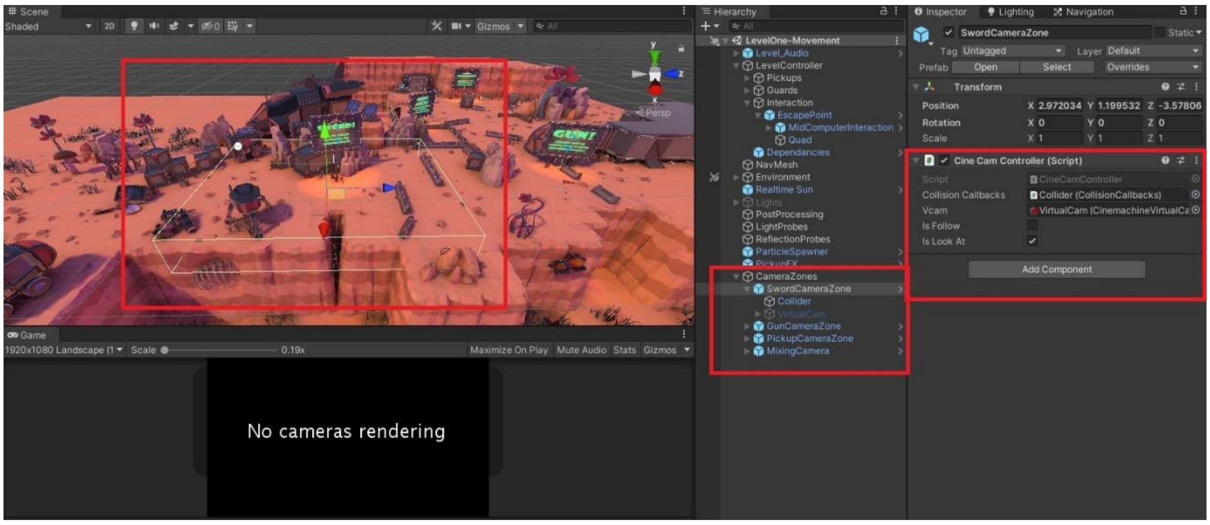
Sprint 03

Assignment 04 : Cinemachine

Step 01: Setup

- I first created a “*CameraZone*” empty GameObject and attached to it the “*CineCamController*” script.
- Then I assigned the “Collider” and Virtual Camera component.
- Then to the Collider component, I attached the CollisionCallBack script and a “Box Collider” and set its Trigger ON and adjusted the bounding box as per requirement.
- Then I made a prefab of this GameObject, and further used it in every camera zone, just updating or renaming it.

