Title

AR Epilepsy Education App:

Learning about the brain with interactive characters.

Subtitle

Documentation for developers and users

Table of Contents

1. Introduction 3

1.1. Overview 3

1.2. Purpose and target audience 3

1.3. System Requirements 3

2. Application Functionality 3

2.1. Image Targets and Recognition 3

2.1.1. List of image targets 3

2.1.2. Triggering mechanism (how are objects rendered/scripts run) 7

2.1.3. Troubleshooting for target recognition issues 11

2.2. Characters and Interactions 12

2.2.1. Introduction of the characters 12

2.2.2. Dialogue structure and content. 12

2.3. Educational Content 14

2.3.1. Topics covered about the brain. 14

2.3.2. Source of information 15

3. Technical Specifications 15

3.1. Development platform and tools 15

3.2. Scripts and Logic 15

3.3. Importing Vuforia 24

1. Introduction

1.1. Overview

An easy-to-use application for learning basic information about the brain parts and their functionality. By using 3D – animation, this app provides visual presentation of the brain.

1.2. Purpose and target audience

The purpose of this application is to provide basic information about the brain to the children.

1.3. System Requirements

Should have at least Android 7.0 and 5 MP camera.

2. Application Functionality

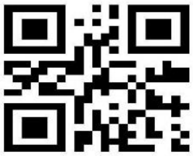
2.1. Image Targets and Recognition

2.1.1. List of image targets

- barcode0022_scaled_scaled

- barcode0033_scaled_scaled

- barcode0044_scaled_scaled



- barcode0055_scaled_scaled

- Image006_scaled

- Image007_scaled

- Image008_scaled

- Image009_scaled

- Image010_scaled

- Image011_scaled

- Image012_scaled

- Image013_scaled

- Image014_scaled

- Image015_scaled

- Image016_scaled

- Image017_scaled

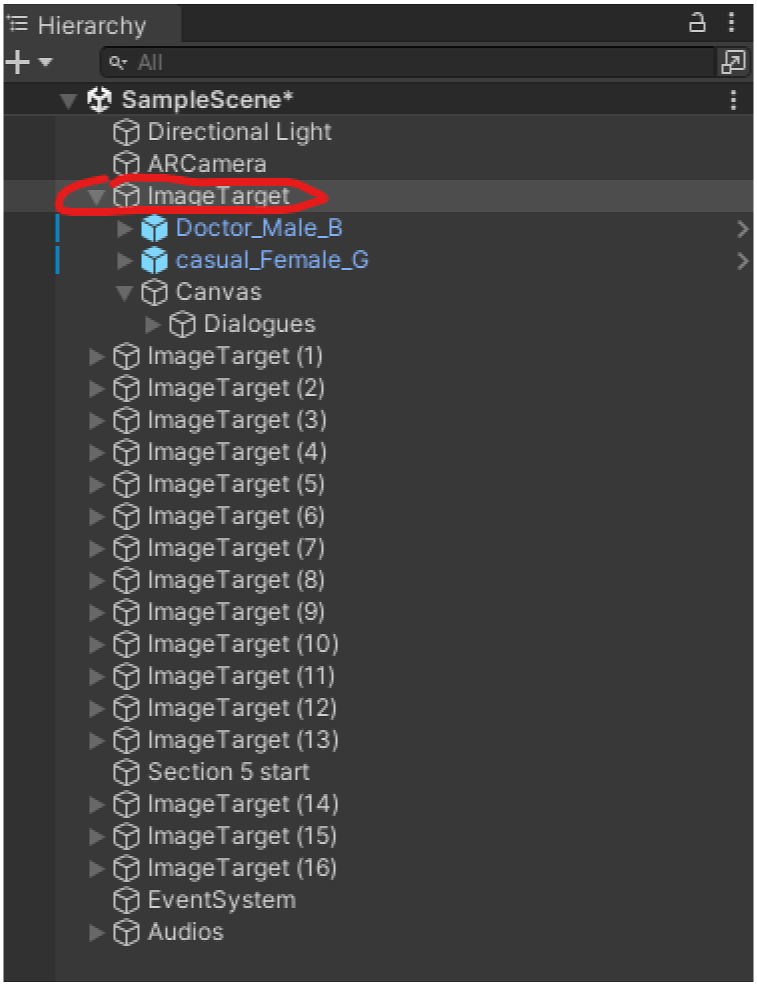
- Image018_scaled

2.1.2. Triggering mechanism (how are objects rendered/scripts run)

Physical images printed with unique patterns are recognized by the camera and act as virtual buttons. When the camera detects and tracks the image, the associated 3D object or animation is triggered. When the 3D object or animation is triggered, it executes the already selected method from a particular script.

Steps:

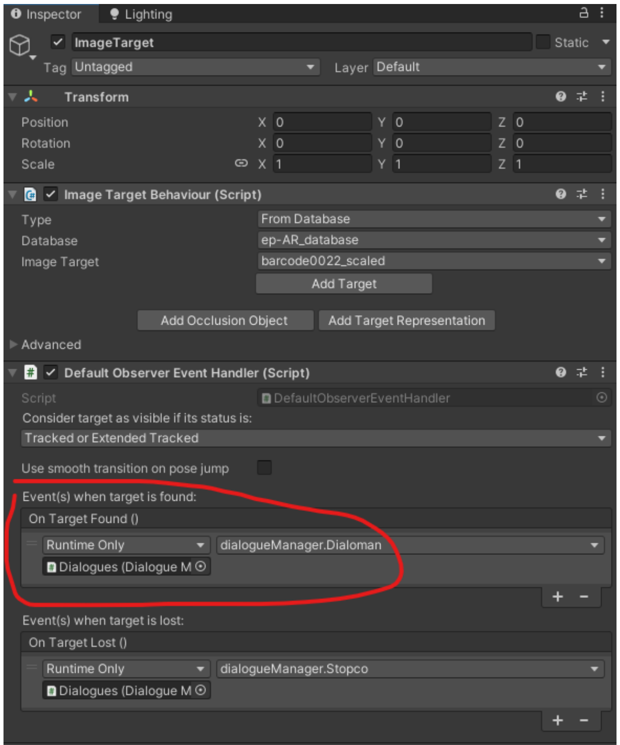
1. Select image target from the **hierarchy** window.



Hierarchy

```
SampleScene*
  Directional Light
  ARCamera
  ImageTarget
    Doctor_Male_B
    casual_Female_G
    Canvas
      Dialogues
  ImageTarget (1)
  ImageTarget (2)
  ImageTarget (3)
  ImageTarget (4)
  ImageTarget (5)
  ImageTarget (6)
  ImageTarget (7)
  ImageTarget (8)
  ImageTarget (9)
  ImageTarget (10)
  ImageTarget (11)
  ImageTarget (12)
  ImageTarget (13)
  Section 5 start
  ImageTarget (14)
  ImageTarget (15)
  ImageTarget (16)
  EventSystem
  Audios
```

2. On the **Inspector** window, there is the functionality to be set when the image target is found.



ImageTarget

Tag Untagged | Layer Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

Image Target Behaviour (Script)

Type: From Database
Database: ep-AR_database
Image Target: barcode0022_scaled

Default Observer Event Handler (Script)

Script: DefaultObserverEventHandler

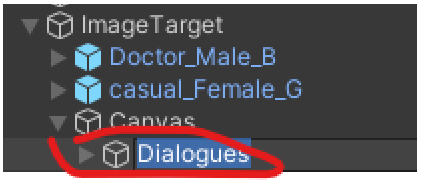
Consider target as visible if its status is: Tracked or Extended Tracked

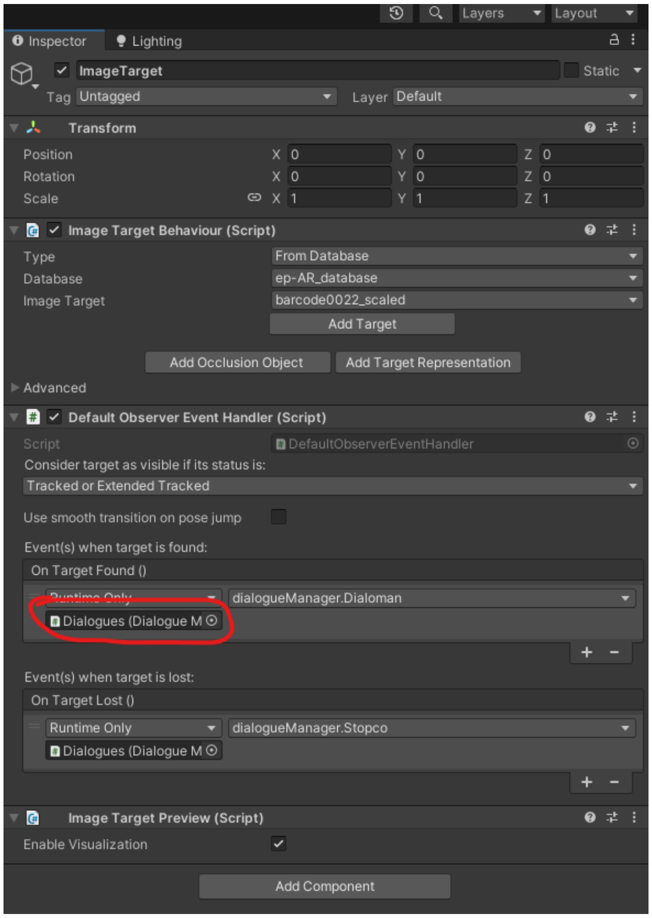
Use smooth transition on pose jump ☐

Event(s) when target is found:
On Target Found ()
Runtime Only | dialogueManager.Dialoman
Dialogues (Dialogue M

Event(s) when target is lost:
On Target Lost ()
Runtime Only | dialogueManager.Stopco
Dialogues (Dialogue M

3. Select **Dialogue** object from **hierarchy** window to the **On Target Found()** in Inspector window and drop it in under the object area below “Runtime Only” option menu





☑ ImageTarget □ Static

Tag Untagged Layer Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

☑ **Image Target Behaviour (Script)**

Type From Database

Database ep-AR_database

Image Target barcode0022_scaled

☑ **Default Observer Event Handler (Script)**

Script DefaultObserverEventHandler

Consider target as visible if its status is:
Tracked or Extended Tracked

Use smooth transition on pose jump □

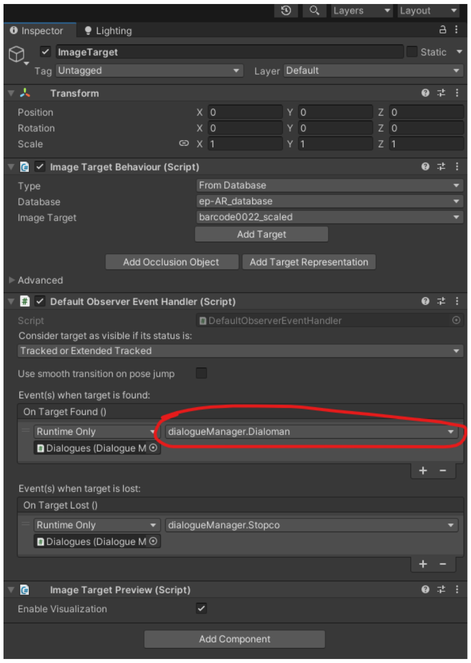
Event(s) when target is found:
On Target Found ()
Runtime Only dialogueManager.Dialoman
Dialogues (Dialogue M

Event(s) when target is lost:
On Target Lost ()
Runtime Only dialogueManager.Stopco
Dialogues (Dialogue M

Image Target Preview (Script)

Enable Visualization ☑

4. Next, click on this menu button and choose your required script, then select the method/function in that script which you want to run.



