**Open Ended Lab**



**Fall 2024**

**CSE-411L Intro to Game Development Lab**

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 January 2025**

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Objective:**

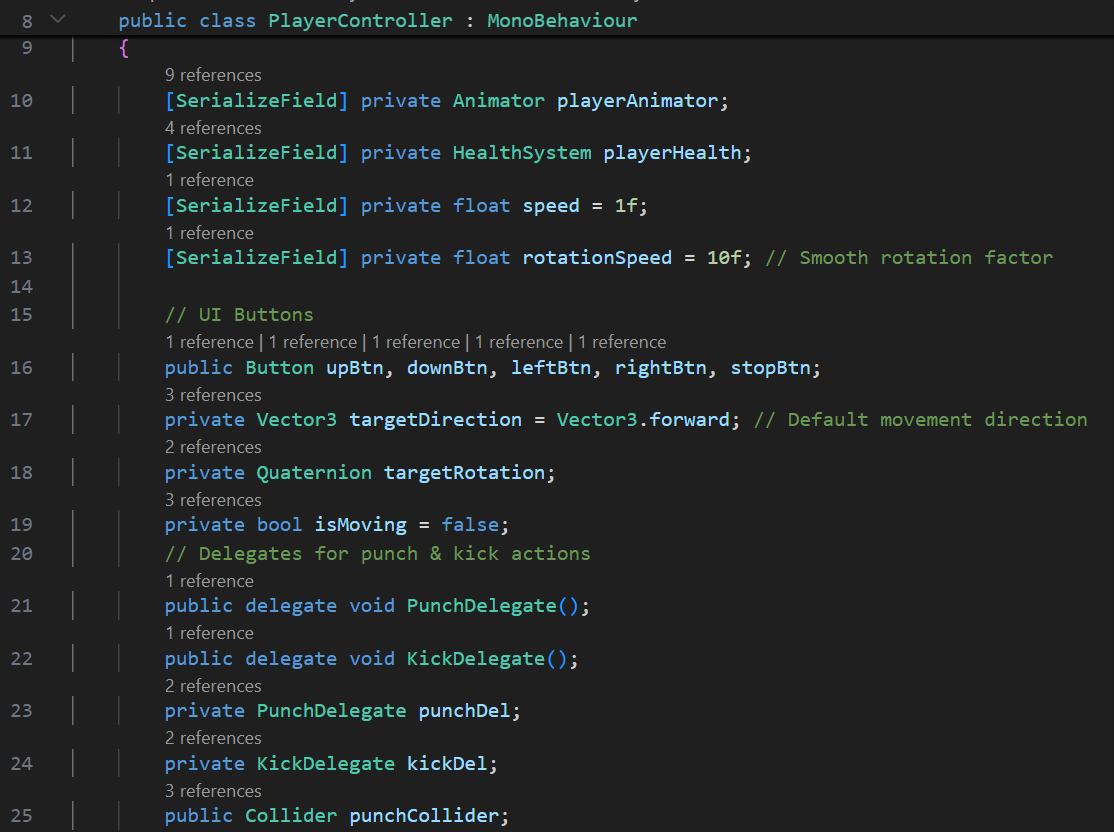
In this lab we further explored the Unity API.

**Tasks:**

1. Scene Setup:
   1. Create a new Unity scene with a thematic environment (e.g., a dojo, forest, or urban street).
   2. Add a textured plane to serve as the ground in the scene. Incorporate additional props such as fences, trees, or barrels to enhance the environment.
2. Character Setup:
   1. Download and import character models for the player and enemy from Mixamo.com.
   2. Assign appropriate animations (walk, idle, punch, kick) to both characters.
   3. Ensure the characters have humanoid rigging for animation compatibility.
3. Player Controls:
   1. Implement on-screen directional buttons using Unity’s UI system for forward, backward, left, and right movement. Bind these buttons to move the player character.
   2. Use the Unity Event System to handle button interactions.
   3. Enable mouse input to trigger the following animations:
   4. Pressing Mouse0 and Q triggers the player’s punch animation.
   5. Pressing Mouse0 and W triggers a player’s kick animation.
4. Enemy Behavior:
   1. Write an AI script that allows the enemy to follow the player if the distance between them is less than 5 units.
   2. When the enemy is close enough (e.g., within 1.5 units), it should automatically trigger the punch animation.
   3. If the player moves away and the distance exceeds 5 units, the enemy should return to its idle state.
5. Delegate System:
   1. Implement a delegate-based input system to handle the player’s actions:
   2. Use delegates for input detection to trigger punch and kick animations based on key combinations.
   3. Ensure the system is modular, allowing for easy addition of new animations or actions.
6. Advanced Animation Features:
   1. Add smooth animation transitions using Unity’s Animator Controller (e.g., transitioning between idle, walk, and attack states).
   2. Use animation events to sync visual effects or sound effects with specific frames of the animations (e.g., a punching sound during a punch).
7. Health System:
   1. Add a health system to both the player and the enemy:
   2. Display health bars above each character or on the UI.
   3. When the player or enemy lands a punch or kick, reduce the opponent’s health.
   4. Trigger a defeat animation when health reaches 0, and display a “Game Over” or “Victory” message.

