**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:**

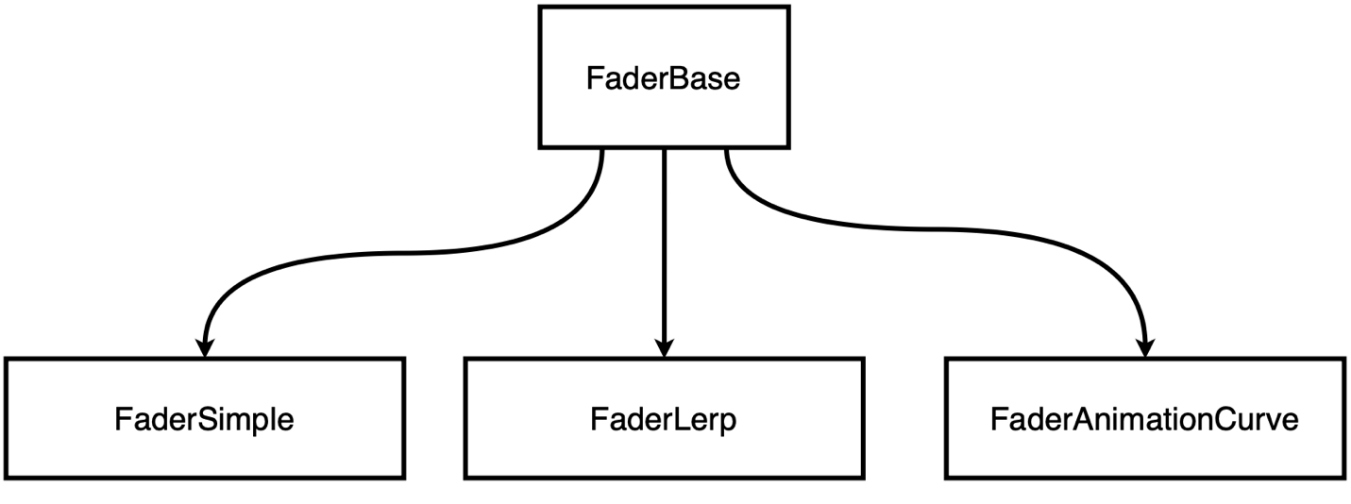


Figure 1: Code Architecture for Implementing Fading Logic using 3 different methods

