Assignment # 3

Fall 2024

CSE-411 Intro to Game Development

Submitted by: **Ali Asghar**

Registration No.: **21PWCSE2059**

Class Section: **A**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Submitted to:

Engr. Abdullah Hamid

Date:

31st December 2024

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Task:

As discussed in class, create logic for fade in and fade out panels in Unity using C# coroutines method.

Solution:

Theory:

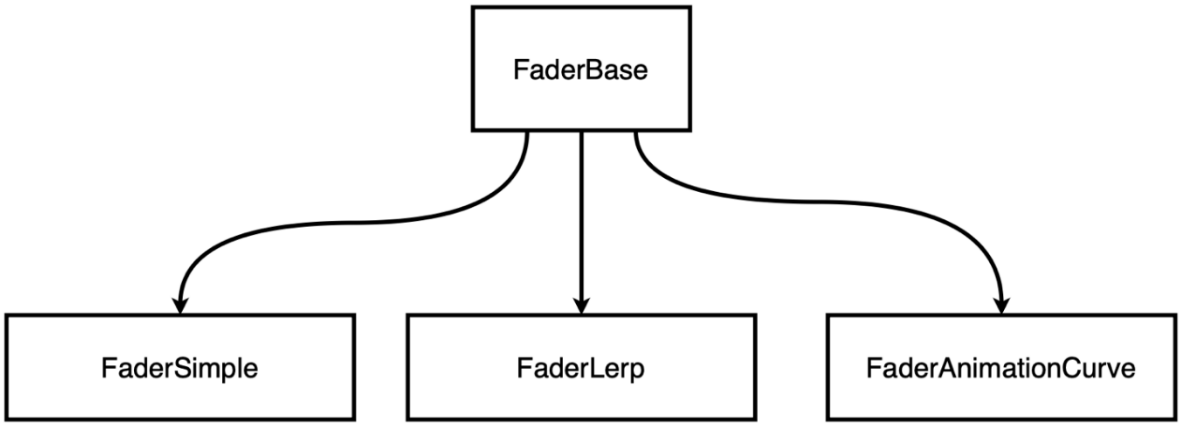


Figure 1: Code Architecture for Implementing Fading Logic using 3 different methods

I have implemented the Fading (Fade in and Fade Out) logic using 3 different approaches. For this reason, I have created a base class **FaderBase** which contains the necessary code common to all the scripts. The **FaderSimple, FaderLerp** and **FaderAnimationCurve** scripts contains the implementation code for each method as described by their name. All the implementations are done by using the Unity's coroutines. Their short explanation is given below.

FaderBase

This is the abstract base class for managing UI fade effects. It provides the basic structure and common functionalities for fade animations, such as references to the Image panel and Button components for triggering fade-in and fade-out actions. The class defines abstract methods FadeIn() and FadeOut(), which derived classes must implement. It initializes the panel, fadeInButton, and fadeOutButton components and sets up button click listeners to trigger the fade effects.

FaderSimple

This is a simpler implementation of the fade effect, also derived from FaderBase. Instead of interpolation, it increments or decrements the alpha value in small steps (0.01 per frame) within a coroutine. It uses WaitForSeconds with a fixed interval (0.01 seconds) to create a stepwise fade effect, which is less smooth compared to the linear interpolation used in FaderLerp.cs.

FaderLerp

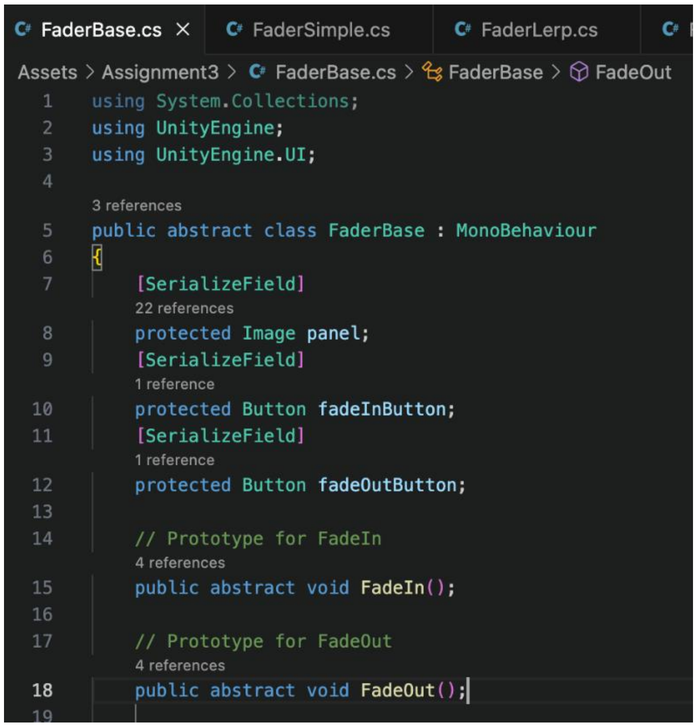
This script inherits from FaderBase and implements fade effects using linear interpolation (Mathf.Lerp). The fading is done over a specified duration (1 second by default) by smoothly interpolating the alpha value of the panel's color between 0 (transparent) and 1 (opaque). The fade effects are handled via IEnumerator coroutines for both FadeIn() and FadeOut().