**Code:**

**PlayerController class**

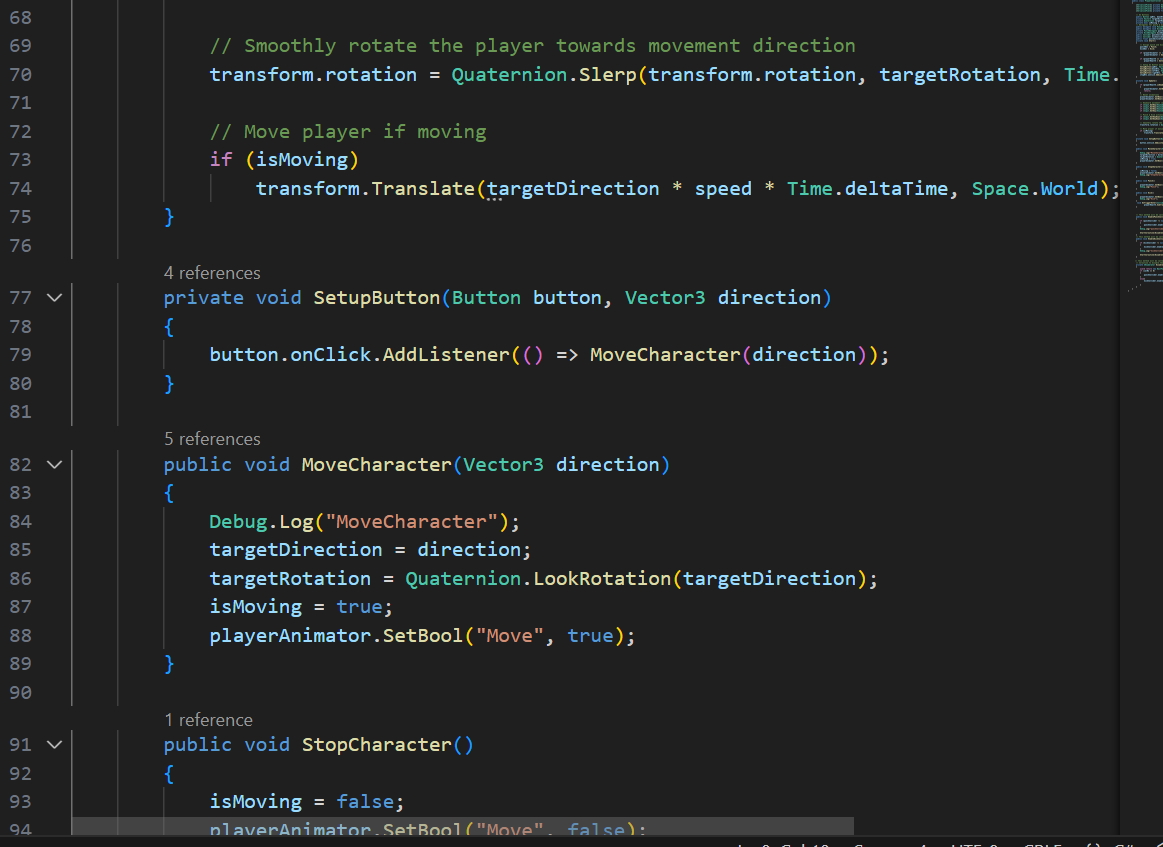
****

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

****

**A screen shot of a computer program

Description automatically generated**

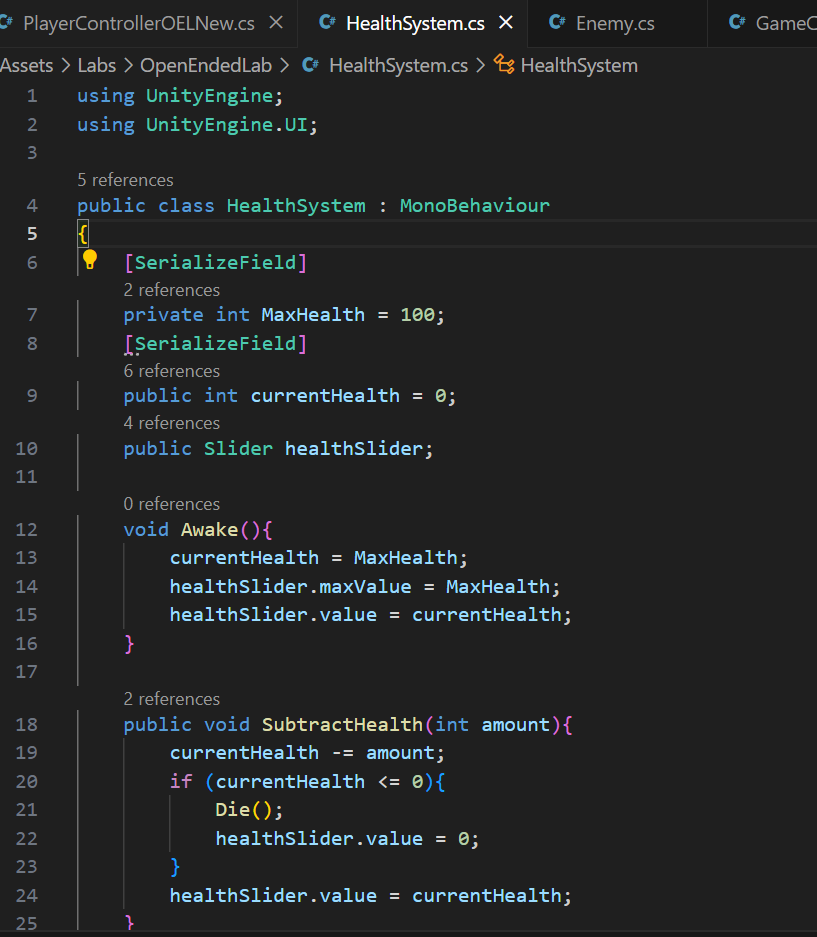
**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

