



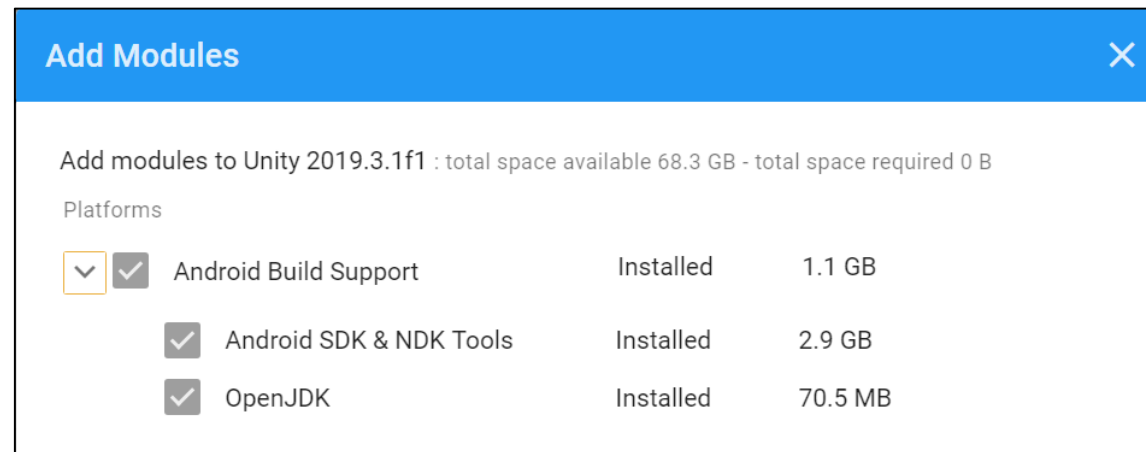
## Oculus Quest Prise en main



**Pr. Fakhreddine Ababsa**  
**Ecole Nationale Supérieure d'Arts et Métiers**

# Oculus Quest sous Unity - Configuration

- Modules supplémentaires à ajouter à partir de Unity Hub
- Android Build Support,
- Android SDK & NDK tools, OpenJDK



# Oculus Quest sous Unity - Configuration

## Outils pour utiliser/visualiser le contenu du casque Oculus Quest sur PC

- Télécharger et installer Oculus Quest usb drivers (Oculus Quest adb drivers):  
<https://developer.oculus.com/downloads/package/oculus-adb-drivers/>
- Télécharger et installer l'outil SideQuest (Advanced installer) : <https://sidequestvr.com/download>

### Test de connexion :

- Lancer l'application SideQuest → Connecter le casque → vérifier que le casque est bien reconnu par l'application
- Tester la fonction Stream

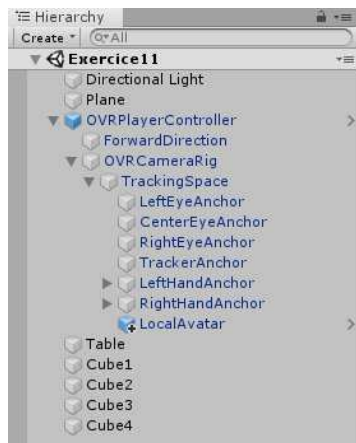
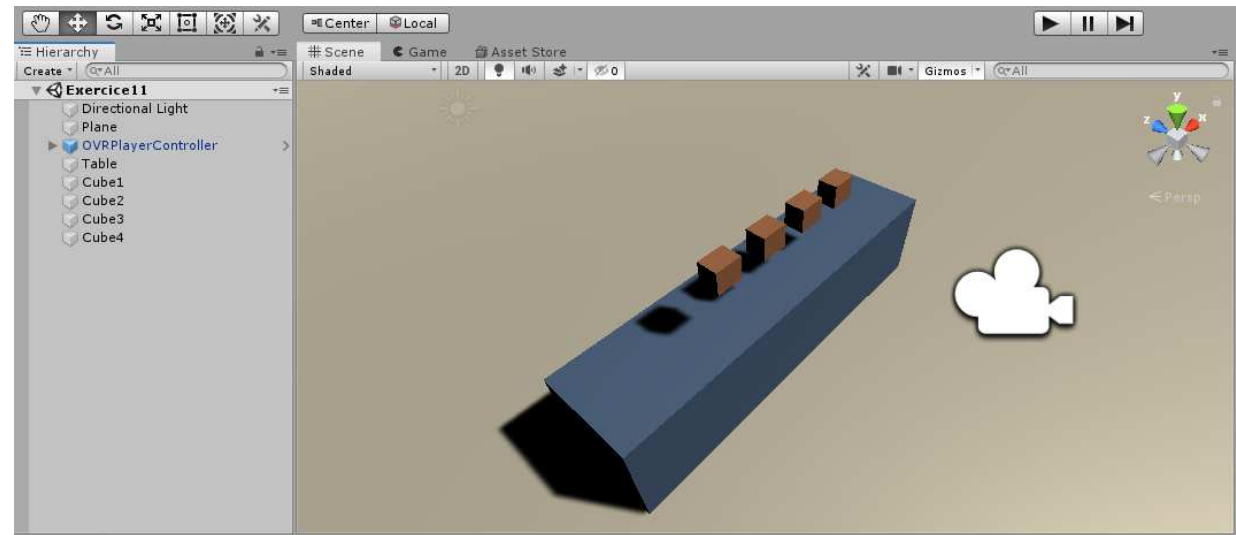