☑ ImageTarget □ Static
Tag Untagged Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

☑ Image Target Behaviour (Script)
Type From Database
Database ep-AR_database
Image Target barcode0022_scaled

☑ Default Observer Event Handler (Script)
Script DefaultObserverEventHandler
Consider target as visible if its status is:
Tracked or Extended Tracked
Use smooth transition on pose jump □
Event(s) when target is found:
On Target Found ()
Runtime Only dialogueManager.Dialoman
Dialogues (Dialogue M
Event(s) when target is lost:
On Target Lost ()
Runtime Only dialogueManager.Stopco
Dialogues (Dialogue M

Image Target Preview (Script)
Enable Visualization ☑

The same goes for the **On Target Lost ()** function.

2.1.3. Troubleshooting for target recognition issues

When your phone fails to detect the image target, then make sure that your camera lens is clean, if that doesn't fix the problem then restart the app. This will probably fix the problem. If the issue persists or the incorrect target is detected, then change the image target.

2.2. Characters and Interactions

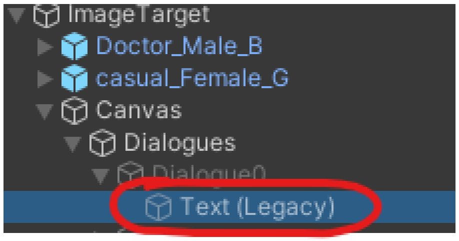
2.2.1. Introduction of the characters

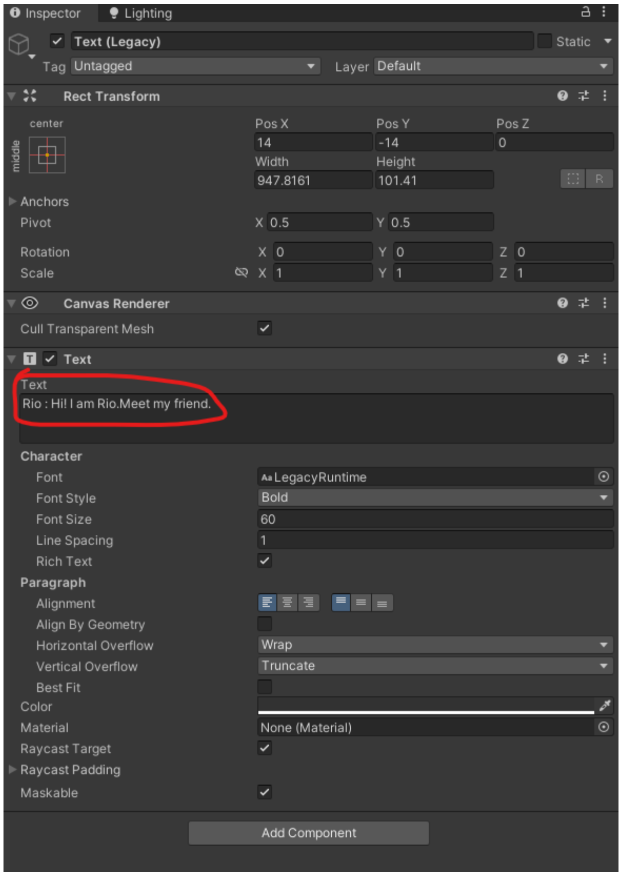
There are two characters, a girl named **Tina** and a doctor named **Rio**. Tina asks questions about human brain and Rio answers them in an understandable way.

Both characters have walking, running and other animations which are executed randomly. But they remain as same place.

2.2.2. Dialogue structure and content.

The dialogues are included as a **3D-Object** in which the text component is included.





Text (Legacy)

Tag Untagged Layer Default

Rect Transform

center middle

	Pos X	Pos Y	Pos Z
	14	-14	0
	Width	Height	
	947.8161	101.41	

Anchors

Pivot X 0.5 Y 0.5

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Canvas Renderer

Cull Transparent Mesh ✓

Text

Text

Rio : Hi! I am Rio.Meet my friend.

Character

Font LegacyRuntime

Font Style Bold

Font Size 60

Line Spacing 1

Rich Text ✓

Paragraph

Alignment

Align By Geometry □

Horizontal Overflow Wrap

Vertical Overflow Truncate

Best Fit □

Color

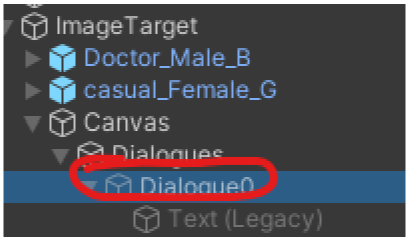
Material None (Material)

Raycast Target ✓

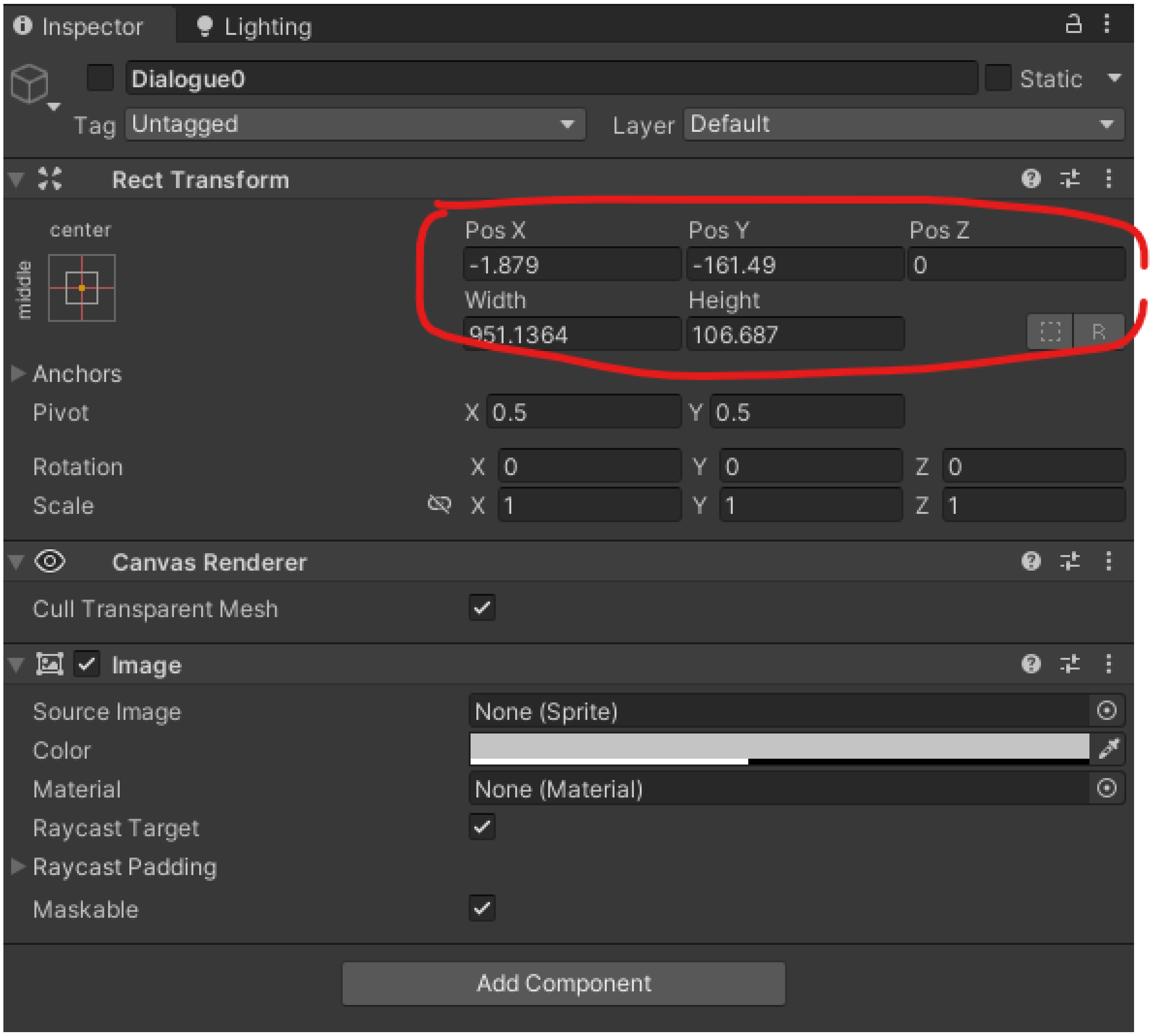
Raycast Padding

Maskable ✓

The orientation of the captions is controlled by the position and scale of this object



ImageTarget
Doctor_Male_B
casual_Female_G
Canvas
Dialogues
Dialogue0
Text (Legacy)



Dialogue0

Tag Untagged

Layer Default

Pos X -1.879

Pos Y -161.49

Pos Z 0

Width 951.1364

Height 106.687

Pivot X 0.5 Y 0.5

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Cull Transparent Mesh ☑

Source Image None (Sprite)

Color

Material None (Material)

Raycast Target ☑

Maskable ☑

2.3. Educational Content

2.3.1. Topics covered about the brain.

The basic knowledge about human brain is taught. The names of the basic parts of the brain were mentioned and their functionality is also discussed.

