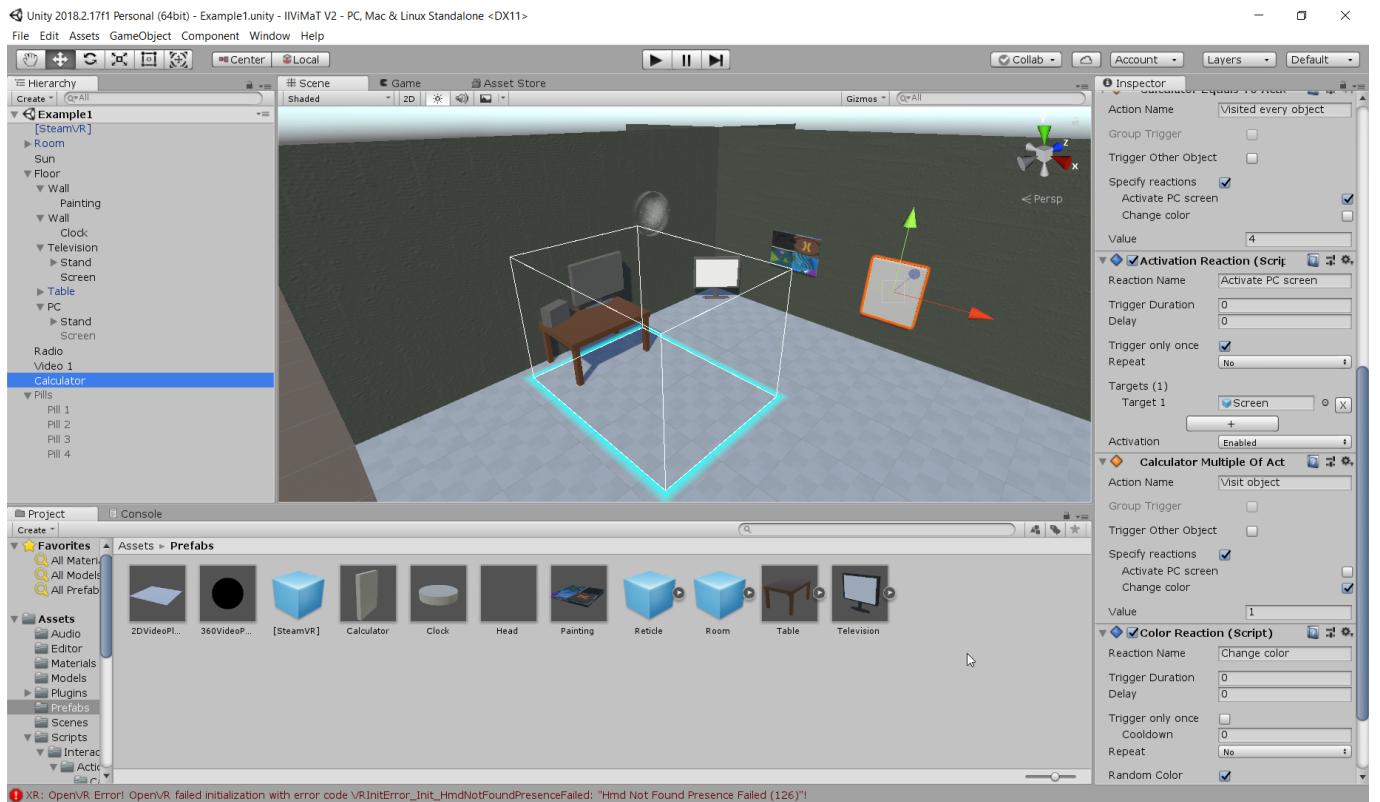
Il faut veiller à désactiver l'écran par défaut, sans quoi il sera déjà activé au lancement de l'expérience. Pour cela il faut décocher la case à côté de son nom en haut à droite.



On souhaite maintenant que notre toile neuve prenne une couleur aléatoire à chaque fois que le spectateur interagit avec un objet de la pièce.

On place alors une **CalculatorMultipleOfAction** et une **ColorReaction**. On indique que l'action doit être provoquée lorsque la valeur de la calculatrice est un multiple de 1, soit à chaque fois que le nombre change (Si on voulait changer la couleur toutes les 2 interactions, il suffirait de remplacer ce nombre par 2).

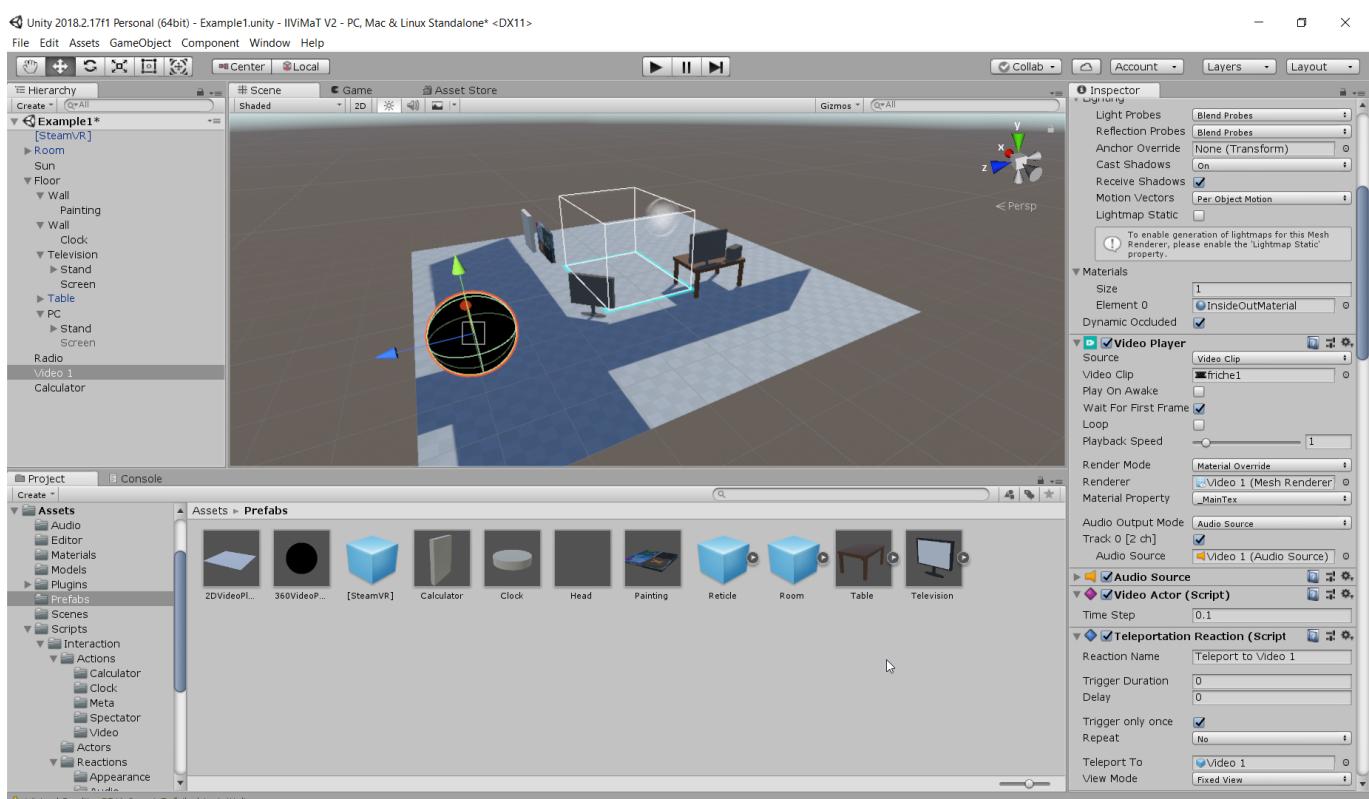
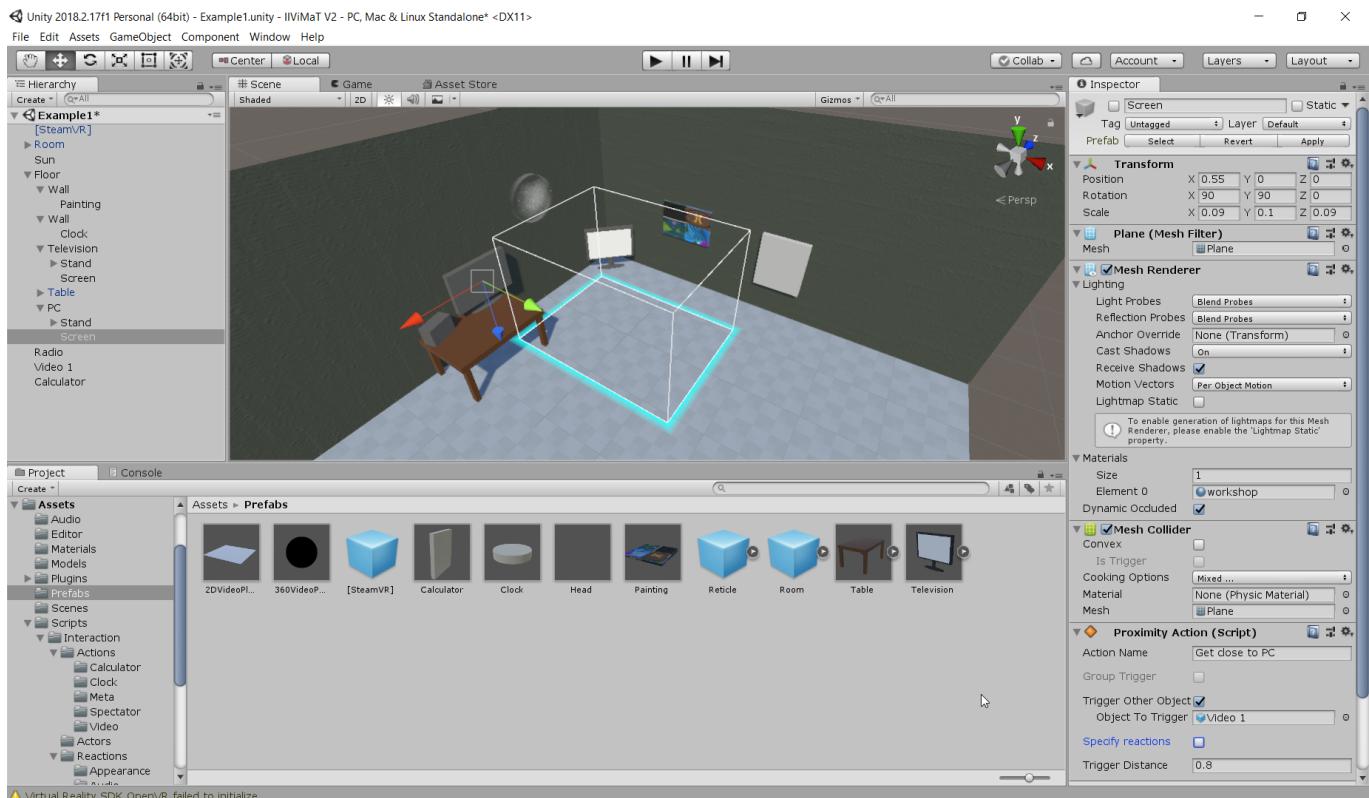
On précise que l'on souhaite une couleur aléatoire. Comme il y a maintenant 2 actions sur cet objet on précise quelle action va avec quelle réaction.



