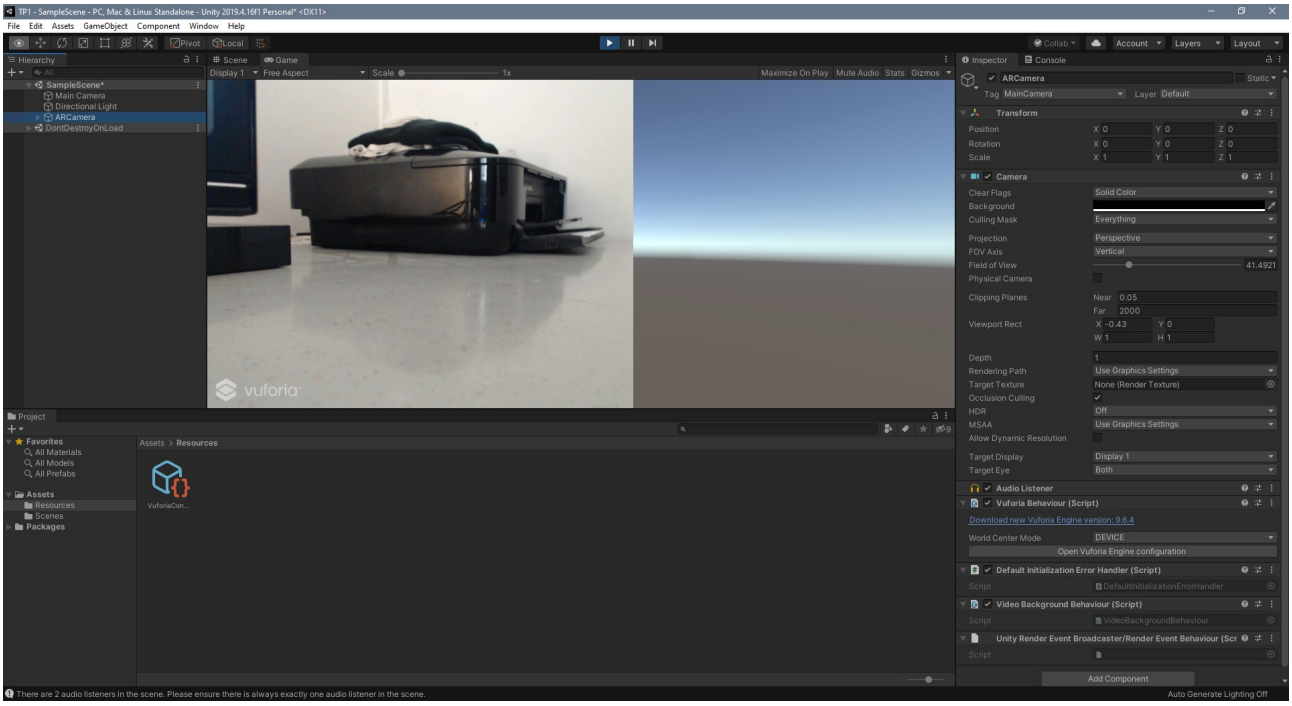
Rapport TP 1 – TIA

Brandon LARGEAU

Utilisation de Vuforia 9.6.4.

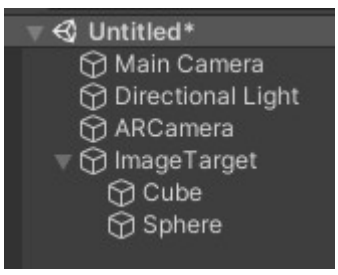
Exercice 1

Voici une capture d’écran avec la caméra qui tourne sur Unity.



Exercice 2

J’ai ajouté l’image “Oxygen” (je n’ai plus d’encre Magenta) avec un cube et une sphere.



Quand on bouge la caméra ou l’image trop rapidement, les objets disparaissent.