2.3.2. Source of information

The information has been collected mainly from the **National Institute of Neurological Disorders and Stroke**.

https://www.ninds.nih.gov/health-information/public-education/brain-basics/brain-basics-know-your-brain

3. Technical Specifications

3.1. Development platform and tools

This application has been developed in **Unity** engine with the help of **Vuforia engine**. All the characters and brain model are made with unity while the AR features and image target detection features are implemented using Vuforia engine.

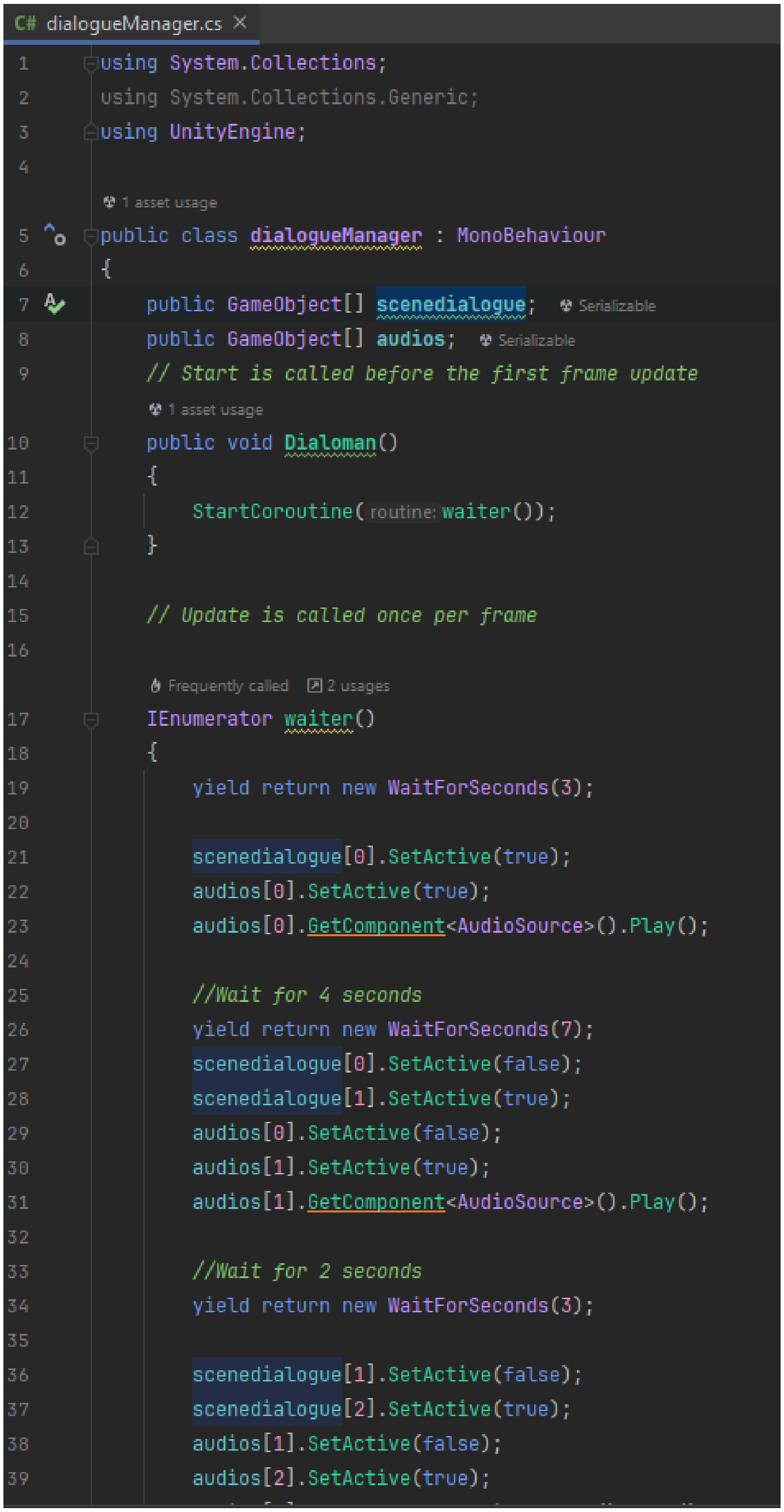
3.2. Scripts and Logic

The basic requirement was to make the characters talk, display subtitles and highlight different parts of the brain.

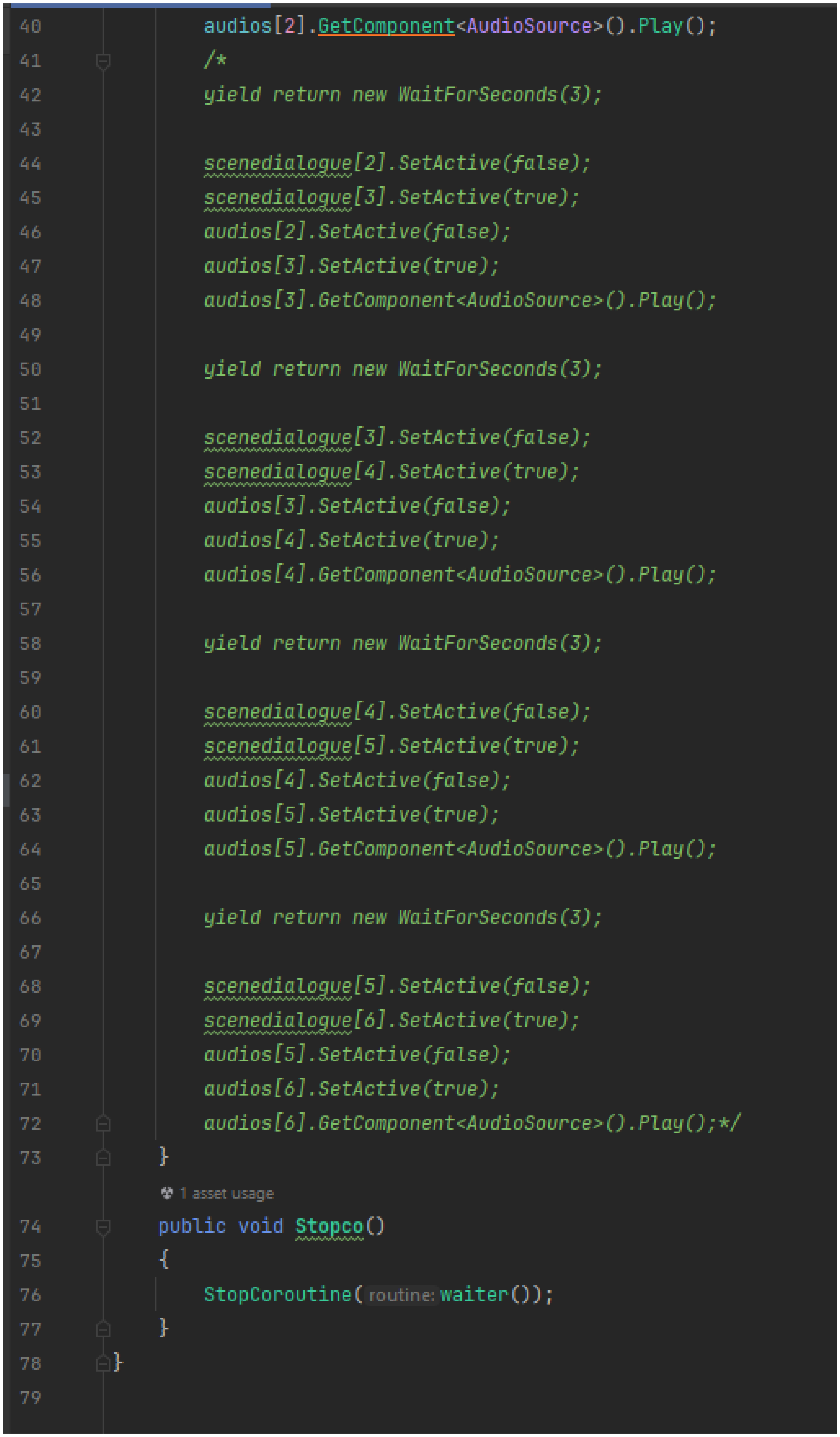
At first, an object named **Dialogues** is made as child object of image target, then a script named **dialogueManager.cs** is attached to it, the dialogueManager.cs has two arrays, One array named as **scenedialogue** stores dialogues while other array named as **audios** stores audios of the characters.

The dialogues in **scenedialogue** and **audios** arrays is put as an object.

The script works in such a way that the second dialogue of the characters is played after some delay, this delay is implemented using **yield** function which pauses all the processes for a certain period of specified time.



