Lab4: VR Bowling

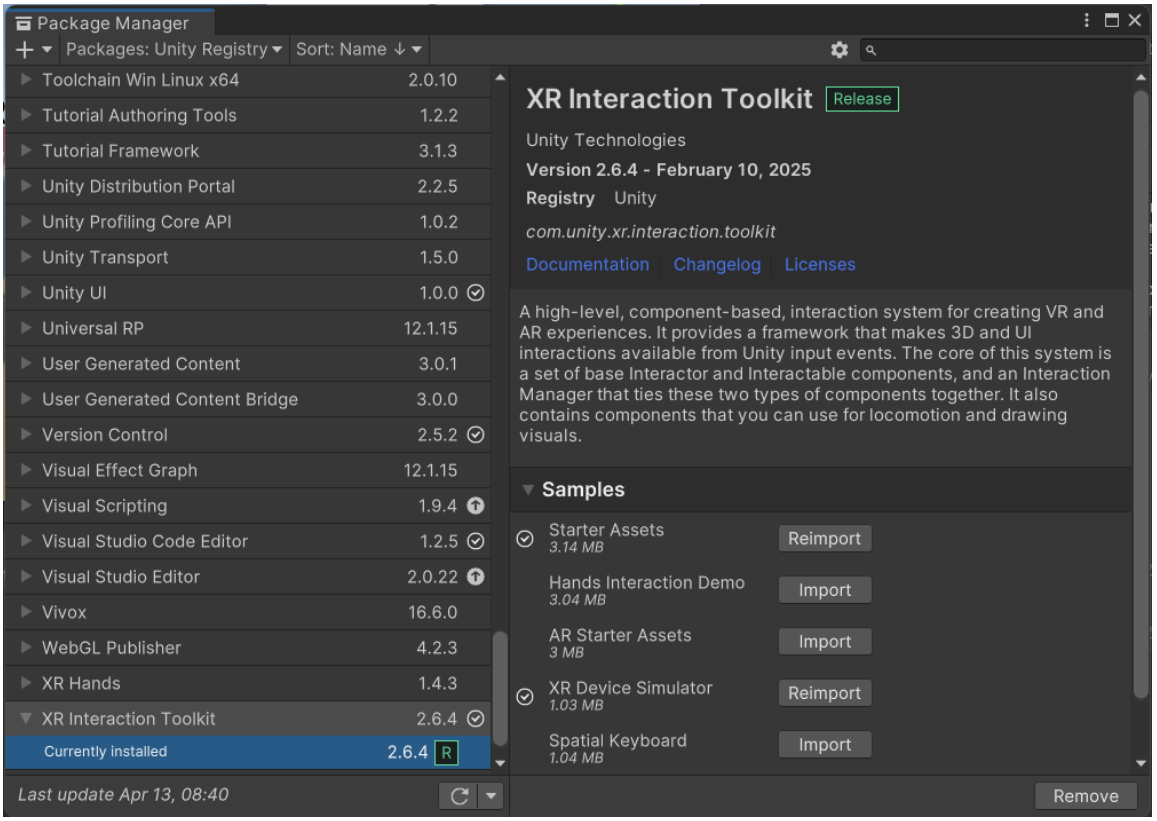
BENGHENIMA Hafsa

IASD

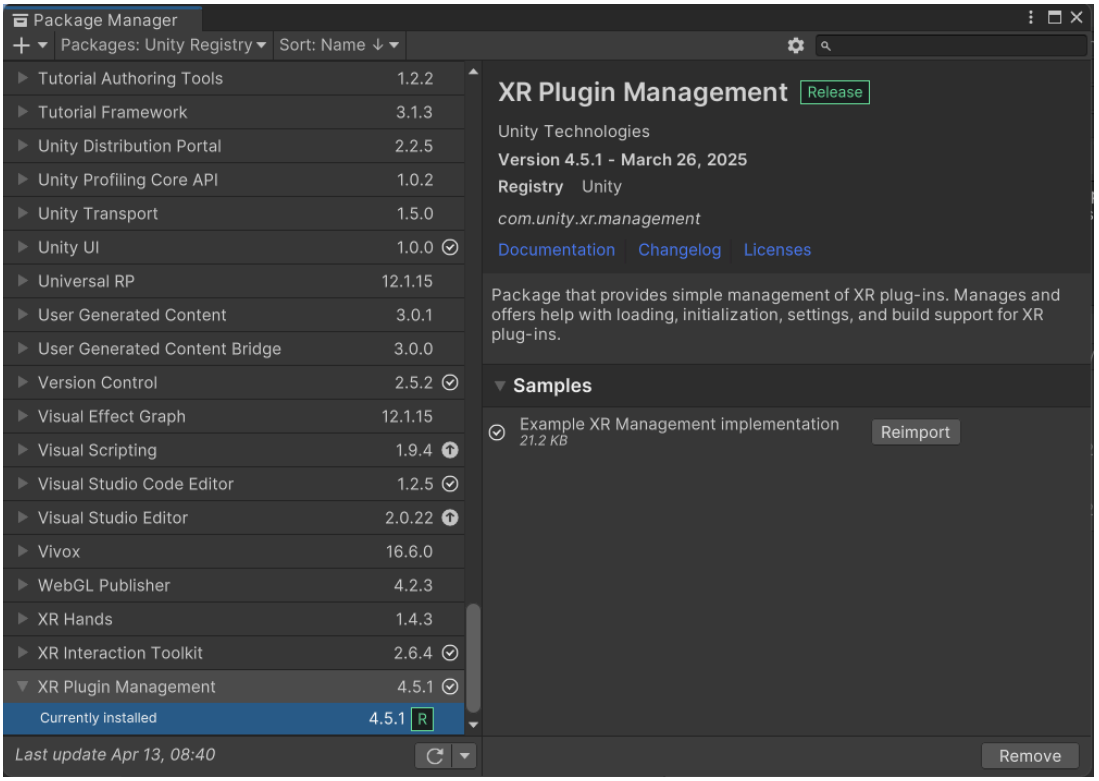
Les étapes

1. Installation des packages XR

1. Menu : `Window > Package Manager`
2. Cherchez "XR Interaction Toolkit" et installez-le et après import :
    - Starter Assets
    - XR Device Simulator