FaderAnimationCurve

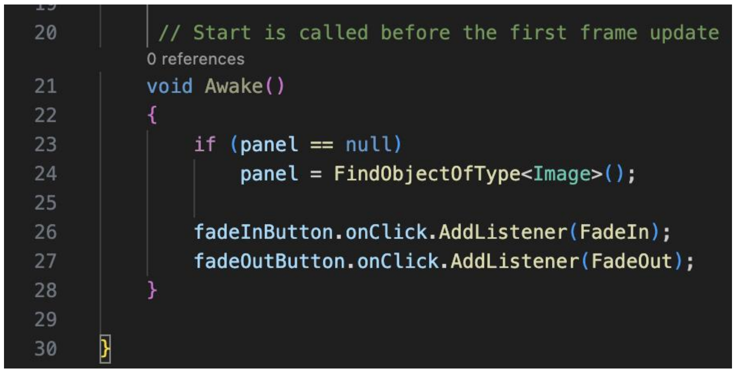
This script extends FaderBase and uses an AnimationCurve to control the fade effect. The curve allows for more complex and customizable fading behaviors, such as easing in and out. The alpha value is calculated by evaluating the curve at normalized time intervals (from 0 to 1) and then interpolating between the start and end alpha values. This approach provides more flexibility in controlling the fade dynamics.

Code:

FaderBase Class

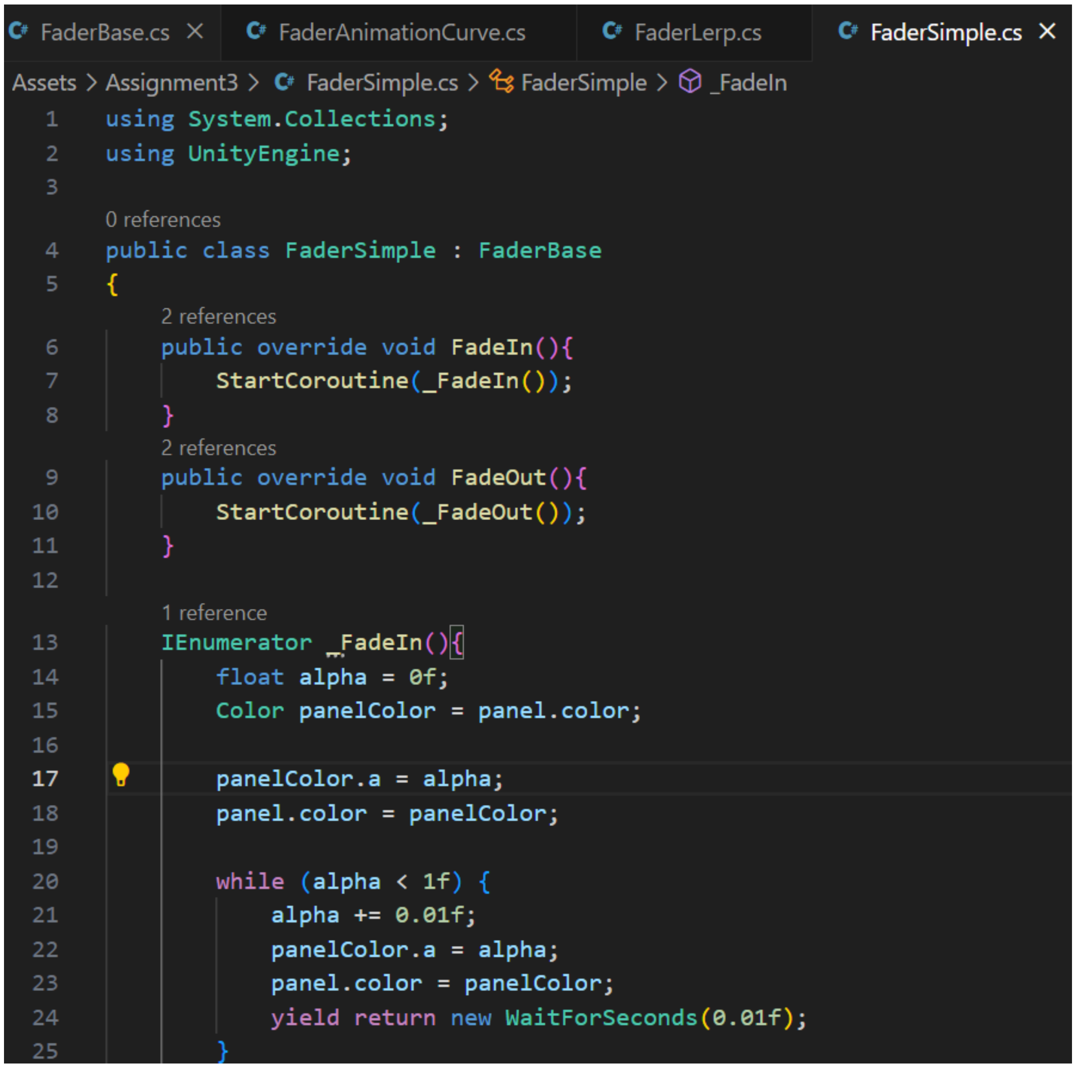


```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;

public abstract class FaderBase : MonoBehaviour
{
    [SerializeField]
    protected Image panel;
    [SerializeField]
    protected Button fadeInButton;
    [SerializeField]
    protected Button fadeOutButton;

    // Prototype for FadeIn
    public abstract void FadeIn();

    // Prototype for FadeOut
    public abstract void FadeOut();
```



```
    // Start is called before the first frame update
    0 references
    void Awake()
    {
        if (panel == null)
            panel = FindObjectOfType<Image>();

        fadeInButton.onClick.AddListener(FadeIn);
        fadeOutButton.onClick.AddListener(FadeOut);
    }

}
```