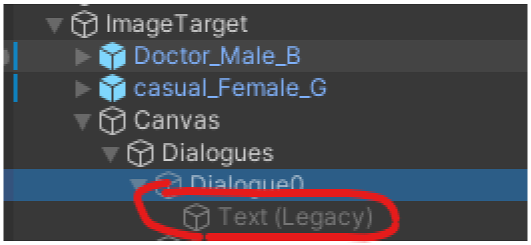
```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class dialogueManager : MonoBehaviour
{
    public GameObject[] scenedialogue;
    public GameObject[] audios;
    // Start is called before the first frame update
    public void Dialoman()
    {
        StartCoroutine( routine: waiter());
    }

    // Update is called once per frame

    IEnumerator waiter()
    {
        yield return new WaitForSeconds(3);

        scenedialogue[0].SetActive(true);
        audios[0].SetActive(true);
        audios[0].GetComponent<AudioSource>().Play();

        //Wait for 4 seconds
        yield return new WaitForSeconds(7);
        scenedialogue[0].SetActive(false);
        scenedialogue[1].SetActive(true);
        audios[0].SetActive(false);
        audios[1].SetActive(true);
        audios[1].GetComponent<AudioSource>().Play();

        //Wait for 2 seconds
        yield return new WaitForSeconds(3);

        scenedialogue[1].SetActive(false);
        scenedialogue[2].SetActive(true);
        audios[1].SetActive(false);
        audios[2].SetActive(true);
```

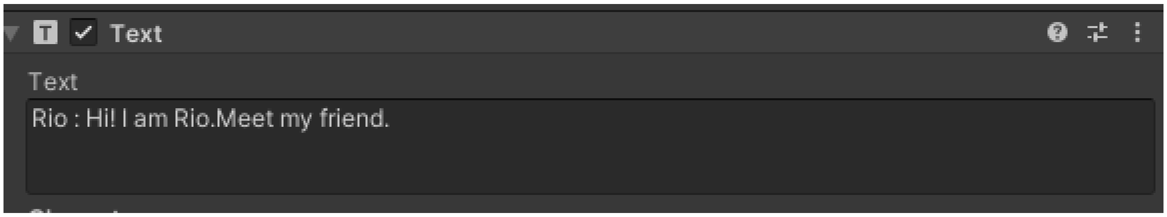


```
            audios[2].GetComponent<AudioSource>().Play();
            /*
            yield return new WaitForSeconds(3);

            scenedialogue[2].SetActive(false);
            scenedialogue[3].SetActive(true);
            audios[2].SetActive(false);
            audios[3].SetActive(true);
            audios[3].GetComponent<AudioSource>().Play();

            yield return new WaitForSeconds(3);

            scenedialogue[3].SetActive(false);
            scenedialogue[4].SetActive(true);
            audios[3].SetActive(false);
            audios[4].SetActive(true);
            audios[4].GetComponent<AudioSource>().Play();

            yield return new WaitForSeconds(3);

            scenedialogue[4].SetActive(false);
            scenedialogue[5].SetActive(true);
            audios[4].SetActive(false);
            audios[5].SetActive(true);
            audios[5].GetComponent<AudioSource>().Play();

            yield return new WaitForSeconds(3);

            scenedialogue[5].SetActive(false);
            scenedialogue[6].SetActive(true);
            audios[5].SetActive(false);
            audios[6].SetActive(true);
            audios[6].GetComponent<AudioSource>().Play();*/
        }
        1 asset usage
        public void Stopco()
        {
            StopCoroutine( routine: waiter());
        }
    }

```

The commented code is written there in case they are needed, but it is a better approach to use loops for it.

After that the **text(legacy)** object is placed in which the subtitles are written





Text

Rio : Hi! I am Rio.Meet my friend.

Now, lets see the part where brain comes in,

In this case, there are some times when we needed different parts of the brain to be separated in order to show them clearly.

This was done by using script named **Two_People_Dialogues.cs** .
This script is similar to **dialogueManager.cs** but it has additional controls for brain model's appearance.

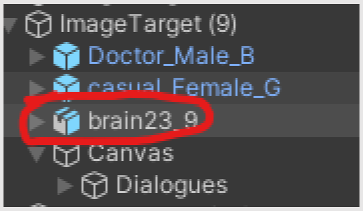
The only useable script used with **Two_People_Dialogues.cs** is **ZoomIn.cs** script.

This **zoomIn.cs** script helps in increasing the size of the brain to show it more clearly. It also takes the brain part's number (for example in this case the brain has 5 parts, and if want to take out part 2) then we drag the brain part and put it in the array named as **brainPartToTakeOut**.

⚠ Important

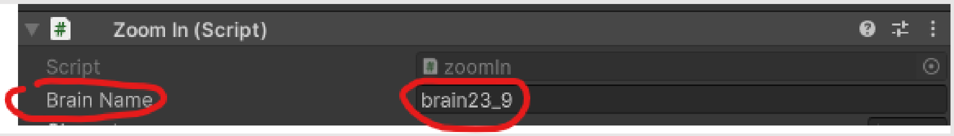
It is very important to write the brain model's name in the input string named **brainName** who's parts are to be separated.

If I have brain model's name **brain23_9**



ImageTarget (9)
Doctor_Male_B
casual_Female_G
brain23_9
Canvas
Dialogues

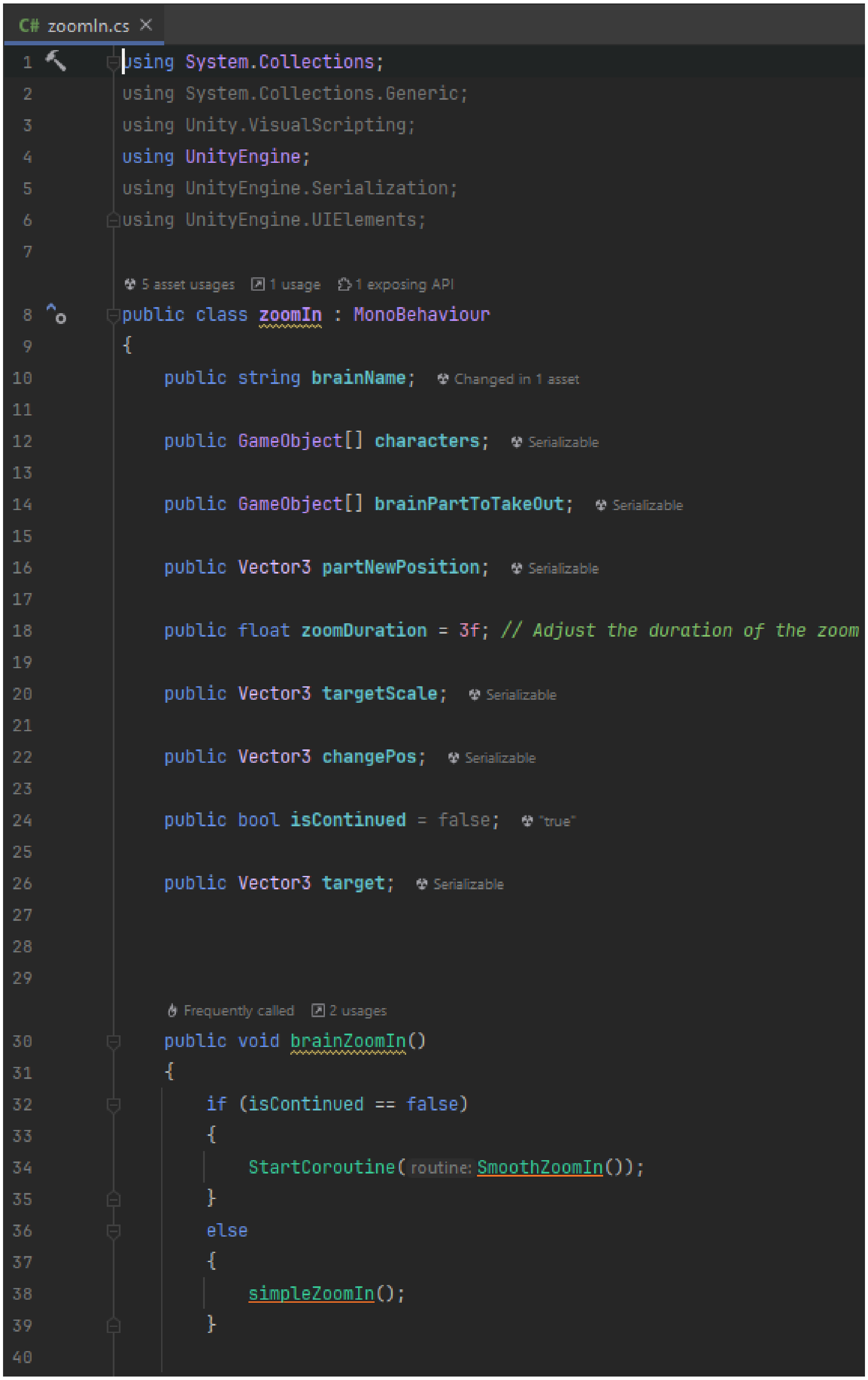
then I must write it correctly in here:



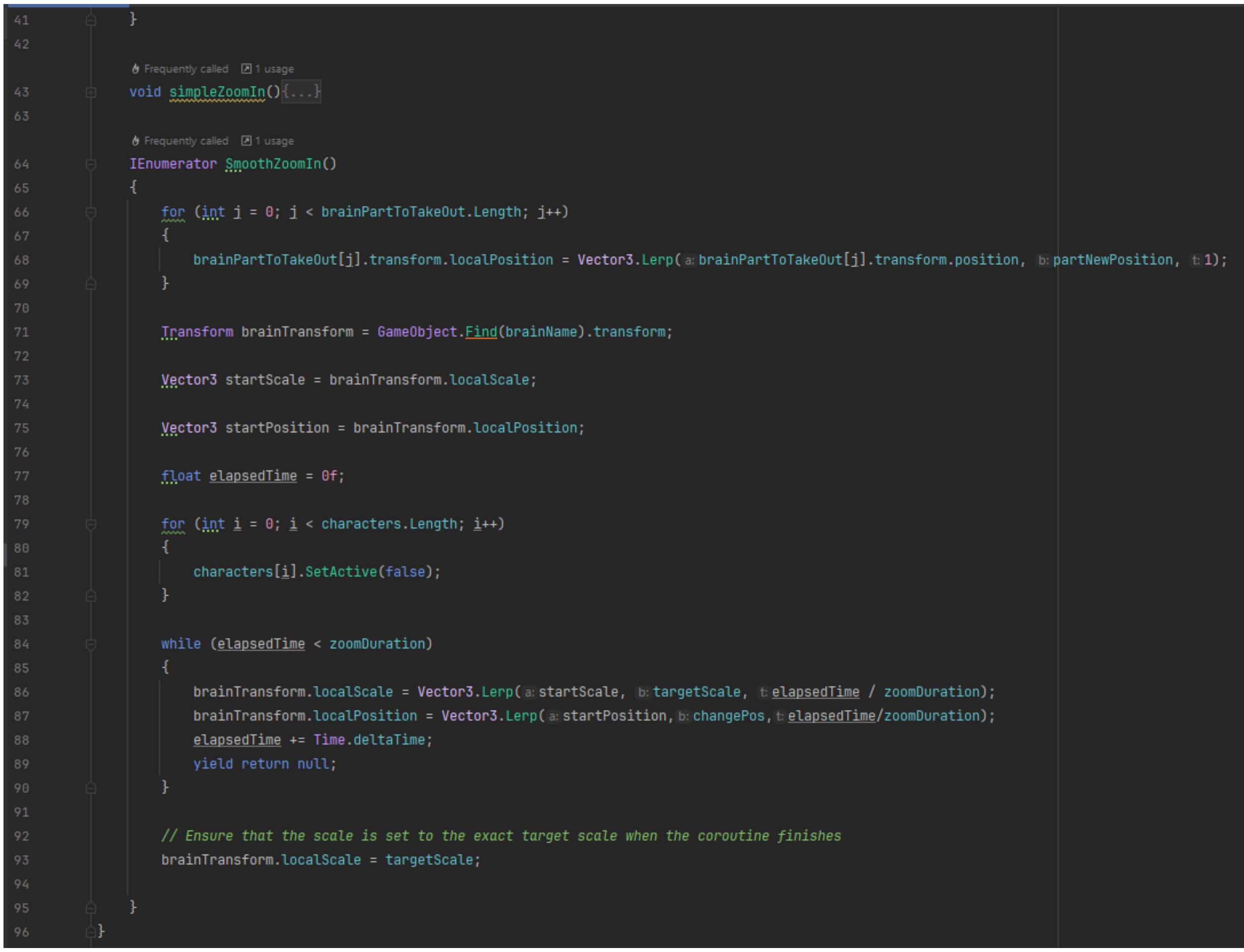
Zoom In (Script)