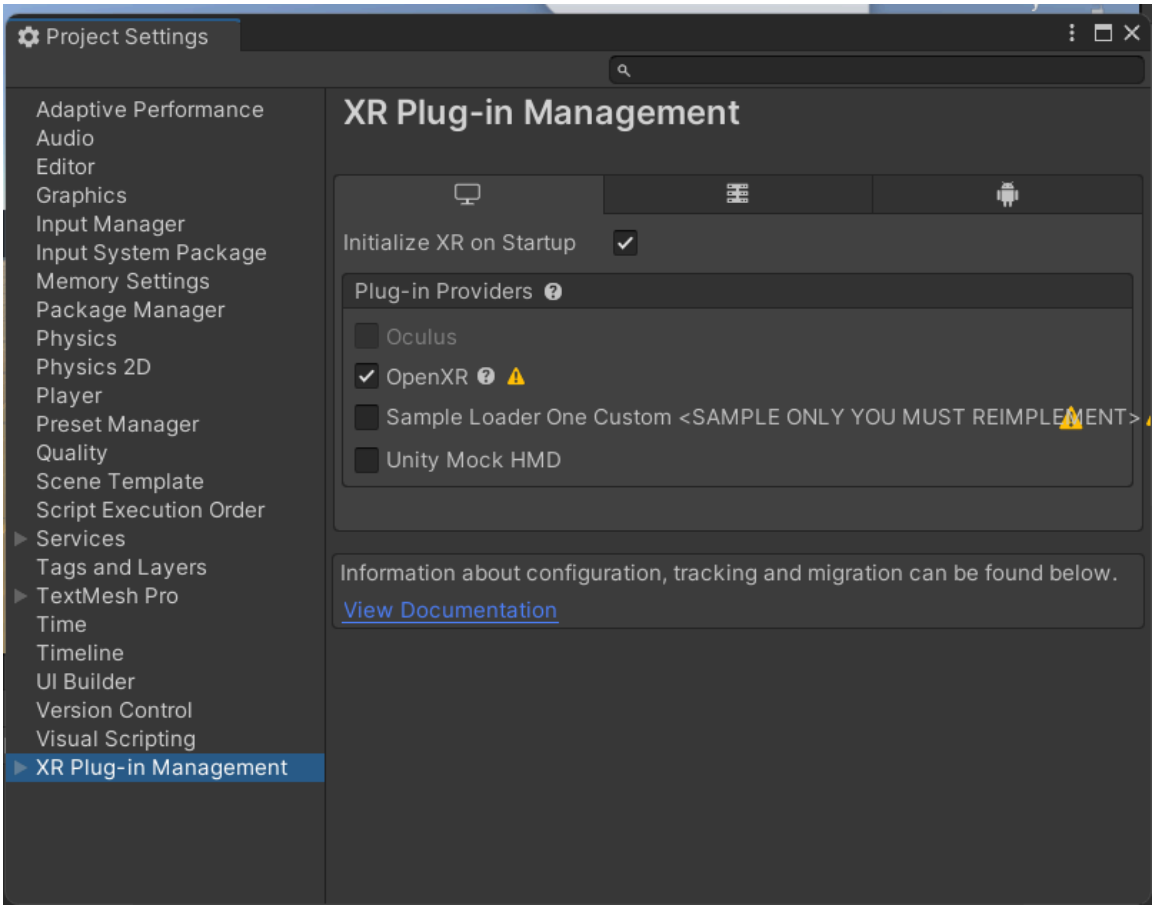


3. Cherchez "XR Plugin Management" et installez-le
    - Import le sample aussi

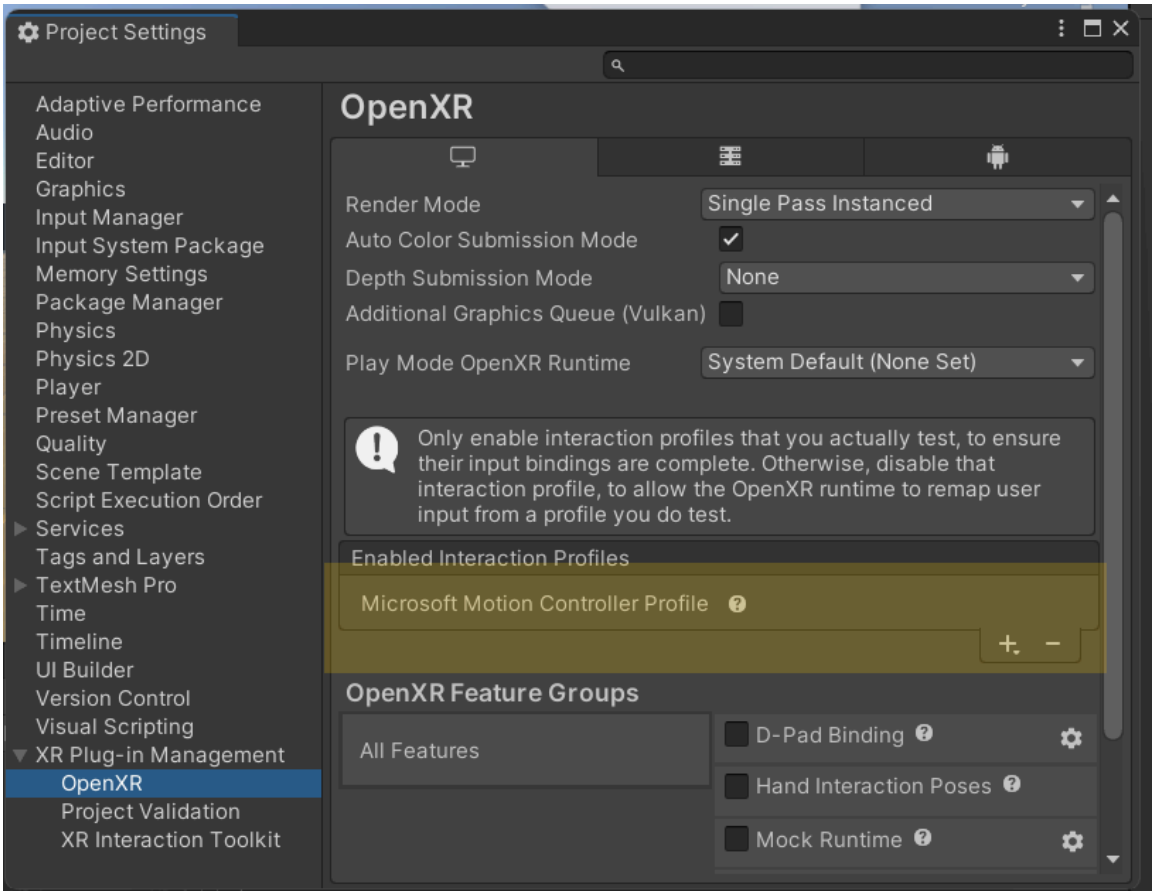


2. Configuration XR

1. Menu : `Edit > Project Settings > XR Plugin Management`
2. Dans l'onglet "Desktop", cochez "OpenXR"

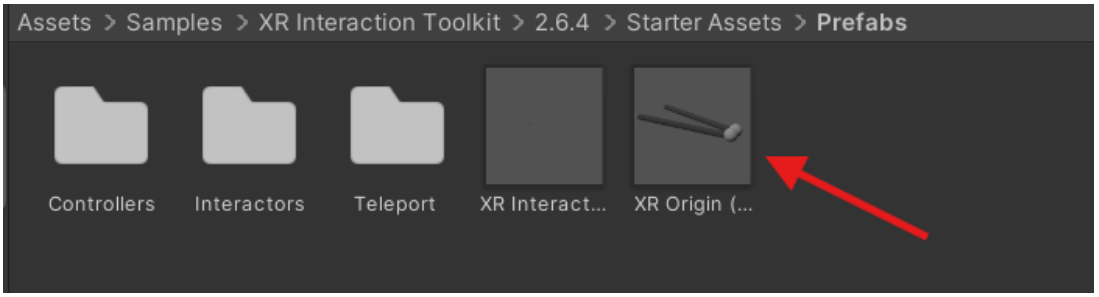


3. Cliquez sur "+" sous "Enable Interaction Profiles" et ajoutez "Microsoft Motion Controller Profile"

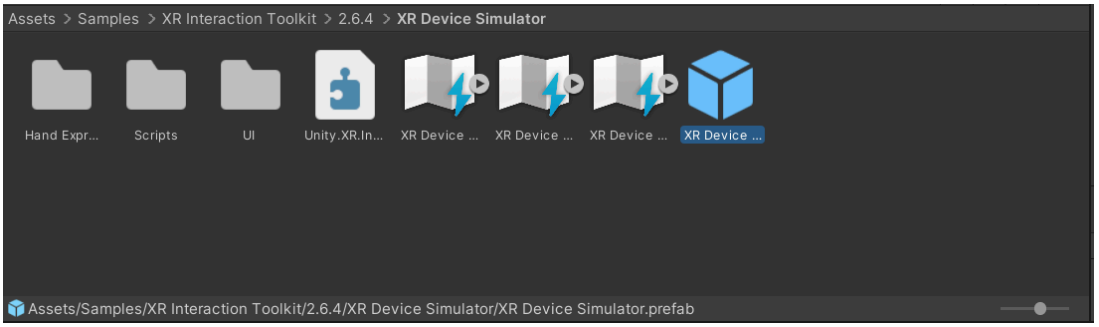


3. Création de la scène

1. Supprimez la caméra Main Camera existante
2. Créez un **Plane.**
3. Ajoutez XR Origin :
    - Dans le dossier Samples/XRI, trouvez le prefab "XR Origin (XR Rig)"

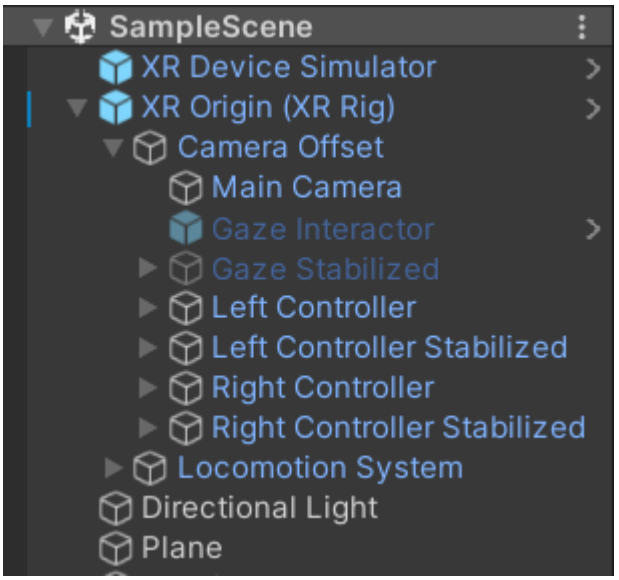


    - Faites-le glisser dans la scène
4. Ajoutez XR Device Simulator :
    - Importez le prefab depuis les exemples XR Interaction