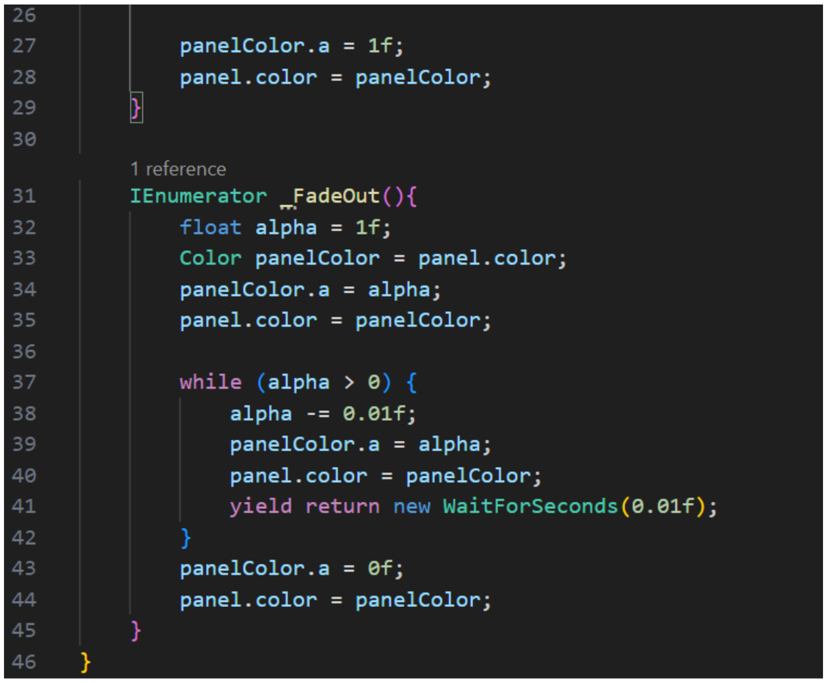
FaderSimple class



FaderBase.cs ✕ FaderAnimationCurve.cs FaderLerp.cs FaderSimple.cs ✕

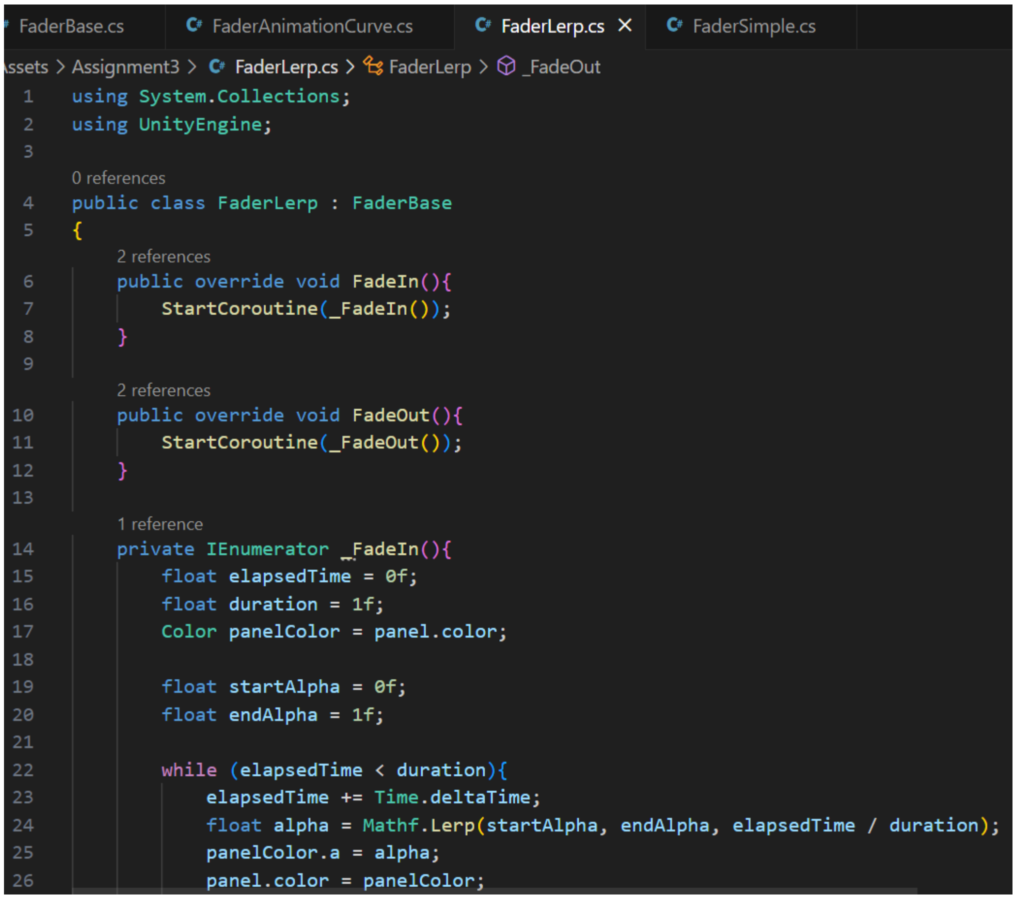
Assets > Assignment3 > FaderSimple.cs > FaderSimple > _FadeIn

```
using System.Collections;
using UnityEngine;

0 references
public class FaderSimple : FaderBase
{
    2 references
    public override void FadeIn(){
        StartCoroutine(_FadeIn());
    }
    2 references
    public override void FadeOut(){
        StartCoroutine(_FadeOut());
    }

    1 reference
    IEnumerator _FadeIn(){
        float alpha = 0f;
        Color panelColor = panel.color;

        panelColor.a = alpha;
        panel.color = panelColor;

        while (alpha < 1f) {
            alpha += 0.01f;
            panelColor.a = alpha;
            panel.color = panelColor;
            yield return new WaitForSeconds(0.01f);
        }
```



```
        panelColor.a = 1f;
        panel.color = panelColor;
    }

    1 reference
    IEnumerator _FadeOut(){
        float alpha = 1f;
        Color panelColor = panel.color;
        panelColor.a = alpha;
        panel.color = panelColor;

        while (alpha > 0) {
            alpha -= 0.01f;
            panelColor.a = alpha;
            panel.color = panelColor;
            yield return new WaitForSeconds(0.01f);
        }
        panelColor.a = 0f;
        panel.color = panelColor;
    }
}
```