3.6 Téléportation dans une sphère 360

On souhaite maintenant que lorsque le spectateur s'approche de l'écran allumé, il soit téléporté dans une vidéo 360. Il faut donc ajouter sur l'écran une **ProximityAction**. Il est faisable d'ajouter une **TeleportationReaction** sur l'écran, en indiquant qu'on souhaite se téléporter sur la sphère 360, cependant pour des raisons qui seront plus claires dans la suite du projet, on choisit de mettre la **TeleportationReaction** directement sur la sphère 360, et de cocher la case **Trigger Other Object** dans la **ProximityAction** afin d'indiquer qu'on souhaite déclencher une réaction qui se trouve sur un autre objet.

On spécifie un mode de vue fixe pour que le spectateur ne puisse pas sortir de la sphère 360. On note que c'est bien parce que l'écran d'ordinateur est désactivé par défaut que le spectateur ne pouvait pas interagir avec avant qu'il soit activé. Si l'écran avait simplement été invisible, le spectateur aurait pu déclencher l'action de proximité avant que la calculatrice ne compte jusqu'à 4.

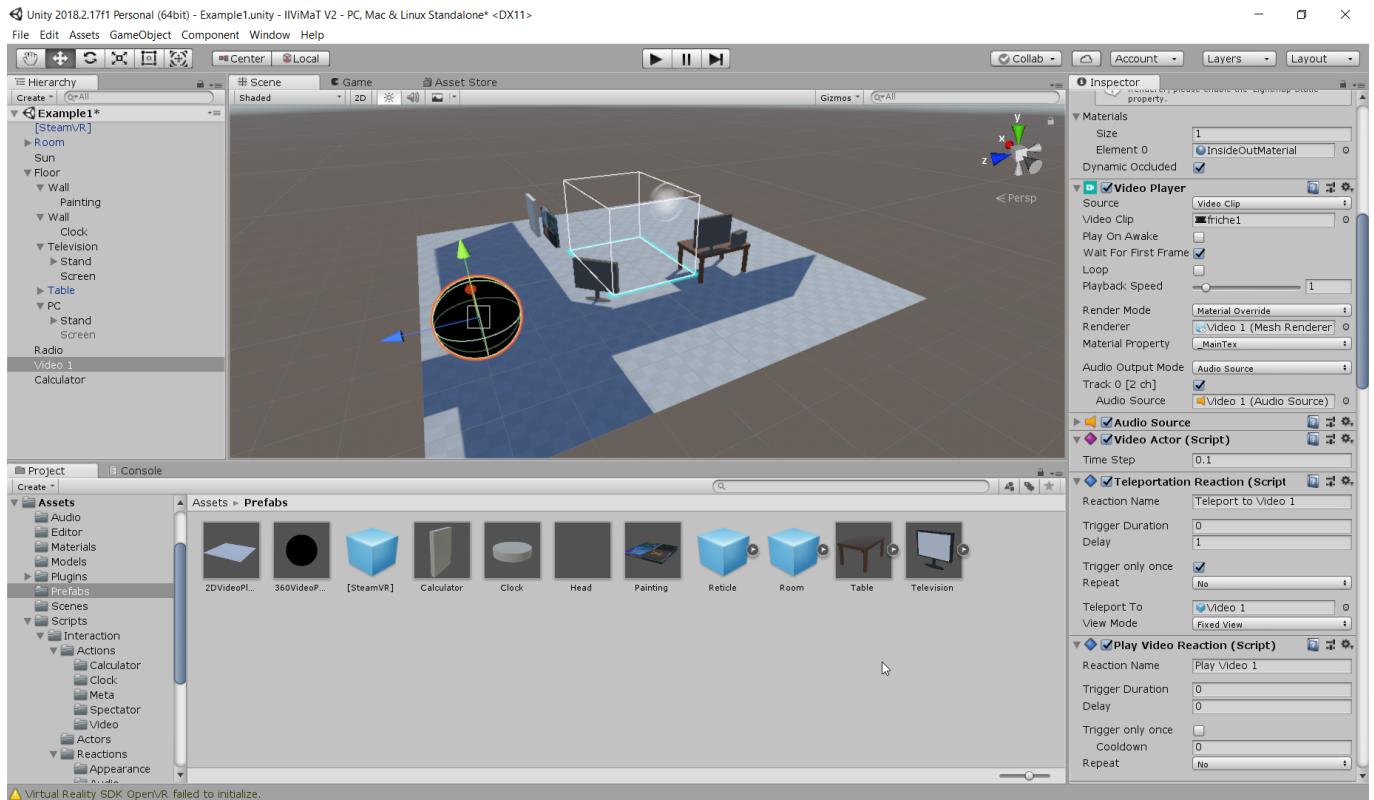


3.7 Lancement d'une vidéo 360 après téléportation

On souhaite maintenant que la vidéo se lance lorsque le spectateur est téléporté dans la sphère 360. On ajoute donc une **PlayVideoReaction** sur la sphère 360, qui sera donc elle aussi déclenchée par la **ProximityAction** sur l'écran d'ordinateur.

La téléportation se fera donc en même temps que la vidéo se lancera. Cependant, comme c'est le premier lancement de la vidéo, elle doit d'abord charger en mémoire avant de pouvoir être lancée. Cela est très rapide pour des vidéos courtes, mais ça reste perceptible et se traduit par le fait que le spectateur se retrouve dans une sphère 360 noire pendant une fraction de seconde avant que la vidéo ne se lance.

Une solution simple pour régler ce problème, est d'ajouter un petit délai sur la réaction de téléportation.