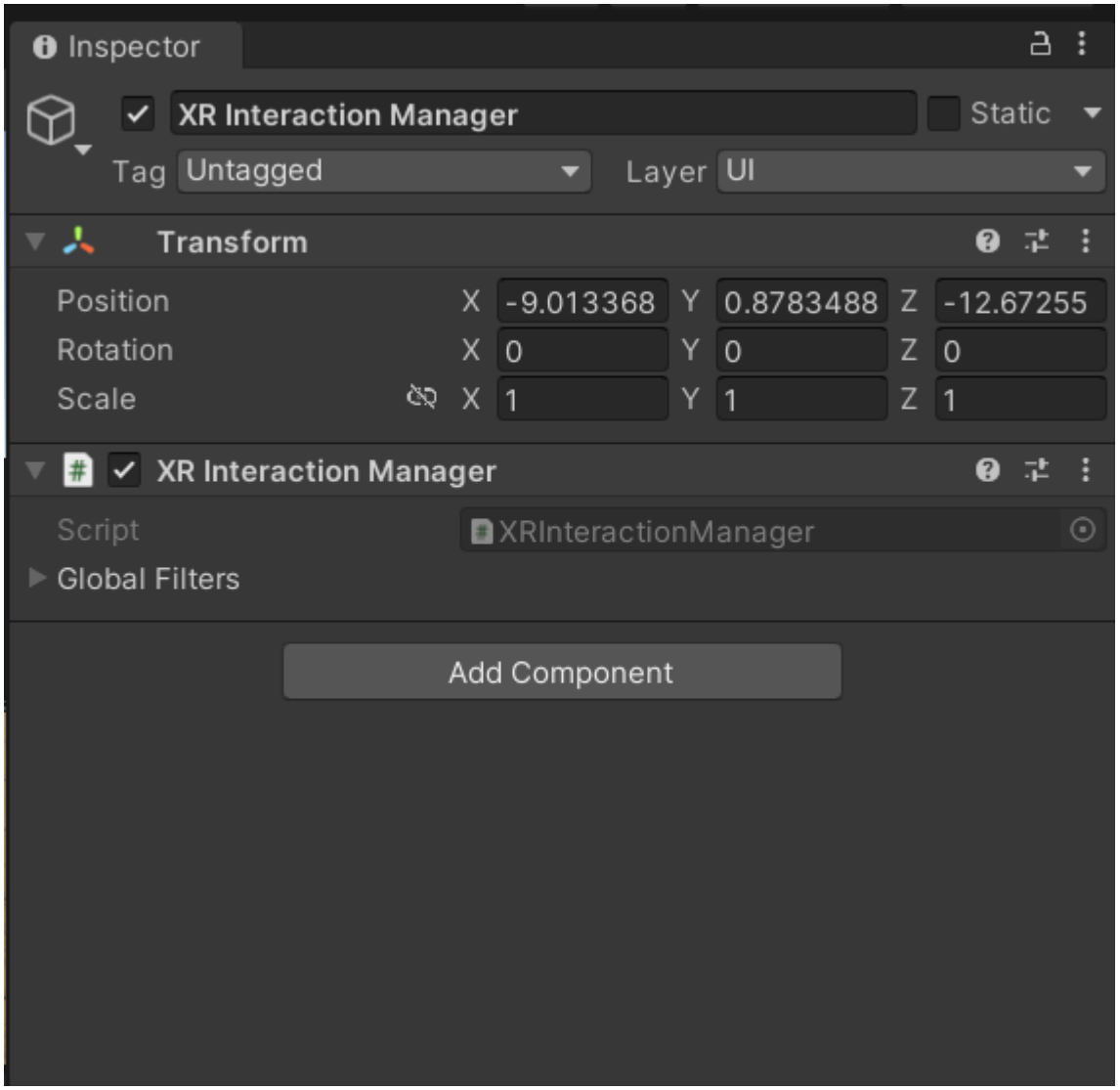
- Faites-le glisser dans la scène

5. Tu devrais avoir quelque chose comme ça dans la hierarchy:



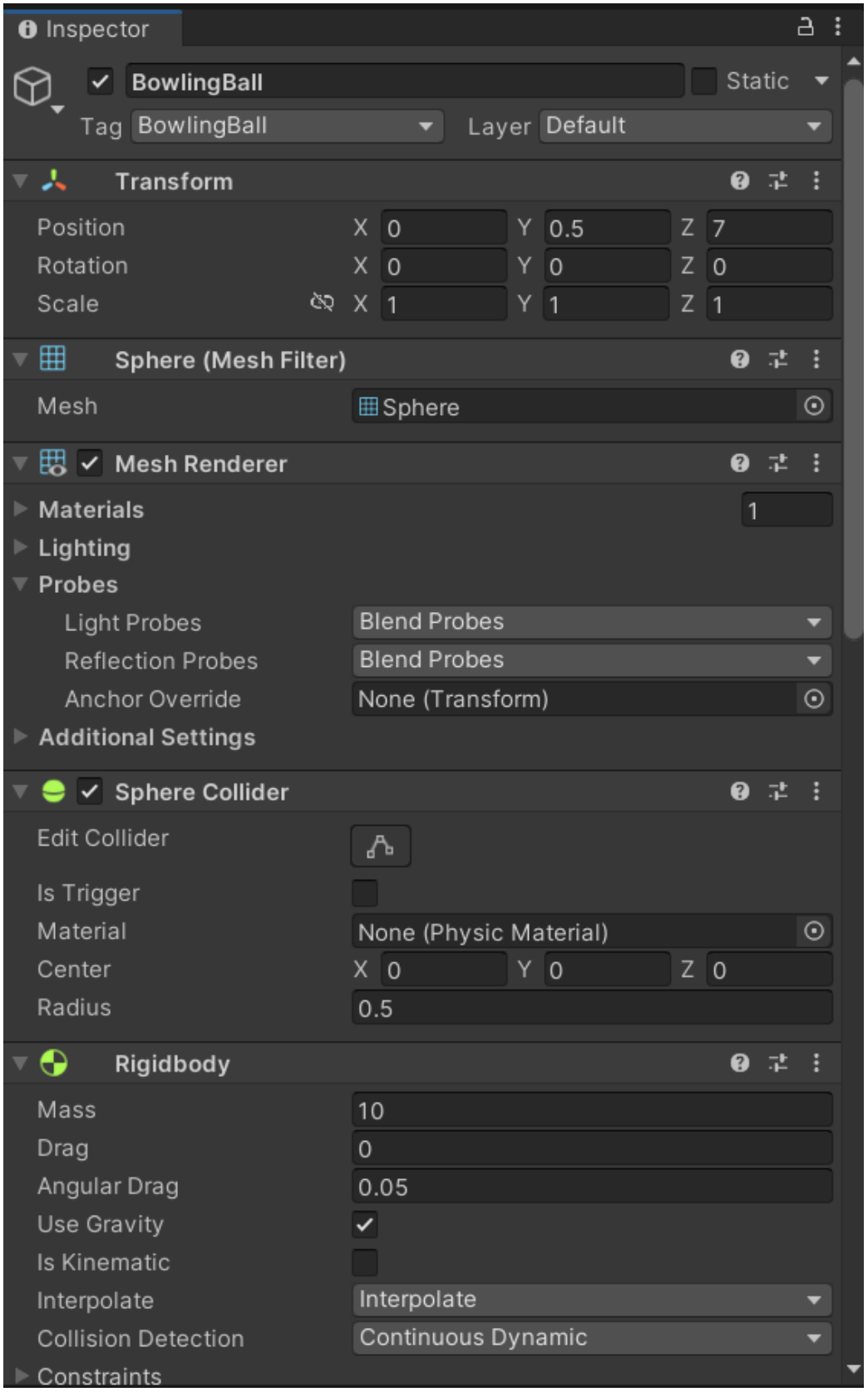
4. Création de XR Interaction Manager

- Créez un nouveau GameObject : clic droit dans la hiérarchie > **Create Empty**.
- Renommez-le **"XR Interaction Manager"**.
- Ajoutez le composant : `Add Component > XR Interaction Manager`.



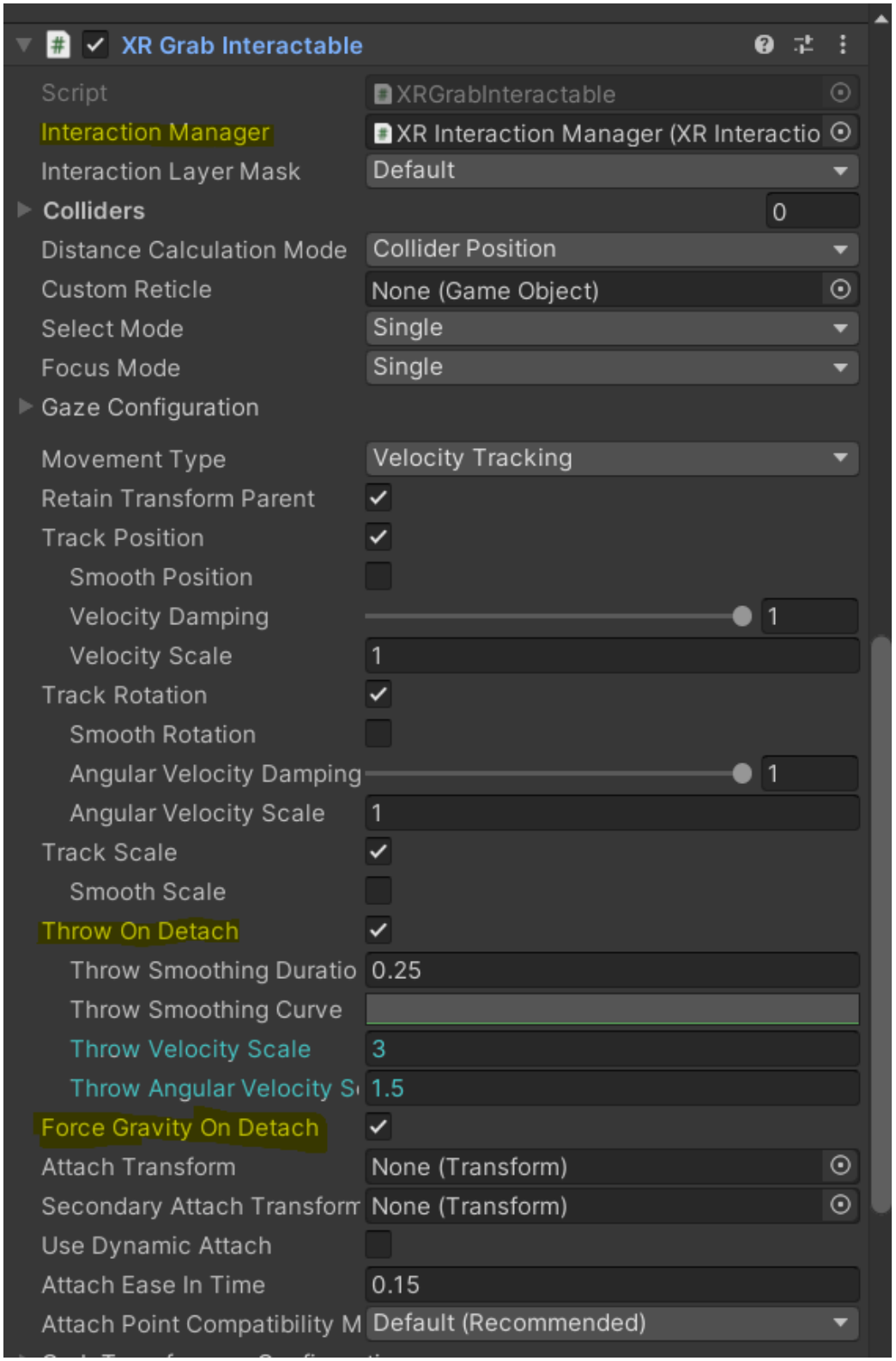
5. Création de la boule de bowling

1. Créez une Sphere
    - Renommez-la "BowlingBall"
2. Ajoutez des composants :
    - `Add Component > Rigidbody`
        - Faire la mass = 10



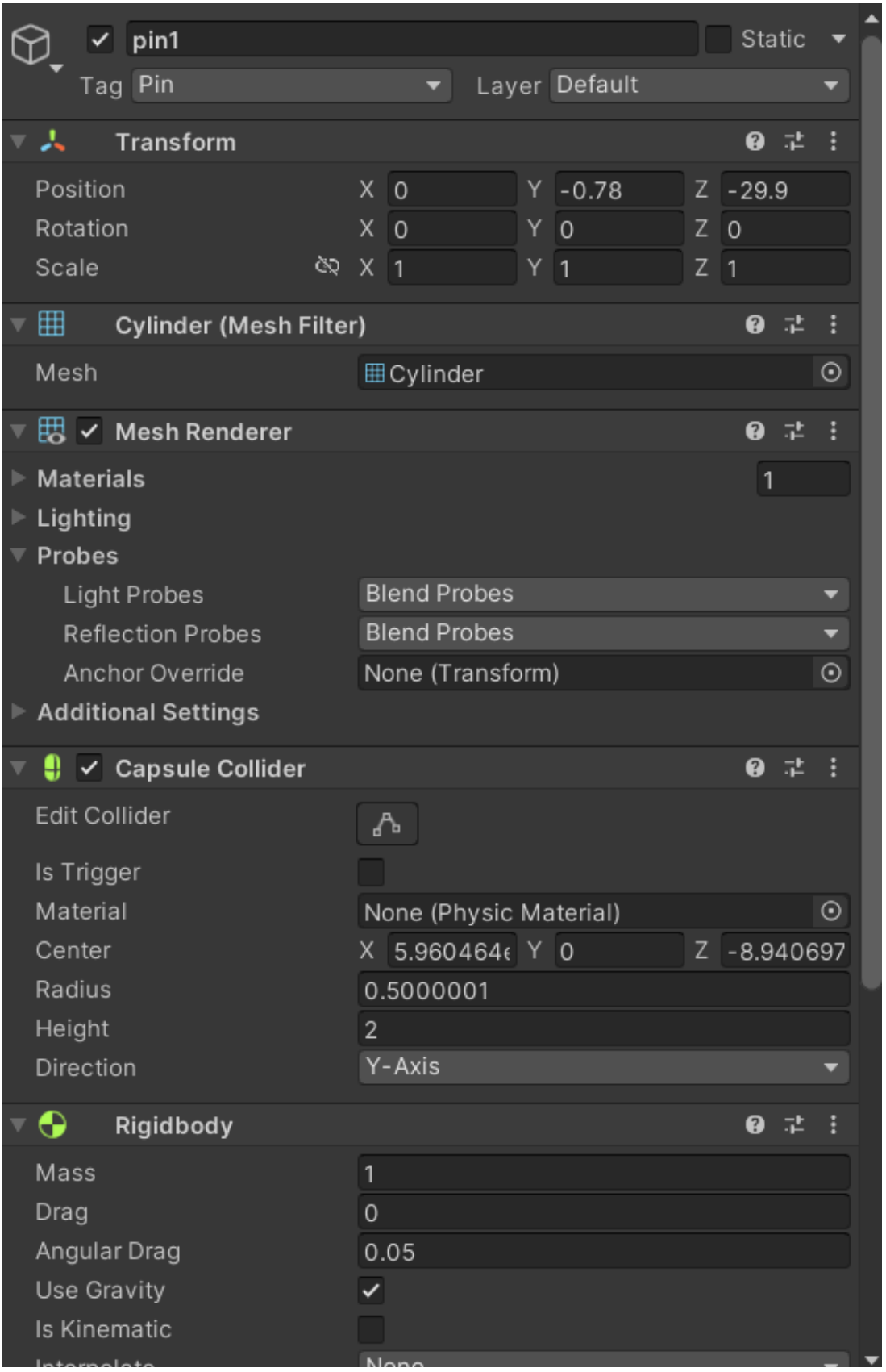
- Add Component > XR Grab Interactable
- Dans XR Grab Interactable :