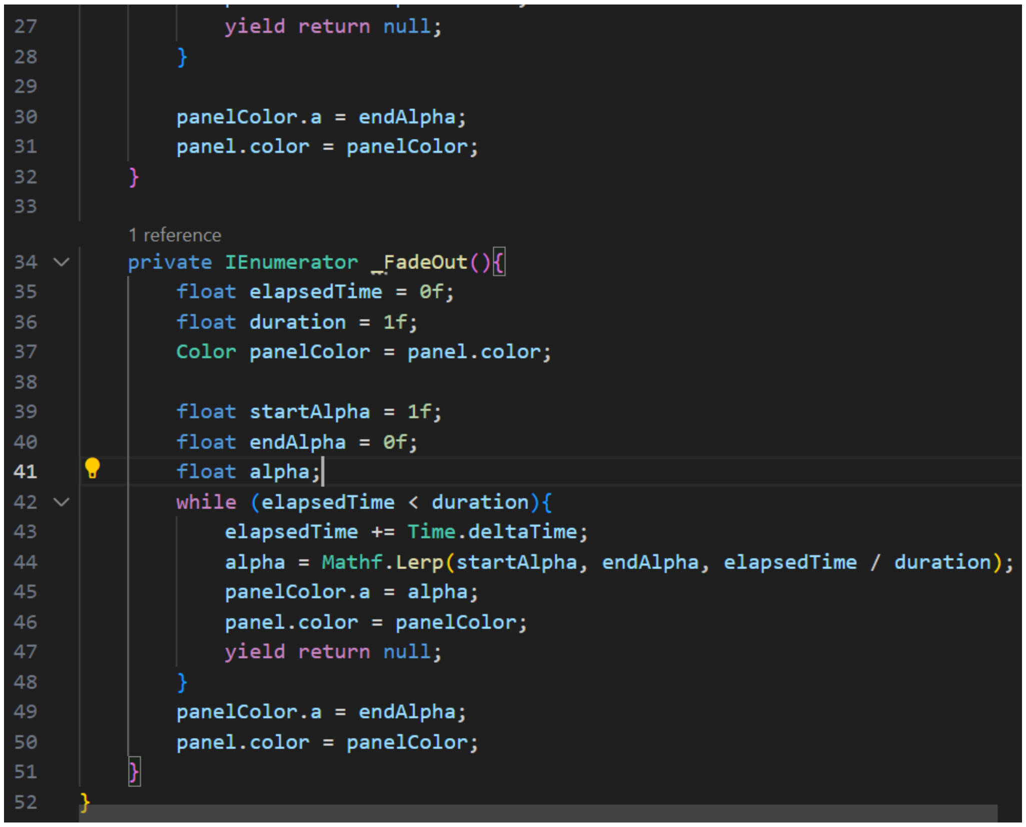
FaderLerp class



FaderBase.cs   FaderAnimationCurve.cs   FaderLerp.cs ×   FaderSimple.cs

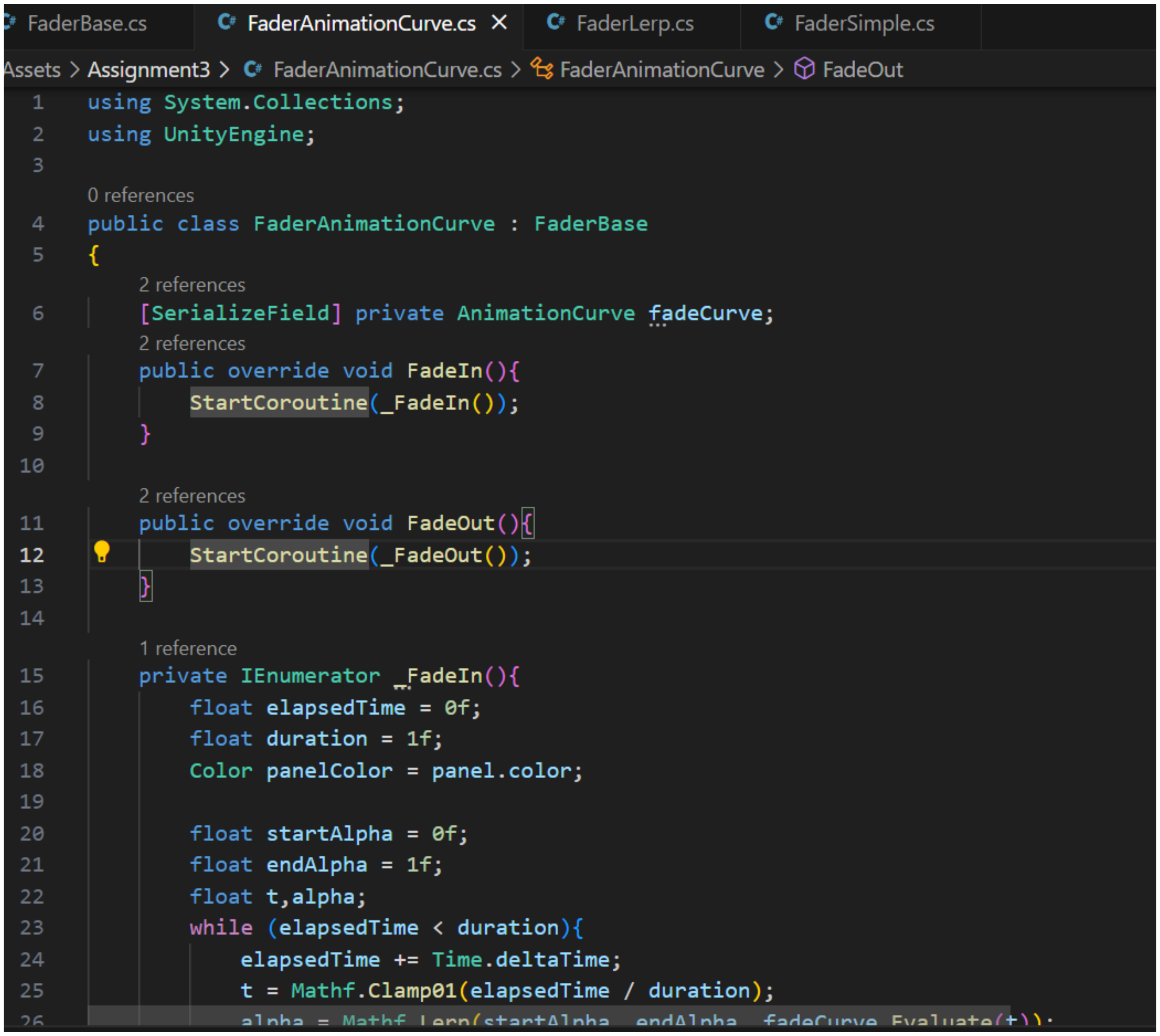
ssets > Assignment3 > FaderLerp.cs > FaderLerp > _FadeOut

```
using System.Collections;
using UnityEngine;

0 references
public class FaderLerp : FaderBase
{
    2 references
    public override void FadeIn(){
        StartCoroutine(_FadeIn());
    }

    2 references
    public override void FadeOut(){
        StartCoroutine(_FadeOut());
    }

    1 reference
    private IEnumerator _FadeIn(){
        float elapsedTime = 0f;
        float duration = 1f;
        Color panelColor = panel.color;

        float startAlpha = 0f;
        float endAlpha = 1f;

        while (elapsedTime < duration){
            elapsedTime += Time.deltaTime;
            float alpha = Mathf.Lerp(startAlpha, endAlpha, elapsedTime / duration);
            panelColor.a = alpha;
            panel.color = panelColor;
```

