HMIN320: Vision, Réalité virtuelle et augmentée

Rendu TP4 : AR & VR

Tianning MA

M2 IMAGINA

17/10/2020

Table de matière

1. Introduction

Ce compte rendu est dédié au TP 4 AR avec Vulforia et VR en Cardboard en Unity dans le cours de « Vision, Réalité virtuelle et augmentée ».

Vous trouvez également tous les dossiers pour ces tps dans le git suivant :

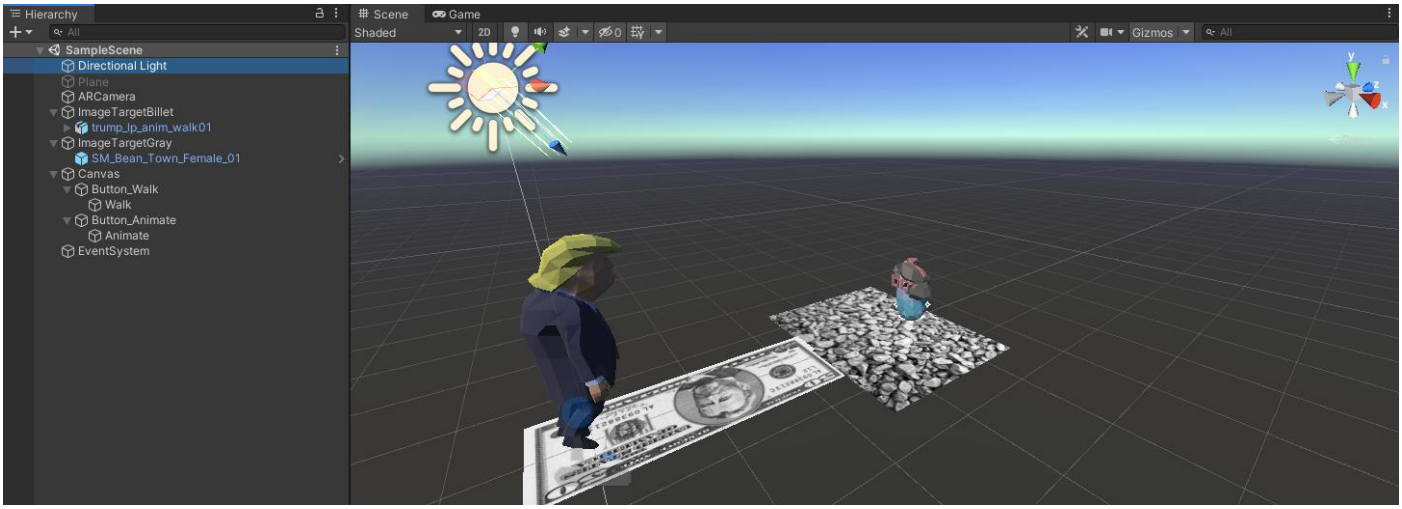
https://github.com/matianning/AR-VR/tree/Tianning