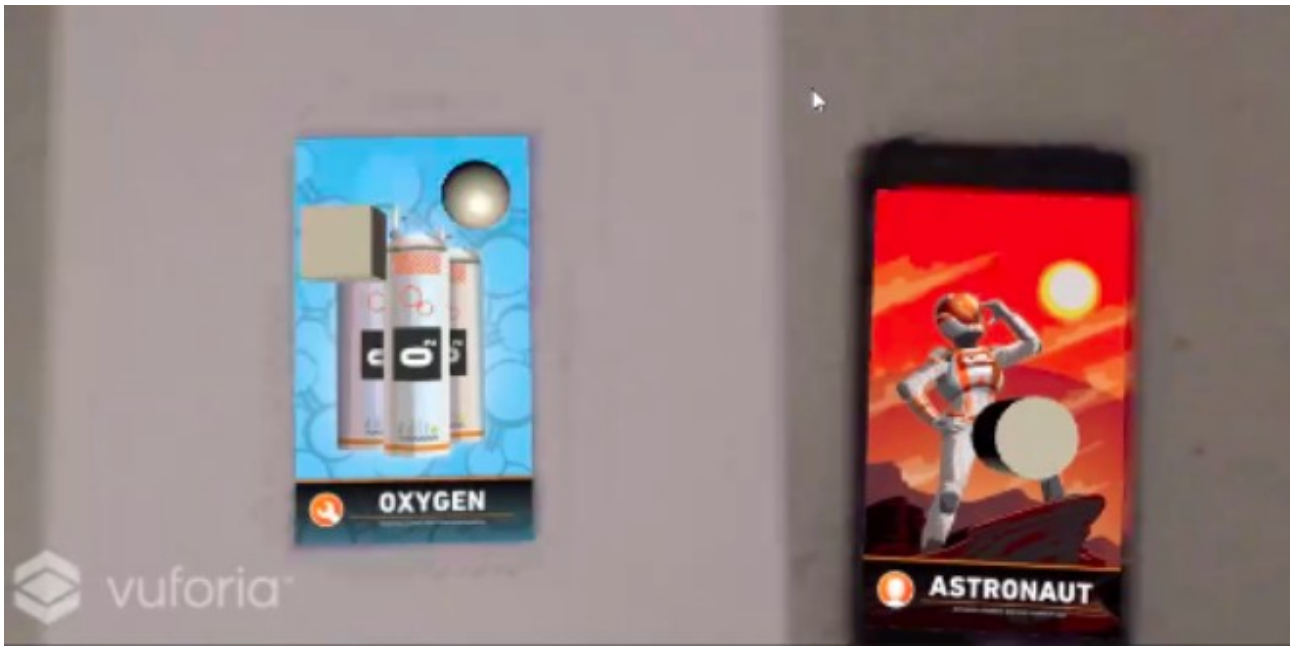
Quand “Turn Off Behaviour“ est désactivé, l’image modélisé apparaît sur la visualisation de la caméra

Si “extended tracking“ est activé les objets sont trackés même quand ils ne sont pas dans le champ de vision de la caméra.

Si je mets un rigidbody sur ma sphere ou mon cube, il n’apparaît plus sur la visualisation sur la caméra, je pense qu’il tombe. Par contre si j’ajoute aussi un rigidbody sur l’image, là tout est bon.

Exercice 3

J’ai donc les 2 images une imprimé et l’autre sur mon téléphone.



En DEVICE l’image sur la scene bouge un peu alors qu’en FIRST_TARGET l’image est statique et donc parallèle.

Exercice 4

Il y a un enum qui défini le type de tracking. On a ensuite les évènements à trigger quand une cible est trouvé ou perdu.

Sur le Start() il y est défini une sorte de listener sur le status du tracking. Le OnDestroy() supprime ce listener.

Quand l'image target apparaît ou disparaît OnTrackableStatusChanged() est trigger et fait appel à HandleTrackableStatusChanged() qui gère le rendering des objet en fonction de s'ils sont visible ou non sur la scène.

OnTrackingFound() et OnTrackingLost() active ou non les différents composants de type renderer, collider et canvas de l'objet.