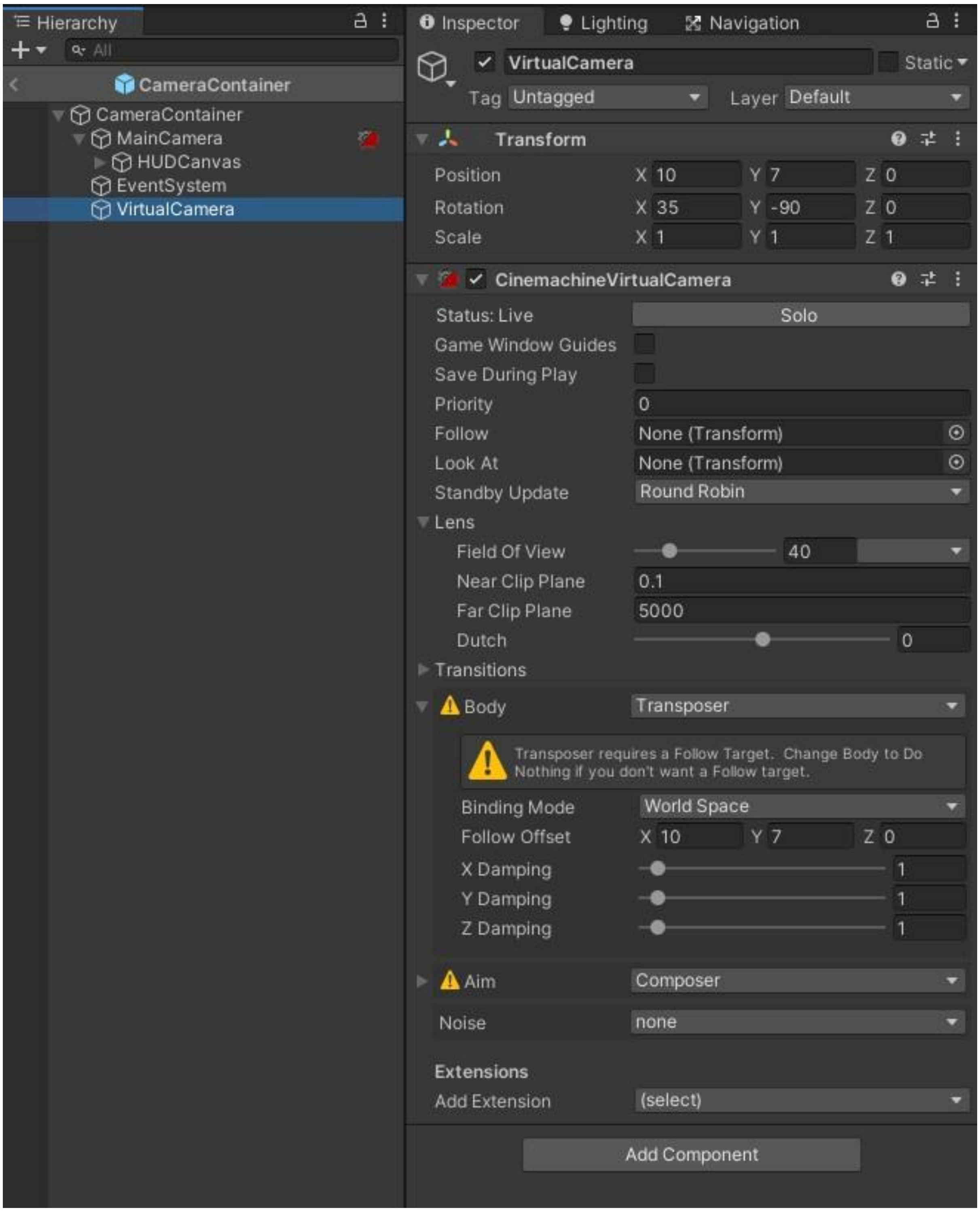


SwordCameraZone
Tag Untagged Layer Default
Prefab Open Select Overrides

Transform
Position X 2.972034 Y 1.199532 Z -3.57806
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Cine Cam Controller (Script)
Script CineCamController
Collision Callbacks Collider (CollisionCallbacks)
Vcam VirtualCam (CinemachineVirtualCa
Is Follow ☐
Is Look At ✓

- Next, I attached a “*VirtualCamera*” to the MainCamera prefab and renamed it accordingly.



VirtualCamera Static
Tag Untagged Layer Default

Transform

Position	X 10	Y 7	Z 0
Rotation	X 35	Y -90	Z 0
Scale	X 1	Y 1	Z 1

CinemachineVirtualCamera

Status: Live	Solo
Game Window Guides	
Save During Play	
Priority	0
Follow	None (Transform)
Look At	None (Transform)
Standby Update	Round Robin
Lens	
Field Of View	40
Near Clip Plane	0.1
Far Clip Plane	5000
Dutch	0
Transitions	
Body	Transposer

Transposer requires a Follow Target. Change Body to Do Nothing if you don't want a Follow target.

Binding Mode	World Space		
Follow Offset	X 10	Y 7	Z 0
X Damping	1		
Y Damping	1		
Z Damping	1		
Aim	Composer		
Noise	none		