2. Exercices

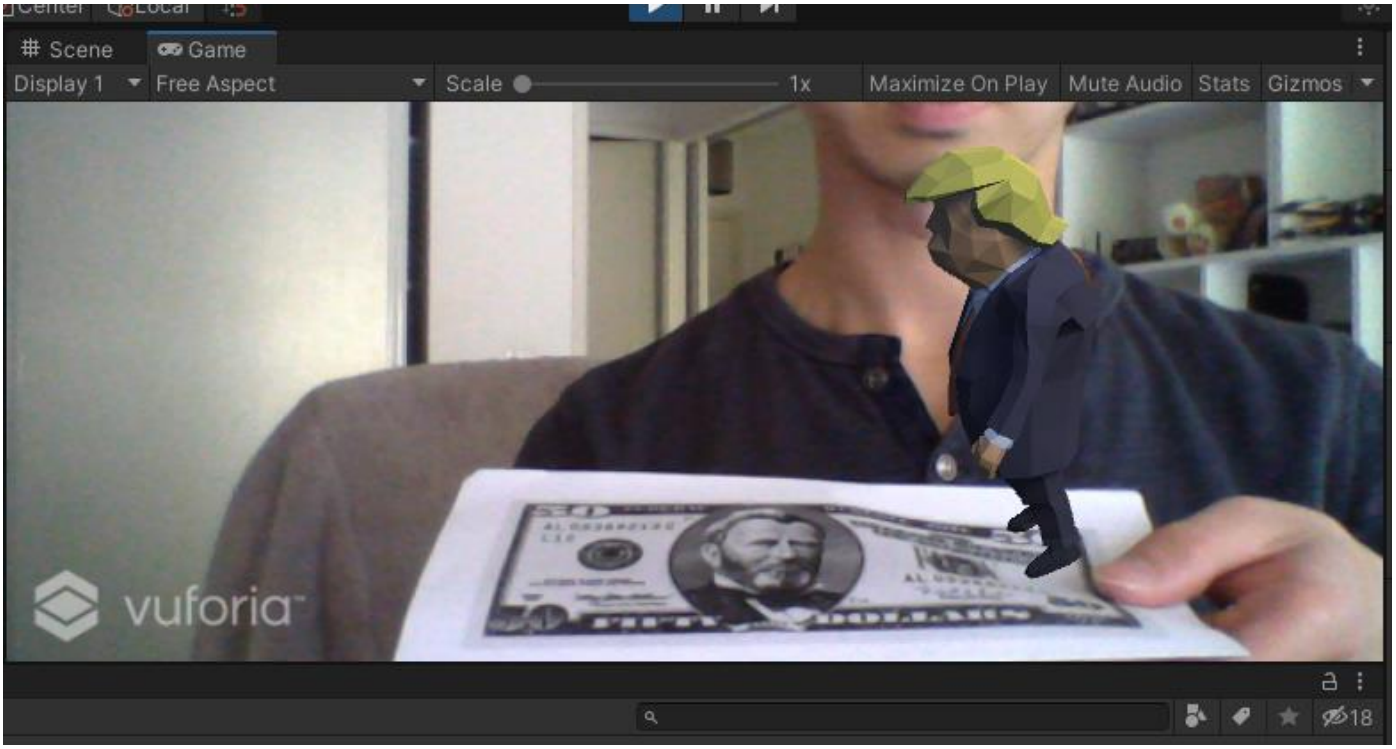
a. Partie 1 AR avec Vulforia

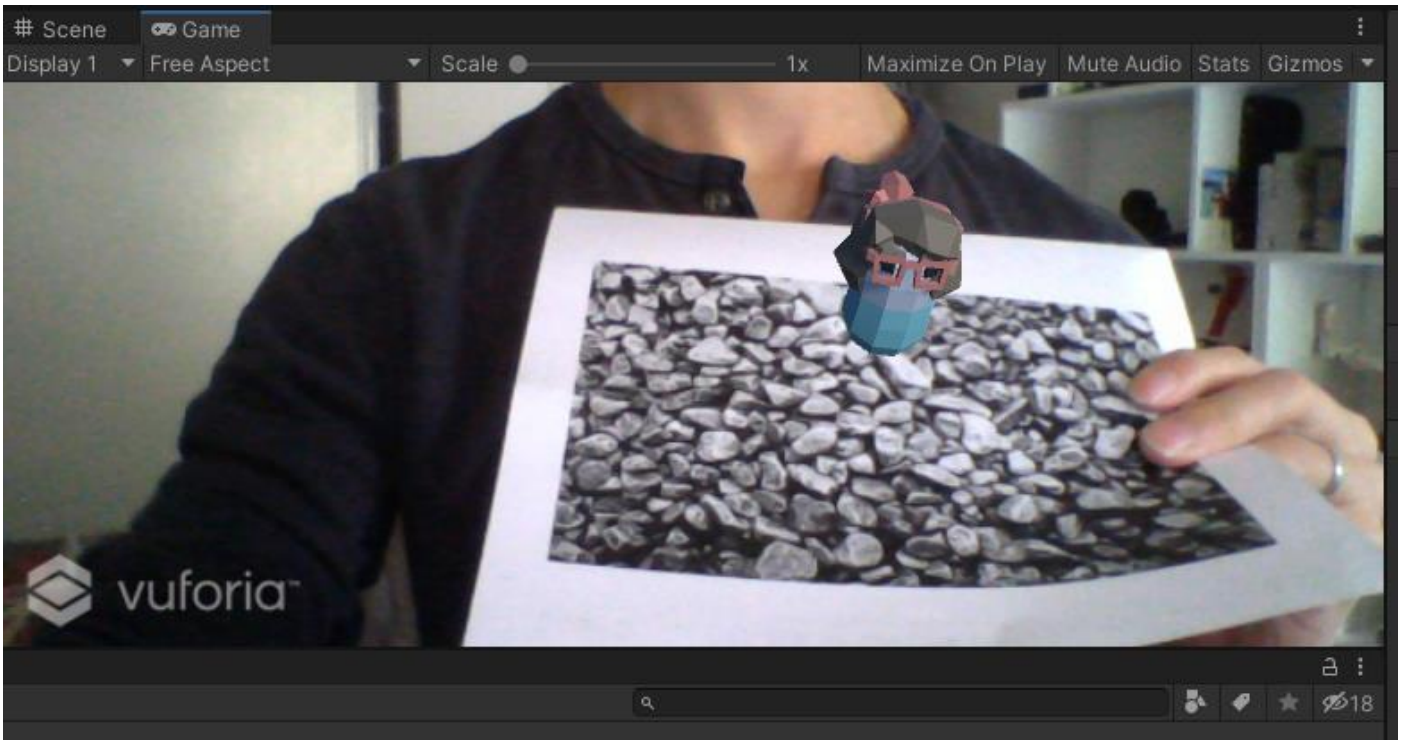
(1) Création de la scène

Comme demandé dans le sujet, la scène est simplement composé des models choisis et les objets avec les marqueurs (billet et grayscale).



Pour tester AR, j'ai imprimer les deux images (celle avec les