Exercice 5

```
public class CustomTrackableEventHandler : MonoBehaviour
```

```
private int cycliqueObjectNumberRenderer = 0;
private int cycliqueObjectNumberCollider = 0;
private int cycliqueObjectNumberCanvas = 0;
```

```
1 référence
protected virtual void OnTrackingFound()
{
    if (mTrackableBehaviour)
    {
        var rendererComponents = mTrackableBehaviour.GetComponentsInChildren<Renderer>(true);
        var colliderComponents = mTrackableBehaviour.GetComponentsInChildren<Collider>(true);
        var canvasComponents = mTrackableBehaviour.GetComponentsInChildren<Canvas>(true);

        if (rendererComponents.Length > 0)
        {
            rendererComponents[cycliqueObjectNumberRenderer].enabled = true;
            cycliqueObjectNumberRenderer = (cycliqueObjectNumberRenderer + 1) % rendererComponents.Length;
        }

        if (colliderComponents.Length > 0)
        {
            colliderComponents[cycliqueObjectNumberCollider].enabled = true;
            cycliqueObjectNumberCollider = (cycliqueObjectNumberRenderer + 1) % colliderComponents.Length;
        }

        if (canvasComponents.Length > 0)
        {
            canvasComponents[cycliqueObjectNumberCanvas].enabled = true;
            cycliqueObjectNumberCanvas = (cycliqueObjectNumberRenderer + 1) % canvasComponents.Length;
        }

        /*
        // Enable rendering:
        foreach (var component in rendererComponents)
            component.enabled = true;

        // Enable colliders:
        foreach (var component in colliderComponents)
            component.enabled = true;

        // Enable canvas':
        foreach (var component in canvasComponents)
            component.enabled = true;
        */
    }

    if (OnTargetFound != null)
        OnTargetFound.Invoke();
}
```

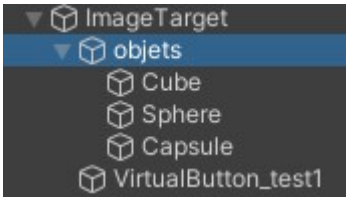
Chaque fois que l'imagetarget est détectée, on va affiché un objet en commençant pas le 0 et ceci que si le composant est disponible (renderer, collider et canvas).

On fait tourner la valeur des variables en avec un modula la taille de la liste des composants de l'objet.

Exercice 6

Avant d'appuyer sur le bouton, tous les objets sont affichés. Quand on appuie u seul est affiché ainsi de suite. Le problème qu'on peut rencontrer c'est si on gère mal les objets, on risque de désactiver le bouton vu qu'il fait parti des enfants de l'image target et qu'il a un Mesh.

Pour régler ce problème on a un empty object qui contient les objets que l'on veut faire disparaître et apparaître de façon cyclique quand on clique sur le bouton.