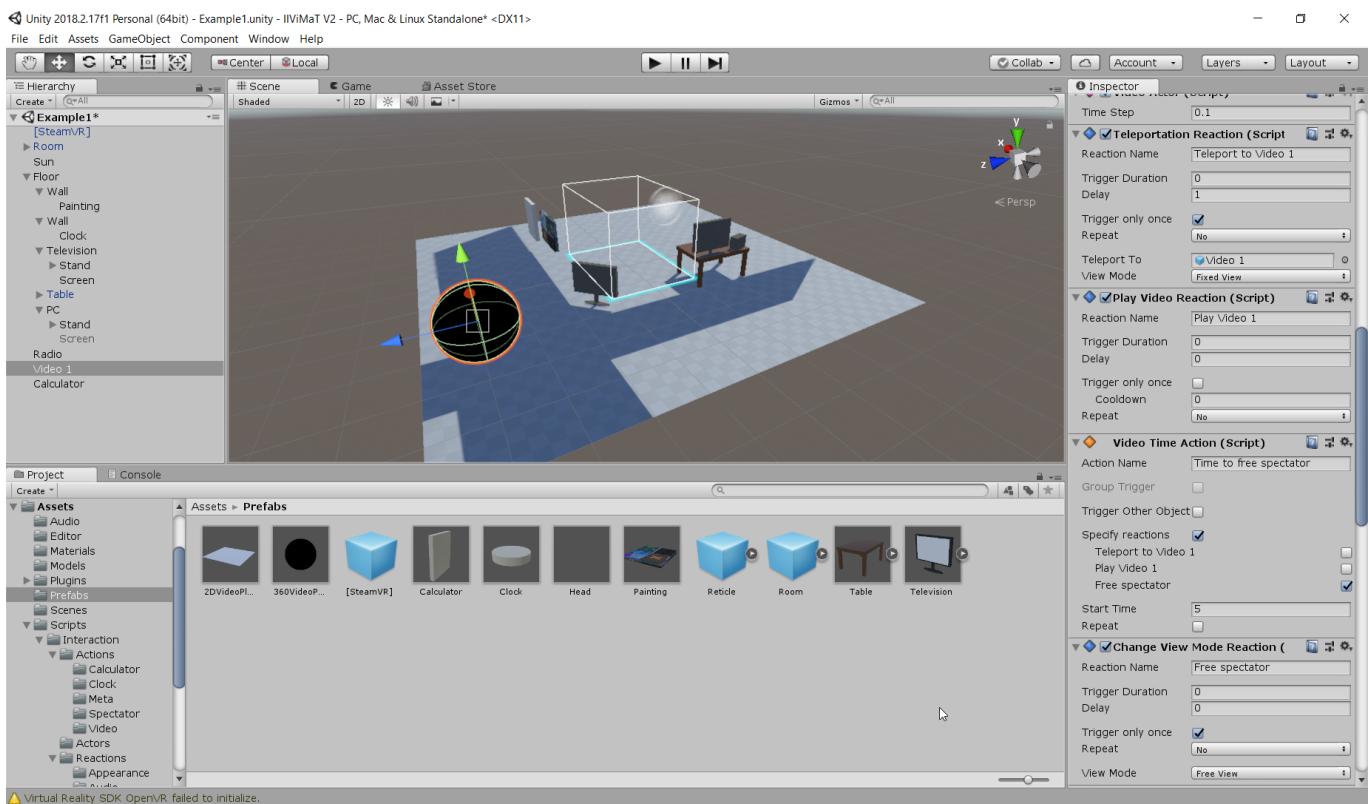


3.8 Changement de mode de vue à un certain temps de la vidéo

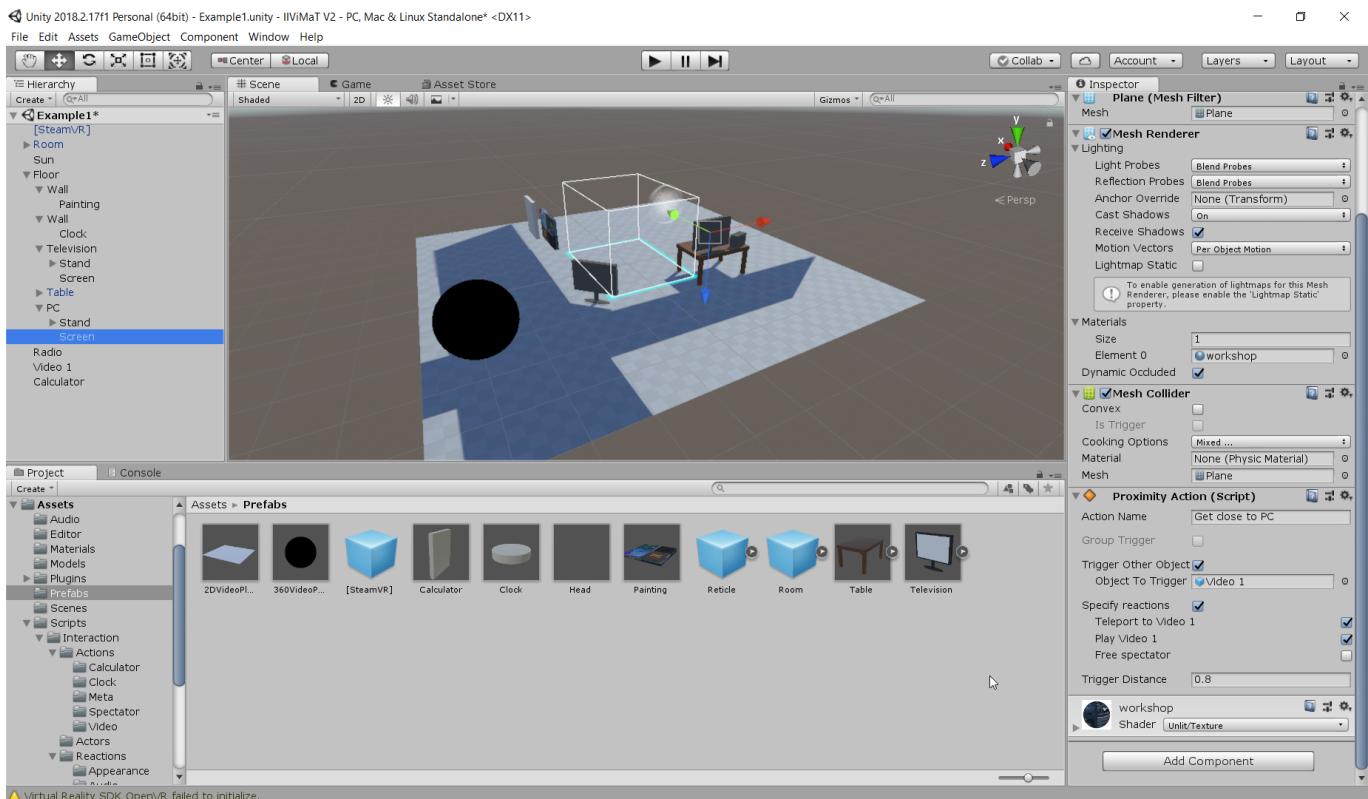
Les murs de la pièces sont fait d'objets planaires ("plane" en anglais). Les plans ont la propriété qu'on peut voir à travers dans un sens mais pas dans l'autre. Ici, cela permettrait au spectateur de voir la pièce depuis l'extérieur. Cependant, le spectateur est coincé à l'intérieur d'une sphère 360 en train de regarder la vidéo et n'a donc pas connaissance de son environnement 3D ni de sa position dans cet environnement. On souhaite alors qu'après 5 secondes à regarder la vidéo, le spectateur puisse sortir de la sphère 360 pour explorer.

Il s'agit alors d'ajouter une **VideoTimeAction** et une **ChangeViewModeReaction**.

On précise sur l'action que l'on veut qu'elle soit provoquée à 5 secondes de la vidéo. On précise sur la réaction que l'on souhaite passer en mode libre. Maintenant qu'il y a 2 actions, il faut préciser sur chacune quelles réactions elles déclenchent.



Pour nos réactions de téléportation et de lancement de vidéo, il faut de même les spécifier sur la **ProximityAction** qui est sur l'écran d'ordinateur.

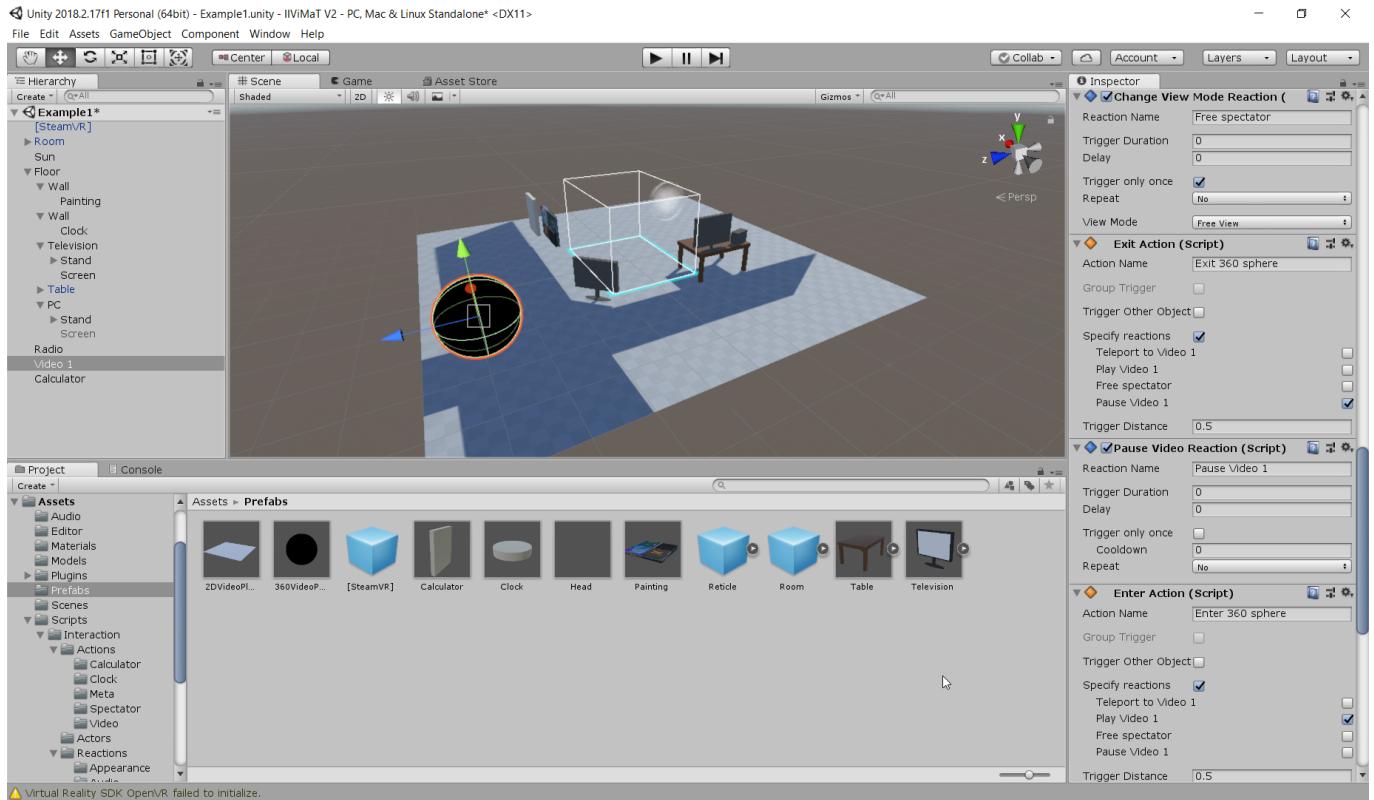


3.9 Lancement et arrêt d'une vidéo 360 par proximité

On souhaite maintenant que si le spectateur décide d'explorer et de sortir de la sphère, la vidéo se mette en pause. On souhaite également que si il entre de nouveau dans la sphère, la vidéo continue.

Pour cela on ajoute une **ExitAction** à laquelle on associe une **PauseVideoReaction** et une **EnterAction** à laquelle on associe la réaction déjà existante **PlayVideoReaction**.

On spécifie une distance de 0.5 car le diamètre de notre sphère mesure 1 mètre.