Script: zoomIn

Brain Name: brain23_9

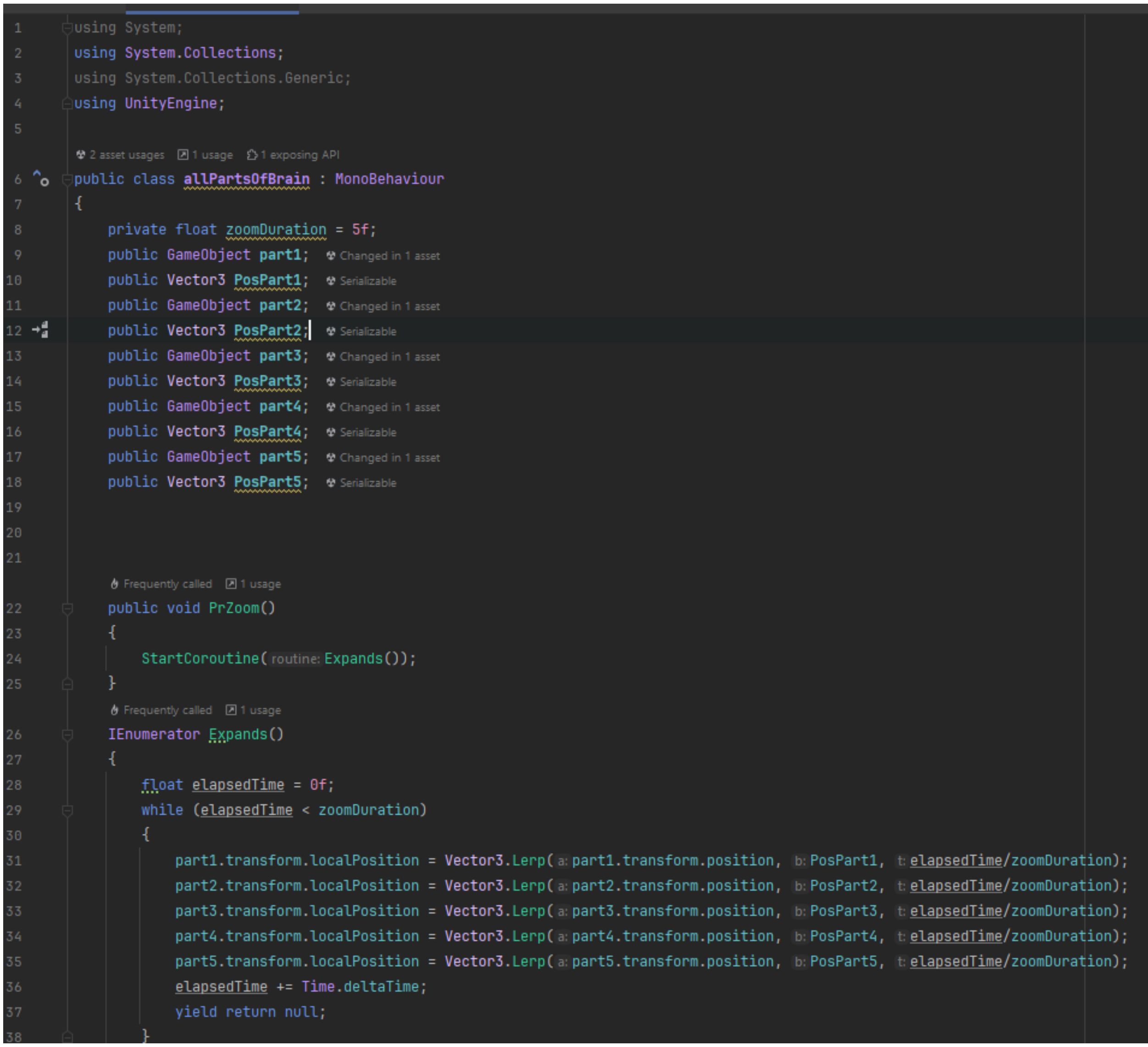


```csharp
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;
using UnityEngine.Serialization;
using UnityEngine.UIElements;

public class zoomIn : MonoBehaviour
{
    public string brainName;

    public GameObject[] characters;

    public GameObject[] brainPartToTakeOut;

    public Vector3 partNewPosition;

    public float zoomDuration = 3f; // Adjust the duration of the zoom

    public Vector3 targetScale;

    public Vector3 changePos;

    public bool isContinued = false;

    public Vector3 target;



    public void brainZoomIn()
    {
        if (isContinued == false)
        {
            StartCoroutine(SmoothZoomIn());
        }
        else
        {
            simpleZoomIn();
        }

```



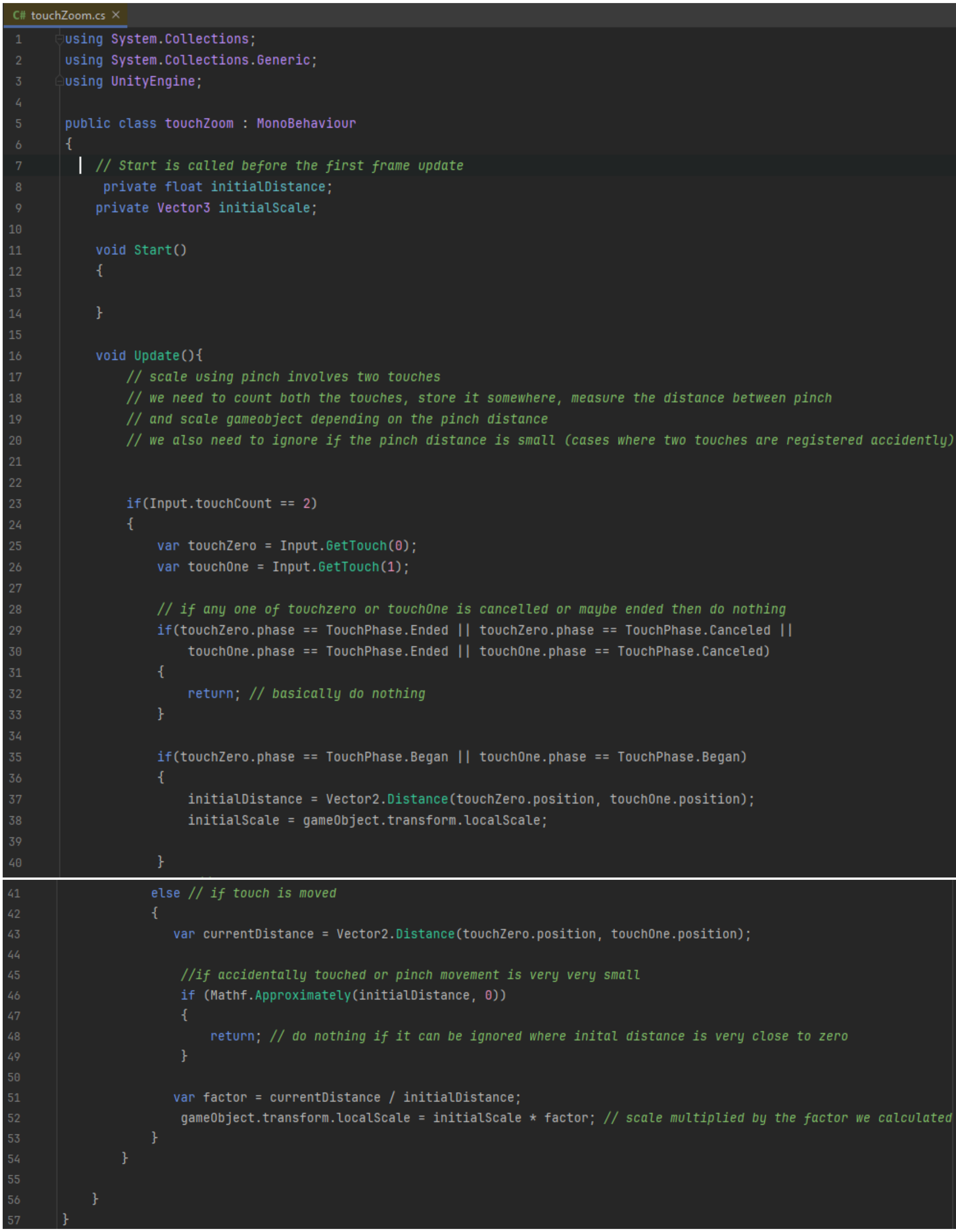
```
    }

    Frequently called  1 usage
    void simpleZoomIn(){...}

    Frequently called  1 usage
    IEnumerator SmoothZoomIn()
    {
        for (int j = 0; j < brainPartToTakeOut.Length; j++)
        {
            brainPartToTakeOut[j].transform.localPosition = Vector3.Lerp( a: brainPartToTakeOut[j].transform.position,  b: partNewPosition,  t: 1);
        }

        Transform brainTransform = GameObject.Find(brainName).transform;

        Vector3 startScale = brainTransform.localScale;

        Vector3 startPosition = brainTransform.localPosition;

        float elapsedTime = 0f;

        for (int i = 0; i < characters.Length; i++)
        {
            characters[i].SetActive(false);
        }

        while (elapsedTime < zoomDuration)
        {
            brainTransform.localScale = Vector3.Lerp( a: startScale,  b: targetScale,  t: elapsedTime / zoomDuration);
            brainTransform.localPosition = Vector3.Lerp( a: startPosition, b: changePos, t: elapsedTime/zoomDuration);
            elapsedTime += Time.deltaTime;
            yield return null;
        }

        // Ensure that the scale is set to the exact target scale when the coroutine finishes
        brainTransform.localScale = targetScale;

    }
}
```

At the last, there is another script named as **allPartsOfBrain.cs** this script is specially made to separate all parts of the brain.