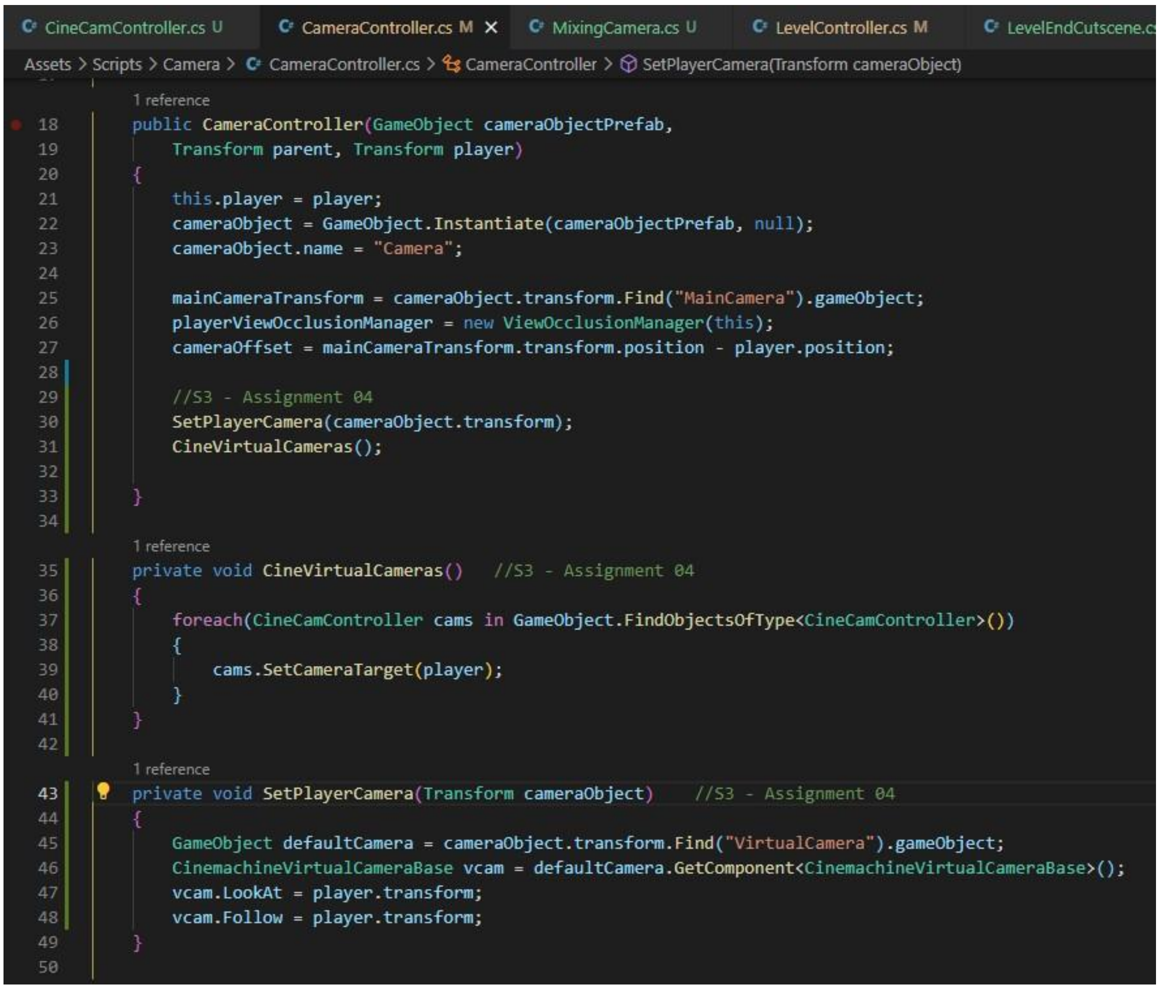
Extensions
Add Extension (select)

Add Component

Step 02: Script & Workflow

- The Scripts workflow is like this:
  - CameraController > CineCamController > MixingCamera.
  - LevelController > EndLevelCutscene > CutsceneSignalListener.

❖ **CameraController.**

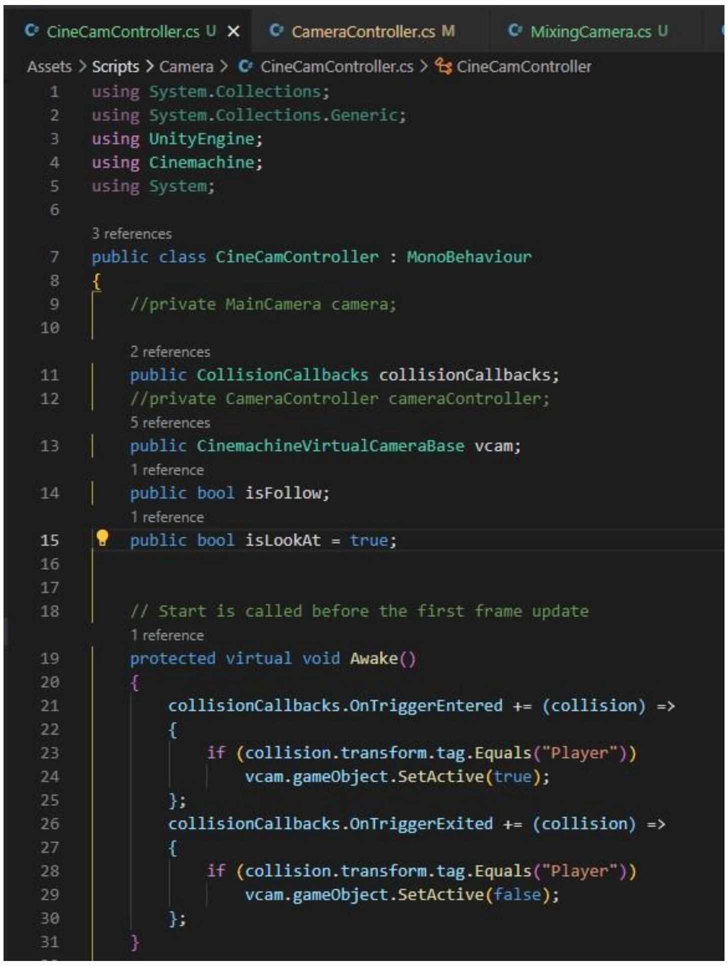


CineCamController.cs U
CameraController.cs M
MixingCamera.cs U
LevelController.cs M
LevelEndCutscene.c
Assets > Scripts > Camera > CameraController.cs > CameraController > SetPlayerCamera(Transform cameraObject)