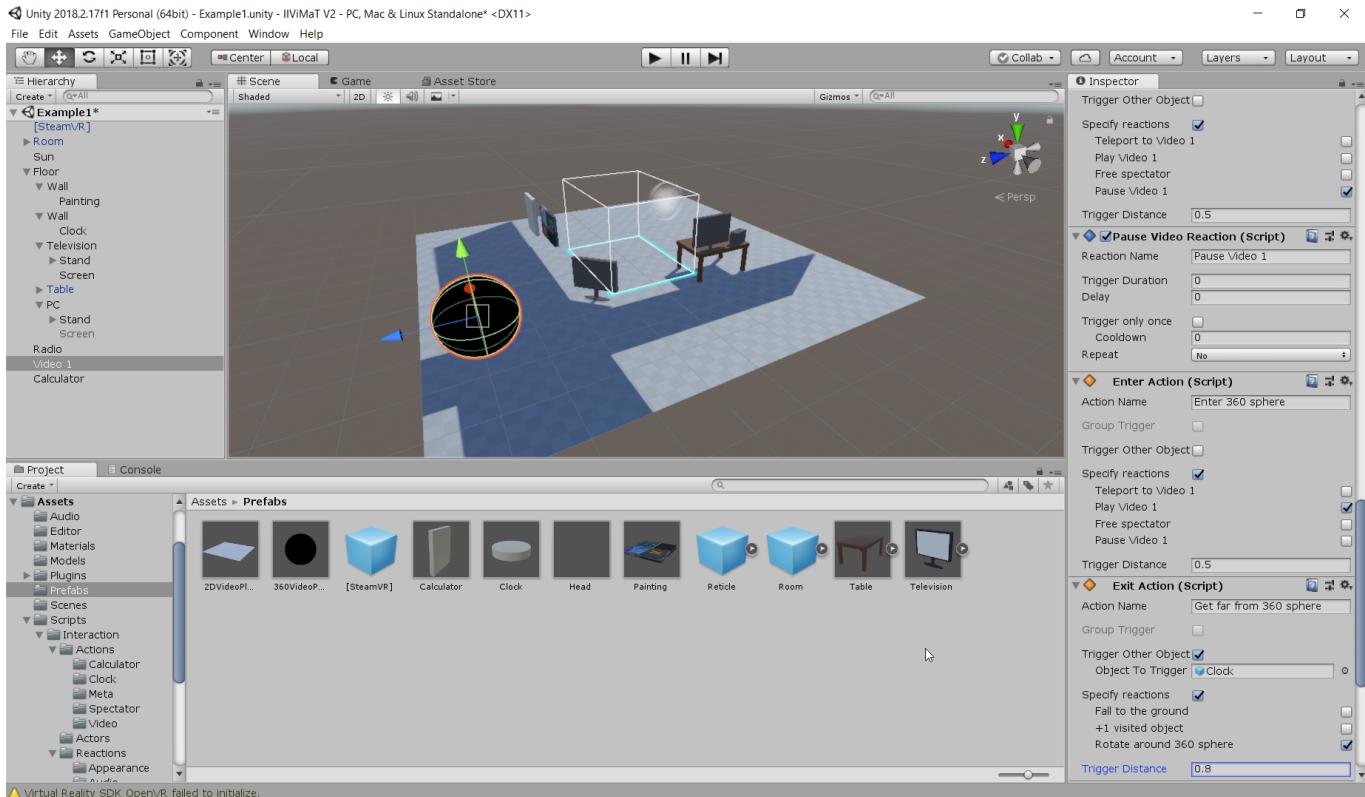


3.10 Rotation d'un objet autour d'un autre

On souhaite maintenant que lorsque le spectateur s'éloigne encore plus de la sphère, l'horloge fasse un tour complet autour de la sphère avant de retourner à sa place d'origine.

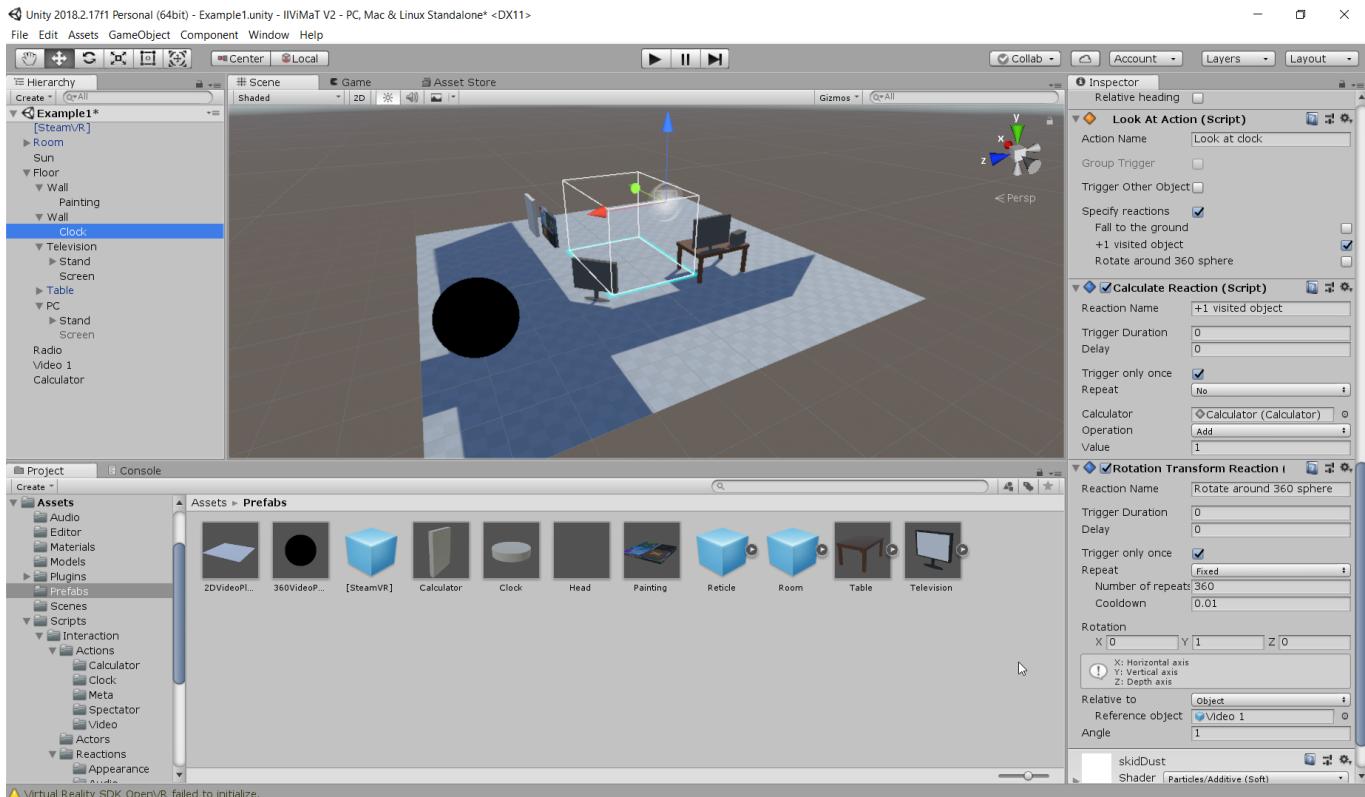
On ajoute alors une autre **ExitAction** sur la sphère, mais cette fois si avec une distance de 0.8 mètres.

Cette action déclenche une **RotationReaction** que l'on place alors sur l'horloge. Il faut donc spécifier qu'on déclenche une réaction spécifique sur un autre objet.



Sur l'horloge, on précise que la réaction doit provoquer une rotation autour de la sphère selon l'axe vertical (l'horloge tourne autour d'une poutre verticale imaginaire plantée dans la sphère 360).

On précise que c'est une rotation de 1° qui doit être répétée 360 fois avec 0.01 seconde entre chaque rotation.