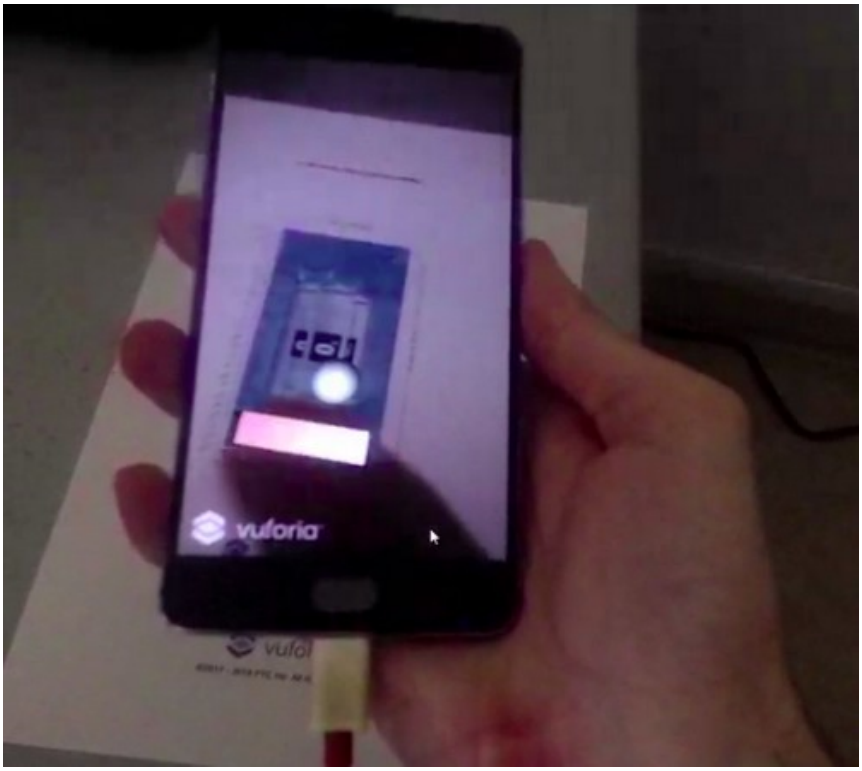


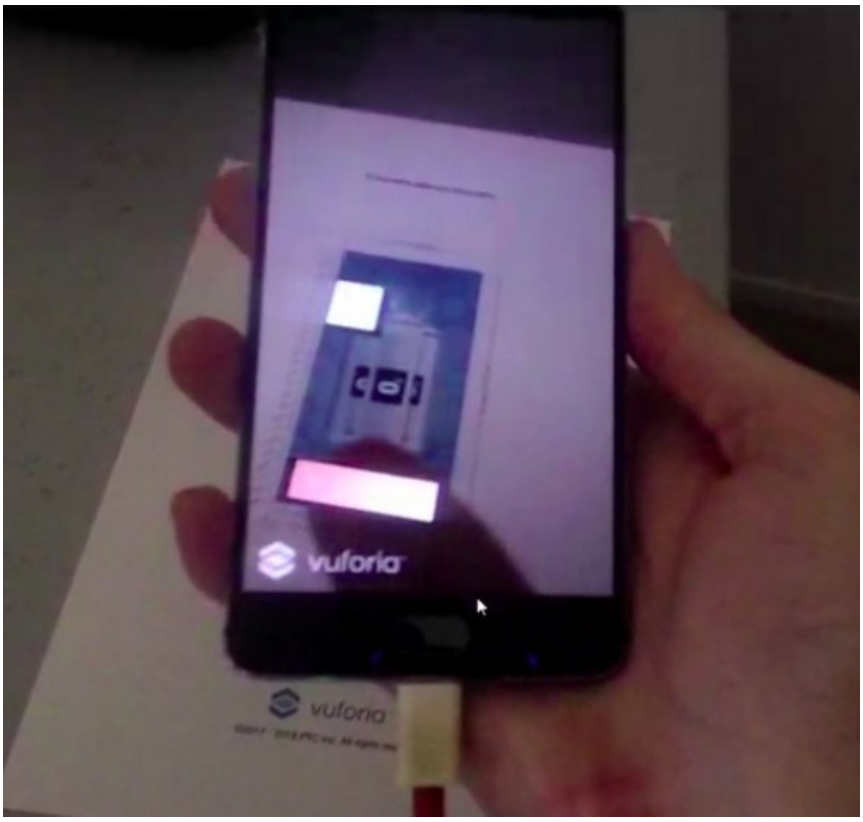

```
public GameObject objets;

protected VirtualButtonBehaviour[] virtualButtonBehaviours;

private int cycliqueObjectNumberRenderer = 0;
private int cycliqueObjectNumberCollider = 0;
private int cycliqueObjectNumberCanvas = 0;
```

```
2 références
public void OnButtonPressed(VirtualButtonBehaviour vb)
{
    var rendererComponents = objets.GetComponentsInChildren<Renderer>(true);
    var colliderComponents = objets.GetComponentsInChildren<Collider>(true);
    var canvasComponents = objets.GetComponentsInChildren<Canvas>(true);

    foreach (var component in rendererComponents)
        component.enabled = false;

    foreach (var component in colliderComponents)
        component.enabled = false;

    foreach (var component in canvasComponents)
        component.enabled = false;

    if (rendererComponents.Length > 0)
    {
        rendererComponents[cycliqueObjectNumberRenderer].enabled = true;
        cycliqueObjectNumberRenderer = (cycliqueObjectNumberRenderer + 1) % rendererComponents.Length;
    }

    if (colliderComponents.Length > 0)
    {
        colliderComponents[cycliqueObjectNumberCollider].enabled = true;
        cycliqueObjectNumberCollider = (cycliqueObjectNumberRenderer + 1) % colliderComponents.Length;
    }

    if (canvasComponents.Length > 0)
    {
        canvasComponents[cycliqueObjectNumberCanvas].enabled = true;
        cycliqueObjectNumberCanvas = (cycliqueObjectNumberRenderer + 1) % canvasComponents.Length;
    }

    Debug.Log("OnButtonPressed::" + vb.VirtualButtonName);
}
```

Exercice 7 : Android Deployment

Tout fonctionne correctement sur mon Oneplus 3 sous Android 9 (LineageOS).