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

public class allPartsOfBrain : MonoBehaviour
{
    private float zoomDuration = 5f;
    public GameObject part1;
    public Vector3 PosPart1;
    public GameObject part2;
    public Vector3 PosPart2;
    public GameObject part3;
    public Vector3 PosPart3;
    public GameObject part4;
    public Vector3 PosPart4;
    public GameObject part5;
    public Vector3 PosPart5;



    public void PrZoom()
    {
        StartCoroutine(routine: Expands());
    }
    IEnumerator Expands()
    {
        float elapsedTime = 0f;
        while (elapsedTime < zoomDuration)
        {
            part1.transform.localPosition = Vector3.Lerp( a: part1.transform.position,  b: PosPart1,  t: elapsedTime/zoomDuration);
            part2.transform.localPosition = Vector3.Lerp( a: part2.transform.position,  b: PosPart2,  t: elapsedTime/zoomDuration);
            part3.transform.localPosition = Vector3.Lerp( a: part3.transform.position,  b: PosPart3,  t: elapsedTime/zoomDuration);
            part4.transform.localPosition = Vector3.Lerp( a: part4.transform.position,  b: PosPart4,  t: elapsedTime/zoomDuration);
            part5.transform.localPosition = Vector3.Lerp( a: part5.transform.position,  b: PosPart5,  t: elapsedTime/zoomDuration);
            elapsedTime += Time.deltaTime;
            yield return null;
        }
```

Now, lastly for zooming in and out of the brain model, the **touchZoom.cs** script has been used. Just apply this script to the brain's model and it will work fine.



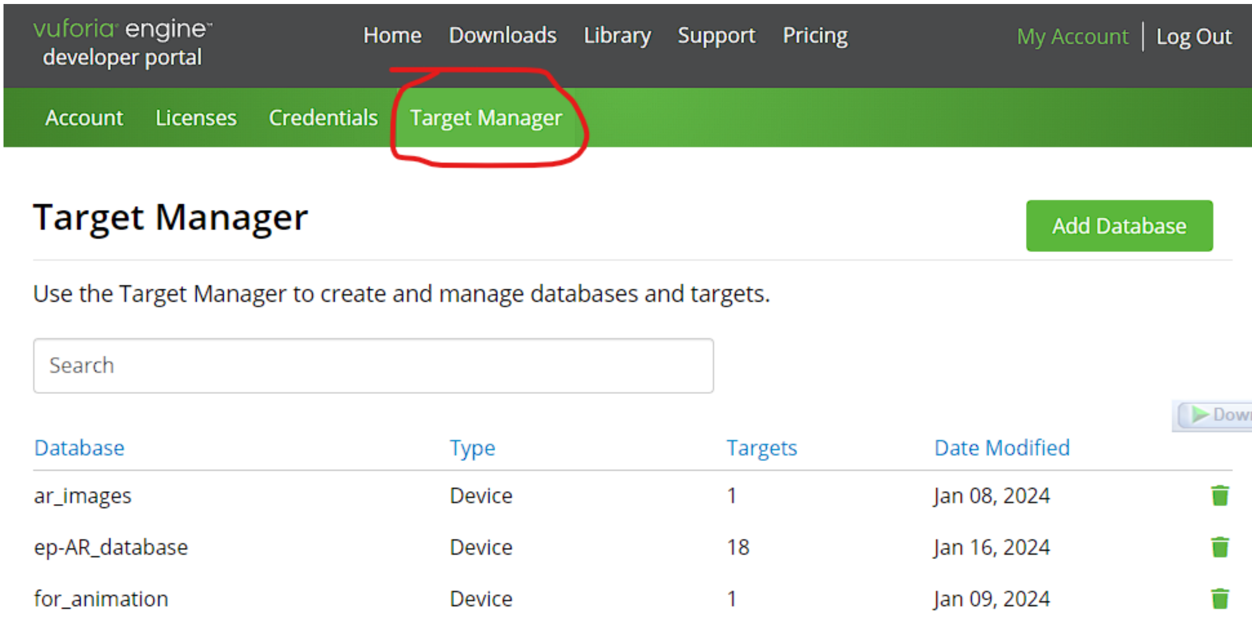
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class touchZoom : MonoBehaviour
{
    // Start is called before the first frame update
     private float initialDistance;
    private Vector3 initialScale;

    void Start()
    {

    }

    void Update(){
        // scale using pinch involves two touches
        // we need to count both the touches, store it somewhere, measure the distance between pinch
        // and scale gameobject depending on the pinch distance
        // we also need to ignore if the pinch distance is small (cases where two touches are registered accidently)


        if(Input.touchCount == 2)
        {
            var touchZero = Input.GetTouch(0);
            var touchOne = Input.GetTouch(1);

            // if any one of touchzero or touchOne is cancelled or maybe ended then do nothing
            if(touchZero.phase == TouchPhase.Ended || touchZero.phase == TouchPhase.Canceled ||
                touchOne.phase == TouchPhase.Ended || touchOne.phase == TouchPhase.Canceled)
            {
                return; // basically do nothing
            }

            if(touchZero.phase == TouchPhase.Began || touchOne.phase == TouchPhase.Began)
            {
                initialDistance = Vector2.Distance(touchZero.position, touchOne.position);
                initialScale = gameObject.transform.localScale;

            }
            else // if touch is moved
            {
               var currentDistance = Vector2.Distance(touchZero.position, touchOne.position);

                //if accidentally touched or pinch movement is very very small
                if (Mathf.Approximately(initialDistance, 0))
                {
                    return; // do nothing if it can be ignored where inital distance is very close to zero
                }

               var factor = currentDistance / initialDistance;
                gameObject.transform.localScale = initialScale * factor; // scale multiplied by the factor we calculated
            }
        }

    }
}
```

3.3. Importing Vuforia

The Vuforia engine is used to make image targets and help display objects on them.

To get Vuforia engine, first we have to go to their website https://developer.vuforia.com/

Then we signup, after that we go to **target manager**



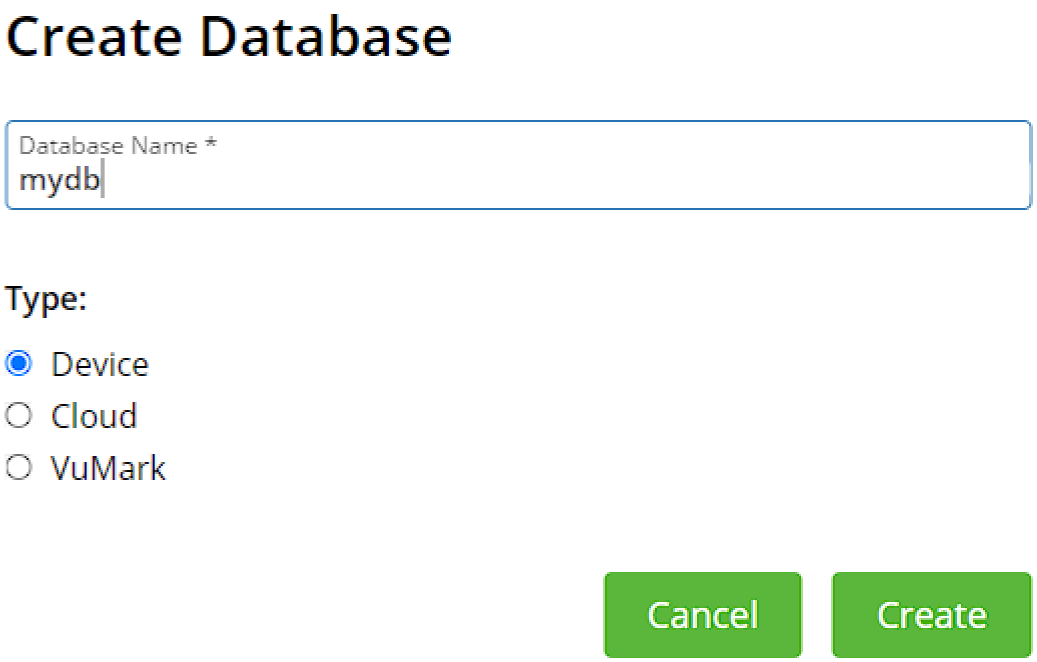
Target Manager

Use the Target Manager to create and manage databases and targets.

Database	Type	Targets	Date Modified
ar_images	Device	1	Jan 08, 2024
ep-AR_database	Device	18	Jan 16, 2024
for_animation	Device	1	Jan 09, 2024

From there, we click on **Add Database.**

We give it a **Name** and choose **Device** then click on **Create**.