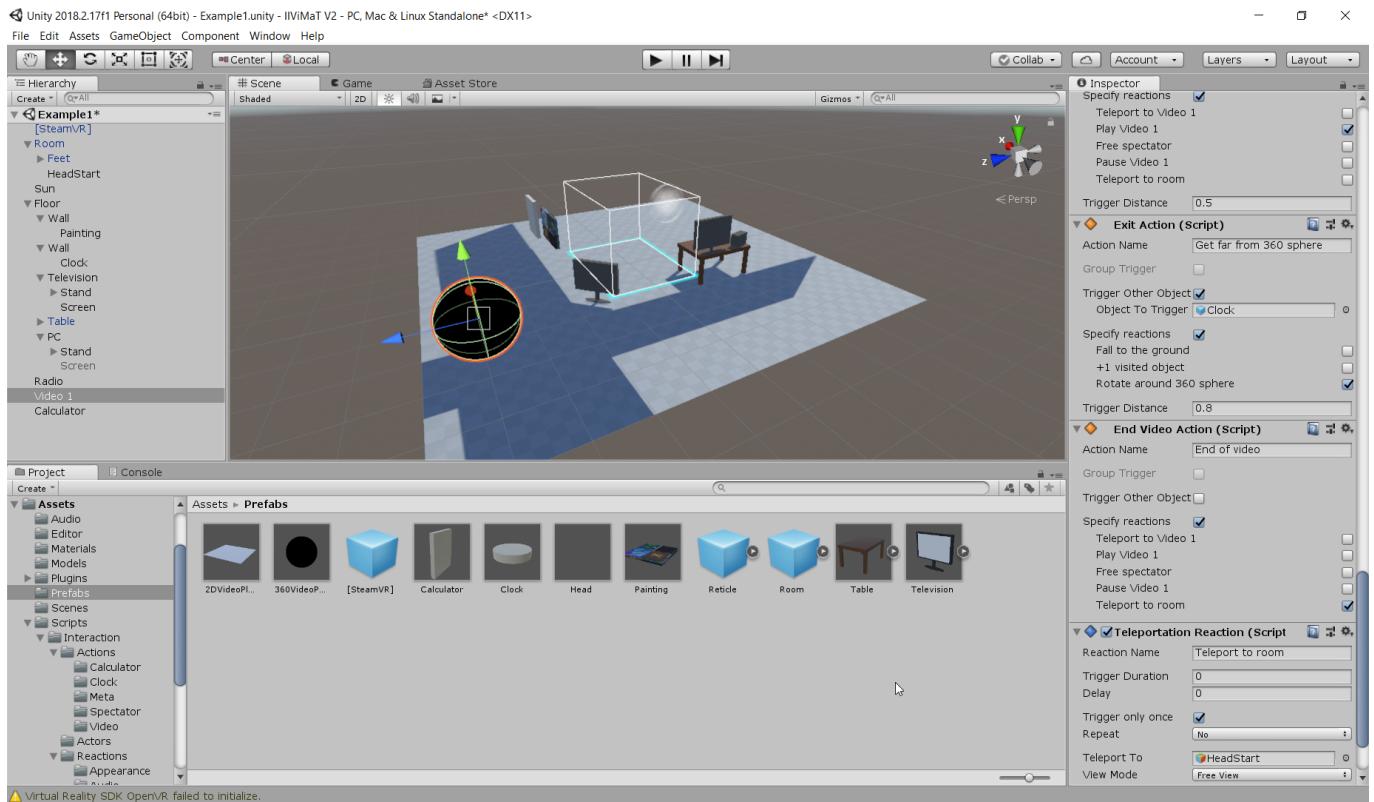


3.11 Téléportation dans un espace 3D

On souhaite maintenant que le spectateur puisse retourner dans la pièce. Il pourrait s'y déplacer physiquement en marchant, cependant il n'a pas la place physiquement de ce déplacer sur une telle distance.

On décide alors de téléporter le spectateur au point de départ lorsque la vidéo 360 se termine. On ajoute alors une **EndVideoAction** et une **TeleportationReaction** sur la sphère 360.

On ajoute un objet invisible que l'on positionne là où on souhaite que la tête du spectateur soit téléportée et on précise alors dans la réaction de téléportation de se téléporter sur cet objet en vue libre.



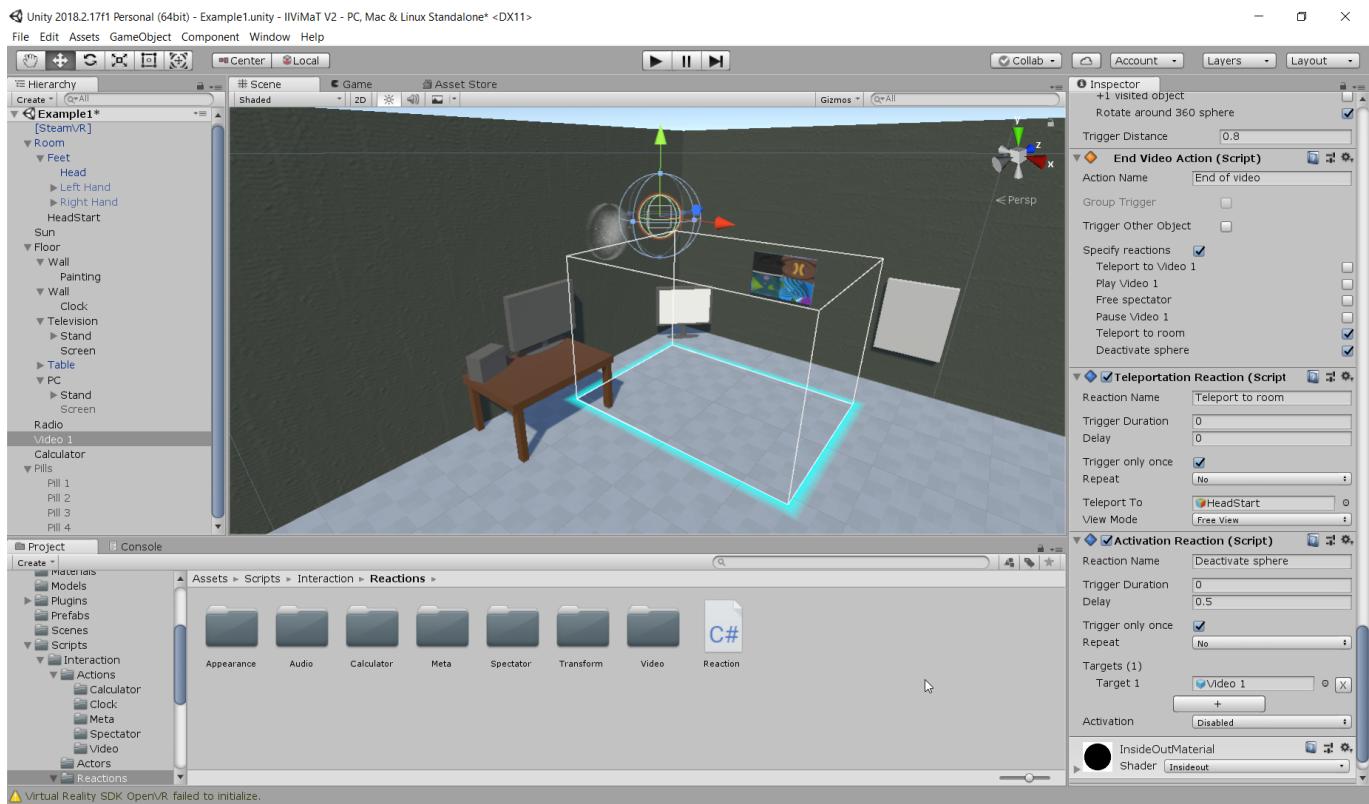
3.12 Désactivation d'objet

Enfin, on souhaite que le spectateur ne puisse pas retourner dans la sphère 360 ou lancer la vidéo en interagissant avec l'écran d'ordinateur. Pour cela on doit désactiver la sphère une fois que le spectateur est retourné dans la pièce.

Pour cela on ajoute une **ActivationReaction** avec la sphère 360 comme cible et le paramètre **Activation** sur **Disabled**.

Enfin il faut faire très attention, il faut absolument ajouter un petit délai sur le déclenchement de cette réaction. En effet si on ne le fait pas, on cours le risque que la sphère se désactive avant d'être téléporté.

Cela est dû au fait que l'action déclenche les réactions une par une dans un ordre aléatoire. Un délai de 0.5 secondes suffit ici.

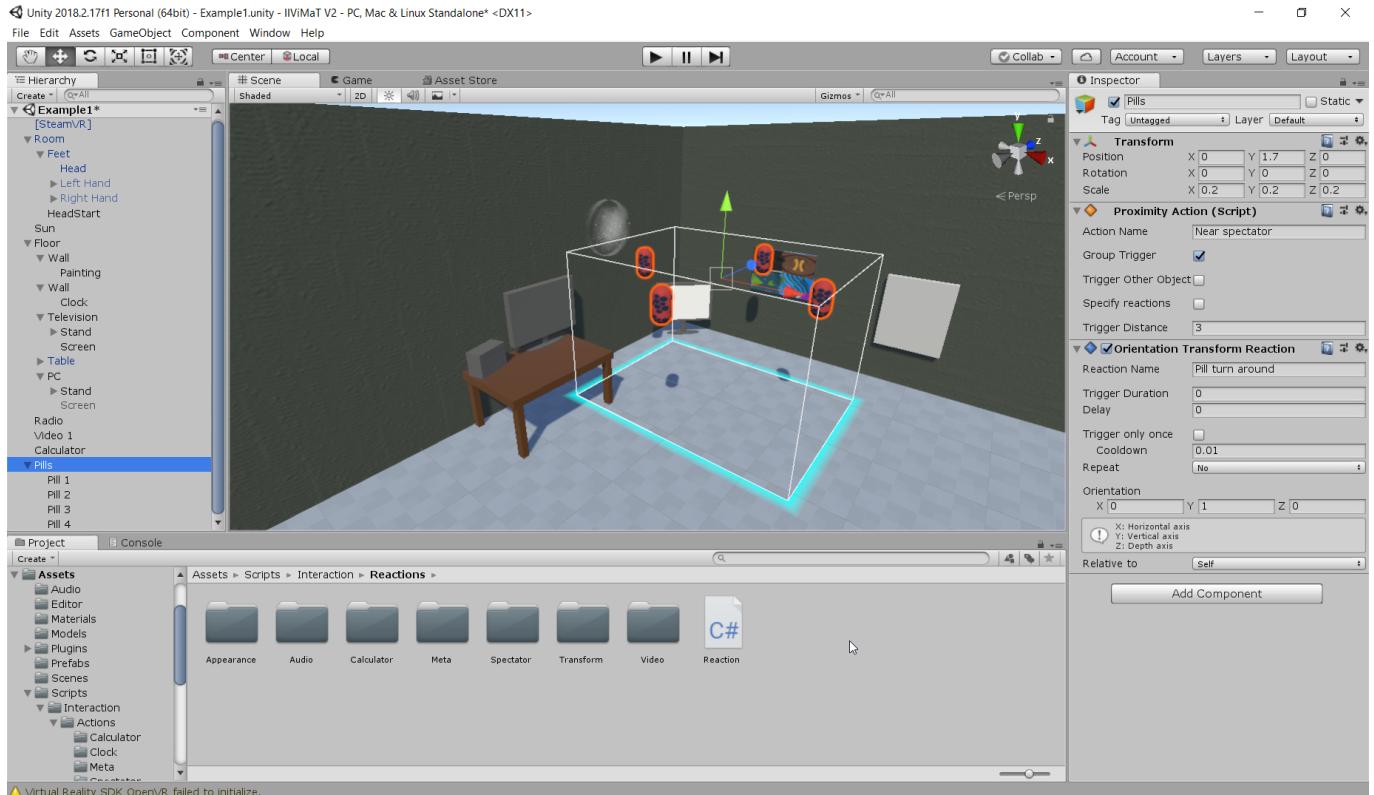


3.13 Réactions de groupe

On s'intéresse maintenant au groupe de 4 pilules. Par groupe, on veut dire que ces 4 objets (Pill 1, 2, 3 et 4) sont placés en tant qu'enfants de l'objet parent (Pills) dans la hiérarchie. Dans Unity, cela signifie qu'une modification du parent implique une modification correspondante de tous les enfants.