# Oculus Quest sous Unity – Prise en main

- Lancer le projet « **HelloOculus** » et ouvrir la scène **Room** qui se trouve dans Oculus → SampleFramework → Usage → AvatarGrab
- Brancher l'Oculus Quest au PC. Pré-requis :
  - ✓ Compte Oculus
  - ✓ Adaptateur usb-c
  - ✓ Mode développeur activé sur le Quest
- Autoriser la connexion depuis le home virtuel
- Générer le fichier . (apk) de votre projet
  - ✓ File → Build Settings → Build
- A partir de Sidequest, téléverser sur le casque le fichier (.apk) ainsi généré.

# Exercice 1 : Saisir un objet

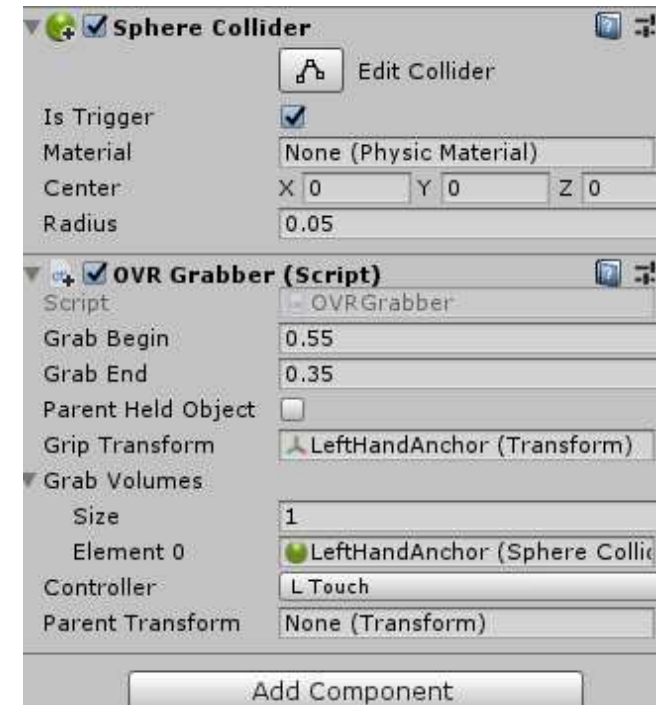
- Créer une nouvelle scène contenant un sol, une table et 4 cubes
- Supprimer la « Main Camera » → ajouter le Prefab « OVRPlayerController » positionner le devant le GameObject « Table »



- Character Controller → Radius = 0.1
- Ajouter un Prefab « LocalAvatar » dans le OVRPlayerController → OVRCameraRig → TrackingSpace
- Build and run (**attention** : ajouter la nouvelle scène dans scenes In Build)

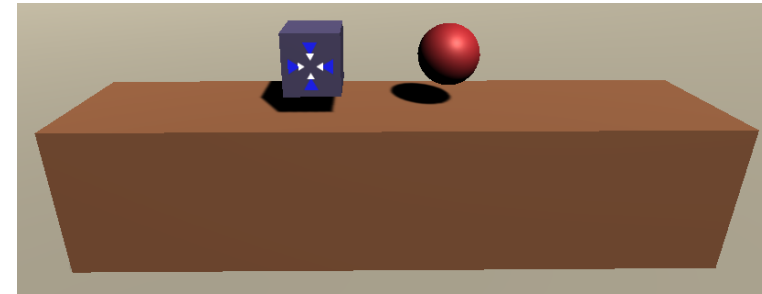
# Exercice 1 : Saisir un objet

- Dans « LeftHandAnchor » et « RightHandAnchor » réaliser les actions suivantes :
  - ✓ Ajouter le composant Sphere Collider : « Is Trigger » activé, radius = 0.05
  - ✓ Ajouter le Script « OVR Grabber »
    - Glisser/Déposer « LeftHandAnchor » et « RightHandAnchor » sur le champs « Grip Transform »
    - Grab Volumes → Size = 1 : Glisser/Déposer « LeftHandAnchor » et « RightHandAnchor » sur le champs « Element 0 »
    - Controller : « L Touch » / « R Touch »
  - ✓ Désactiver « Use Gravity » du Composant « Rigid Body »
- Sélectionner les 4 cubes et ajouter :
  - ✓ Le script « OVR Grabbable »
  - ✓ Un « Rigid Body » avec l'option « Use Gravity » activée
- Build and run