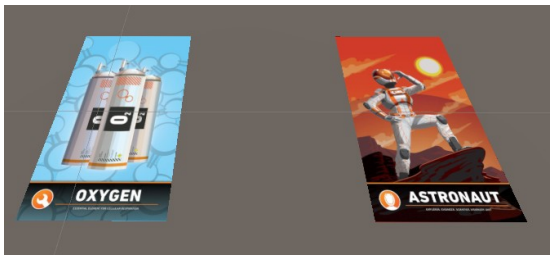


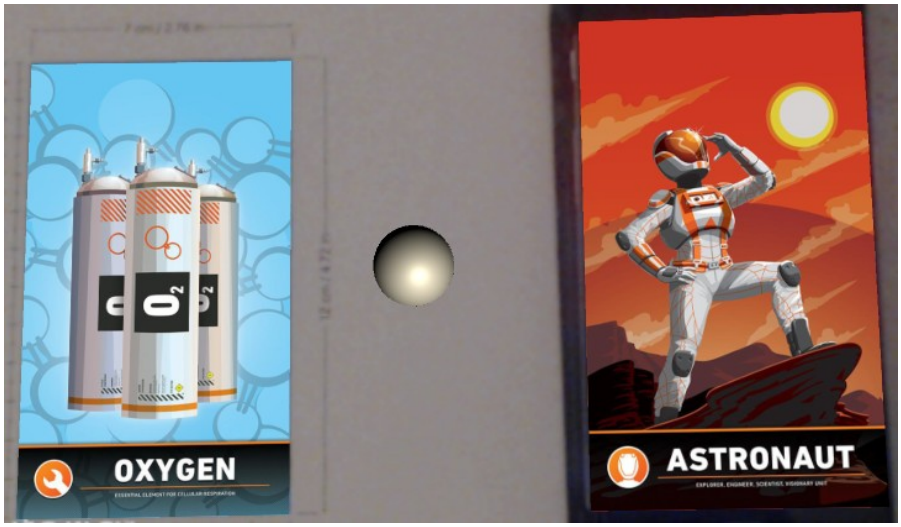


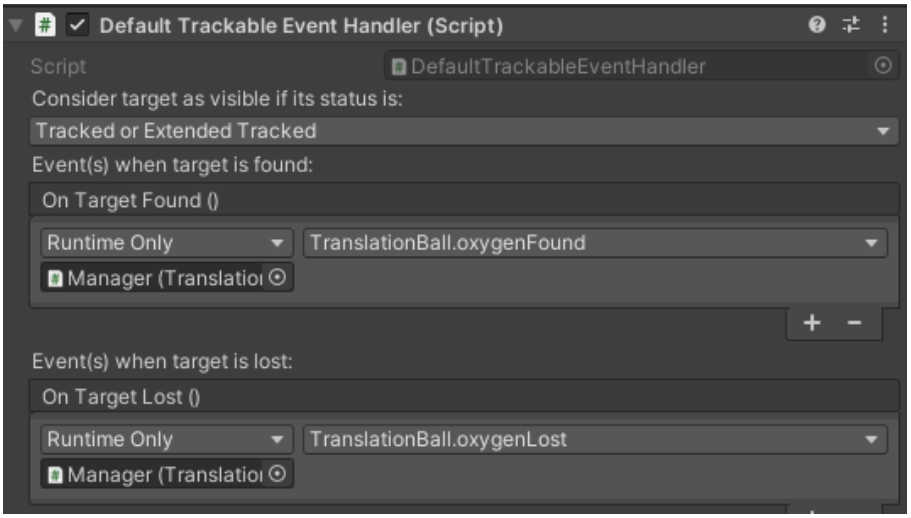
Additional Exercises

Exercice 8

On a créé un script qui s'occupe de générer une sphere sur l'imageTarget astronaute et qui fait son déplacement vers l'imageTarget oxygen. Pour cela on a créé un GameObject « Manager » qui prend en entré les deux imageTarget (ce qui nous permet de récupérer les transforms). On spawn une fois la sphere dans le update lorsque les 2 imageTarget sont visibles sur la caméra. Pour vérifier cela on a ajouter un Event sur le « On Target Found () » (pareil pour le Lost) de chaque imageTarget qui fait appelle à une fonction « astronautFound() » (ou la même pour oxygen et pareil pour Lost) du script attaché au Manager ce qui active un boolean sur le script. Quand les 2 booleans sont activés fait déplacer la boule d'une image à l'autre.