marqueurs), j'ai le résultat comme les images suivantes :





Donc, Application est bien marché.

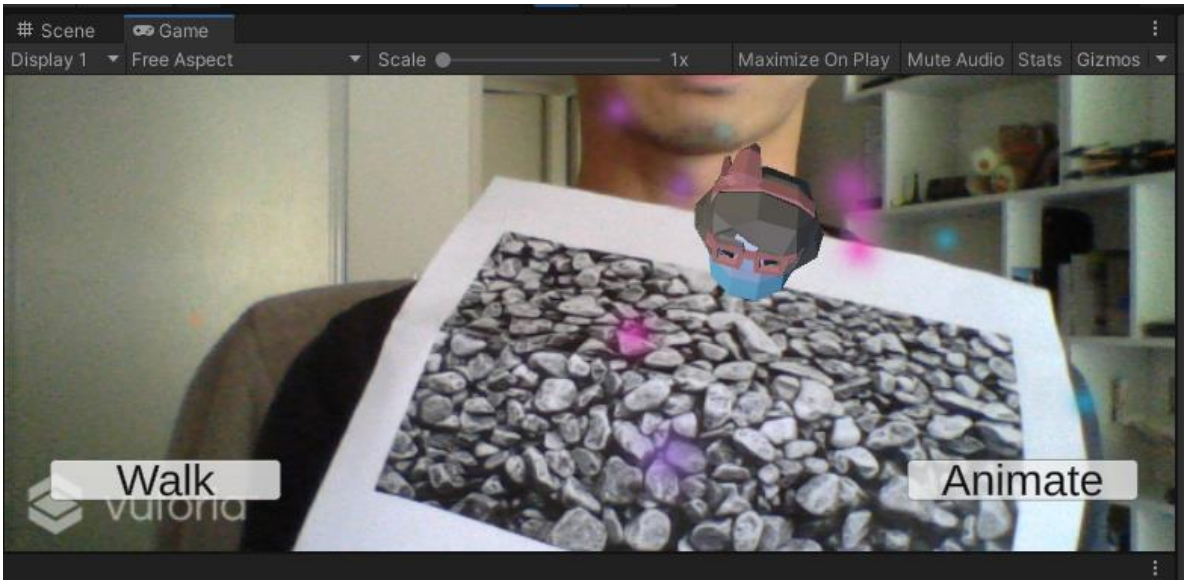
(2) Ajout des intéractions avec les éléments de canvas

Ensuite, pour réaliser un canvas avec les boutons qui vont intéragir avec l'utilisateur, j'ai ajouté simplement un canvas dans la scène et associer le script ci-dessous.