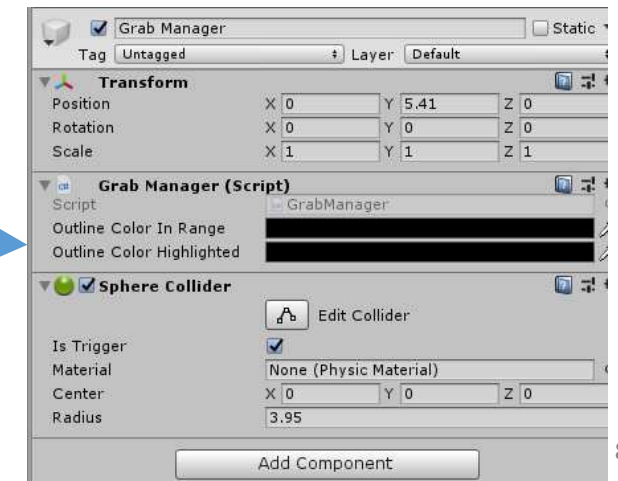
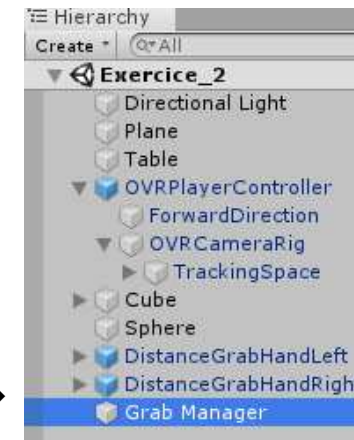
# Exercice 2 : Saisir un objet distant

- Charger et tester la scène :  
Oculus/SampleFramework/Usage/DistanceGrab
- Créer une nouvelle scène contenant un plan, une table, un cube et une sphère
- Supprimer la « Main Camera » → ajouter le Prefab  
« OVRPlayerController » positionner le devant le GameObject  
« Table »
- Sélectionner les GameObjects « Cube » et « Sphere » et réaliser les actions suivantes:
  - ✓ Rigidbody → Collision Detection : Continuous Dynamic
  - ✓ Ajouter le script « Distance Grabbable »
- Ajouter dans la scène les deux prefabs « DistanceGrabHandLeft » et « DistanceGrabHandRight »
  - ✓ Glisser/Déposer « OVRPlayerController » sur le champs « Player » et « Tracking Space » sur le champs « Parent Transform »



# Exercice 2 : Saisir un objet distant

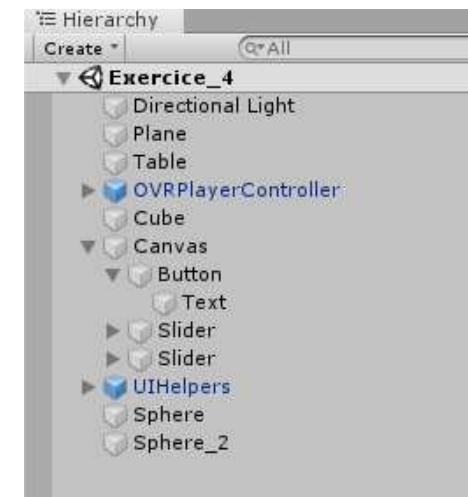
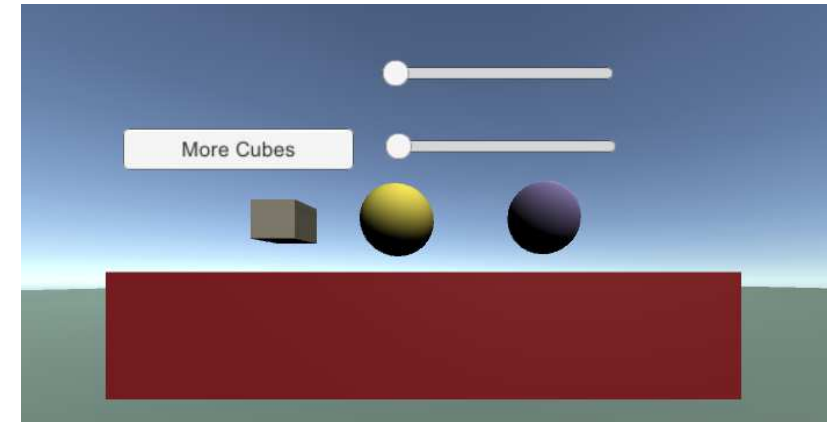
- Définir un « Grabbable Layer » rassemblant les objets que vous souhaitez attraper à distance :
  1. Sélectionner l'objet « Cube » → Inspector → Layer → Add Layer → User Layer 8 : Grabbable
  2. Sélectionner les objets « Cube » et « Sphere » → Layer → Grabbable
- Dans « DistanceGrabHandLeft » et « DistanceGrabHandRight » → « Grab Object in Layers » = 8 et « Obstruction Layer » = -1
- Créer un GameObject vide, nommer le « Grab Manager » et ajouter
  - ✓ Le script « Grab Manager »
  - ✓ Un composant « Sphere Collider » : activer « Is Trigger », et choisir le paramètre « Radius » pour définir la taille de zone les objets
- Build and run





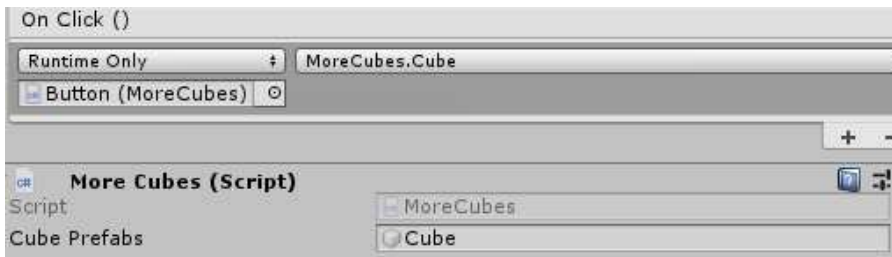
# Exercice 3 : Interface Utilisateur (UI)