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:**

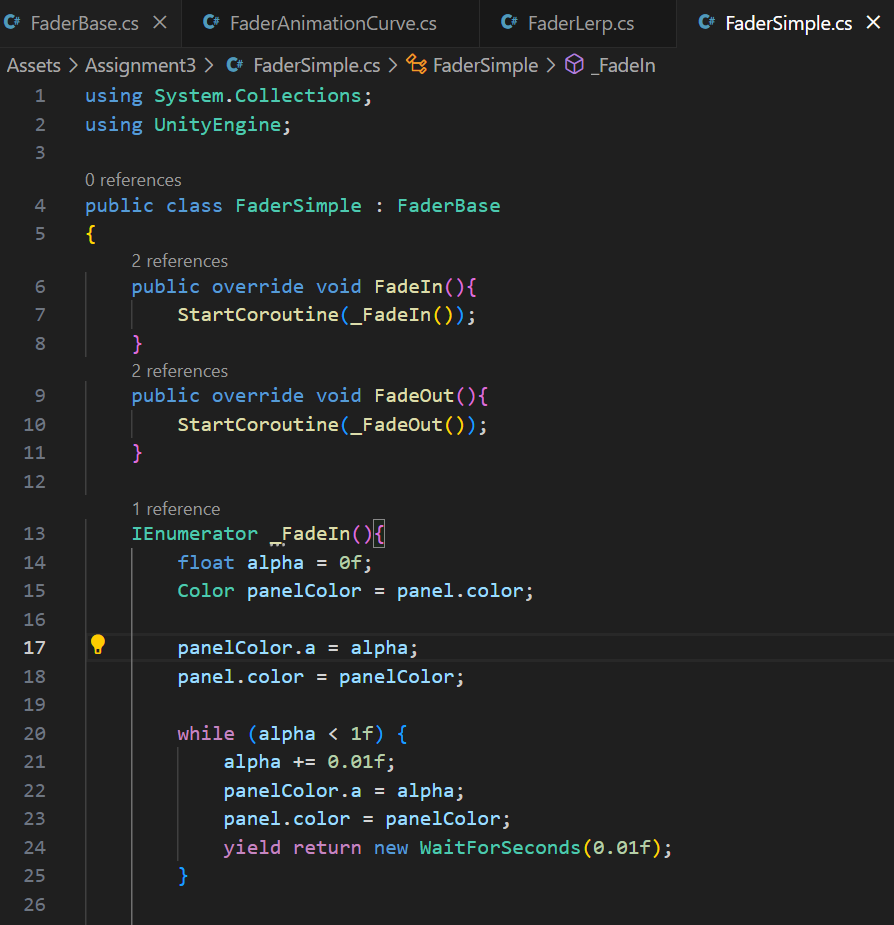
**A screenshot of a computer program

Description automatically generatedFaderBase Class**

**A screenshot of a computer program

Description automatically generated**

**FaderSimple class**

****

**A screen shot of a computer program

Description automatically generated**

**FaderLerp class**

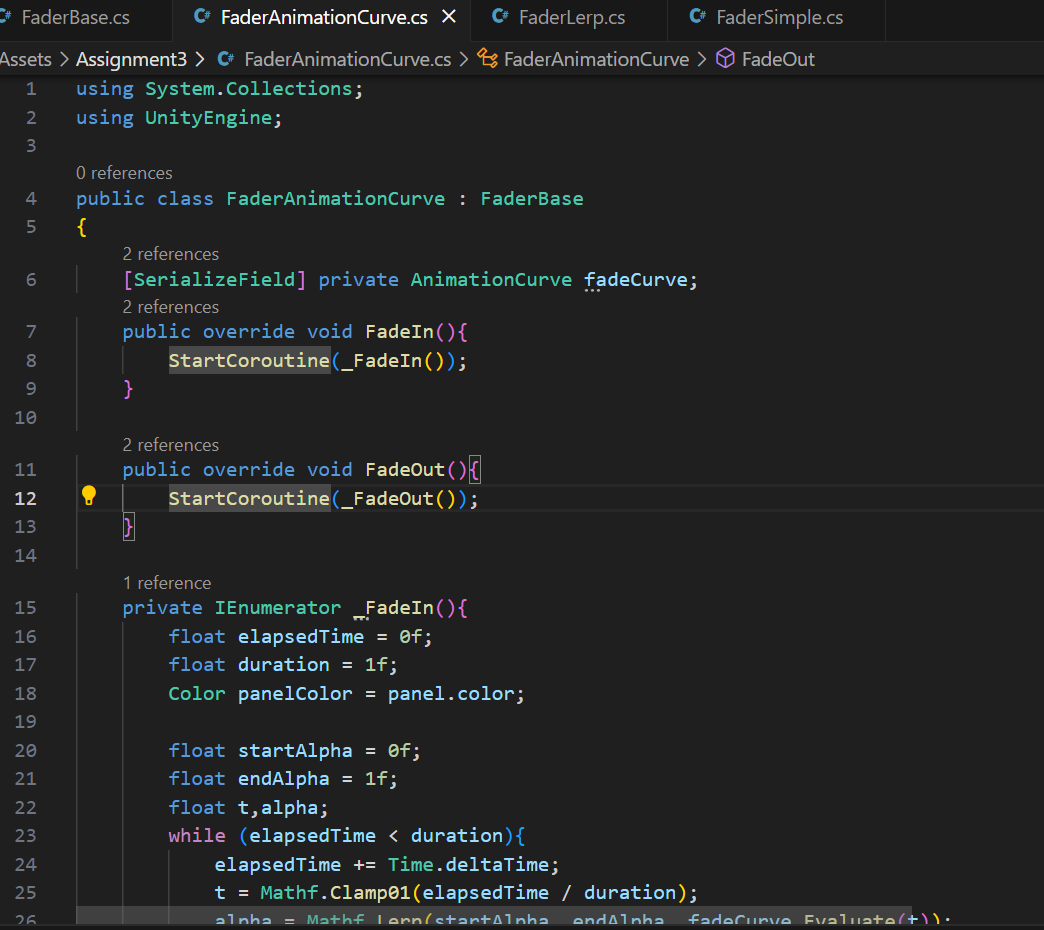
**A screenshot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**FaderAnimationCurve class**

****

**A screen shot of a computer program

Description automatically generated**

**Output:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**