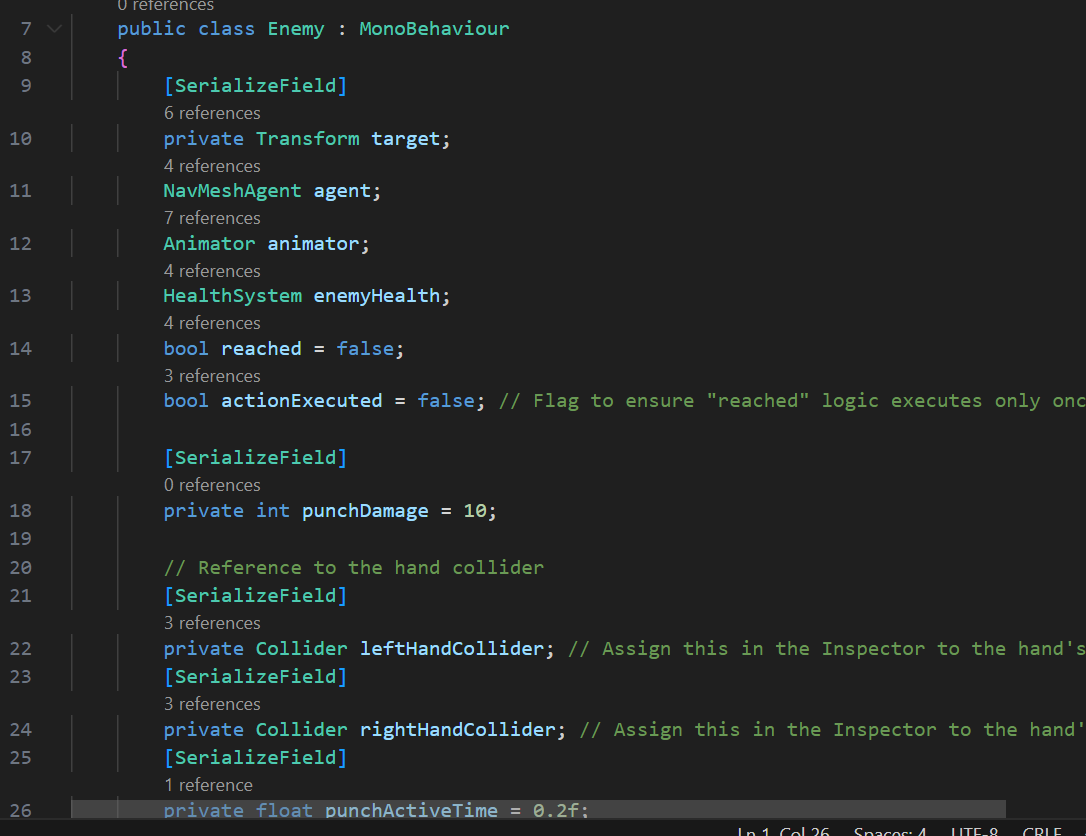
**HealthSystem class**

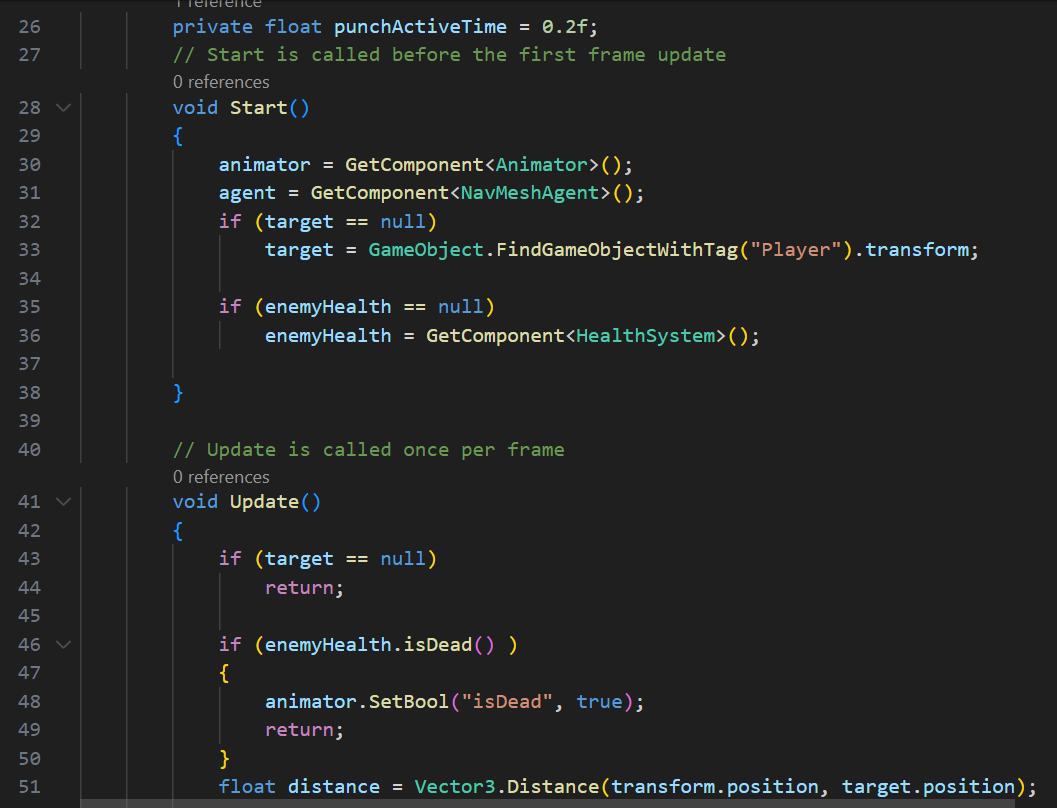
****

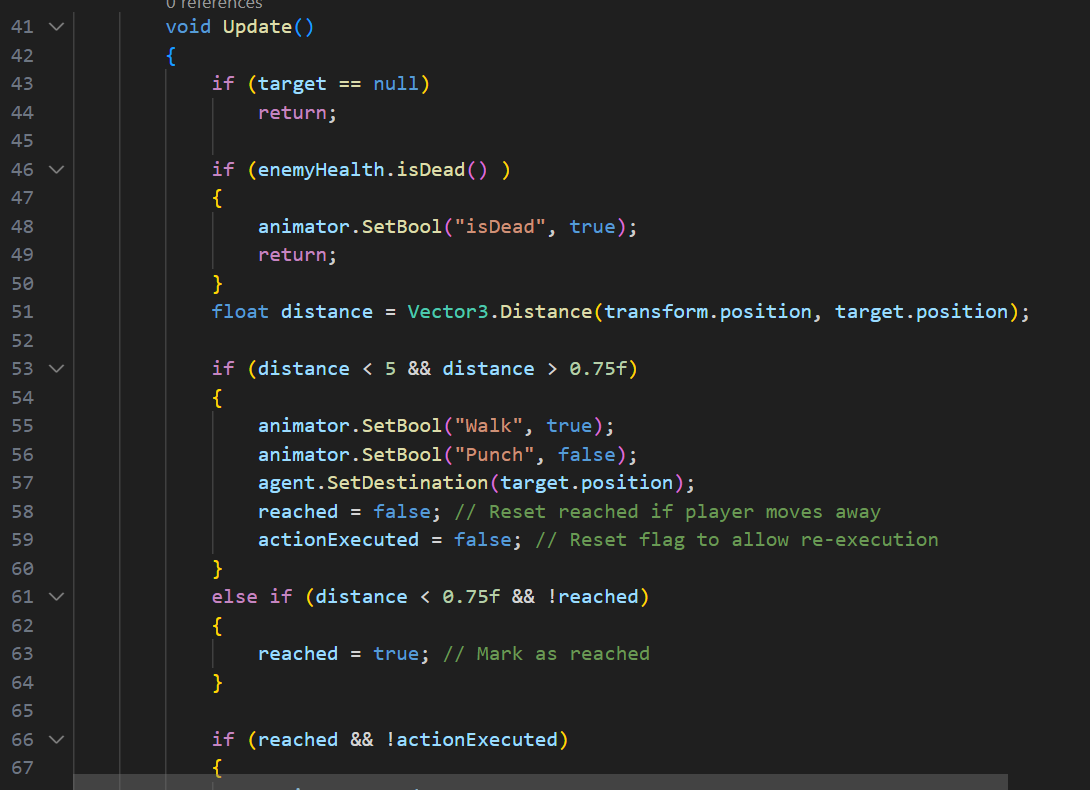