- Créer une nouvelle scène contenant un plan, un cube et deux sphères
- Supprimer la « Main Camera » → ajouter le Prefab « OVRPlayerController » positionner le devant le GameObject « Cube »
- Character Controller → Radius = 0.1
- Ajouter un Prefab « LocalAvatar » dans OVRPlayerController → OVRCameraRig → TrackingSpace
- Ajouter le GameObject UI → Canvas ayant les propriétés suivantes:
  - *Render mode = Word Space*
  - *Event Camera = CenterEyeAnchor*
  - *Scale = 0.03, Height=193, Width=700*
- Dans ce canvas effectuer les actions suivantes:
  - ✓ Ajouter un « Button » et deux « Slider »
  - ✓ Supprimer le script « Graphic Raycaster » et le remplacer par le script « OVR Raycaster »
- Chercher le Prefab « UIHelpers » et ajouter le dans votre scène, supprimer l'objet « EventSystem » associé avec le Canvas
- Drag/Drop UIHelpers → LaserPointer sur Canvas → OVR Raycaster → Pointer
- EventSystem → Joy Pad Click Button → Secondary Index trigger (main droite)

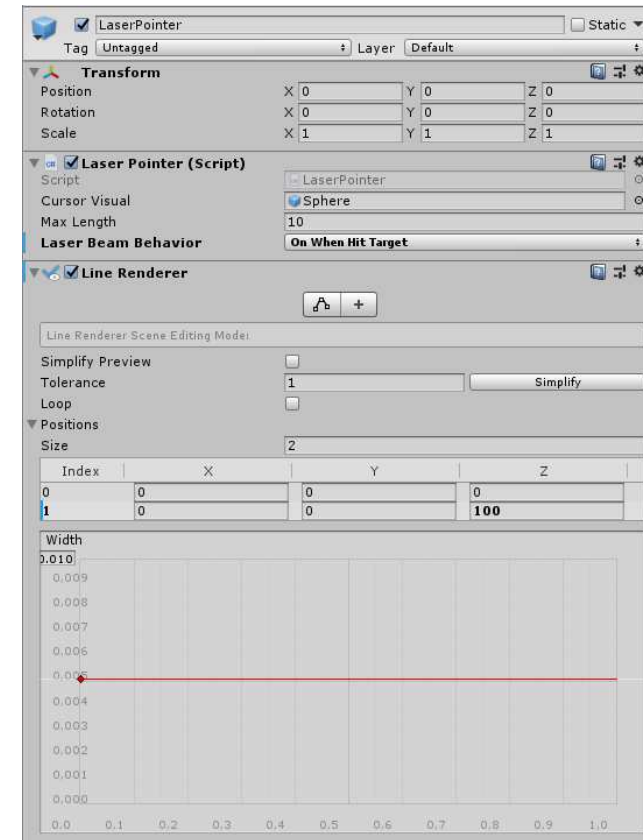


# Exercice 3 : Interface Utilisateur (UI)

- Dans le GameObject UIHelpers → LaserPointer, faire:
  - ✓ Activer le composant « Line Renderer »
  - ✓ Editer le script « LaserPointer » et modifier la ligne 30 en changeant le type de la variable LaserBeamBehavior de *private* à *public*.
    - Dans l'Inspector, associer à « Laser Beam Behavior » l'évènement « On »
- Dans le GameObject « Button », ajouter le script « MoreCubes »
- Glisser/Déposer le GameObject « Cube » sur la variable « Cube Prefab »
- Ajouter un évènement On Click()
- Glisser/Déposer Button sur le champs « None Object »
- Sélectionner la fonction « MoreCubes »
- Jouer la scène

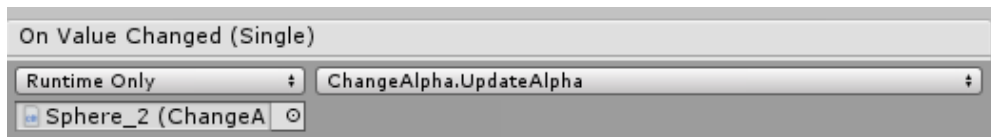


```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 [O références]
6 public class MoreCubes : MonoBehaviour
7 {
8     public GameObject CubePrefabs;
9
10    [O références]
11    public void Cube()
12    {
13        Instantiate(CubePrefabs);
14    }
15 }
```



# Exercice 3 : Interface Utilisateur (UI)

- Dans le GameObject « Sphere1 », ajouter le script « Change Alpha »
- Glisser/Déposer le GameObject « Slider » sur « Main Slider » et « Sphere » sur Sphere1
- Valeur du Slider : Min Value = 0, Max Value = 1
- Ajouter un évènement « On Value Changed (Single) »
- Glisser/Déposer la sphère sur le champs « None Object »
- Sélectionner la fonction ChangeAlpha()→UpdateAlpha()
- Jouer la scène