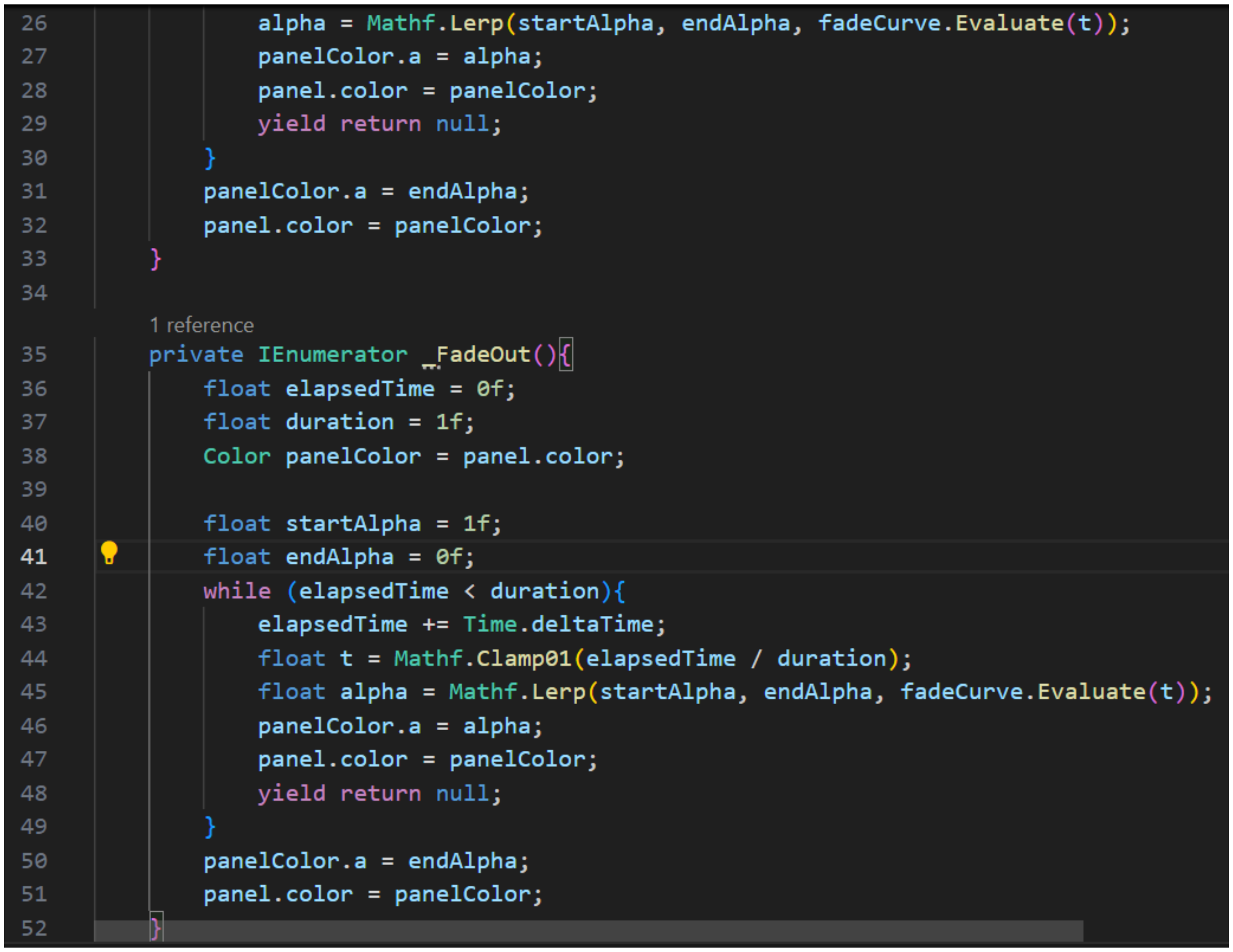


```
            yield return null;
        }

        panelColor.a = endAlpha;
        panel.color = panelColor;
    }

    1 reference
    private IEnumerator _FadeOut(){
        float elapsedTime = 0f;
        float duration = 1f;
        Color panelColor = panel.color;

        float startAlpha = 1f;
        float endAlpha = 0f;
        float alpha;
        while (elapsedTime < duration){
            elapsedTime += Time.deltaTime;
            alpha = Mathf.Lerp(startAlpha, endAlpha, elapsedTime / duration);
            panelColor.a = alpha;
            panel.color = panelColor;
            yield return null;
        }
        panelColor.a = endAlpha;
        panel.color = panelColor;
    }
}
```