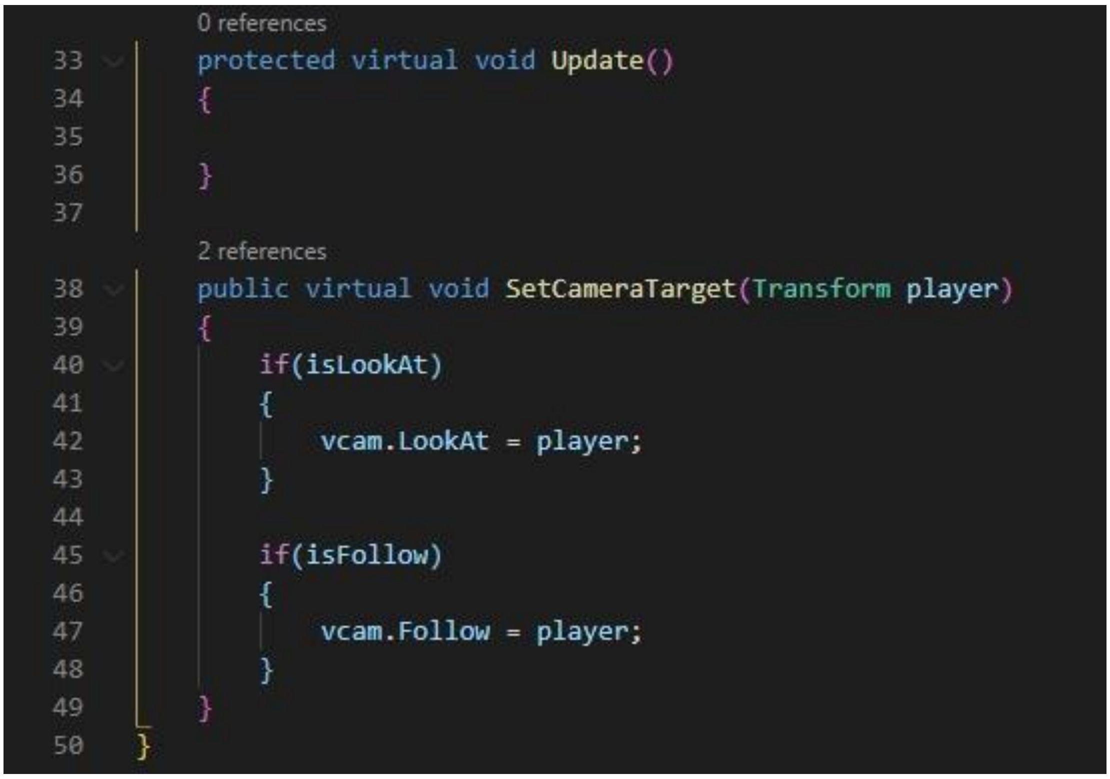
```
1 reference
public CameraController(GameObject cameraObjectPrefab,
    Transform parent, Transform player)
{
    this.player = player;
    cameraObject = GameObject.Instantiate(cameraObjectPrefab, null);
    cameraObject.name = "Camera";

    mainCameraTransform = cameraObject.transform.Find("MainCamera").gameObject;
    playerViewOcclusionManager = new ViewOcclusionManager(this);
    cameraOffset = mainCameraTransform.transform.position - player.position;

    //S3 - Assignment 04
    SetPlayerCamera(cameraObject.transform);
    CineVirtualCameras();

}

1 reference
private void CineVirtualCameras()    //S3 - Assignment 04
{
    foreach(CineCamController cams in GameObject.FindObjectsOfType<CineCamController>())
    {
        cams.SetCameraTarget(player);
    }
}

1 reference
private void SetPlayerCamera(Transform cameraObject)     //S3 - Assignment 04
{
    GameObject defaultCamera = cameraObject.transform.Find("VirtualCamera").gameObject;
    CinemachineVirtualCameraBase vcam = defaultCamera.GetComponent<CinemachineVirtualCameraBase>();
    vcam.LookAt = player.transform;
    vcam.Follow = player.transform;
}
```

- The "*CineVirtualCamera*" function gets the all the Components who have this script attached, from the Scene-Hierarchy.
- The "*SetPlayerCamera*" function gets the "*VirtualCamera*" attached to the MainCamera prefab, and assigns the LookAt and Follow target as Player.