**A screen shot of a computer code

Description automatically generated**

**Enemy class**

****

****

****

**A screen shot of a computer program

Description automatically generated**

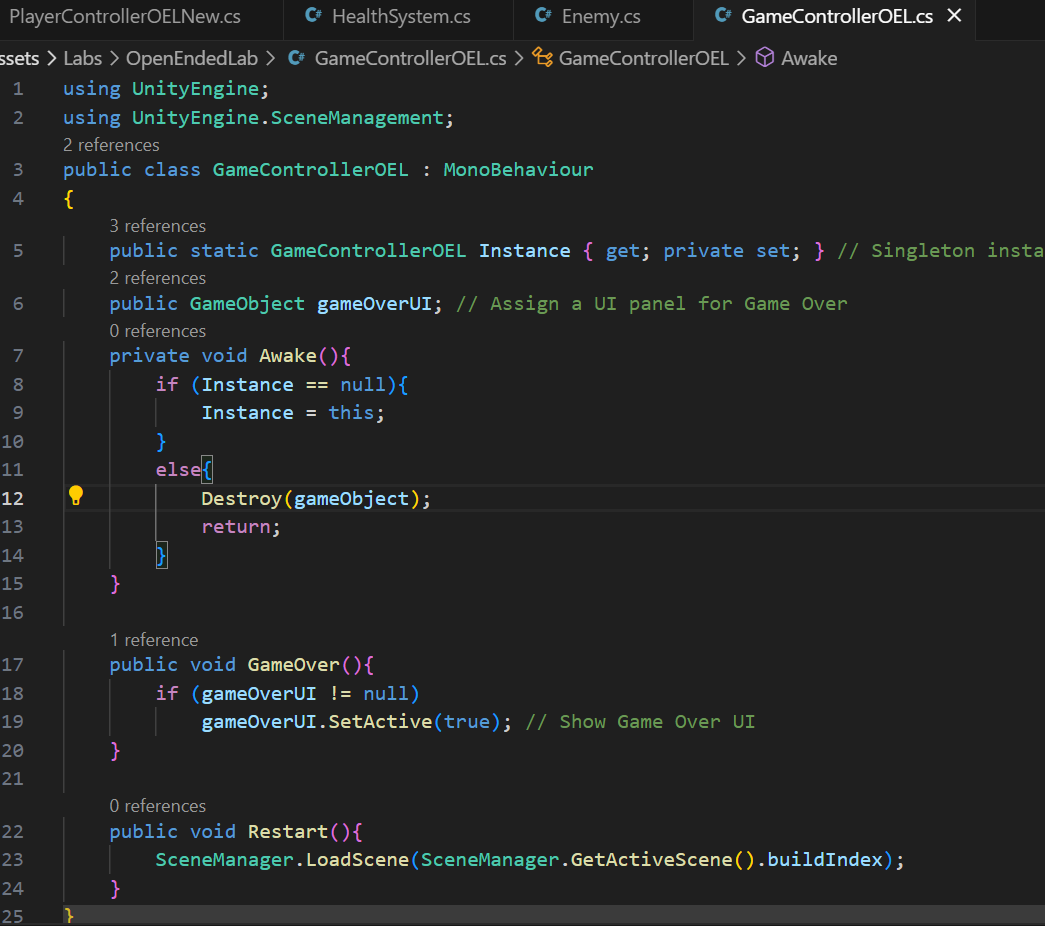
**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**GameController**