- Cochez "Force Gravity"
- Cochez "Throw on Detach"
- N'oublie pas d'ajouter le Interaction Manager avec le tien dans la scène

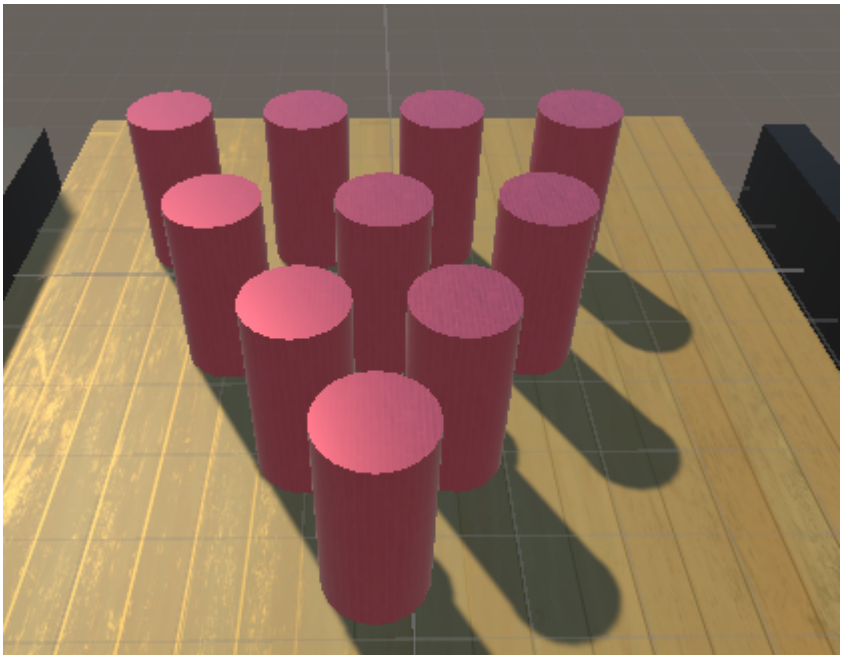


6. Création des quilles

- Créez un nouveau GameObject et Renommez-le "Pins"
- Créez un Cylinder
  - Renommez-le "Pin"
  - Ajoutez un `Rigidbody` et un `Capsule Collider`

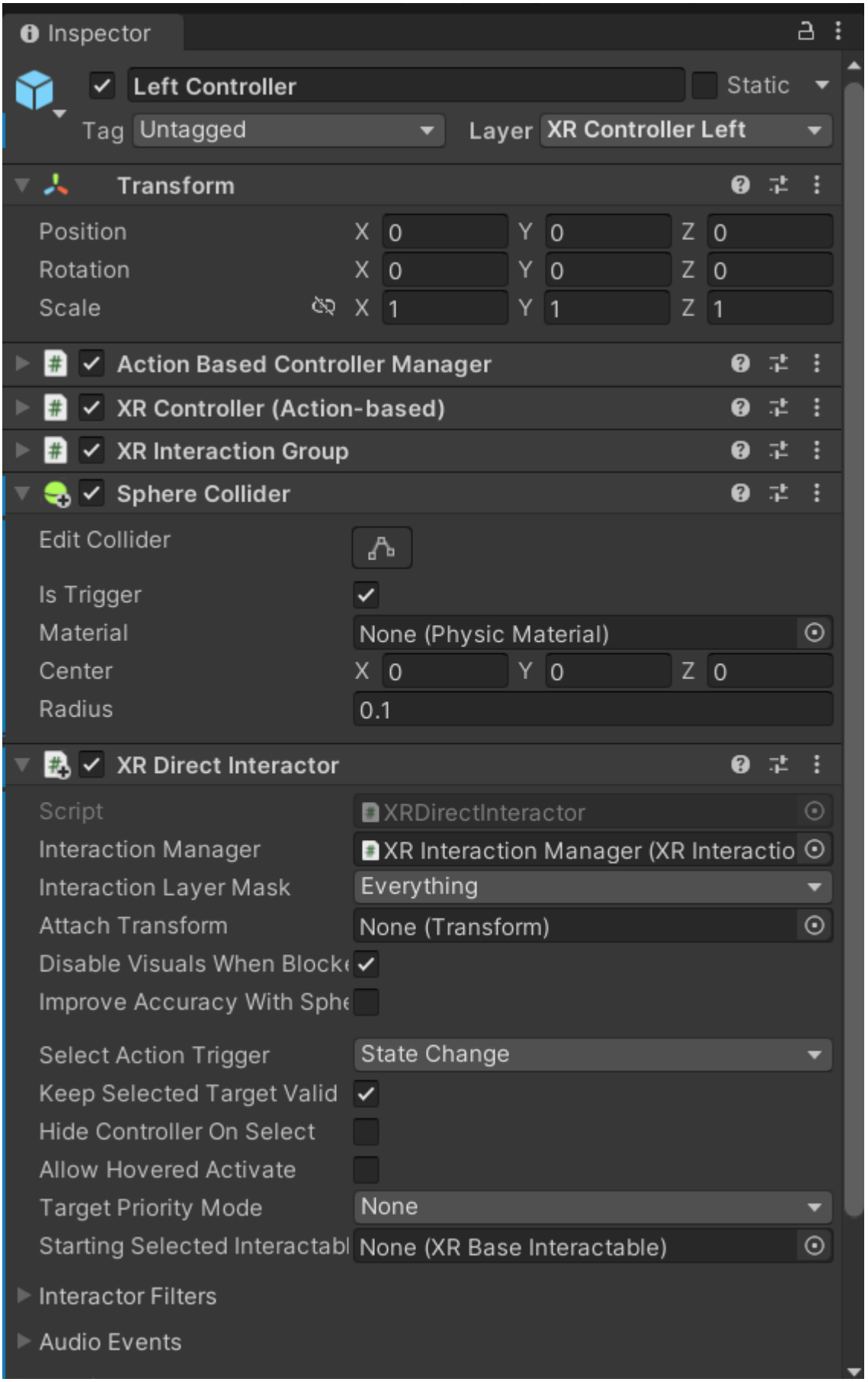


- Dupliquez la quille 9 fois
- Disposez-les en triangle :



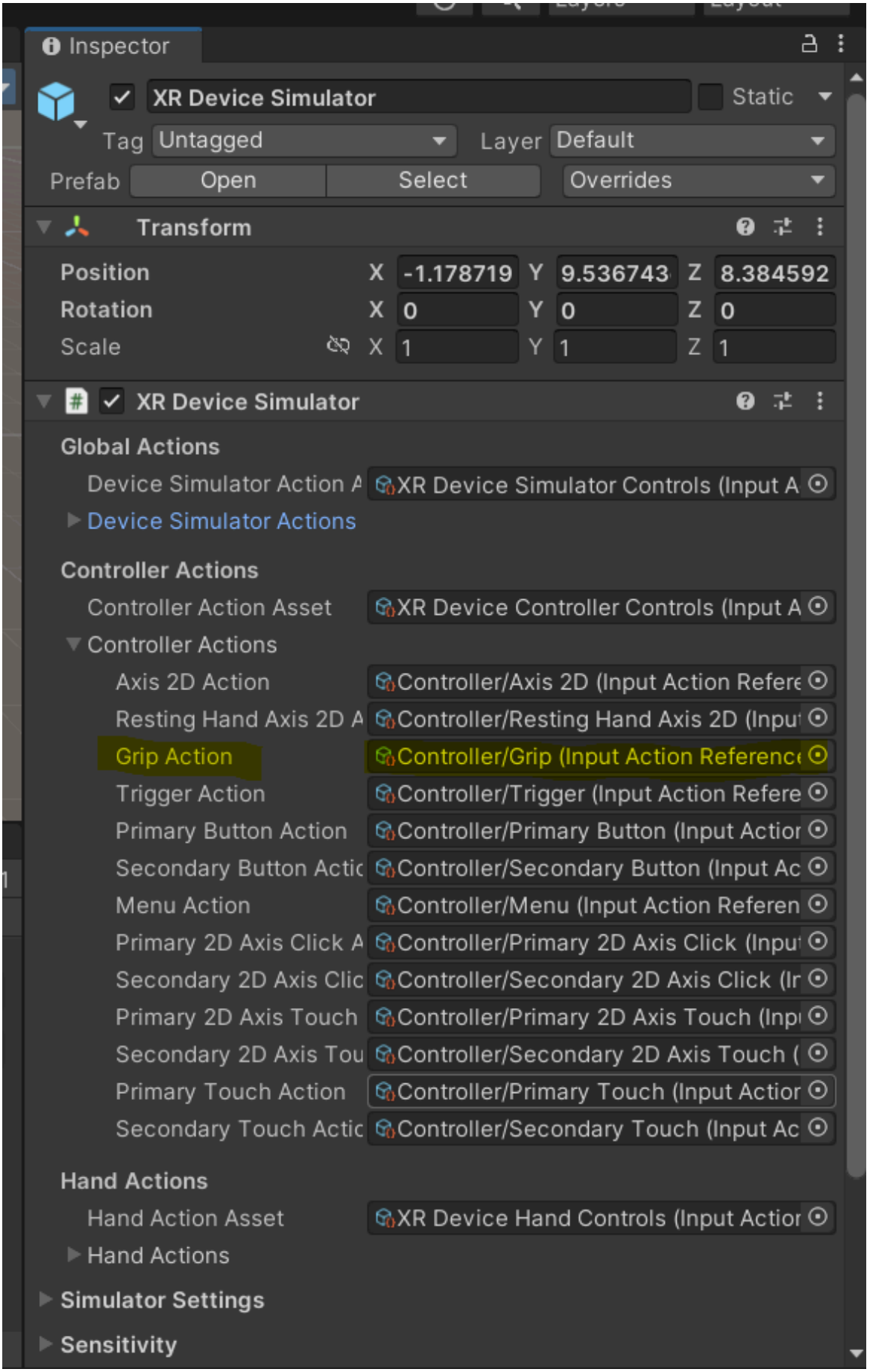
7. Configuration des controlleurs

- Sélectionnez le **XR Origin** dans la hiérarchie.
- Sélectionnez à tour de rôle :
  - **LeftHand Controller**
  - **RightHand Controller**
- Pour chaque contrôleur :
  1. Cliquez sur **"Add Component"**.
  2. Ajoutez un **Sphere Collider** (ou **Capsule Collider**).
  3. Cochez **"Is Trigger"** dans les propriétés du collider
- Ajouter XR Direct Intractor et assigné **XR Interaction Manager (le même que celui de la boule).**
- On devrait avoir tous ces composants dans les 2 contrôleurs:

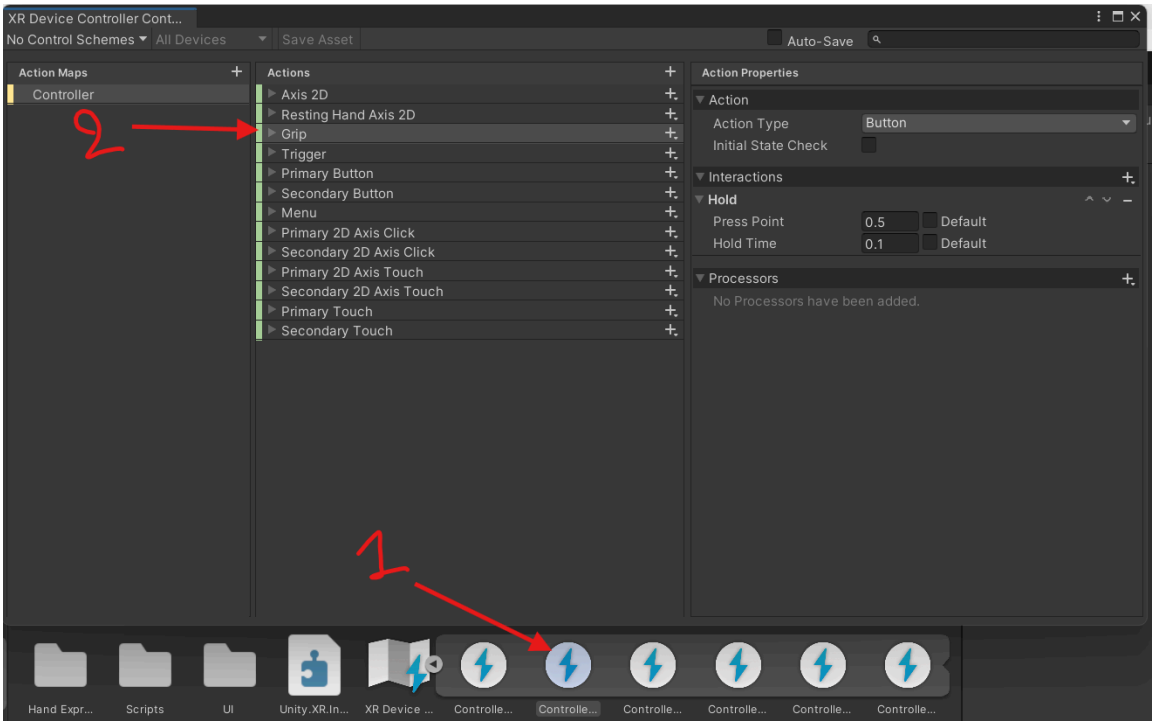

Inspector
Left Controller
Static
Tag Untagged
Layer XR Controller Left
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Action Based Controller Manager
XR Controller (Action-based)
XR Interaction Group
Sphere Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 0 Y 0 Z 0
Radius 0.1
XR Direct Interactor
Script XRDirectInteractor
Interaction Manager XR Interaction Manager (XR Interactio
Interaction Layer Mask Everything
Attach Transform None (Transform)
Disable Visuals When Block
Improve Accuracy With Sphe
Select Action Trigger State Change
Keep Selected Target Valid
Hide Controller On Select
Allow Hovered Activate
Target Priority Mode None
Starting Selected Interactabl None (XR Base Interactable)
Interactor Filters
Audio Events

8. Configurer les Input Actions

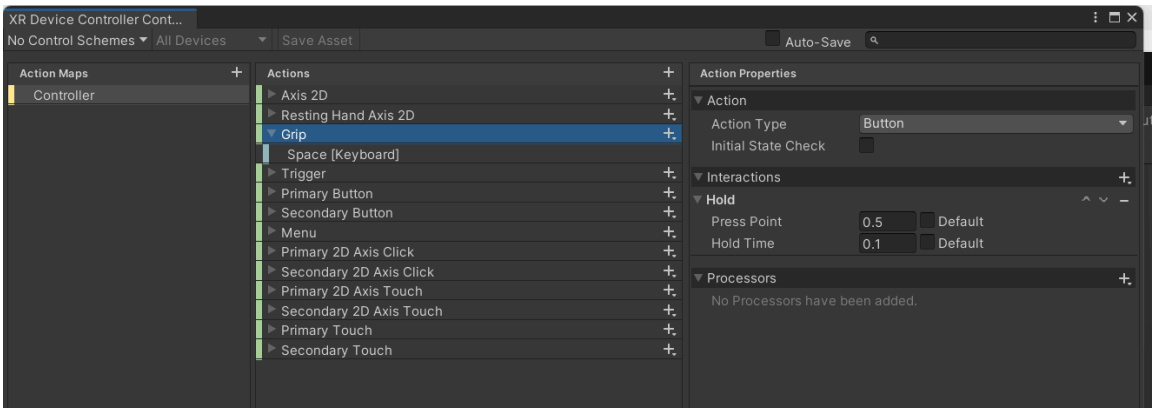
1. Ouvrez **XR Device Simulator** et va ici :


Inspector
XR Device Simulator
Static
Tag Untagged
Layer Default
Prefab Open Select Overrides
Transform
Position X -1.178719 Y 9.536743 Z 8.384592
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
XR Device Simulator
Global Actions
Device Simulator Action A XR Device Simulator Controls (Input A
Device Simulator Actions
Controller Actions
Controller Action Asset XR Device Controller Controls (Input A
Controller Actions
Axis 2D Action Controller/Axis 2D (Input Action Refere
Resting Hand Axis 2D A Controller/Resting Hand Axis 2D (Inpu
Grip Action Controller/Grip (Input Action Referenc
Trigger Action Controller/Trigger (Input Action Refere
Primary Button Action Controller/Primary Button (Input Actior
Secondary Button Actic Controller/Secondary Button (Input Ac
Menu Action Controller/Menu (Input Action Referen
Primary 2D Axis Click A Controller/Primary 2D Axis Click (Inpu
Secondary 2D Axis Clic Controller/Secondary 2D Axis Click (Ir
Primary 2D Axis Touch Controller/Primary 2D Axis Touch (Inp
Secondary 2D Axis Tou Controller/Secondary 2D Axis Touch (
Primary Touch Action Controller/Primary Touch (Input Actior
Secondary Touch Actic Controller/Secondary Touch (Input Ac
Hand Actions
Hand Action Asset XR Device Hand Controls (Input Actior
Hand Actions
Simulator Settings
Sensitivity

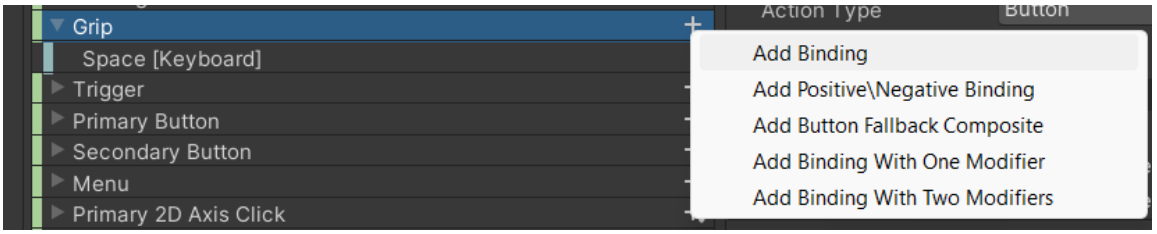
2. Et puis:



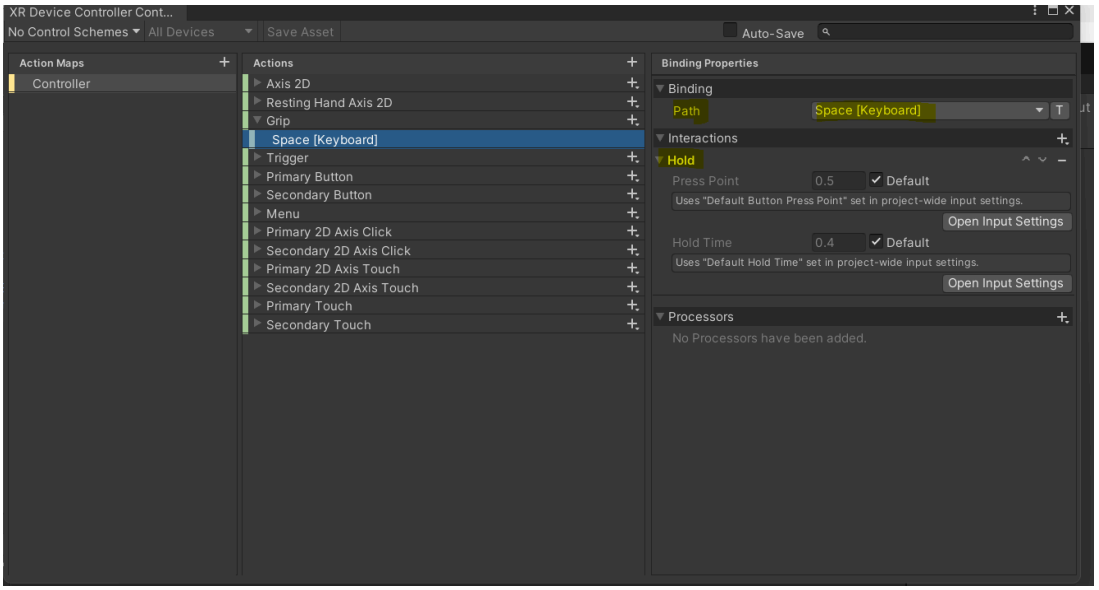
3. Dans la section **Interactions** pour Grip :
   - Clique sur `+ pour ajouter une interaction`
   - Sélectionne **Hold** dans le menu déroulant