❖ **CineCamController.**

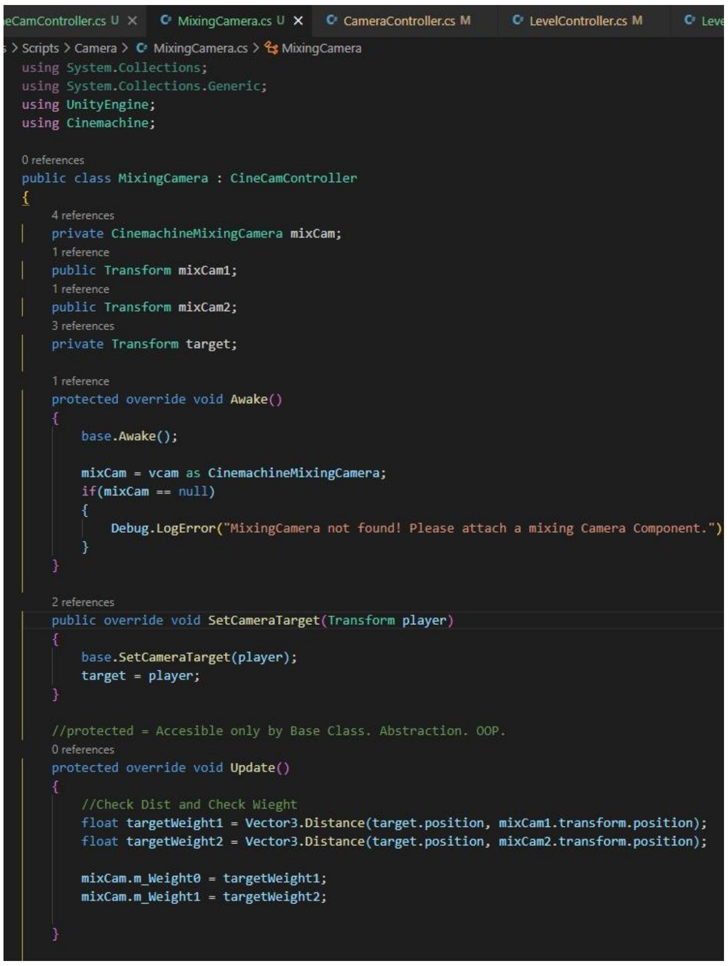

CineCamController.cs U ×
CameraController.cs M
MixingCamera.cs U
Assets > Scripts > Camera > CineCamController.cs > CineCamController

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

3 references
public class CineCamController : MonoBehaviour
{
    //private MainCamera camera;

    2 references
    public CollisionCallbacks collisionCallbacks;
    //private CameraController cameraController;
    5 references
    public CinemachineVirtualCameraBase vcam;
    1 reference
    public bool isFollow;
    1 reference
    public bool isLookAt = true;


    // Start is called before the first frame update
    1 reference
    protected virtual void Awake()
    {
        collisionCallbacks.OnTriggerEntered += (collision) =>
        {
            if (collision.transform.tag.Equals("Player"))
                vcam.gameObject.SetActive(true);
        };
        collisionCallbacks.OnTriggerExited += (collision) =>
        {
            if (collision.transform.tag.Equals("Player"))
                vcam.gameObject.SetActive(false);
        };
    }
```



```
    0 references
    protected virtual void Update()
    {

    }

    2 references
    public virtual void SetCameraTarget(Transform player)
    {
        if(isLookAt)
        {
            vcam.LookAt = player;
        }