****

**Player Animator**

**A screenshot of a computer

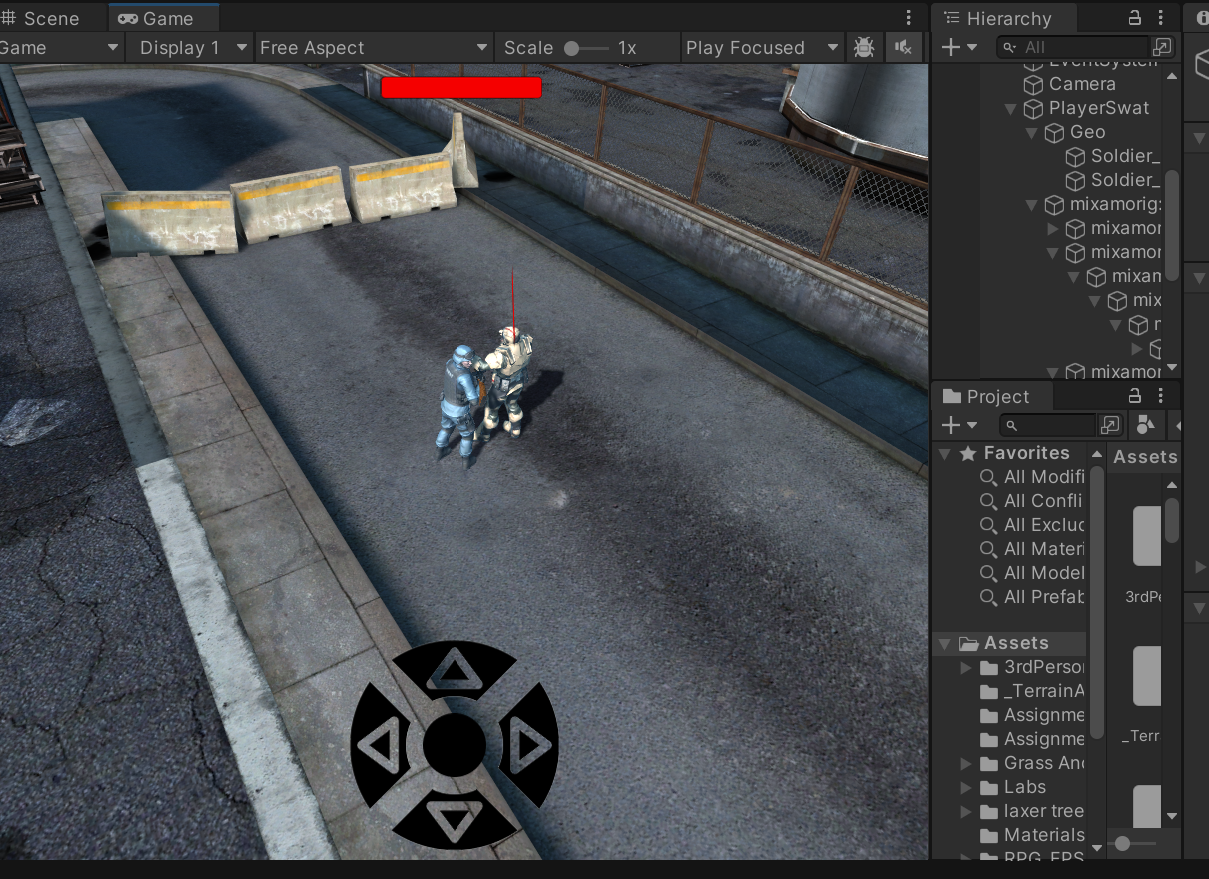
Description automatically generated**

**Enemy Animator**

**A screenshot of a computer

Description automatically generated**

**Output:**

****