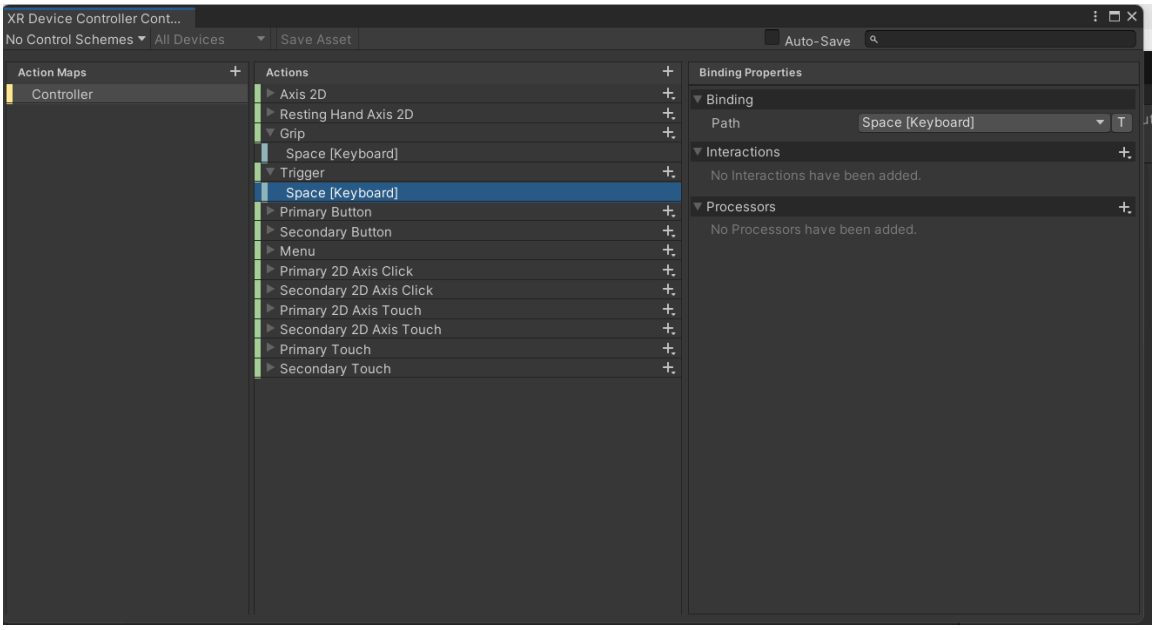
4. Vérifiez les bindings et cliqué sur **Add Binding**:



5. Et pour le **Space keyboard:**
    - Changer le path à `Keyboard/space`
    - ajouter a Interactions **Hold:**



6. Faire le meme pour Trigger **mais sans hold!**



Et maintenant, on a :

Grab: Maintenir Espace (0,2s) → Les mains s'attachent à la balle

Throw: Relâcher Espace → La balle est lancée avec la physique

9. Creation de side Lines

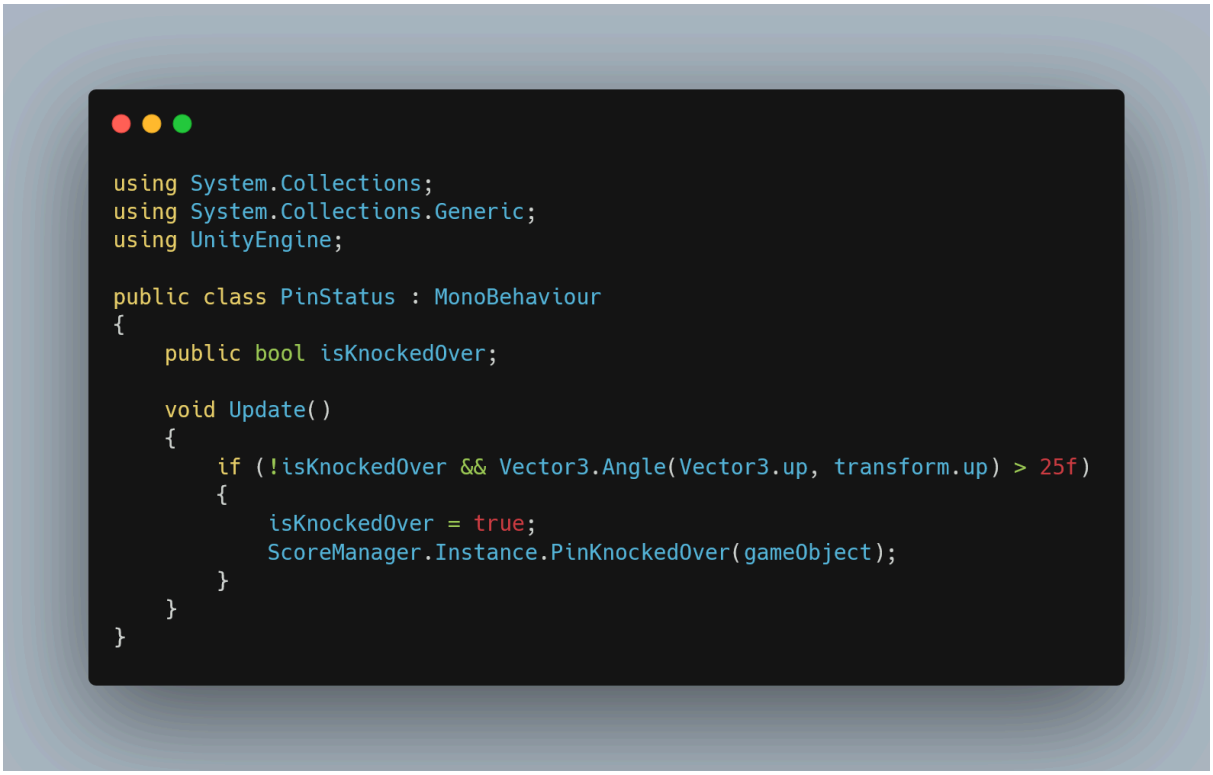
- Créer un **objet vide** et le renommer **Lines**
- Créer **2 cubes**
- Chaque cube doit avoir un **Box Collider**

10. Créer le score manager:

Step 1: Créer un nouveau script C# `ScoreManager.cs` :

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


public class ScoreManager : MonoBehaviour
{
    public static ScoreManager Instance;
    private void Awake() => Instance = this;

    [Header("UI Configuration")]
    [SerializeField] private TMP_Text scoreText;

    [Header("Pin Settings")]
    [SerializeField] private string pinTag = "Pin";

    private List<GameObject> standingPins = new List<GameObject>();
    private Vector3[] pinInitialPositions;
    private Quaternion[] pinInitialRotations;
    private int currentScore;

    void Start()
    {
        InitializePins();
        UpdateScoreDisplay();
    }

    void InitializePins()
    {
        GameObject[] pins = GameObject.FindGameObjectsWithTag(pinTag);
        pinInitialPositions = new Vector3[pins.Length];
        pinInitialRotations = new Quaternion[pins.Length];

        for (int i = 0; i < pins.Length; i++)
        {
            standingPins.Add(pins[i]);
            pinInitialPositions[i] = pins[i].transform.position;
            pinInitialRotations[i] = pins[i].transform.rotation;
        }
    }

    public void PinKnockedOver(GameObject pin)
    {
        if (!standingPins.Contains(pin)) return;

        standingPins.Remove(pin);
        currentScore++;
        UpdateScoreDisplay();

        if (standingPins.Count == 0)
        {
            Debug.Log("All pins knocked down!");
        }
    }

    public void ResetGame()
    {
        currentScore = 0;
        UpdateScoreDisplay();
        ResetAllPins();
    }

    void ResetAllPins()
    {
        GameObject[] pins = GameObject.FindGameObjectsWithTag(pinTag);

        for (int i = 0; i < pins.Length; i++)
        {
            Rigidbody rb = pins[i].GetComponent<Rigidbody>();
            rb.velocity = Vector3.zero;
            rb.angularVelocity = Vector3.zero;

            rb.isKinematic = true;
            pins[i].transform.position = pinInitialPositions[i];
            pins[i].transform.rotation = pinInitialRotations[i];
            rb.isKinematic = false;

            PinStatus pinStatus = pins[i].GetComponent<PinStatus>();
            pinStatus.isKnockedOver = false;
            standingPins.Add(pins[i]);
        }
    }