        if(isFollow)
        {
            vcam.Follow = player;
        }
    }
}
```

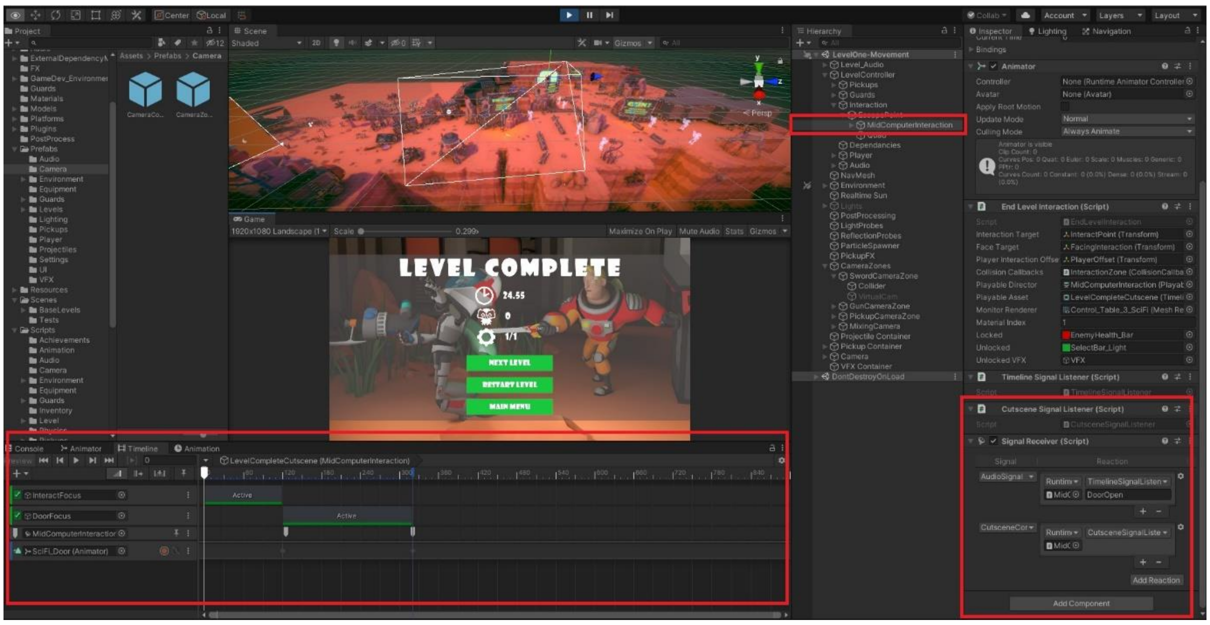
❖ MixingCamera.



```
eCamController.cs U ×   MixingCamera.cs U ×   CameraController.cs M   LevelController.cs M   Leve
> Scripts > Camera > MixingCamera.cs > MixingCamera
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Cinemachine;

0 references
public class MixingCamera : CineCamController
{
    4 references
    private CinemachineMixingCamera mixCam;
    1 reference
    public Transform mixCam1;
    1 reference
    public Transform mixCam2;
    3 references
    private Transform target;

    1 reference
    protected override void Awake()
    {
        base.Awake();

        mixCam = vcam as CinemachineMixingCamera;
        if(mixCam == null)
        {
            Debug.LogError("MixingCamera not found! Please attach a mixing Camera Component.")
        }
    }

    2 references
    public override void SetCameraTarget(Transform player)
    {
        base.SetCameraTarget(player);
        target = player;
    }

    //protected = Accesible only by Base Class. Abstraction. OOP.
    0 references
    protected override void Update()
    {
        //Check Dist and Check Wieght
        float targetWeight1 = Vector3.Distance(target.position, mixCam1.transform.position);
        float targetWeight2 = Vector3.Distance(target.position, mixCam2.transform.position);

        mixCam.m_Weight0 = targetWeight1;
        mixCam.m_Weight1 = targetWeight2;