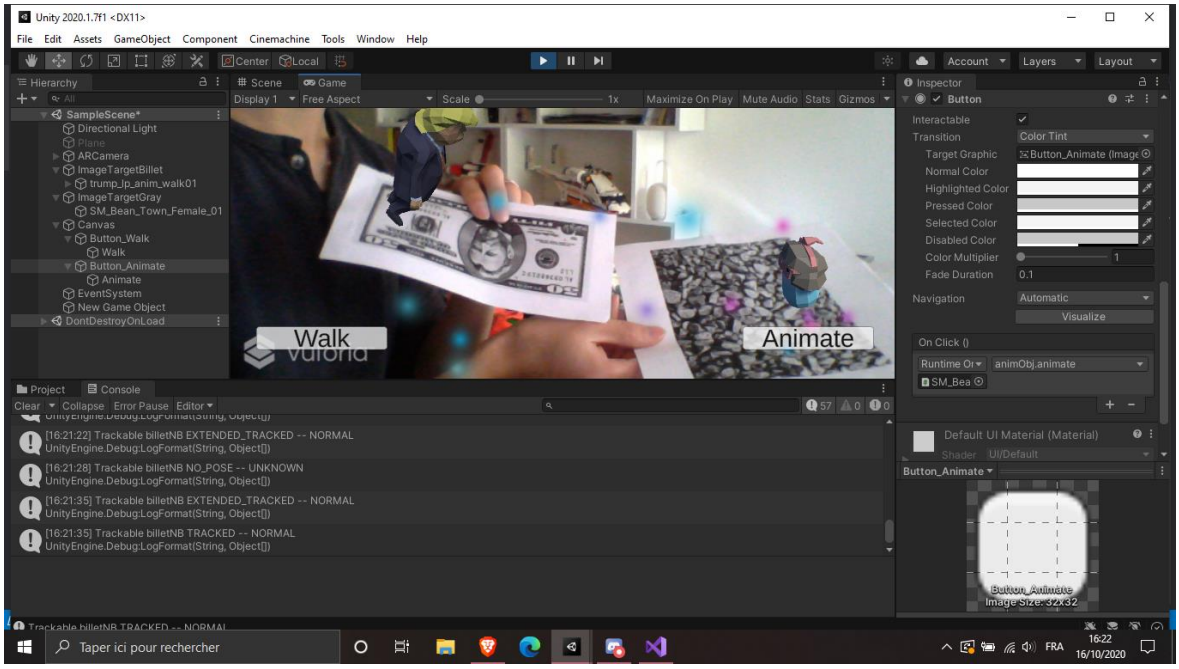
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class animObj : MonoBehaviour
{
    private Animation anim;
    public string animName;
    public ParticleSystem ps;
```

```
    void Start()
    {
        anim = GetComponent<Animation>();
        ps = GetComponent<ParticleSystem>();
        if (ps != null) ps.Stop();