    void UpdateScoreDisplay()
    {
        scoreText.text = $"SCORE: {currentScore}";
    }
}
```

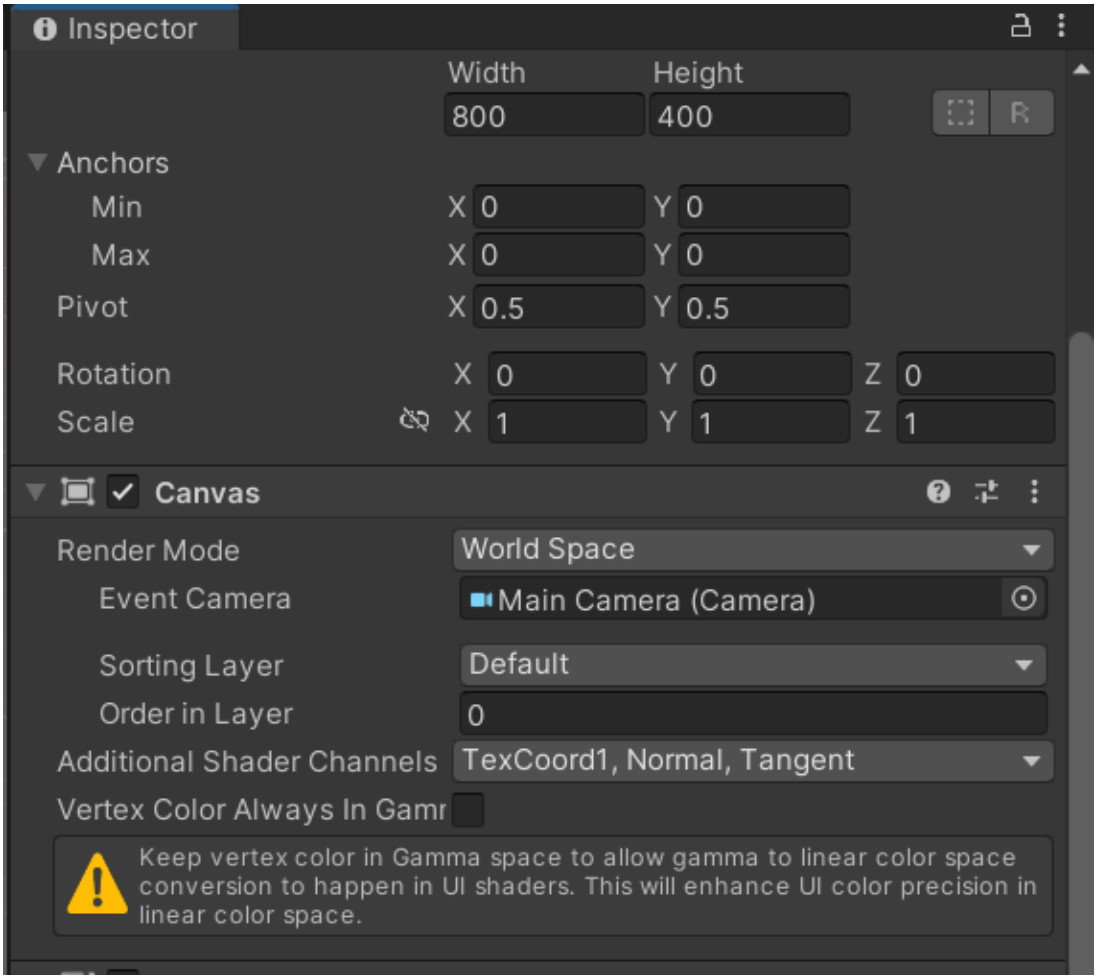
Étape 2: Créer PinStatus.cs et l'attacher à chaque pin :



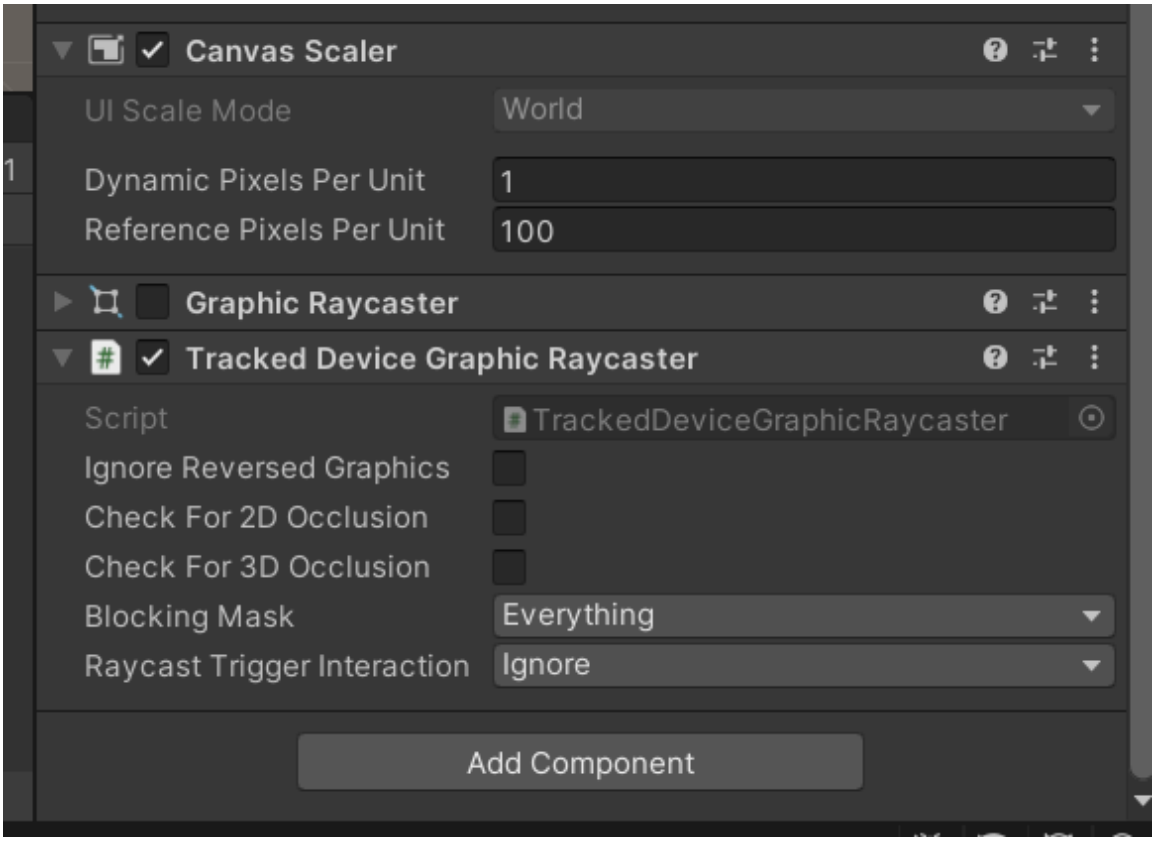
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PinStatus : MonoBehaviour
{
    public bool isKnockedOver;

    void Update()
    {
        if (!isKnockedOver && Vector3.Angle(Vector3.up, transform.up) > 25f)
        {
            isKnockedOver = true;
            ScoreManager.Instance.PinKnockedOver(gameObject);
        }
    }
}
```

Étape 3: Configurer l'UI

- Clic droit sur la Hiérarchie → UI → Canvas
- Configurer les paramètres du Canvas :
  - Render Mode: World Space
  - Event Camera: XR Origin Camera