    }
```

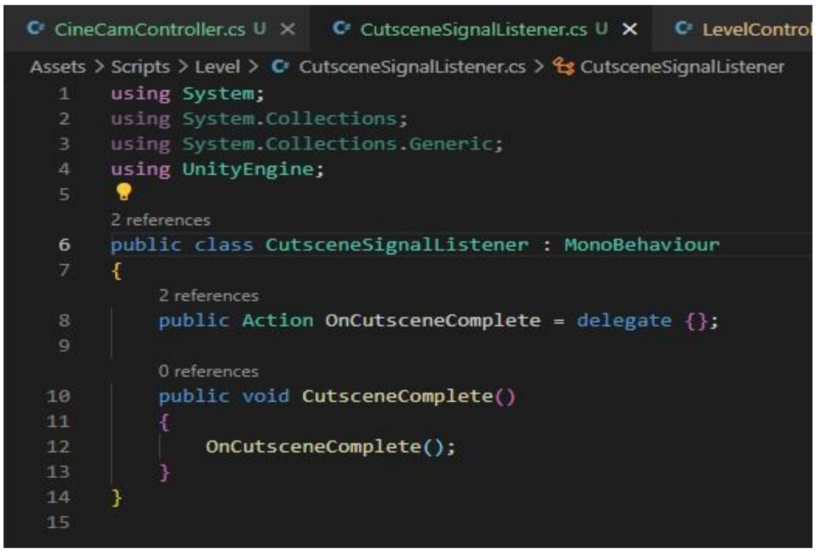
- This script inherits the "*CineCamController*" script.
- It inherits the same functions and their implementations, and in addition implements its own logic too.
- "***Override***" is used to use the same function from the Base Class.
- "***base***" is used to implement the functionalities from that function of the Base Class.

❖ Timeline Setup.



P.S: I have created a folder called "*Timeline*" and kept the Signals and other components in there.

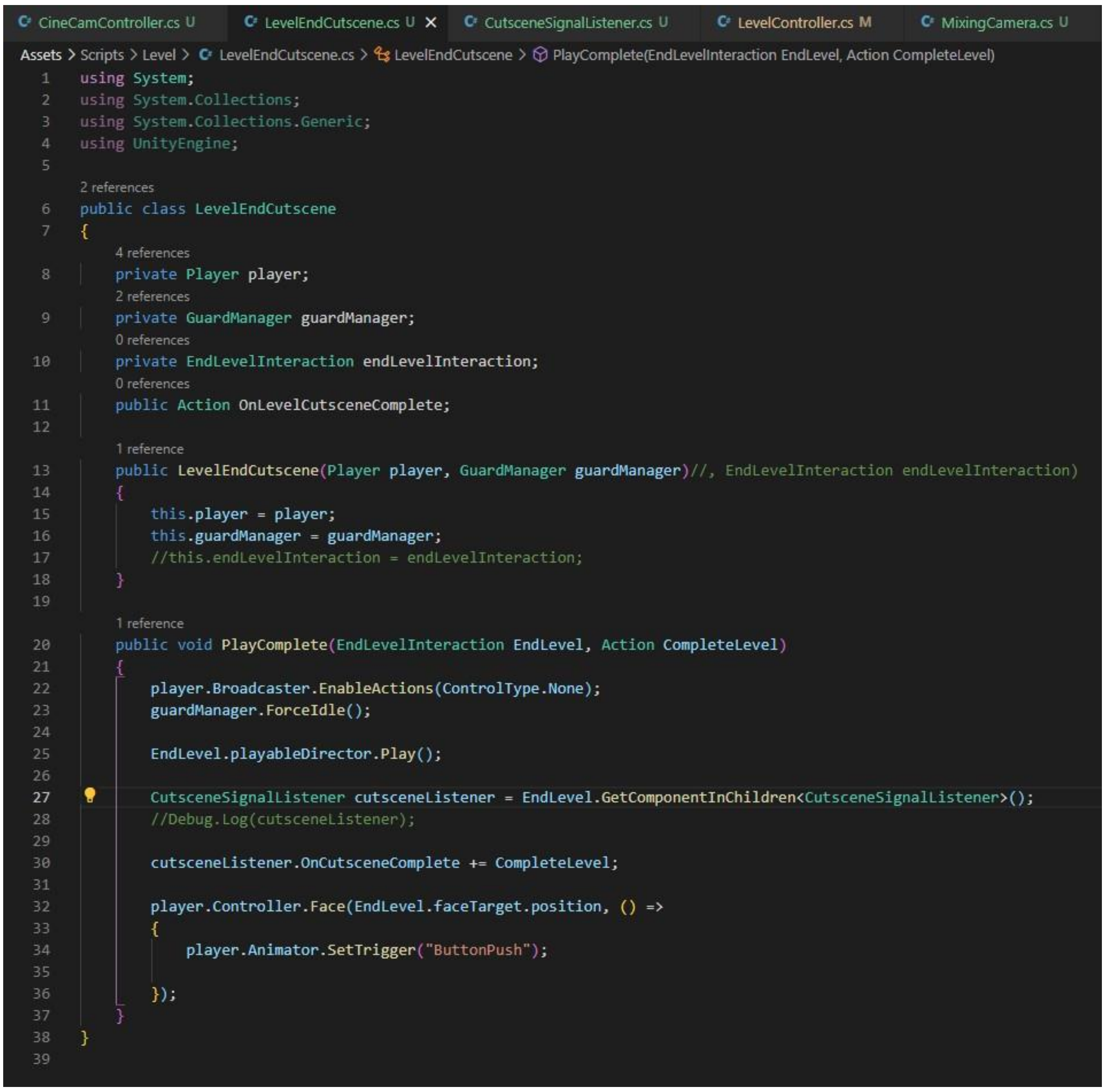
❖ CutsceneSignalListener.


CineCamController.cs U
CutsceneSignalListener.cs U
LevelControl
Assets > Scripts > Level > CutsceneSignalListener.cs > CutsceneSignalListener

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

2 references
public class CutsceneSignalListener : MonoBehaviour
{
    2 references
    public Action OnCutsceneComplete = delegate {};

    0 references
    public void CutsceneComplete()
    {
        OnCutsceneComplete();
    }
}

```

- This script is attached to the "*MidComputerInteraction*" GameObject.
- And its Action event is called by other scripts.