Par exemple si le parent tourne, les 4 pilules tournent autour du parent afin de rester au même endroit relativement à leur parent. On souhaite justement utiliser cela pour faire tourner ces 4 pilules autour de leur parent.

Il suffit alors d'ajouter une **OrientationReaction** sur le parent. On décide que les pilules tournent tant que le spectateur est à portée, on ajoute donc une **ProximityAction**.



On souhaite donner à chacune des pilules un effet propre à elle, on place donc sur chacune d'elle la réaction souhaitée.

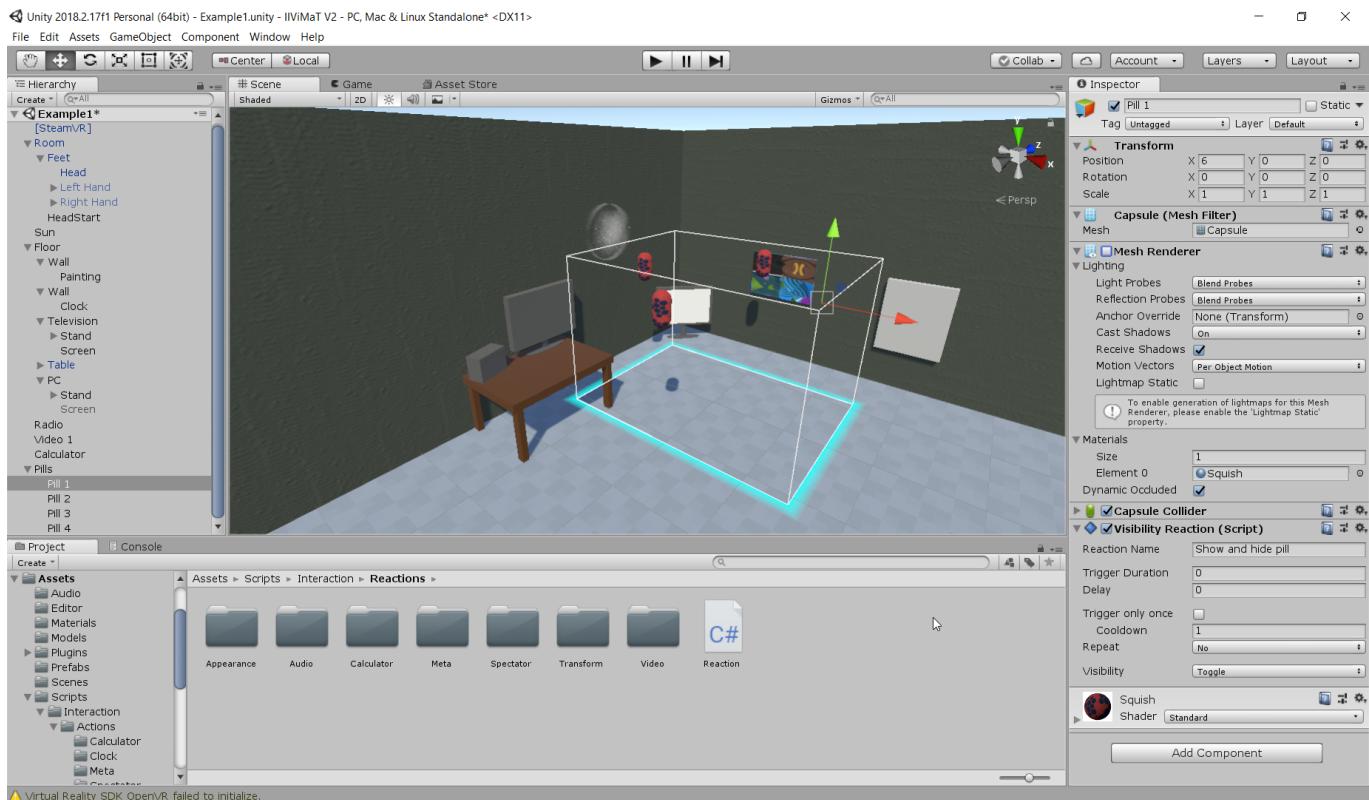
On souhaite également que ces effets se déclenchent en même temps que leur rotation autour de leur parent. On pourrait ajouter une **ProximityAction** sur toutes les pilules, mais il y a une solution bien plus simple. Il suffit de cocher la case **Group Trigger** de la **ProximityAction** sur Pills.

Cela a pour effet que cette action déclenche non seulement les réactions sur Pills, mais également les réactions sur les enfants de Pills.

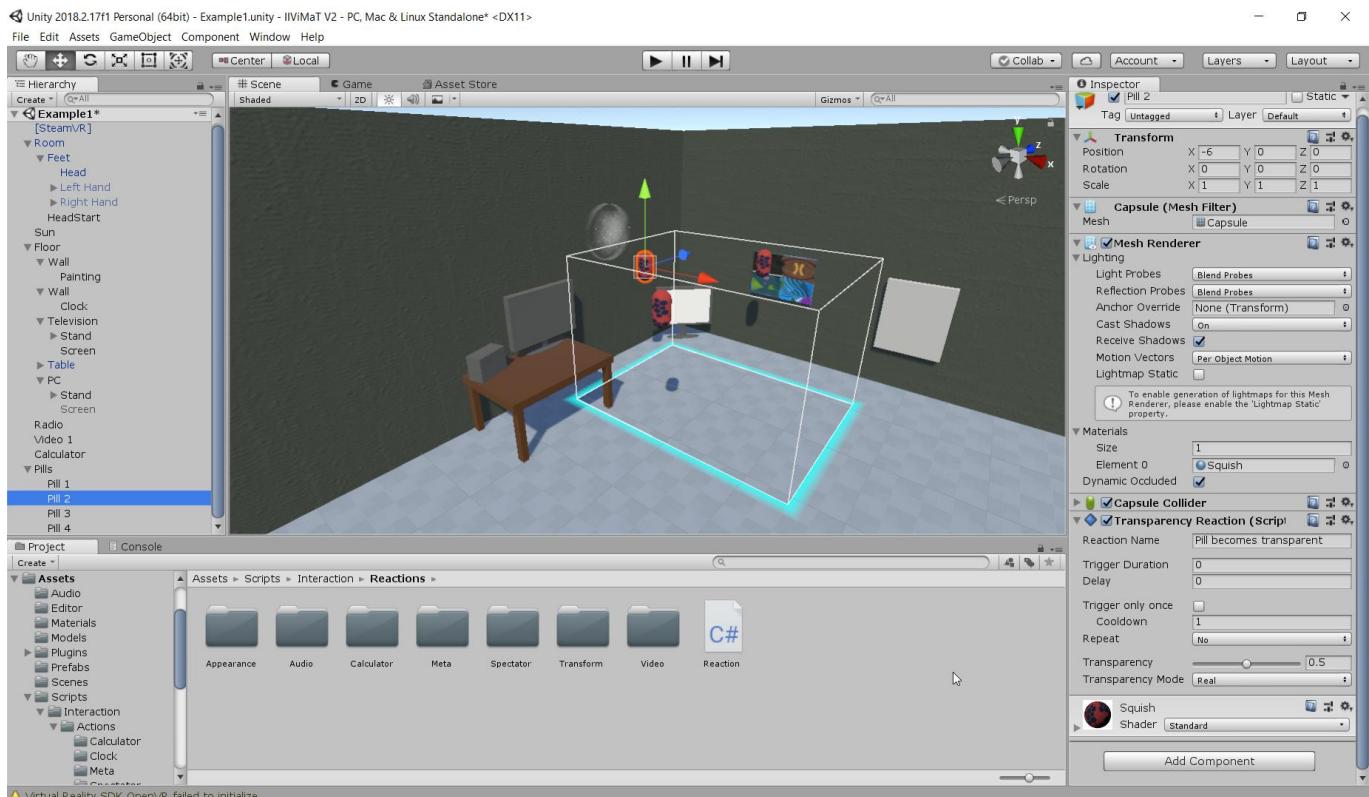
On ajoute donc les réactions sur les pilules. Pour la première, on souhaite qu'elle apparaisse et disparaisse toutes les secondes.

Pour cela on lui ajoute une **VisibilityReaction** en précisant le paramètre **Cooldown** à 1 seconde et le paramètre **Visibility** à **Toggle** pour alterner entre visible et invisible.