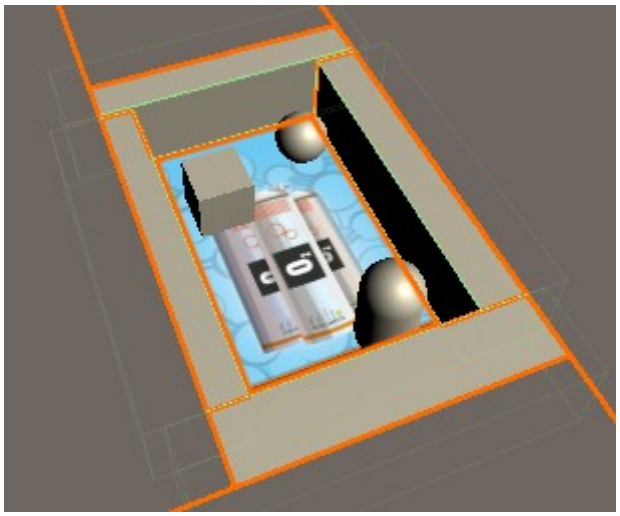


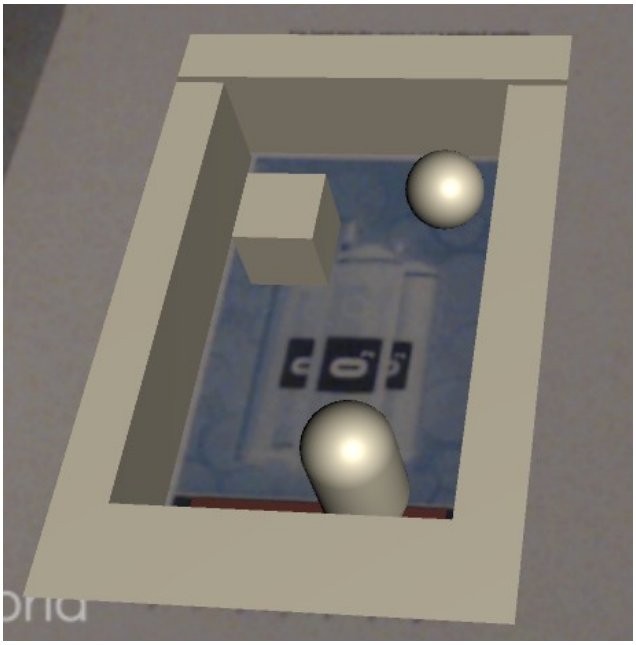
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Script Unity | 0 références
public class TranslationBall : MonoBehaviour
{
    public GameObject oxygenImageTarget;
    public GameObject astronautImageTarget;

    private GameObject theBall;
    private bool bAstronautFound = false;
    private bool bOxygenFound = false;
    private bool firstTime = true;

    // Start is called before the first frame update
    Message Unity | 0 références
    void Start()
    {

    }

    // Update is called once per frame
    Message Unity | 0 références
    void Update()
    {
        if (bAstronautFound && bOxygenFound)
        {
            if (firstTime)
            {
                firstTime = false;

                theBall = GameObject.CreatePrimitive(PrimitiveType.Sphere);

                theBall.transform.localScale = new Vector3(0.02f, 0.02f, 0.02f);
                theBall.transform.position = astronautImageTarget.transform.position + new Vector3(0.0f, 0.02f, 0.0f);
            }

            theBall.transform.position = Vector3.MoveTowards(theBall.transform.position, oxygenImageTarget.transform.position, 0.05f * Time.deltaTime);
        }
    }

    0 références
    public void astronautFound()
    {
        bAstronautFound = true;
    }

    0 références
    public void astronautLost()
    {
        bAstronautFound = false;
    }
```

```
    0 références
    public void oxygenFound()
    {
        bOxygenFound = true;
    }