    }

    void Update()
    {
    }

    public void animate()
    {
        if(anim!=null) anim.Play(animName);
        if (ps != null) ps.Play();
    }
}
```

Pour le model « Trump » j'ai asscocié un bouton pour le faire marcher. Et l'autre model , un bouton pour lancer un système de particule comme montré l'image ci-dessous.



Donc les résultats finales sont comme l'image ci-dessous (avec deux models ensemble)

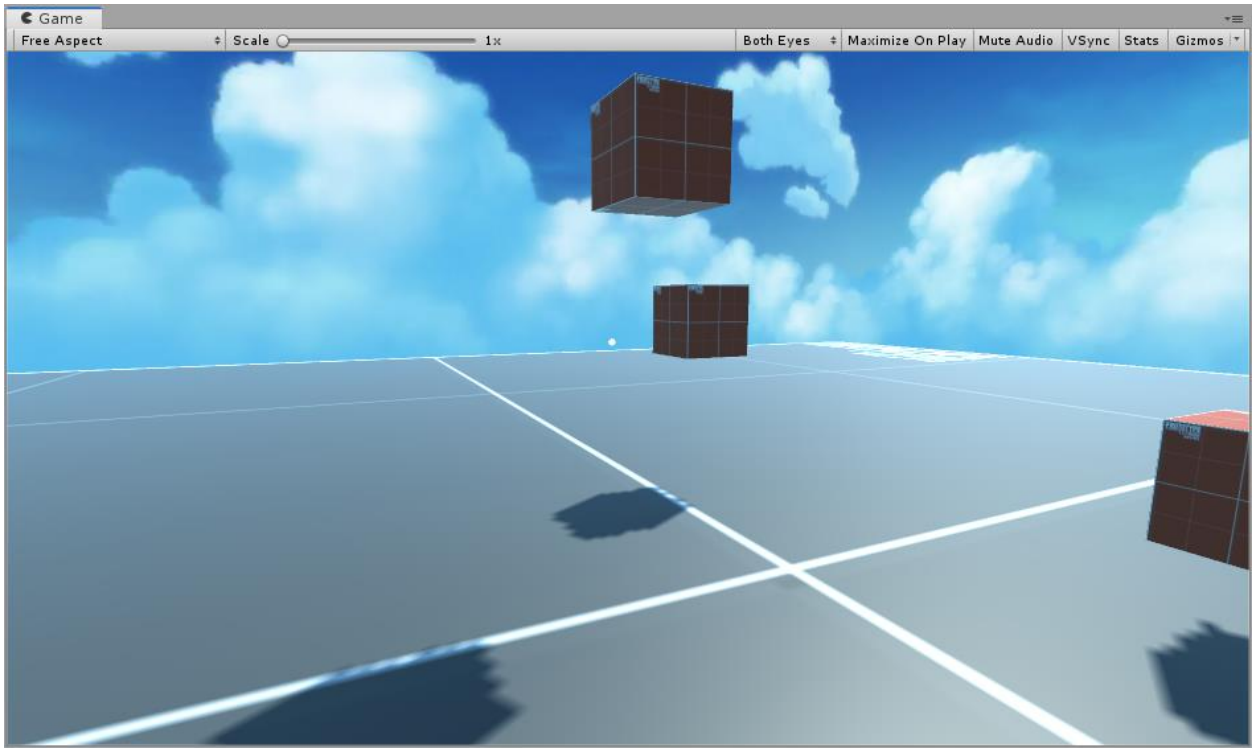


b. Partie 2 VR avec cardboard

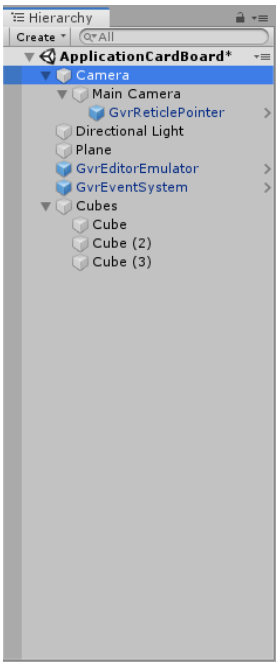
(1) Création de la scène

Je n'ai pas utilisé la scène de labyrinthe, pour ce tp, j'ai crée une nouvelle scène comme l'image ci-dessous. Dans cette scène, on a un plan pour le joueur marche dessus et un ensemble de cube pour faire tester l'interaction avec les commandes du joueur.

J'ai également ajouté un skybox et changé les matérials (des packages d'assetstore) des objets pour que ce soit plus joli.



(2) Utilisation des prefabs de Google



Comme demandé, j'ai ajouté les préfabs de GoogleVR comme montrée l'image ci-contre.

GvrReticlePointer consiste à configurer le pointeur du joueur pour réaliser l'intéraction. GvrEventSystem a pour but de gérer les évènements capturé par les pointeur du joueur. De plus, j'ai ajouté le script « Gvr Pointer Physics Raycaster » à la caméra pour la détection des objets pointés.

(3) Interaction avec la scène

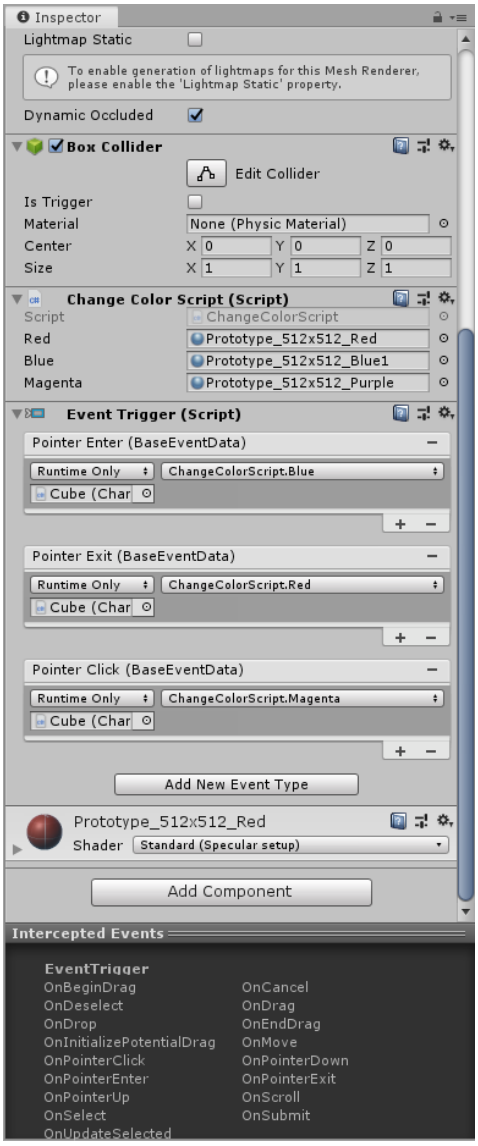
Intéraction possible :

Enter : Changement de couleur en Bleu

Exit : Changement de couleur en Rouge à nouveau

Click : Changement de couleur en Violet

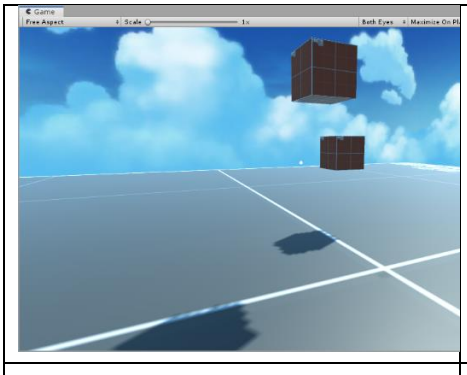
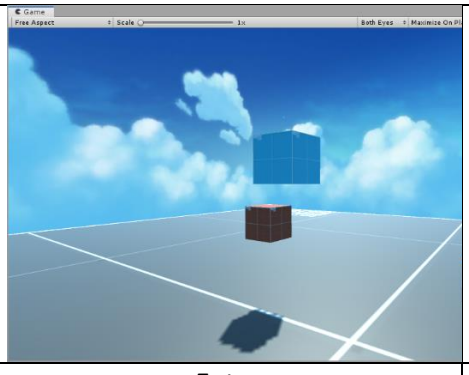
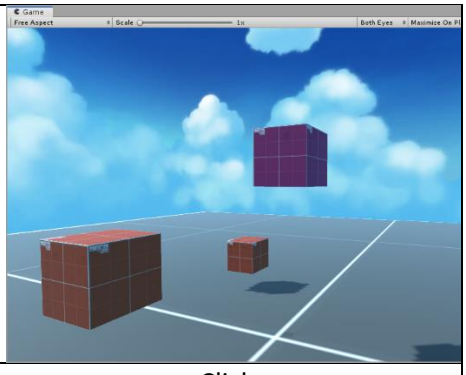
J'ai mis les matérials en prédéfinis dans l'inspector pour changer les matérials.



ChangeColorScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class ChangeColorScript : MonoBehaviour
{
    public Material red;
    public Material blue;
    public Material magenta;

    public void Red()
    {
        Debug.Log("red");
        GetComponent<MeshRenderer>().material = red;
    }

    public void Blue()
    {
        Debug.Log("blue");
        GetComponent<MeshRenderer>().material = blue;
    }

    public void Magenta()
    {
        Debug.Log("purple");
        GetComponent<MeshRenderer>().material = magenta;
    }
}
```