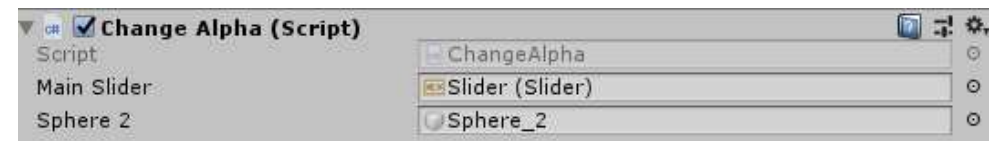


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

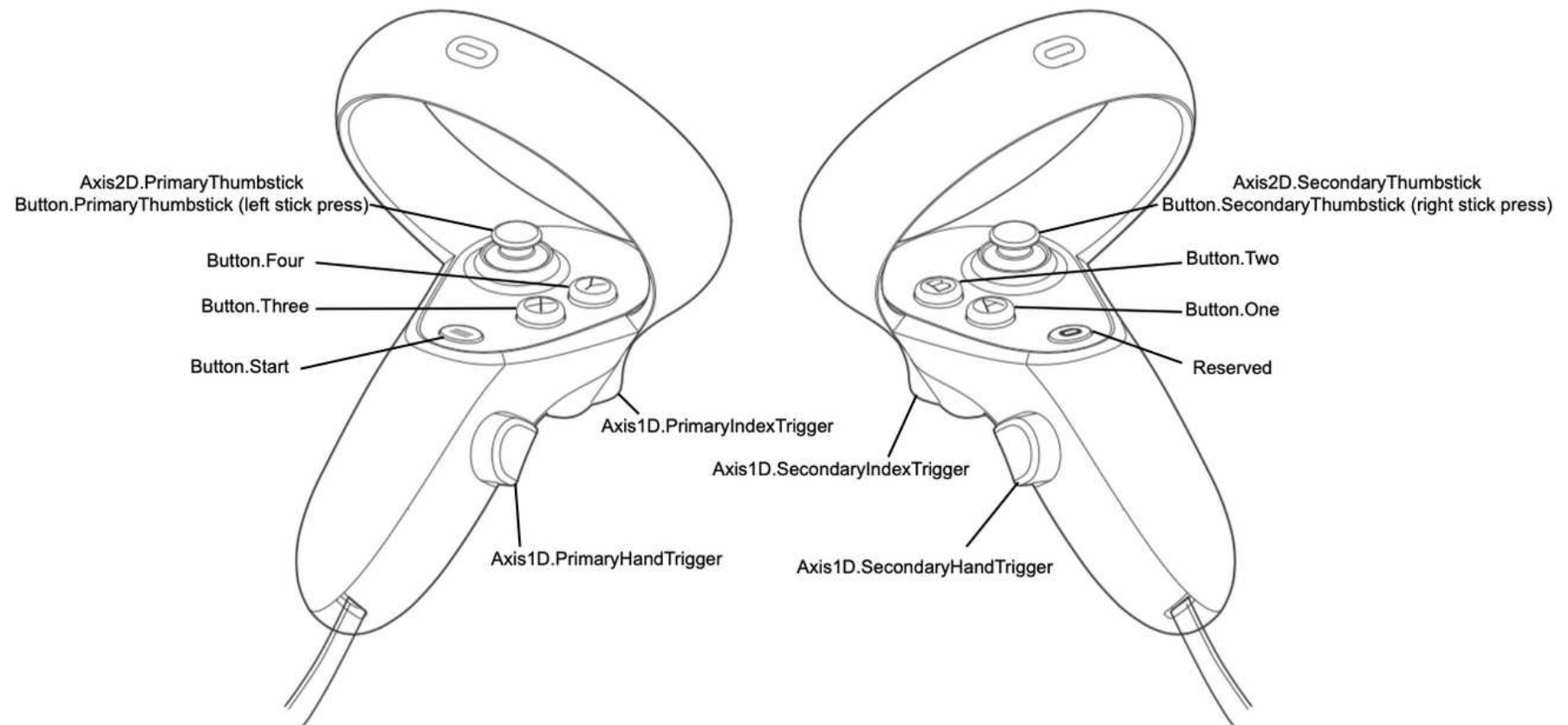
public class ChangeAlpha : MonoBehaviour
{
    public Slider mainSlider;
    public GameObject currentGameObject;
    private Material currentMat;
    private float alpha;

    // Start is called before the first frame update
    void Start()
    {
        currentGameObject = gameObject;
        currentMat = currentGameObject.GetComponent<Renderer>().material;
        mainSlider.value = currentMat.color.a;
    }