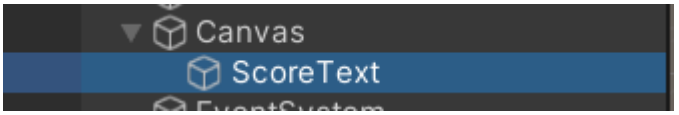


- Ajouter un composant **: Tracked Device Graphic Raycaster**



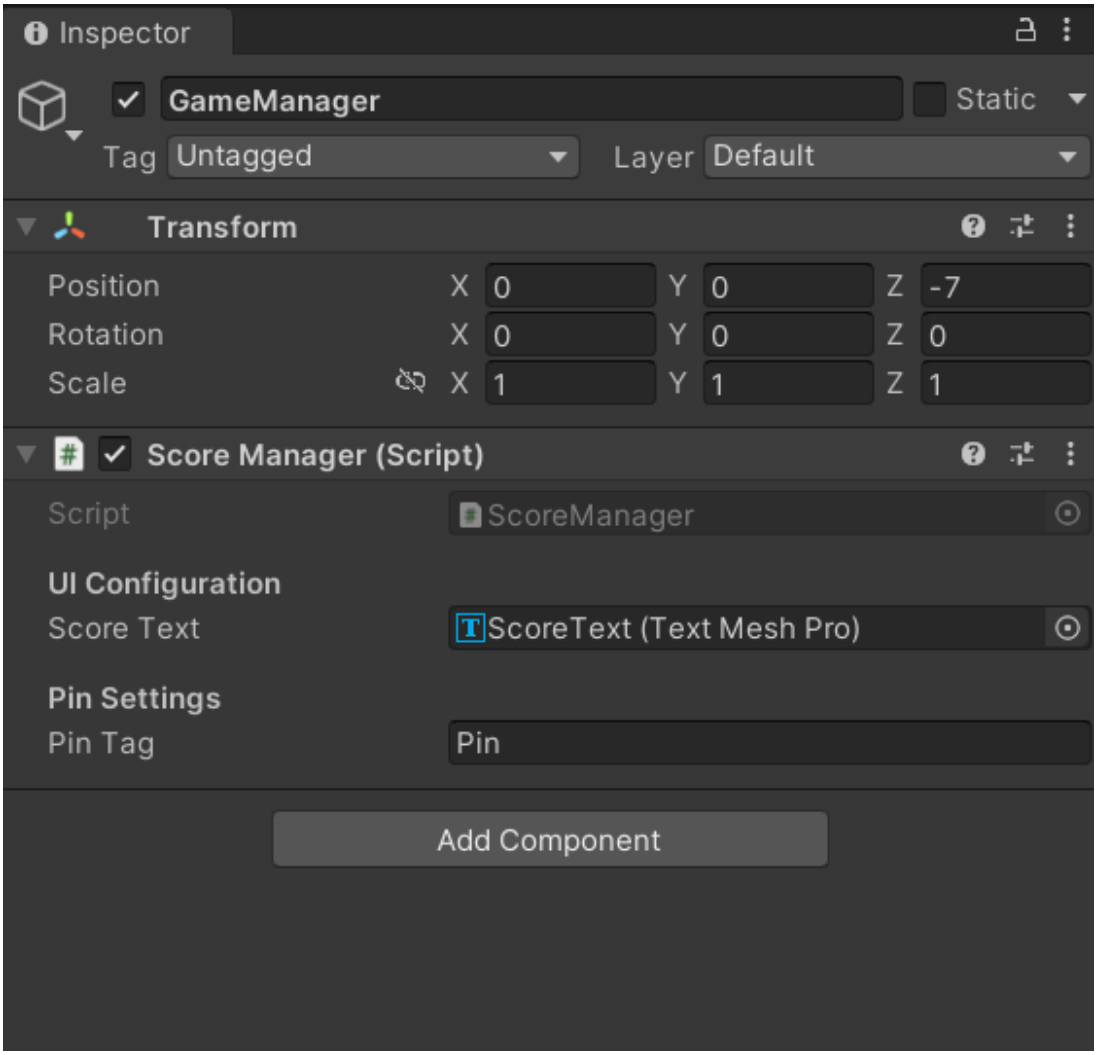
Étape 4 : Ajouter des éléments UI

1. Créer un enfant de type TMP Text:
    - Name: ScoreText
    - Text: SCORE: 0



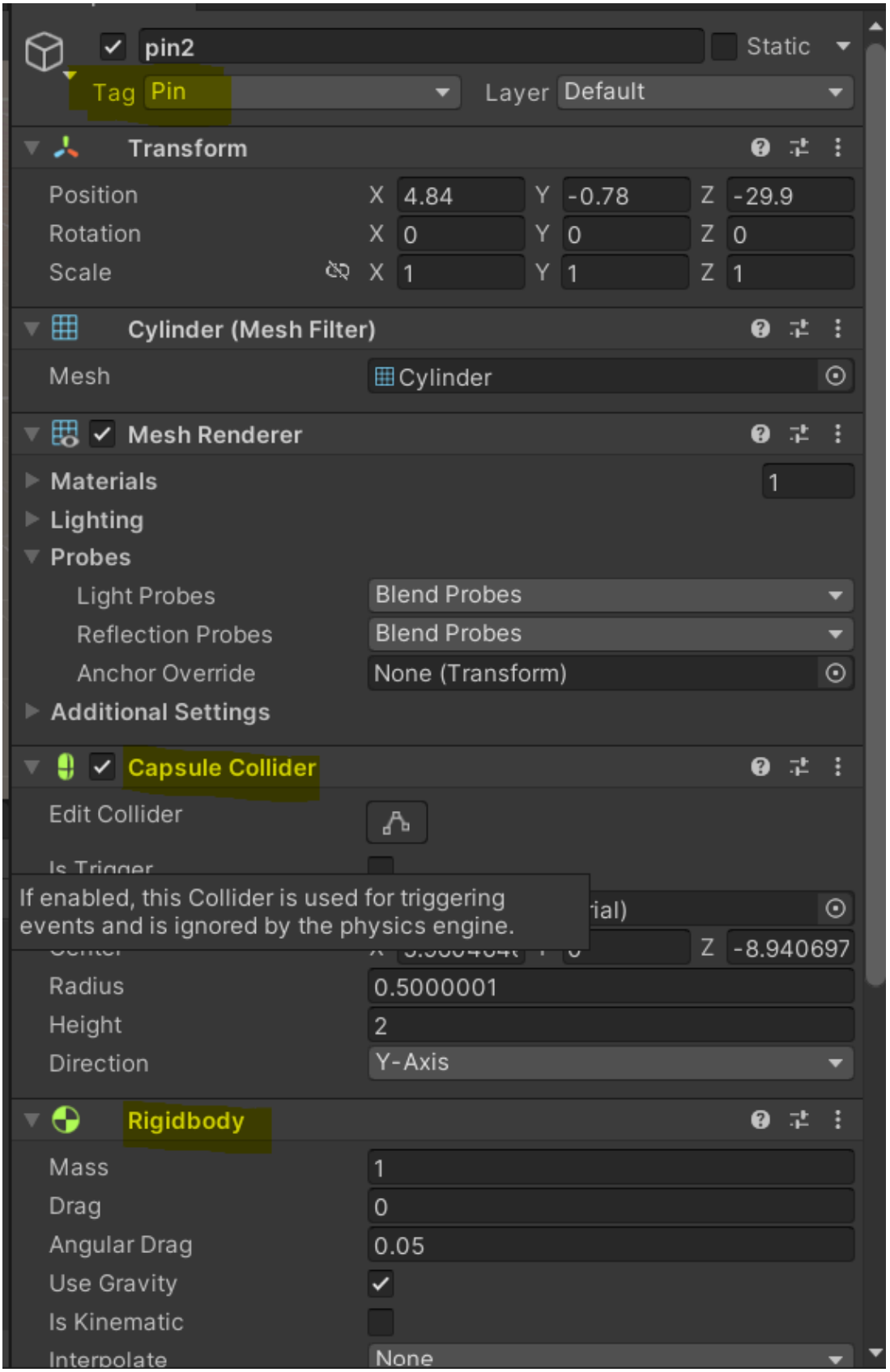
Étape 5 : Assignation du `ScoreManager.cs` :

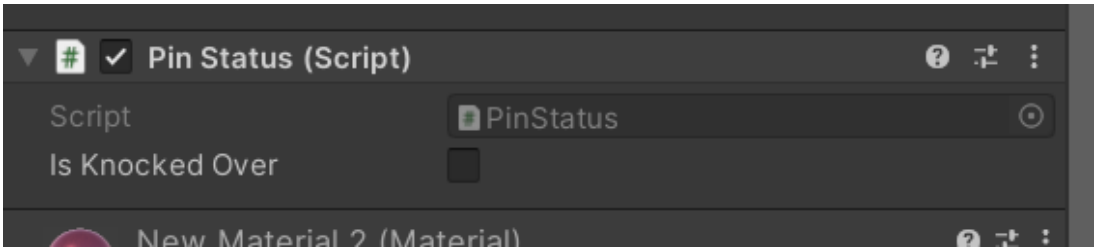
- Clic droit sur la Hiérarchie → **Créer Empty** → Renommer en "GameManager"
- Glisser le script `ScoreManager.cs` dessus
- Dans l'Inspector, assigner les références :
    - **Score Text** : Glisser l'élément UI Text (ScoreText) ici



Étape 6 : Assignation du `PinStatus.cs` :

- Sélectionner toutes les pins dans la Hiérarchie
- Glisser le script **PinStatus.cs** sur tous ces pins
- Vérifier que toutes les pins:
  - Ont le tag **"Pin"** (important !)
  - Ont le composant **Rigidbody** (pour la détection des physiques)


pin2
Static
Tag Pin
Layer Default
Transform
Position X 4.84 Y -0.78 Z -29.9
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Cylinder (Mesh Filter)
Mesh Cylinder
Mesh Renderer
Materials 1
Lighting
Probes
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Additional Settings
Capsule Collider
Edit Collider
If enabled, this Collider is used for triggering events and is ignored by the physics engine.
Z -8.940697
Radius 0.5000001
Height 2
Direction Y-Axis
Rigidbody
Mass 1
Drag 0
Angular Drag 0.05
Use Gravity
Is Kinematic
None

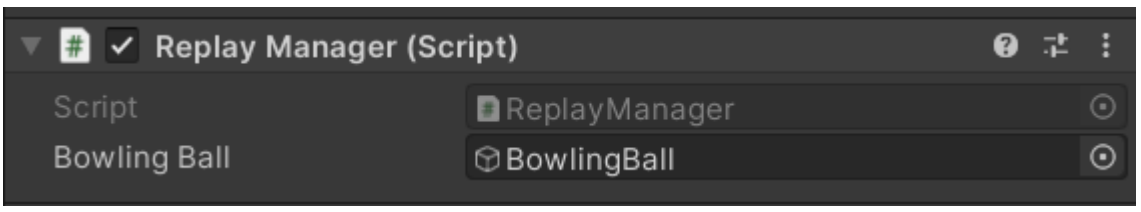


11. Créer le replay manager:

1. Create new C# script `ReplayManager.cs`:

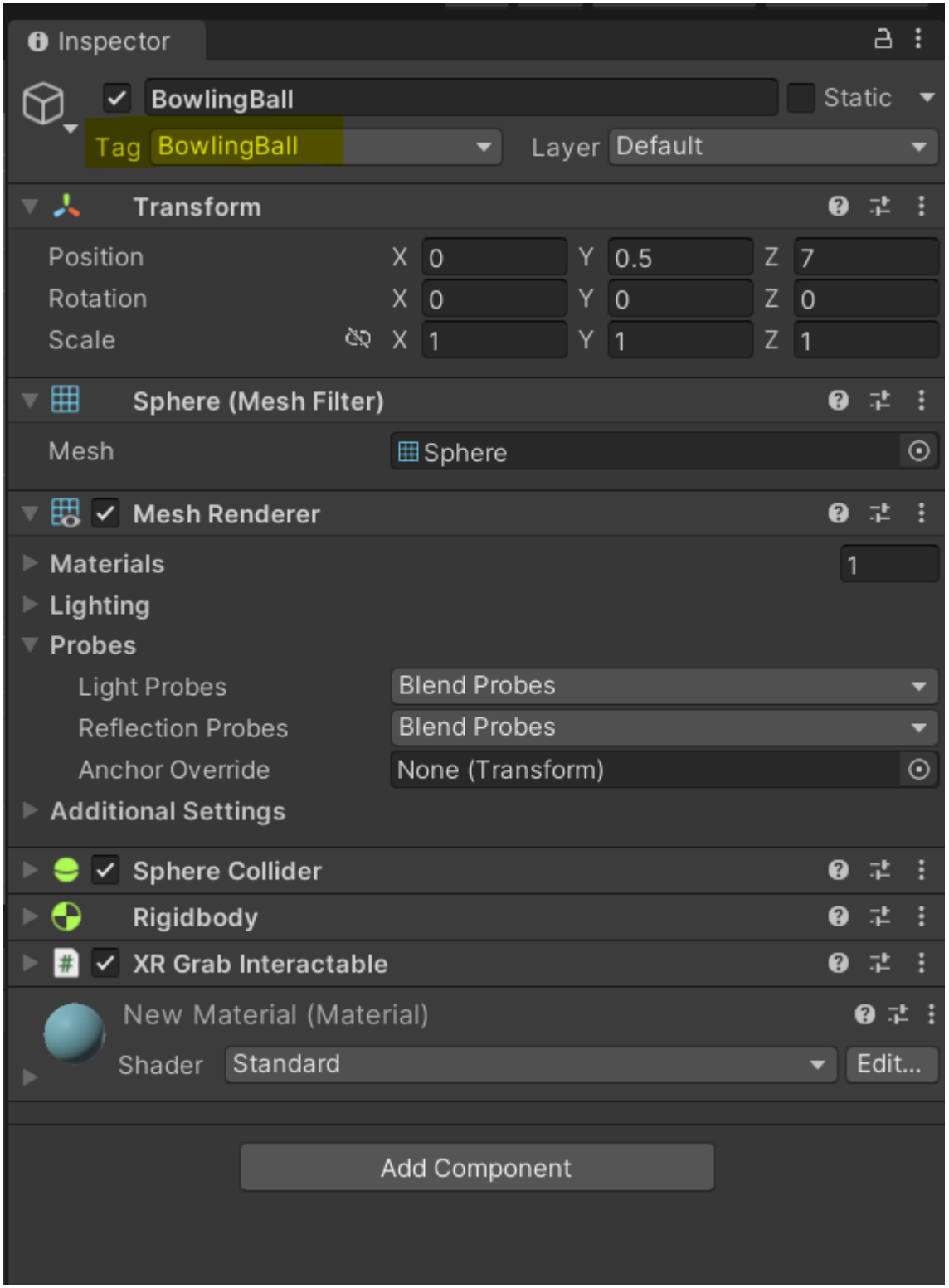
```
using UnityEngine;
using UnityEngine.XR.Interaction.Toolkit;

public class ReplayManager : MonoBehaviour
{
    [SerializeField] private GameObject bowlingBall;
    private Vector3 ballStartPosition;
    private Rigidbody ballRb;

    void Start()
    {
        ballRb = bowlingBall.GetComponent<Rigidbody>();
        ballStartPosition = bowlingBall.transform.position;
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.R))
        {
            ResetBall();
            ScoreManager.Instance.ResetGame();
        }
    }

    void ResetBall()
    {
        ballRb.velocity = Vector3.zero;
        ballRb.angularVelocity = Vector3.zero;
        ballRb.isKinematic = true;
        bowlingBall.transform.position = ballStartPosition;
        ballRb.isKinematic = false;
    }
}
```

2. Glisser le script `ReplayManager.cs` sur le **game manager**
3. Ajouter le ballbowling dans inspector de **game manager**



4. Taguer la boule de bowling (si ce n’est pas déjà fait) :

- Sélectionner l’objet **BowlingBall**
- Dans l’**Inspector** → Menu déroulant **Tag** → **Add Tag...**
- Créer un nouveau tag nommé **"BowlingBall"**
- Assigner ce tag à **BowlingBall**



5. **Remarque :** les quilles doivent être **taguées avec le tag "Pin"**.

Donc, si j'appuie sur R → la balle et les quilles doivent revenir à leur position de départ et le texte du score doit être remis à 0.

Heirarchy

- À la fin, la hiérarchie sera :