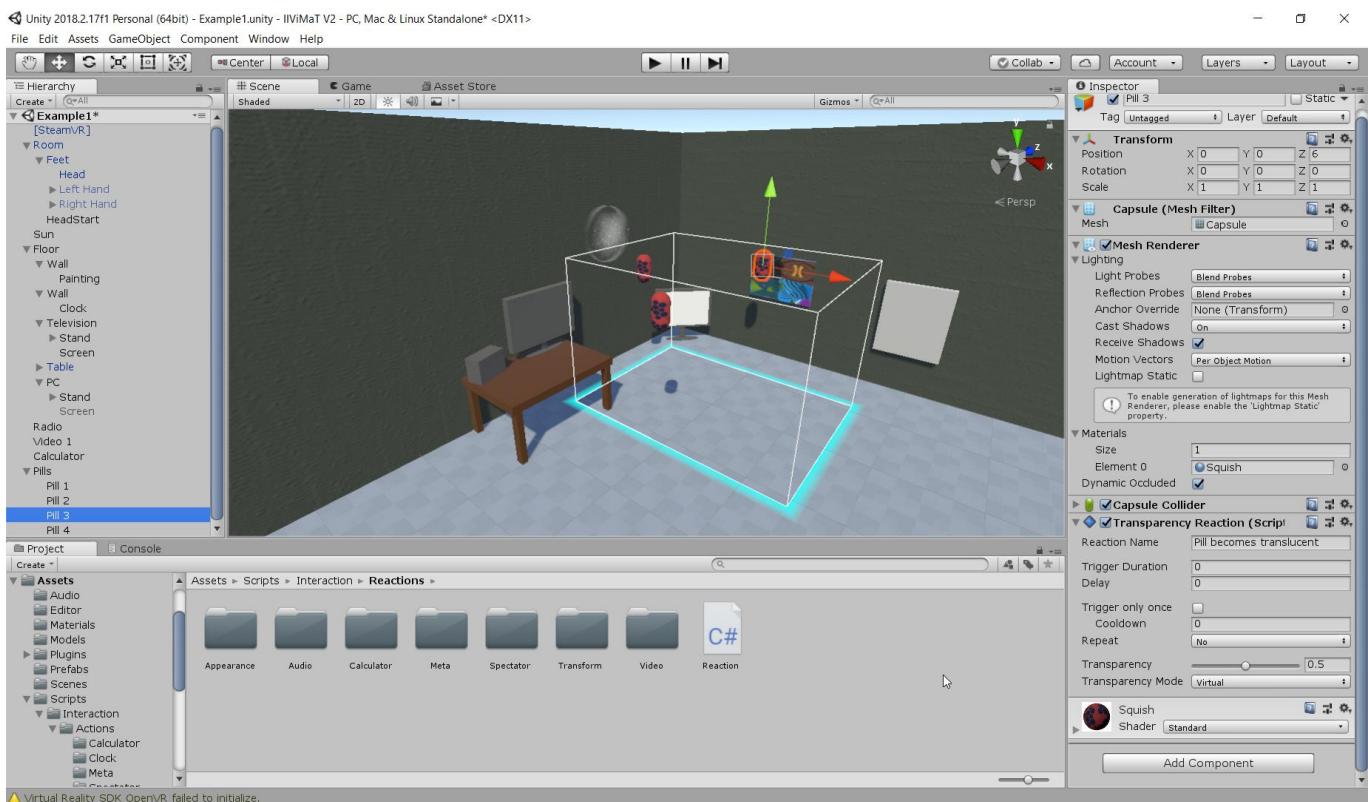
On souhaite également que la pilule soit invisible au lancement, et ne devienne visible qu'au moment de la première réaction. Pour cela il faut décocher la case à coté du composant **Mesh Renderer** sur la pilule.



Pour la deuxième on souhaite qu'elle apparaisse semi-transparente de manière réaliste comme du verre. On place alors une **TransparencyReaction** en précisant le paramètre **Transparency Mode** à **Real**.

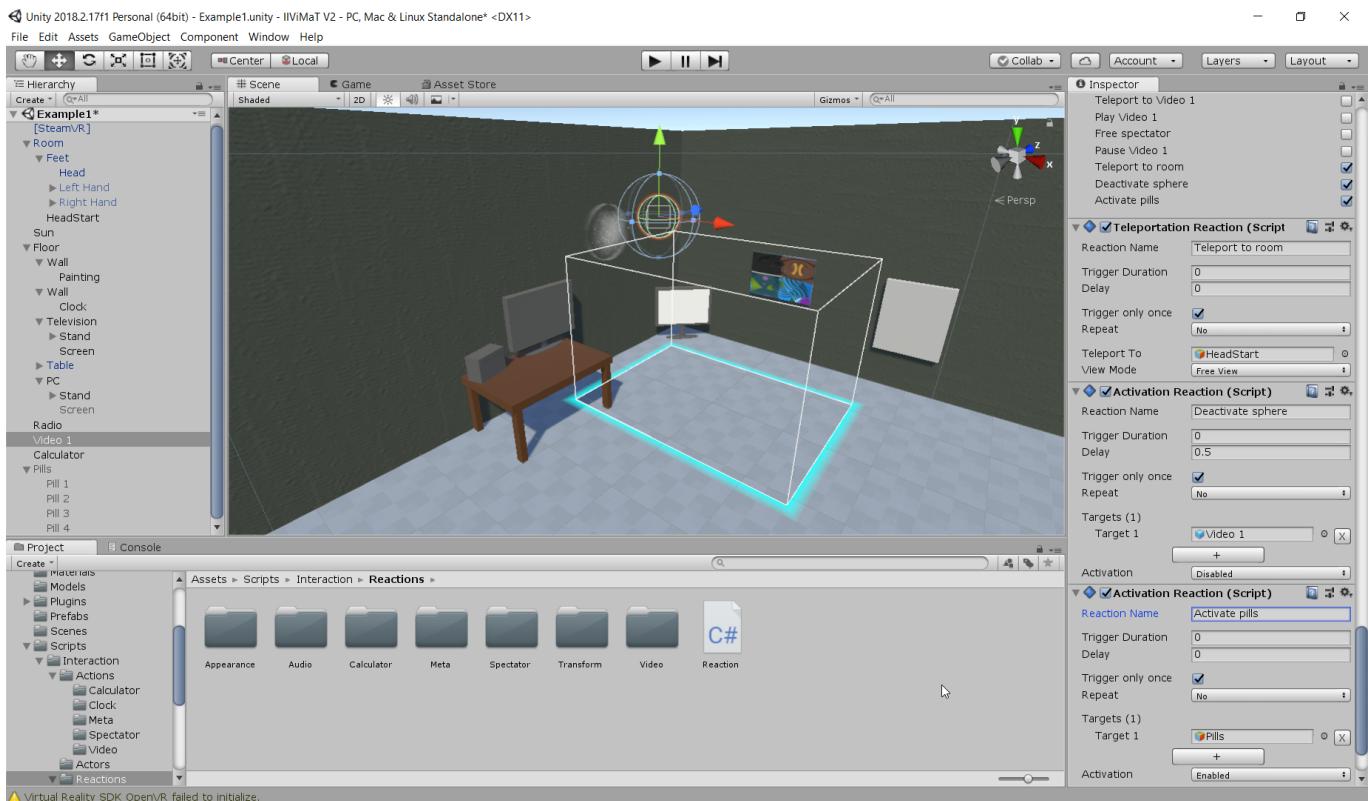


Pour la troisième on souhaite qu'elle apparaisse semi-invisible comme un fantôme. On place alors une **TransparencyReaction** en précisant le paramètre **Transparency Mode** à **Virtual**.



Enfin la dernière ne fera rien.

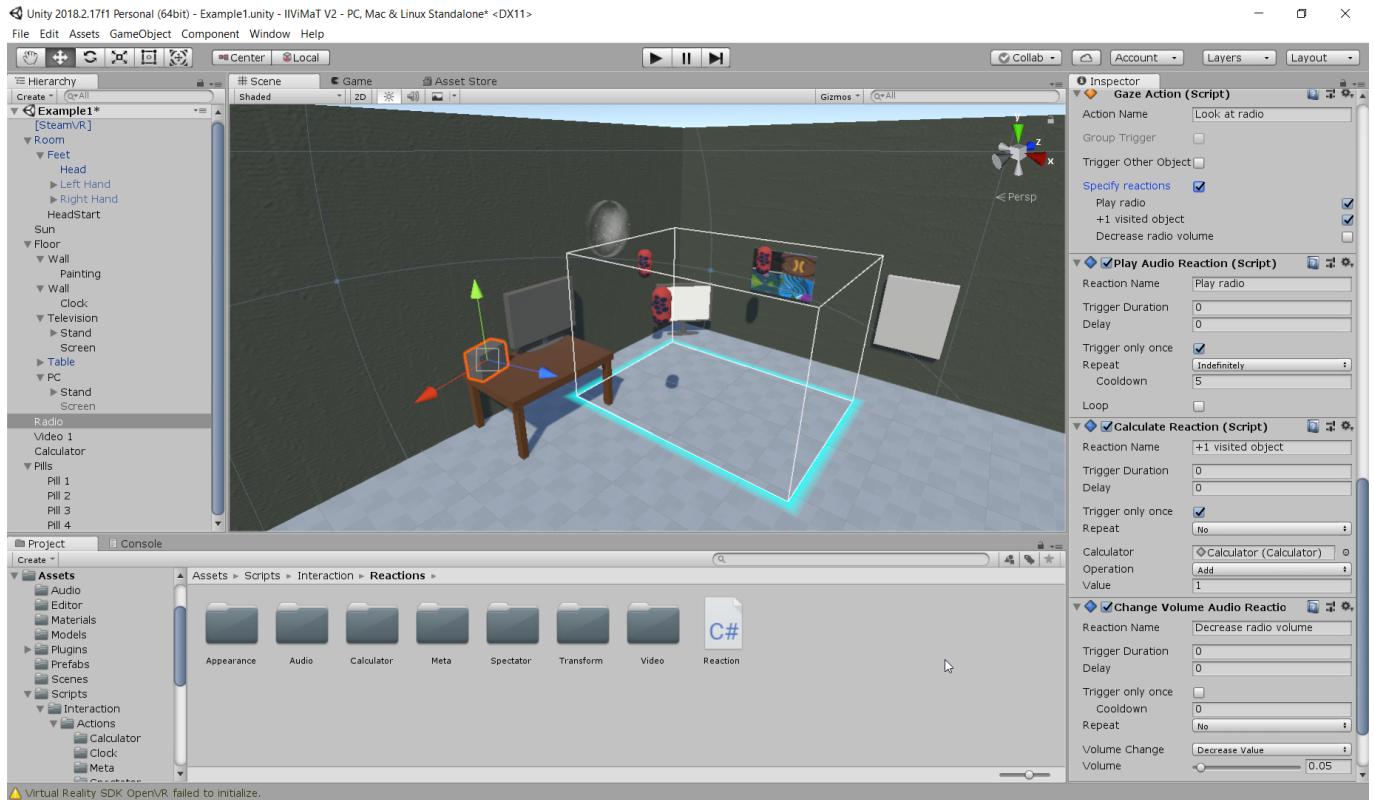
On souhaite que ces pilules soient désactivées jusqu'à ce que le spectateur revienne dans la pièce après avoir vu la vidéo 360. Pour cela on ajoute une **ActivationReaction** sur la vidéo 360 pour activer Pill (et donc ses enfants en même temps). On oublie pas de cocher la réaction dans la liste sur **EndVideoAction** ni de décocher la case sur l'objet Pills pour qu'il soit désactivé par défaut.