    public void UpdateAlpha()
    {
        Color oldColor = currentMat.color;
        alpha = mainSlider.value;
        Color newColor = new Color(oldColor.r, oldColor.g, oldColor.b, alpha);
        currentMat.SetColor("_Color", newColor);
    }
}
```

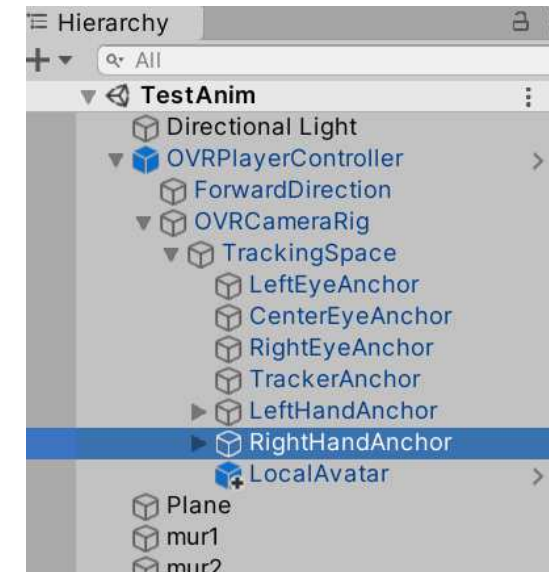
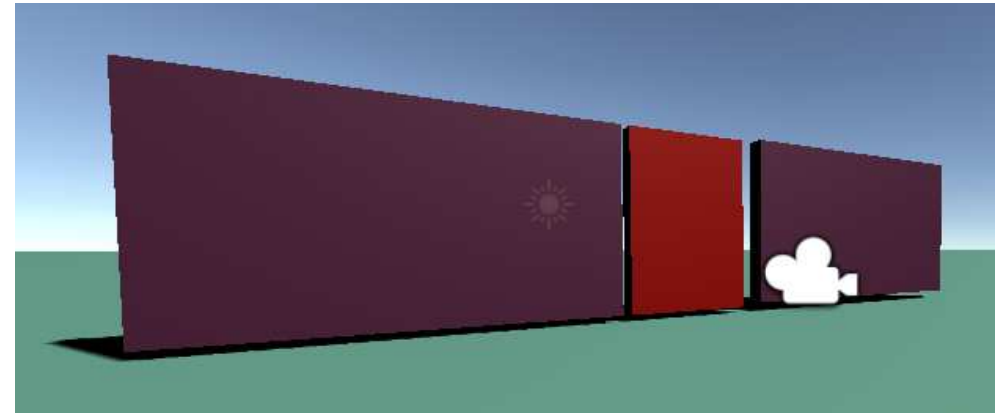


# Exercice 3 : Interface Utilisateur (UI)




# Exercice 4 : RayCast + Animation

- Créer une nouvelle scène contenant un plan, deux murs et une porte
- Supprimer la « Main Camera » → ajouter le Prefab « OVRPlayerController » positionner le devant le GameObject « Porte »
- Character Controller → Radius = 0.1
- Ajouter un Prefab « LocalAvatar » dans OVRPlayerController → OVRCameraRig → TrackingSpace



# Exercice 4 : RayCast + Animation

## Animation du GameObject « Porte »

- GameObject « Porte » → window → Animation → Animation
- Create new animation → DoorOpen
- Add Property → Transform → Position and Rotation
- Pour enregistrer l'animation, appuyer sur , puis définir les transformations de début et fin de l'animation

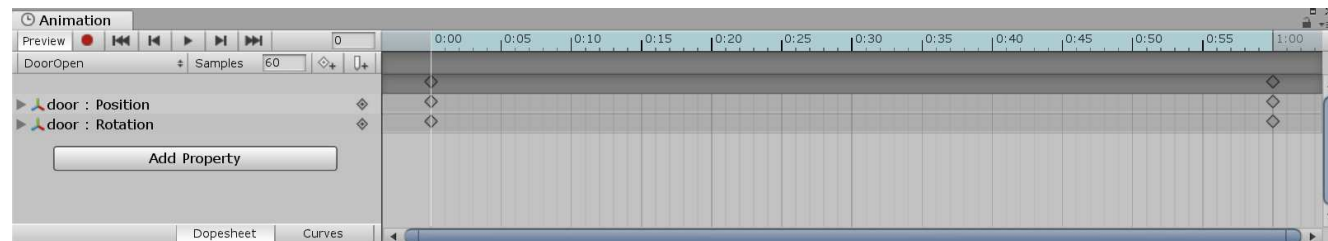
→  $t = 0$

Position	X	0.07	Y	-0.03	Z	-5.39
Rotation	X	0	Y	0	Z	0

$t = 1$

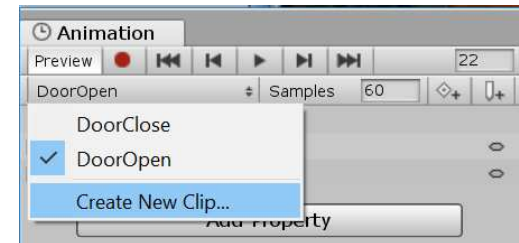
Position	X	1.42	Y	-0.03	Z	-4.39
Rotation	X	0	Y	90	Z	0

- Appuyer sur  pour jouer et vérifier l'animation



# Exercice 4 : RayCast + Animation

- Create New Clip → DoorClose
- Add Property → Transform → Position and Rotation
- Enregistrer l'animation → inverser les transformations précédentes



→ t=0

Position	X	1.42	Y	-0.03	Z	-4.39
Rotation	X	0	Y	90	Z	0

t=1

Position	X	0.07	Y	-0.03	Z	-5.39
Rotation	X	0	Y	0	Z	0

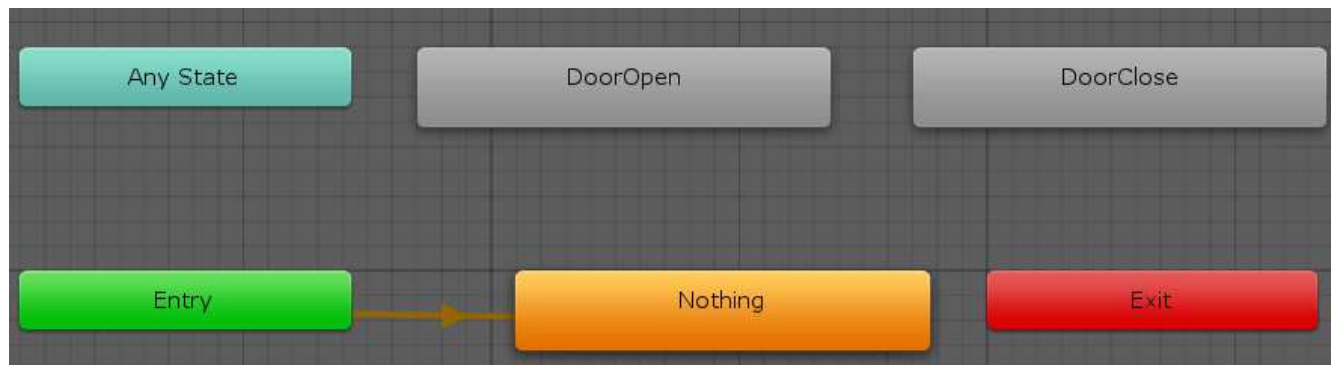
- Unity crée un Contrôleur d'animation nommé « porte.controller »



# Exercice 4 : RayCast + Animation

## Animation du GameObject « porte »

- GameObject « door » → Composant « Animator » → Cliquer deux fois sur Controller « Door »
- Bouton droit de la souris → Create New state « Nothing » → Set as Layer Default State : ceci évite le lancement des animations au démarrage du jeu.





# Exercice 4 : RayCast + Animation

- Dans OVRPlayerController → OVRCameraRig → TrackingSpace → RightHandAnchor, ajouter les composants suivants:

1. Line Renderer ayant une largeur de 0.01
2. Un script que vous nommerez **RayCastLine1** dont le rôle est de générer un RayCast afin d'interagir avec la porte grâce à la touche « A » de la manette droite. Si le Raycast entre en collision avec la porte alors un clique sur A permettra de l'ouvrir. Un 2<sup>ème</sup> clique sur A fermera la porte. Ci-dessous le contenu de ce script.

## RayCastLine1.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class RayCastLine1 : MonoBehaviour
6  {
7
8      public LineRenderer drawLine;
9
10     public float lineWidth = 0.1f;
11     public float lineMaxLength = 100f;
12
13     private bool etatF = true;
14     private bool etatO = false;
15
16     private float HandRight = OVRInput.Get(OVRInput.Axis1D.SecondaryHandTrigger);
17
18     public bool enemyHit = false;
19
20     private GameObject enemy;
```