❖ **EndLevelCutscene.**



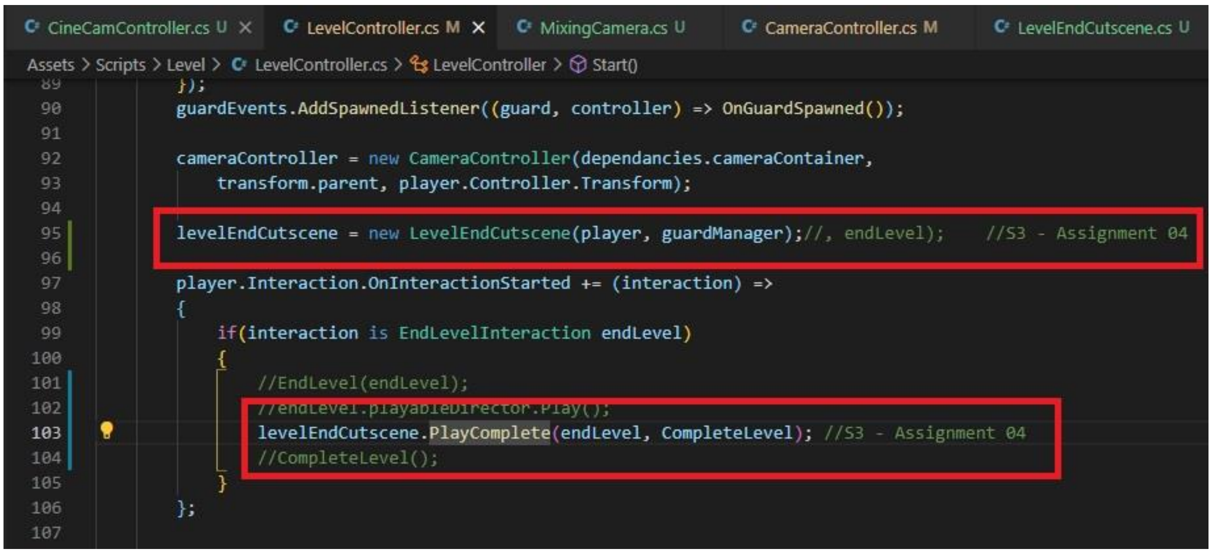
CineCamController.cs U | LevelEndCutscene.cs U × | CutsceneSignalListener.cs U | LevelController.cs M | MixingCamera.cs U

Assets > Scripts > Level > LevelEndCutscene.cs > LevelEndCutscene > PlayComplete(EndLevelInteraction EndLevel, Action CompleteLevel)

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

2 references
public class LevelEndCutscene
{
    4 references
    private Player player;
    2 references
    private GuardManager guardManager;
    0 references
    private EndLevelInteraction endLevelInteraction;
    0 references
    public Action OnLevelCutsceneComplete;

    1 reference
    public LevelEndCutscene(Player player, GuardManager guardManager)//, EndLevelInteraction endLevelInteraction)
    {
        this.player = player;
        this.guardManager = guardManager;
        //this.endLevelInteraction = endLevelInteraction;
    }

    1 reference
    public void PlayComplete(EndLevelInteraction EndLevel, Action CompleteLevel)
    {
        player.Broadcaster.EnableActions(ControlType.None);
        guardManager.ForceIdle();

        EndLevel.playableDirector.Play();

        CutsceneSignalListener cutsceneListener = EndLevel.GetComponentInChildren<CutsceneSignalListener>();
        //Debug.Log(cutsceneListener);

        cutsceneListener.OnCutsceneComplete += CompleteLevel;

        player.Controller.Face(EndLevel.faceTarget.position, () =>
        {
            player.Animator.SetTrigger("ButtonPush");

        });
    }
}
```

- I shifted all the "*EndLevel*" function from LevelController to here.
- Then passed the EndLevelInteraction and an Action Event, to call the "*CompleteLevel*" function from LevelController.
- The Action event is triggered by the event of CutsceneSignalListener script's event.

❖ **LevelController.**



```
        });
        guardEvents.AddSpawnedListener((guard, controller) => OnGuardSpawned());

        cameraController = new CameraController(dependancies.cameraContainer,
            transform.parent, player.Controller.Transform);

        levelEndCutscene = new LevelEndCutscene(player, guardManager);//, endLevel);    //S3 - Assignment 04

        player.Interaction.OnInteractionStarted += (interaction) =>
        {
            if(interaction is EndLevelInteraction endLevel)
            {
                //EndLevel(endLevel);
                //endLevel.playableDirector.Play();
                levelEndCutscene.PlayComplete(endLevel, CompleteLevel); //S3 - Assignment 04
                //CompleteLevel();
            }
        };
```

- Here, I instantiate the LevelEndCutscene script and pass in the required values, and then call the "*PlayComplete*" function and then pass the "*CompleteLevel*" function also.

---------------------THE END---------------------