Create Database

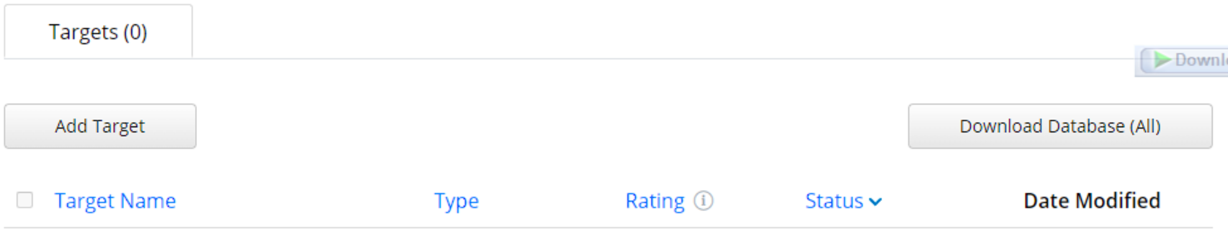
Database Name *
mydb

Type:

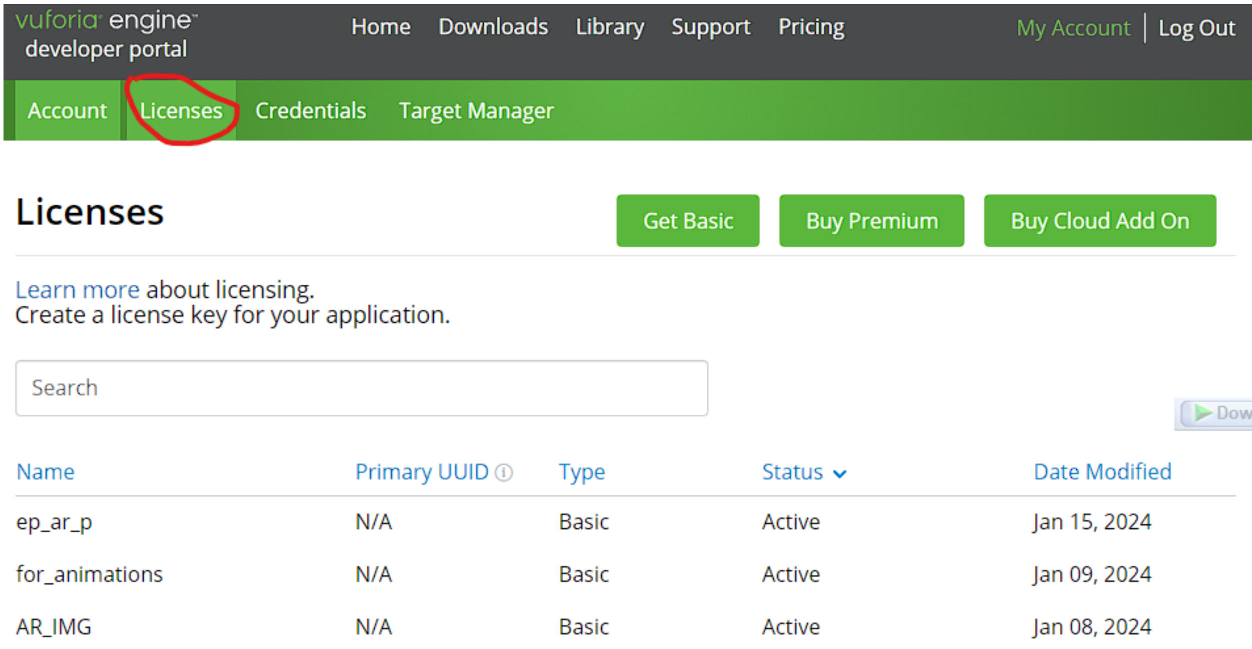
● Device
○ Cloud
○ VuMark

Then we click on **Add Target** button. Then we upload an image which could be a good target (don't use too much decorated image or too dull image, this won't be a better image target).

Then after adding desired amount of images, click on **download database** button



After that go to **Licenses** tab

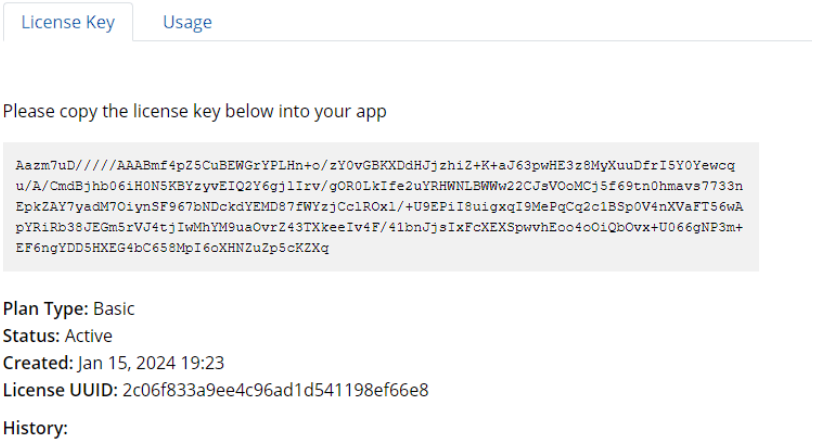


Licenses

Learn more about licensing.
Create a license key for your application.

Name	Primary UUID ⓘ	Type	Status	Date Modified
ep_ar_p	N/A	Basic	Active	Jan 15, 2024
for_animations	N/A	Basic	Active	Jan 09, 2024
AR_IMG	N/A	Basic	Active	Jan 08, 2024

Click on **Get Basic** and give your license a name, after creating, you'll get something like that



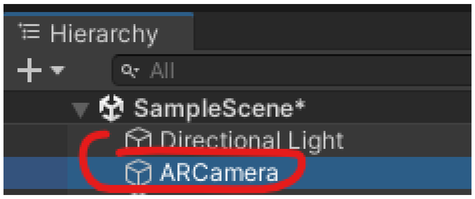
Please copy the license key below into your app

```
Aazm7uD/////AAABmf4pZ5CuBEWGrYPLHn+o/zY0vGBKXDdHJjzhiZ+K+aJ63pwHE3z8MyXuuDfrI5Y0Yewcq
u/A/CmdBjhb06iH0N5KBYzyvEIQ2Y6gjlIrv/gOR0LkIfe2uYRHWNLBWWw22CJsVOoMCj5f69tn0hmavs7733n
EpkZAY7yadM7OiynSF967bNDckdYEMD87fWYzjCclROxl/+U9EPiI8uigxqI9MePqCq2c1BSp0V4nXVaFT56wA
pYRiRb38JEGm5rVJ4tjIwMhYM9uaOvrZ43TXkeeIv4F/41bnJjsIxFcXEXSpwvhEoo4oOiQbOvx+U066gNP3m+
EF6ngYDD5HXEG4bC658MpI6oXHNZuZp5cKZXq
```

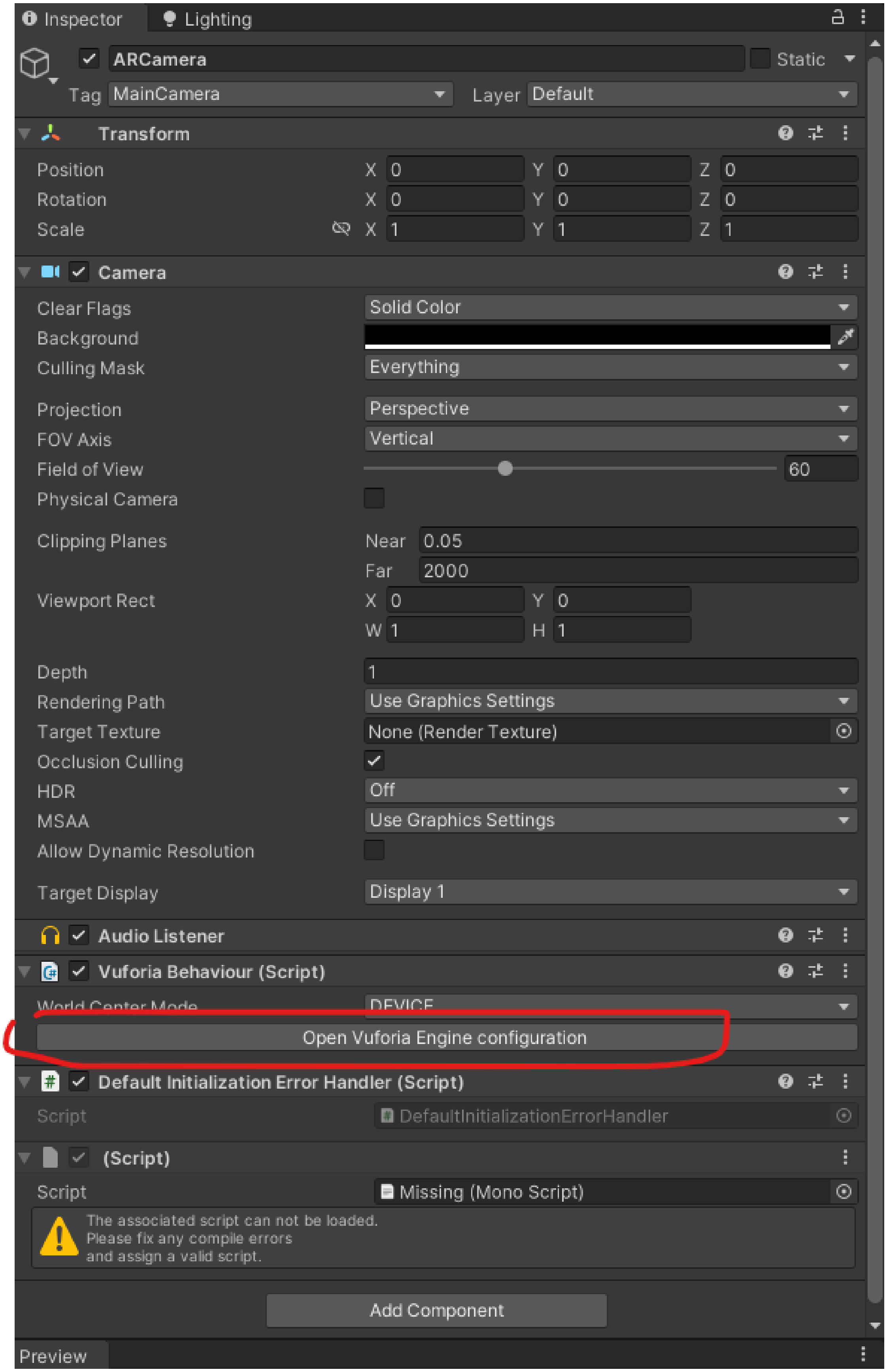
Plan Type: Basic
Status: Active
Created: Jan 15, 2024 19:23
License UUID: 2c06f833a9ee4c96ad1d541198ef66e8

History:

Copy the license key and paste it in the **AR Camera** in unity engine.