    0 références
    public void oxygenLost()
    {
        bOxygenFound = false;
    }
}
```

Exercice 9

On utilise des plane avec un material « VR/SpatialMapping/Occlusion » pour faire l’effet de profondeur.





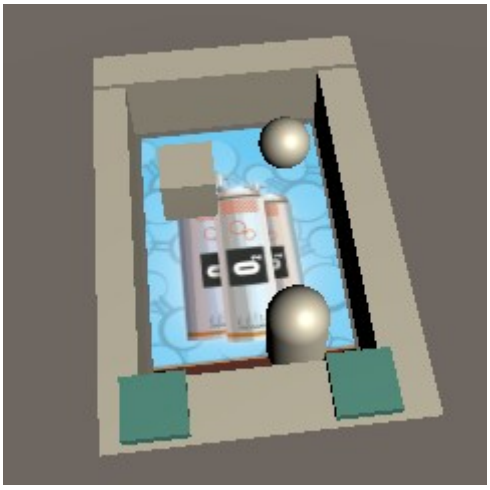
Exercice 10

L’avantage des Virtual Button c’est qu’on peut interagir directement dans la vrai vie, pas besoin d’une surface tactile électronique (un téléphone, directement sur la simulation Unity, …). Le désavantage est sa fiabilité.

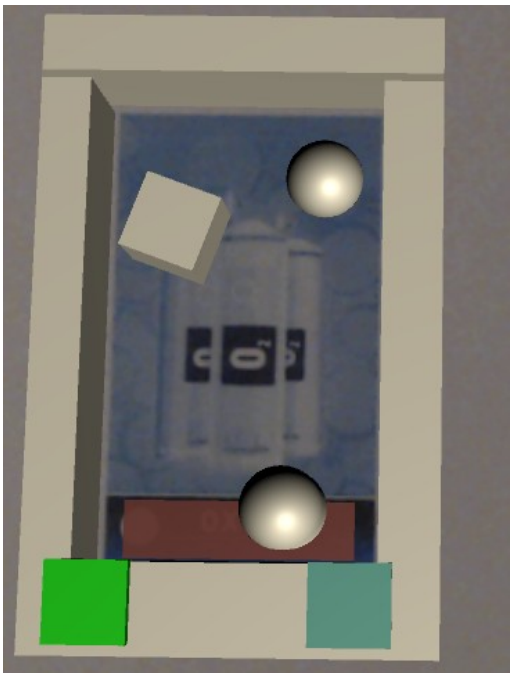
L’avantage des boutons directement sur Unity avec des triggers c’est la facilité de test.

On a 2 boutons, le bouton de gauche fait tourner l’objet vers la gauche et le bouton de droite fait tourner l’objet vers la droite.

On a un script qui a en entré l’objet à faire tourner. Le script est appliqué au 2 boutons et pour savoir quel bouton est appuyé on a créé un tag par bouton « button_rotate_left » et « button_rotate_right ». Quand un bouton est appuyé on change la couleur de son material et quand le bouton est relâché on remet sa couleur qu’il avait de base.



Quand j’appuie sur le bouton de gauche :



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

Script Unity | 0 références
public class RotateScript : MonoBehaviour
{
    public GameObject objectForRotation;
    private Renderer buttonRender;
    private Collider buttonCollider;
    private Color previousColor;

    private bool rotateLeft = false;
    private bool rotateRight = false;

    // Start is called before the first frame update
    Message Unity | 0 références
    void Start()
    {
        buttonRender = GetComponent<Renderer>();
        buttonCollider = GetComponent<Collider>();
    }

    // Update is called once per frame
    Message Unity | 0 références
    void Update()
    {
        if (rotateLeft)
        {
            objectForRotation.transform.Rotate(Vector3.up, Time.deltaTime * 10.0f, Space.Self);
        }

        if (rotateRight)
        {
            objectForRotation.transform.Rotate(Vector3.up, Time.deltaTime * -10.0f, Space.Self);
        }
    }

    Message Unity | 0 références
    void OnMouseDown()
    {
        previousColor = buttonRender.materials[0].color;
        buttonRender.materials[0].color = Color.green;

        if (buttonCollider.gameObject.tag == "button_rotate_left")
        {
            print("START left");
            rotateLeft = true;
        }
```

```
        }

        if (buttonCollider.gameObject.tag == "button_rotate_right")
        {
            print("START right");
            rotateRight = true;
        }
    }
    Message Unity | 0 références
    void OnMouseOver()
    {
    }
    Message Unity | 0 références
    void OnMouseUp()
    {
        buttonRender.materials[0].color = previousColor;

        if (buttonCollider.gameObject.tag == "button_rotate_left")
        {
            print("STOP left");
            rotateLeft = false;
        }

        if (buttonCollider.gameObject.tag == "button_rotate_right")
        {
            print("STOP right");
            rotateRight = false;
        }
    }
}
```