FaderAnimationCurve class

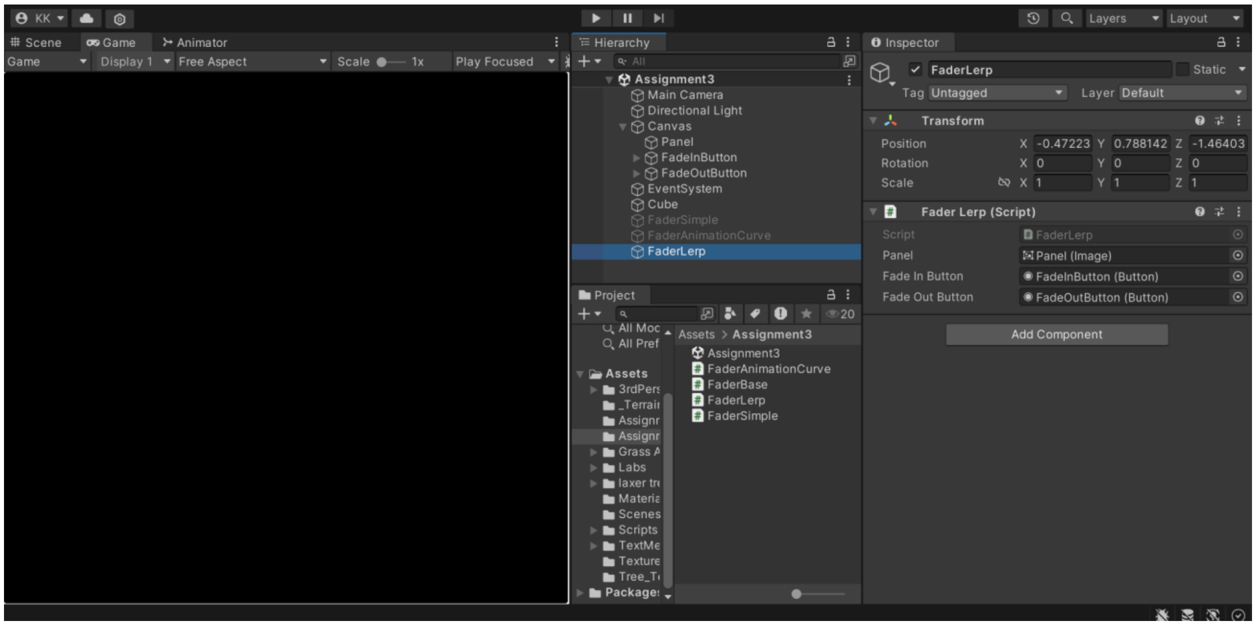

FaderBase.cs
FaderAnimationCurve.cs
FaderLerp.cs
FaderSimple.cs
Assets > Assignment3 > FaderAnimationCurve.cs > FaderAnimationCurve > FadeOut

```
using System.Collections;
using UnityEngine;

0 references
public class FaderAnimationCurve : FaderBase
{
        2 references
        [SerializeField] private AnimationCurve fadeCurve;
        2 references
        public override void FadeIn(){
            StartCoroutine(_FadeIn());
        }

        2 references
        public override void FadeOut(){
            StartCoroutine(_FadeOut());
        }

        1 reference
        private IEnumerator _FadeIn(){
            float elapsedTime = 0f;
            float duration = 1f;
            Color panelColor = panel.color;

            float startAlpha = 0f;
            float endAlpha = 1f;
            float t,alpha;
            while (elapsedTime < duration){
                elapsedTime += Time.deltaTime;
                t = Mathf.Clamp01(elapsedTime / duration);
                alpha = Mathf.Lerp(startAlpha, endAlpha, fadeCurve.Evaluate(t));
```



```
            alpha = Mathf.Lerp(startAlpha, endAlpha, fadeCurve.Evaluate(t));
            panelColor.a = alpha;
            panel.color = panelColor;
            yield return null;
        }
        panelColor.a = endAlpha;
        panel.color = panelColor;
    }

    1 reference
    private IEnumerator FadeOut(){
        float elapsedTime = 0f;
        float duration = 1f;
        Color panelColor = panel.color;

        float startAlpha = 1f;
        float endAlpha = 0f;
        while (elapsedTime < duration){
            elapsedTime += Time.deltaTime;
            float t = Mathf.Clamp01(elapsedTime / duration);
            float alpha = Mathf.Lerp(startAlpha, endAlpha, fadeCurve.Evaluate(t));
            panelColor.a = alpha;
            panel.color = panelColor;
            yield return null;
        }
        panelColor.a = endAlpha;
        panel.color = panelColor;
    }
```

Output:



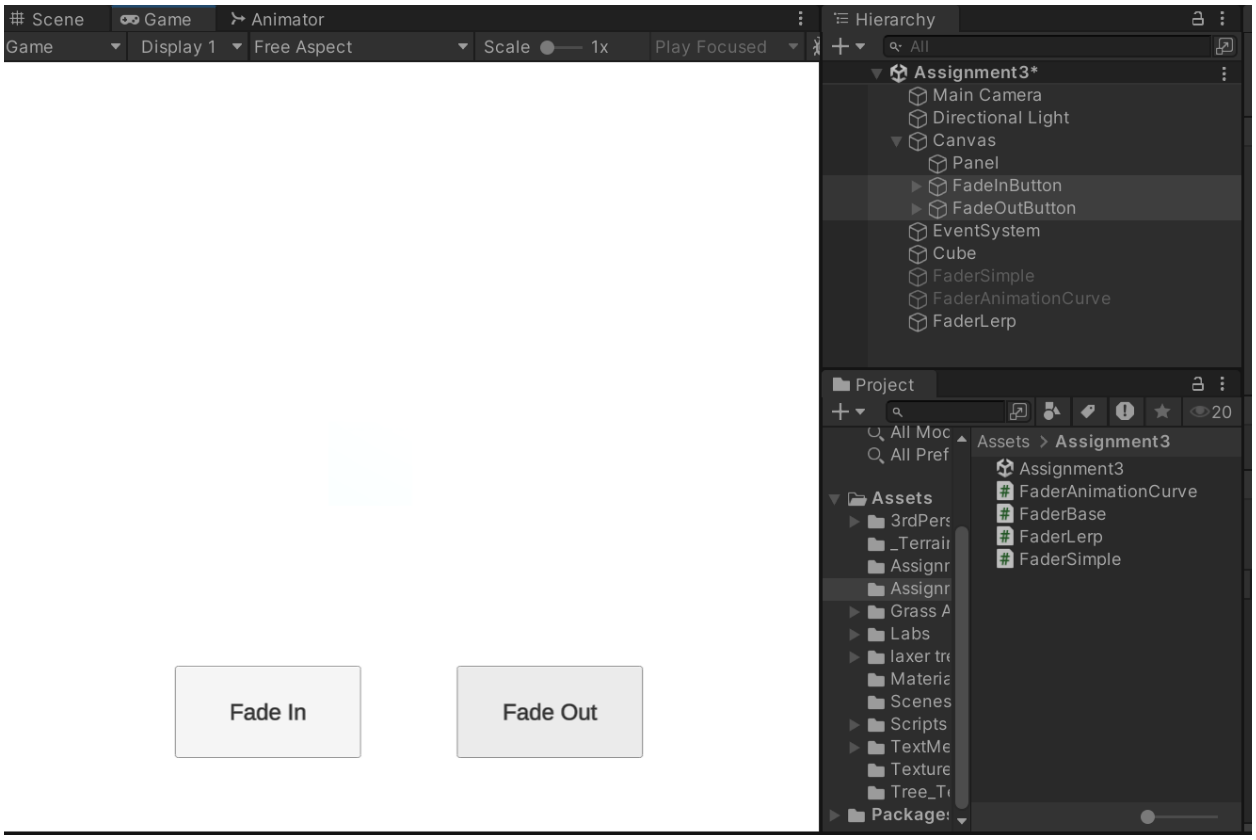
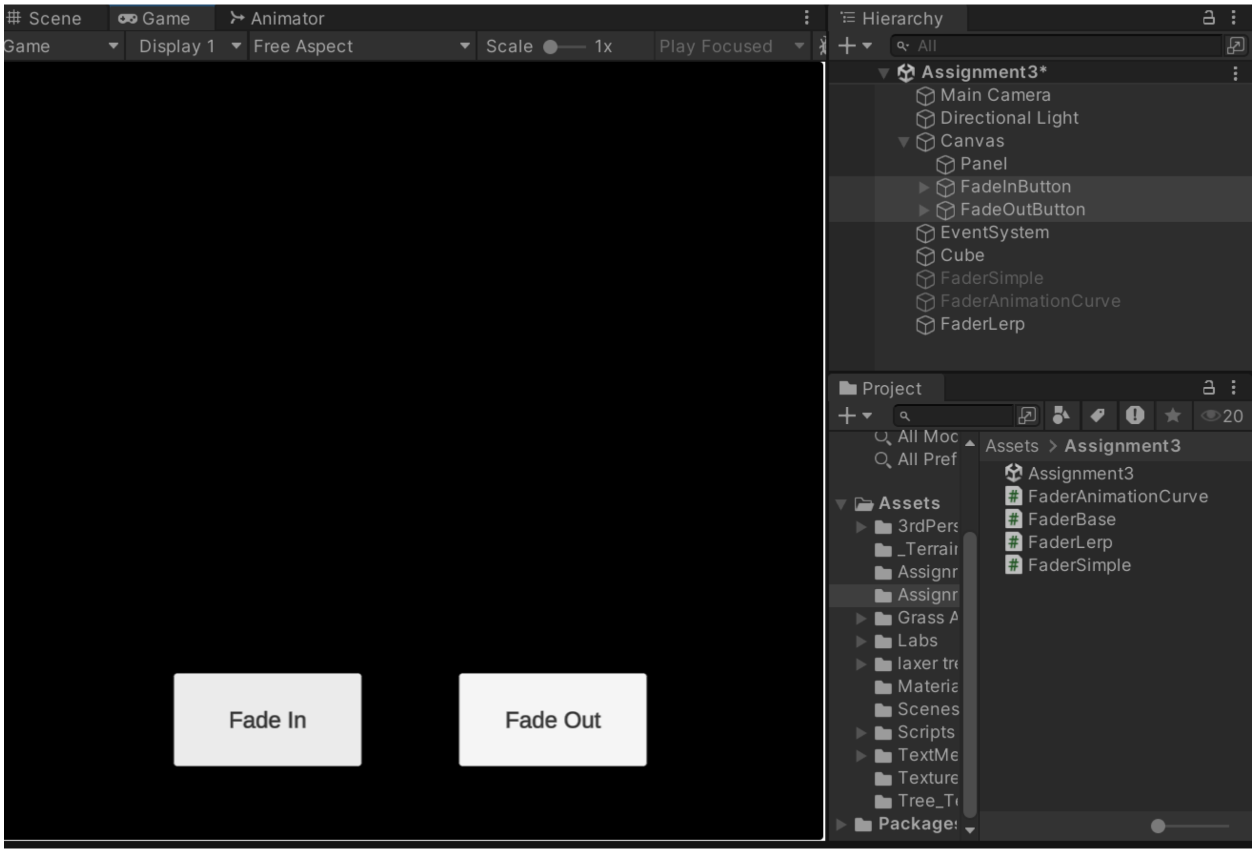
Hierarchy
- Assignment3
  - Main Camera
  - Directional Light
  - Canvas
    - Panel
    - FadeInButton
    - FadeOutButton
  - EventSystem
  - Cube
  - FaderSimple
  - FaderAnimationCurve
  - FaderLerp

Inspector: FaderLerp — Tag Untagged — Layer Default

Transform
- Position X -0.47223 Y 0.788142 Z -1.46403
- Rotation X 0 Y 0 Z 0
- Scale X 1 Y 1 Z 1

Fader Lerp (Script)
- Script: FaderLerp
- Panel: Panel (Image)
- Fade In Button: FadeInButton (Button)
- Fade Out Button: FadeOutButton (Button)

Project: Assets > Assignment3
- Assignment3
- FaderAnimationCurve
- FaderBase
- FaderLerp
- FaderSimple


Scene
Game
Animator
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Hierarchy
Assignment3*
Main Camera
Directional Light
Canvas
Panel
FadeInButton
FadeOutButton
EventSystem
Cube
FaderSimple
FaderAnimationCurve
FaderLerp
Project
Assets > Assignment3
Assignment3
FaderAnimationCurve
FaderBase
FaderLerp
FaderSimple
Fade In
Fade Out

Scene
Game
Animator
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Hierarchy
Assignment3*
Main Camera
Directional Light
Canvas
Panel
FadeInButton
FadeOutButton
EventSystem
Cube
FaderSimple
FaderAnimationCurve
FaderLerp
Project
Assets > Assignment3
Assignment3
FaderAnimationCurve
FaderBase
FaderLerp
FaderSimple
Fade In
Fade Out