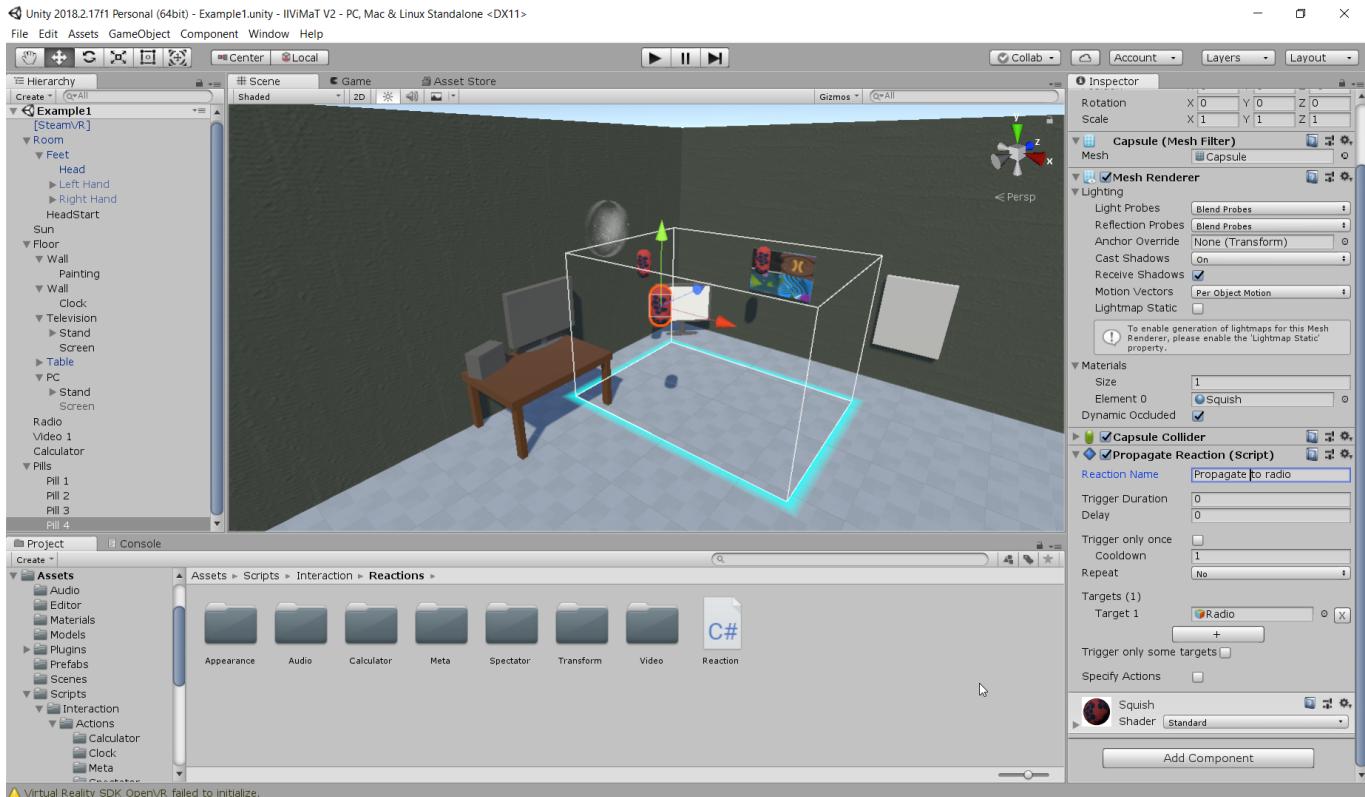
3.14 Modifier le volume d'une source audio par propagation

On souhaiterait que le volume de la radio diminue en même temps que les pilules réagissent.

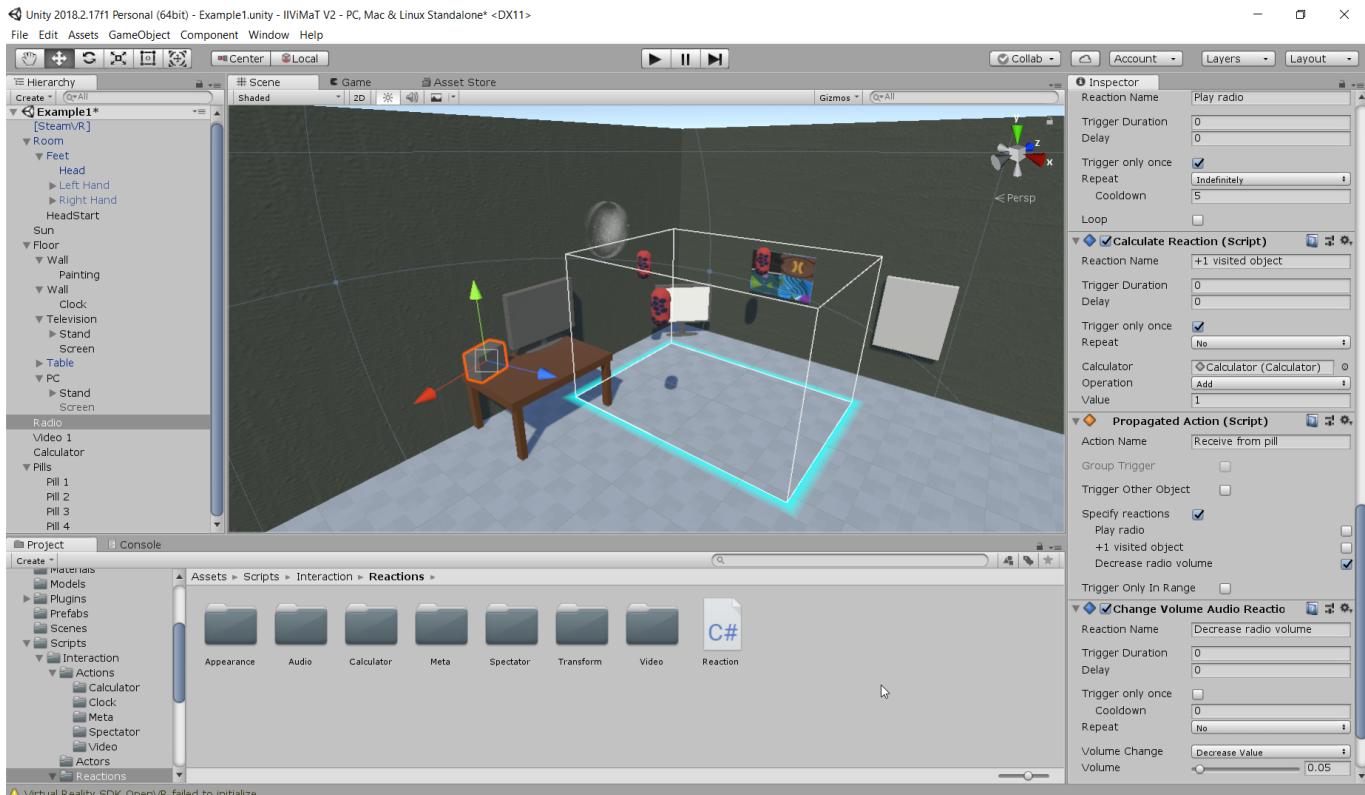
On ajoute alors sur la radio une **ChangeVolumeAudioReaction** en précisant qu'on souhaite diminuer le volume de 0.05 à chaque réaction. Sans oublier de cocher **Specify Reactions** pour la **GazeAction** sur la radio pour qu'elle ne déclenche pas cette nouvelle réaction.



Pour déclencher cette réaction de volume, on ne peut pas utiliser la **ProximityAction** de Pills, car on ne peut pas à la fois déclencher des réactions sur l'objet et un autre. On pourrait par contre placer une deuxième **ProximityAction** sur Pills qui elle déclencherait la réaction de volume. Il existe une autre solution, que nous utiliseront ici pour l'illustrer. Il s'agit d'utiliser le fait que notre 4ème pilule ne fait rien. On ajouterait alors une réaction à notre pilule, dont le seul but est de relayer le déclenchement à un autre objet. Pour cela on utilise une **PropagateReaction**. On précise que notre cible est la radio.



La **PropagateReaction** transmet le signal à la radio, il faut maintenant ajouter une **PropagatedAction** à la radio qui reçoit ce signal pour l'utiliser. On spécifie que cette action déclenche la réaction de volume.



4 Conseils

Tous les temps et durées sont exprimés en secondes, toutes les distances sont exprimées en mètres et tous les angles sont exprimés en degrés.

Un objet désactivé ne peut pas interagir, attention donc à bien activer l'objet avant d'essayer d'interagir avec. Utilisez donc un délai si nécessaire pour vous assurer de l'ordre de déclenchement.

Les réactions appliqués directement aux parents provoquent parfois des effets non désirés chez les enfants. Pour palier à cela, placez vos réactions sur les enfants, placez l'action sur le parent, et sélectionnez l'option **Group Trigger** pour que le parent déclenche les réactions des enfants en plus des siennes.