Click here


Inspector
Lighting
Static
Transform
Camera
Audio Listener
Vuforia Behaviour (Script)
Open Vuforia Engine configuration
Default Initialization Error Handler (Script)
(Script)
Add Component
Preview

☑ ARCamera

Tag MainCamera

Layer Default

Position X 0 Y 0 Z 0

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Clear Flags: Solid Color

Background

Culling Mask: Everything

Projection: Perspective

FOV Axis: Vertical

Field of View: 60

Physical Camera: □

Clipping Planes: Near 0.05, Far 2000

Viewport Rect: X 0 Y 0, W 1 H 1

Depth: 1

Rendering Path: Use Graphics Settings

Target Texture: None (Render Texture)

Occlusion Culling: ☑

HDR: Off

MSAA: Use Graphics Settings

Allow Dynamic Resolution: □

Target Display: Display 1

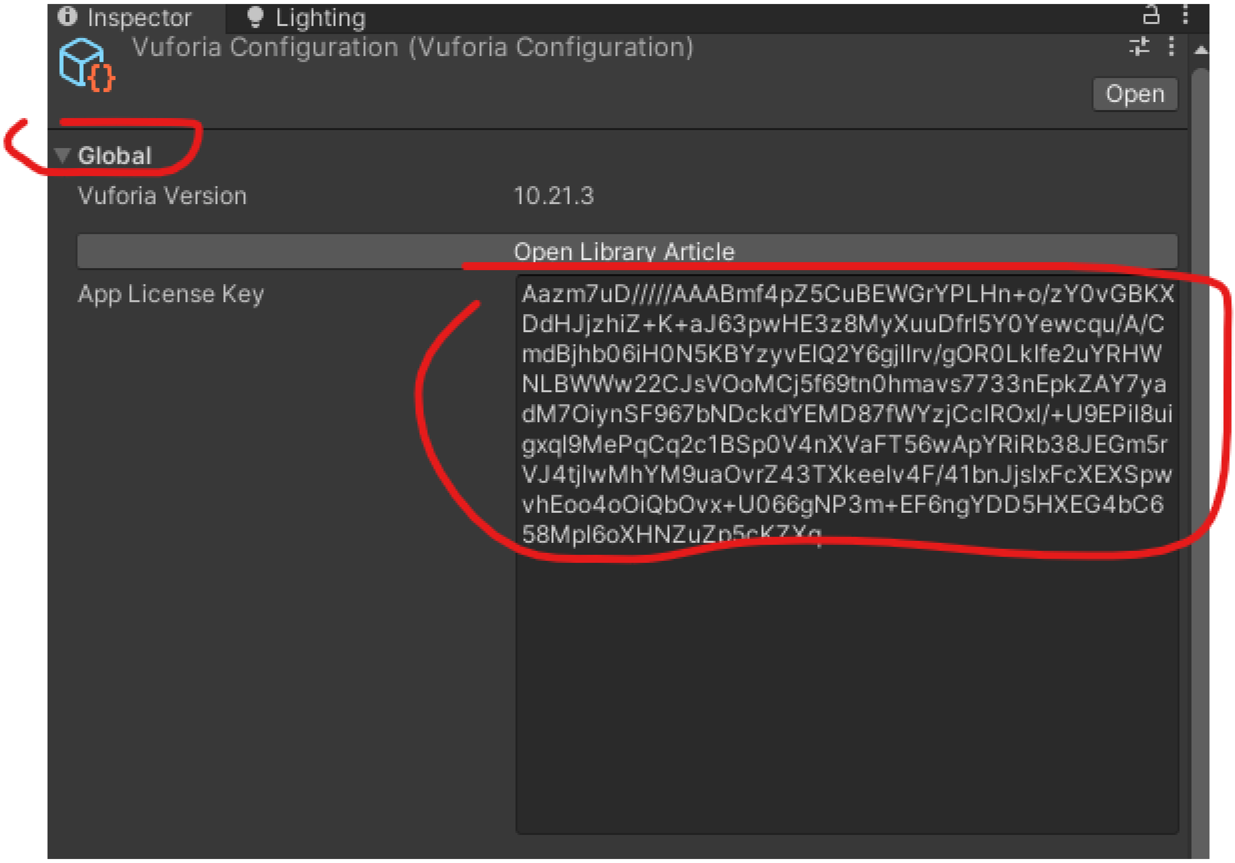
World Center Mode: DEVICE

Script: DefaultInitializationErrorHandler

Script: Missing (Mono Script)

The associated script can not be loaded.
Please fix any compile errors
and assign a valid script.

Then drop down **Global** and paste your **License key** there

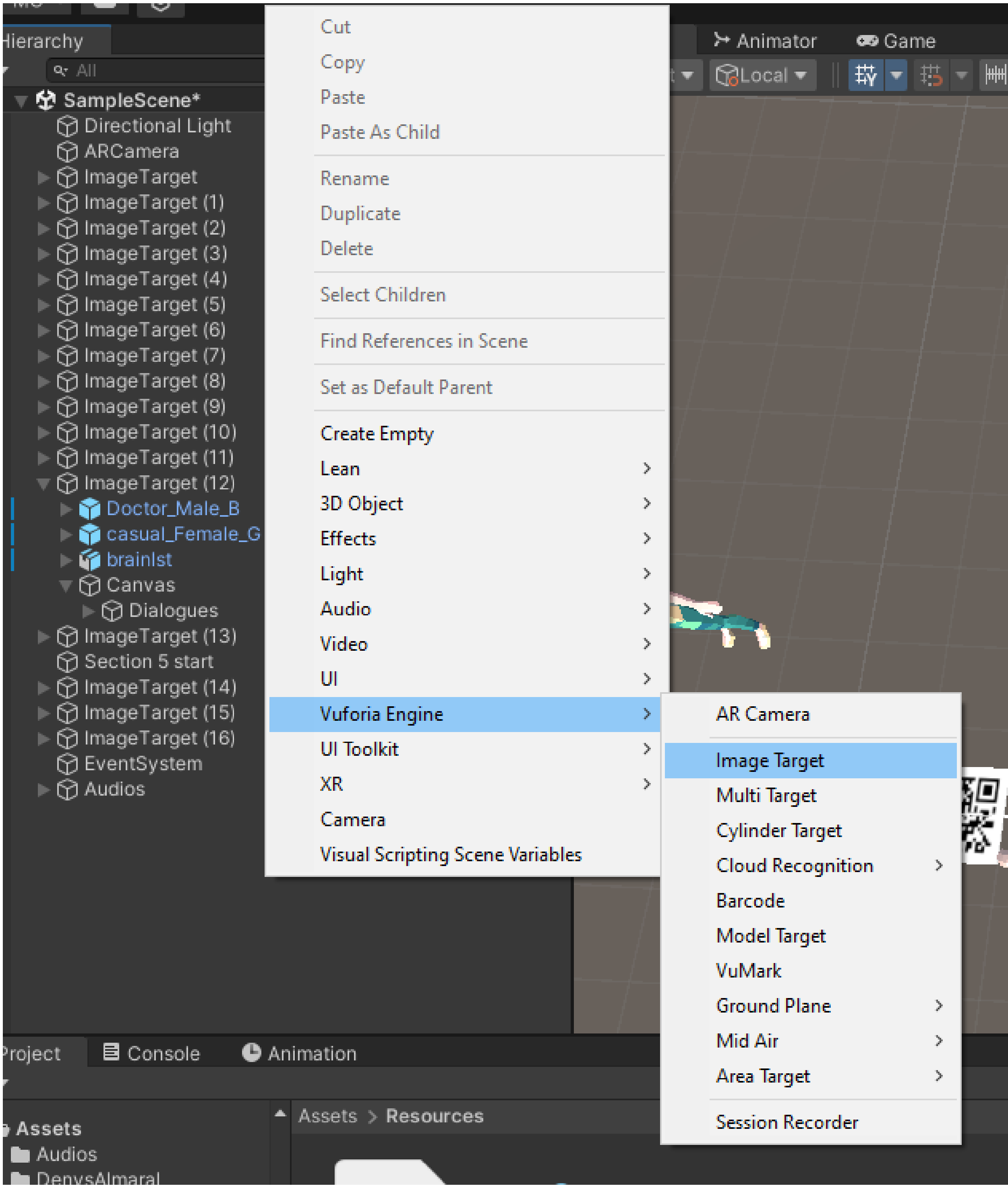


Vuforia Version 10.21.3

App License Key

```
Aazm7uD/////AAABmf4pZ5CuBEWGrYPLHn+o/zY0vGBKX
DdHJjzhiZ+K+aJ63pwHE3z8MyXuuDfrl5Y0Yewcqu/A/C
mdBjhb06iH0N5KBYzyvElQ2Y6gjIlrv/gOR0Lklfe2uYRHW
NLBWWw22CJsVOoMCj5f69tn0hmavs7733nEpkZAY7ya
dM7OiynSF967bNDckdYEMD87fWYzjCclROxl/+U9EPil8ui
gxql9MePqCq2c1BSp0V4nXVaFT56wApYRiRb38JEGm5r
VJ4tjlwMhYM9uaOvrZ43TXkeelv4F/41bnJjslxFcXEXSpw
vhEoo4oOiQbOvx+U066gNP3m+EF6ngYDD5HXEG4bC6
58Mpl6oXHNZuZp5cKZXq
```

Then to create image targets, simply **Right-Click** in hierarchy section and do the following



In image target's properties on right side of screen, you can select your imported Vuforia database and select your image targets from there

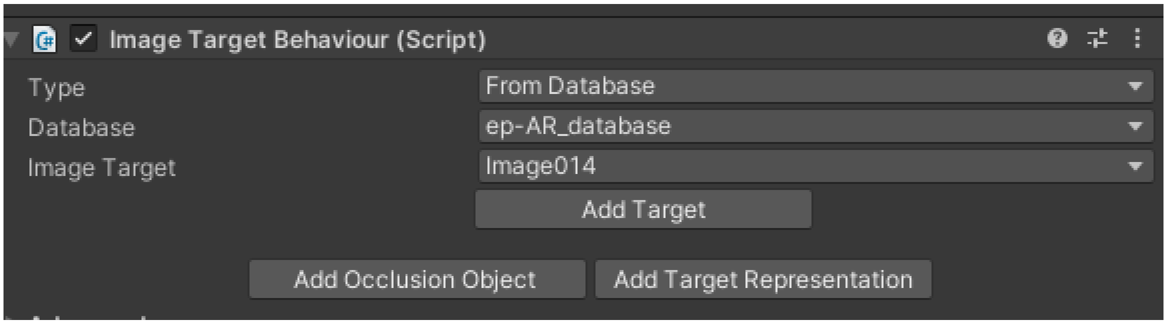


Image Target Behaviour (Script)

Type: From Database

Database: ep-AR_database

Image Target: Image014

The End