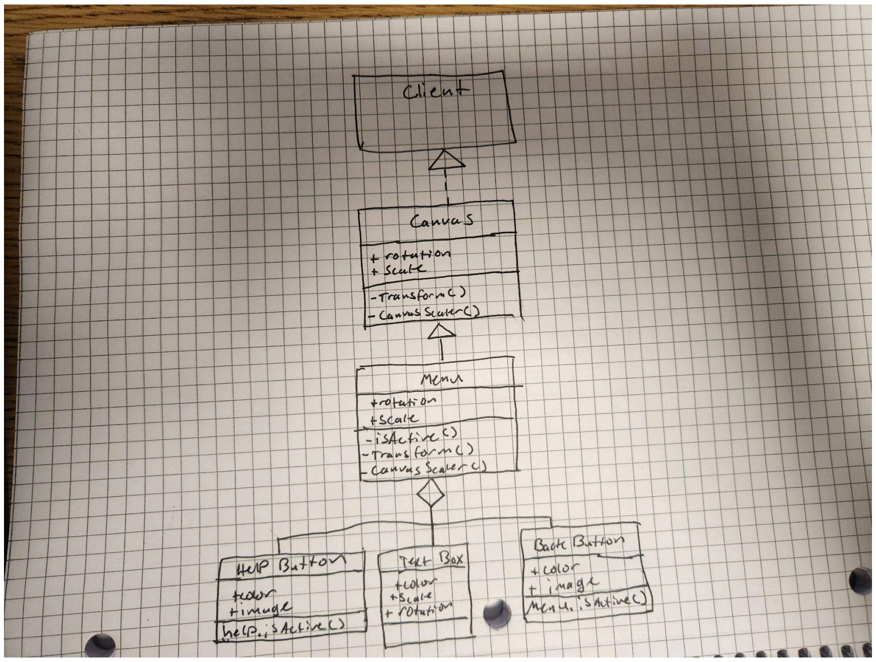
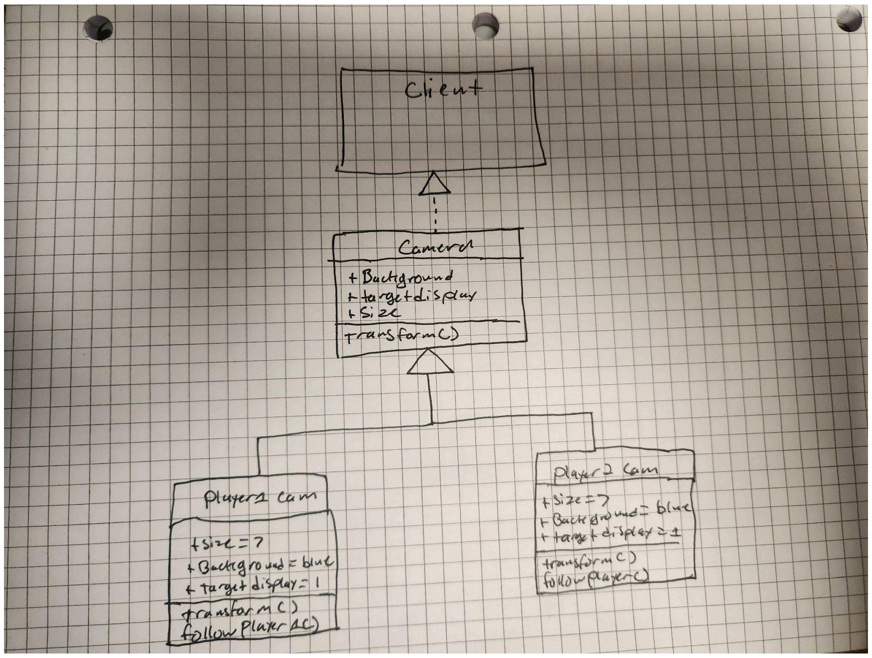
Chosen Design Pattern: Adapter Design Pattern

Justification for the Adapter Design Pattern. Unity has a lot of built in classes and features that us as the Software engineers don't need to use. Almost every class we use has 10's of features that are obsolete to us, and either would hinder our program or not affect it in a meaningful way. So we adapt it to fit in our program how we need. For the Camera, we take the big overarching class that is the camera system in Unity, and adapt it to be able to use 2 instances of it, change its limits of view, change how it reacts to move it, and adapt it to our use. For a menu, we take an overarching class called a "canvas" and we adapt it to our needs to act as a pause and help menu.





Addressing the Comments from the Assignment 4 feedback:

How classes are interacting with each other:
In the top image, the Menu mentioned is an adaptation or subclass of the Canvas class. It inherits the traits, but then some of them are changed with slightly different definitions.

In the bottom image, the Player 1 and Player 2 cameras are an adaptation of the Camera class in Unity, just changed to each cover only ½ of the screen. They are essentially the same adaptation of the camera, it's just 2 iterations of them.

Methods in the classes:

“help.isActive()” under “Help Button”: This method sets the screen showing the help information as active or not, essentially showing it to the player or not

“Transform()” under “Camera”: This method controls the position of the camera in space, given specific parameters or actions it can change the position.

“Menu.isActive()” under “Back Button”: Similar to the help.isActive() method, but is instead referring to the menu, so the back button would close the menu when clicked, which would stop showing it to the player.

“isActive()” unser “Menu”: Similar to the help.isActive() method, but is instead just referring to itself as whether it can be seen by the player or not.