# Exercise 4 : RayCast + Animation

RayCastLine1.cs

```
22 void Start()
23 {
24     Vector3[] startLinePositions = new Vector3[]{Vector3.zero, Vector3.zero};
25     drawLine.SetPositions(startLinePositions);
26 }
27
28 // Update is called once per frame
29 void Update()
30 {
31     drawLine.enabled = true;
32
33     RaycastHit hit;
34     Ray telekinesisOut = new Ray(transform.position, transform.forward);
35     Vector3 endPosition = transform.position + (lineMaxLength * transform.forward);
36
37     if (Physics.Raycast(telekinesisOut, out hit))
38     {
39         endPosition = hit.point;
40
41         var selection = hit.transform;
42         var selectionRenderer = selection.GetComponent<Renderer>();
```

# Exercice 4 : RayCast + Animation

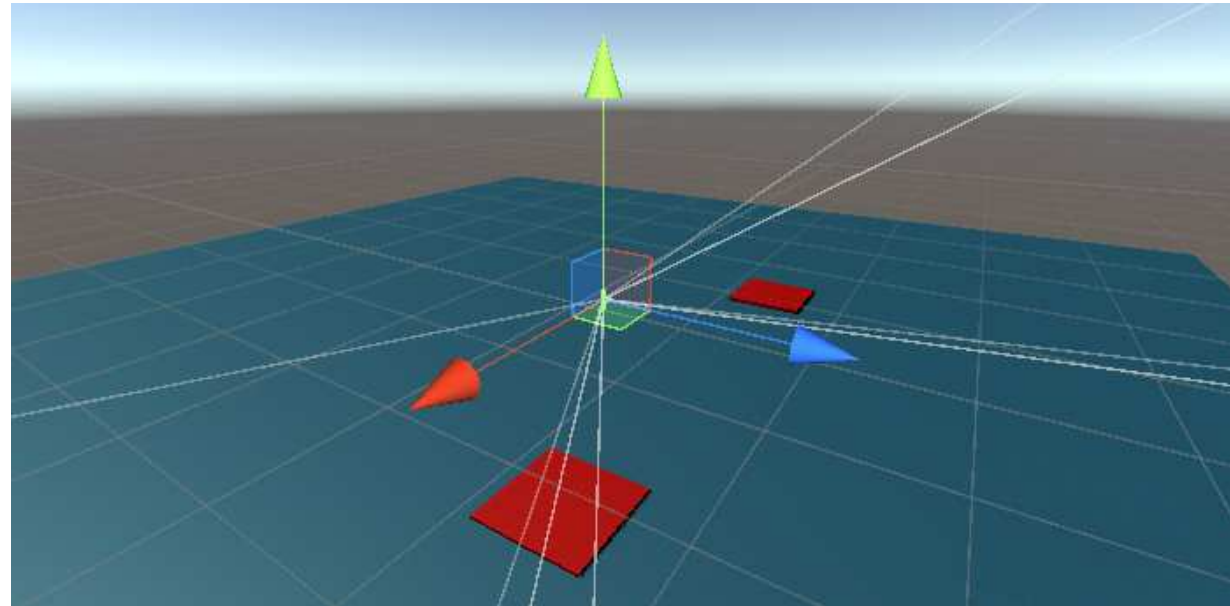
## RayCastLine1.cs

```
43
44     if (hit.collider.gameObject.name == "porte" && OVRInput.GetDown(OVRInput.Button.One))
45     {
46         Animator anim = hit.transform.GetComponent<Animator>();
47         if (etatF)
48         {
49             anim.Play("OuvrirPorte");
50             etatF = false;
51             etatO = true;
52         }else if (etatO)
53         {
54             anim.Play("FermerPorte");
55             etatO = false;
56             etatF = true;
57         }
58     }
59
60     drawLine.SetPosition(0, transform.position);
61     drawLine.SetPosition(1, endPosition);
62
63 }
```