Les résultats sont comme les images ci-dessous :

		
	Enter	Click

(4) Avancement du joueur

Le joueur avance automatiquement avec une vitesse personalisable. Il peut choisir s'arreter quand il regarde sur le sol. Pour réaliser cela, j'ai utiliser Raycasting pour voir l'intersection de ray avec les objets de la scène. Si l'objet intersecté est le sol (dans mon projet est « Plane »), on met le boolean « walking » en false, donc le joueur s'arrete à avancer, sinon on remet à true.

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

public class PlayerController : MonoBehaviour
{
    public bool walking = true;
    private Vector3 spawnPoint;
    private Camera mainCamera;
    public float speed = 1.5f;

    void Start()
    {
        mainCamera = Camera.main;
        spawnPoint = transform.position;
    }

    void Update()
    {
        if (walking)
        {
            transform.position += speed * mainCamera.transform.forward * 0.5f * Time.deltaTime;
        }

        if (transform.position.y < -10f)
        {
            transform.position = spawnPoint;
        }

        Ray ray = Camera.main.ViewportPointToRay(new Vector3(0.5f, 0.5f, 0.0f));
        RaycastHit hit;

        if(Physics.Raycast(ray, out hit))
        {
            if (hit.collider.name.Contains("Plane"))
            {
                walking = false;
            }
            else
            {
                walking = true;
            }
        }
        else
        {
            walking = true;
        }
    }
}
```

Après la configuration de téléphone et Unity, l’application est bien marché sur mon téléphone et cardboard.