- Glisser/Déposer le GameObject « RightHandAnchor » sur la variable « Draw Line » du script « RayCastLine1.cs »
- Build la scène

# Exercice 5 : RayCast + Teleportation

- Créer une nouvelle scène contenant un plan, deux cubes nommés « platform1 » et « platform2 ». Ces deux cubes joueront le rôle de plateformes de téléportation
- Supprimer la « Main Camera » → ajouter le Prefab « OVRPlayerController » positionner le devant le GameObject « Porte »
- Character Controller → Radius = 0.1
- Ajouter un Prefab « LocalAvatar » dans OVRPlayerController → OVRCameraRig → TrackingSpace



# Exercice 5 : RayCast + Teleportation

- Dans OVRPlayerController → OVRCameraRig → TrackingSpace → RightHandAnchor, ajouter les composants suivants:
  1. Line Renderer ayant une largeur de 0.01
  2. Un script que vous nommerez **RayCastTeleportation** dont le rôle est de générer un RayCast et d'interagir avec les deux plateformes de téléportation grâce au bouton « SecondaryIndexTrigger » de la manette Droite. Ce script est donné ci-dessous :

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class RayCastTeleportation : MonoBehaviour
6  {
7      protected CharacterController character_controller;
8      public GameObject player;
9      public LineRenderer telekinesisLine;
10     public float lineWidth = 0.1f;
11     public float lineMaxLength = 100f;
12     private float HandRight = OVRInput.Get(OVRInput.Axis1D.SecondaryHandTrigger);
13 }
```

**RayCastTeleportation.cs**

# Exercise 5 : RayCast + Teleportation

*RayCastTeleportation.cs*

```
14 void Start()
15 {
16     character_controller = player.GetComponent<CharacterController>();
17     Vector3[] startLinePositions = new Vector3[]{Vector3.zero, Vector3.zero};
18     telekinesisLine.SetPositions(startLinePositions);
19 }
20 void Update()
21 {
22     telekinesisLine.enabled = true;
23     RaycastHit hit;
24     Ray telekinesisOut = new Ray(transform.position, transform.forward);
25     Vector3 endPosition = transform.position + (lineMaxLength * transform.forward);
26     HandRight = OVRInput.Get(OVRInput.Axis1D.SecondaryIndexTrigger);
```

# Exercise 5 : RayCast + Teleportation

## *RayCastTeleportation.cs*

```
27     if (Physics.Raycast(telekinesisOut, out hit))
28     {
29         endPosition = hit.point;
30         var selection = hit.transform;
31         var selectionRenderer = selection.GetComponent<Renderer>();
32         if (hit.collider.gameObject.name == "Teleport1" && HandRight > 0.9)
33         {
34             character_controller.Move(hit.transform.position - player.transform.position);
35         }
36         else if(hit.collider.gameObject.name == "Teleport2" && HandRight > 0.9)
37         {
38             character_controller.Move(hit.transform.position - player.transform.position);
39         }
40     }
41     telekinesisLine.SetPosition(0, transform.position);
42     telekinesisLine.SetPosition(1, endPosition);
43 }
44 }
```

# Exercice 5 : RayCast + Teleportation

- Glisser/Déposer le GameObject « RightHandAnchor » sur la variable « Draw Line » du script « RayCastTeleportation.cs »
- Glisser/Déposer le GameObject « OVRPlayerController » sur la variable « Player » du script « RayCastTeleportation.cs »
- Build la scène



# Exercice en autonomie 2

## Réaliser une application de RV à partir d'un scénario

- Réaliser une pièce avec une porte
- Ajouter des objets à l'intérieur de la pièce (cubes, sphères, à partir de l'asset store, etc.)
- Le joueur est à l'extérieur, il se rapproche de la pièce puis ouvre la porte grâce à un laser. Il entre dans la pièce et avance en direction des objets.
- Le joueur peut alors interagir avec son EV, il a la possibilité de:
  - Se téléporter d'un endroit à un autre.
  - Saisir et déplacer les objets, leur changer de couleur et d'échelle.
  - Viser un objet et de lancer des sphères dans sa direction.