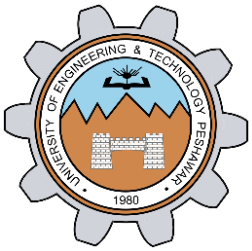
Open Ended Lab



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:
    a. Create a new Unity scene with a thematic environment (e.g., a dojo, forest, or urban street).
    b. 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:
    a. Download and import character models for the player and enemy from Mixamo.com.
    b. Assign appropriate animations (walk, idle, punch, kick) to both characters.
    c. Ensure the characters have humanoid rigging for animation compatibility.
3. Player Controls:
    a. 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.
    b. Use the Unity Event System to handle button interactions.
    c. Enable mouse input to trigger the following animations:
    d. Pressing Mouse0 and Q triggers the player's punch animation.
    e. Pressing Mouse0 and W triggers a player's kick animation.
4. Enemy Behavior:
    a. Write an AI script that allows the enemy to follow the player if the distance between them is less than 5 units.
    b. When the enemy is close enough (e.g., within 1.5 units), it should automatically trigger the punch animation.
    c. If the player moves away and the distance exceeds 5 units, the enemy should return to its idle state.
5. Delegate System:
    a. Implement a delegate-based input system to handle the player's actions:

    b. Use delegates for input detection to trigger punch and kick animations based on key combinations.
    c. Ensure the system is modular, allowing for easy addition of new animations or actions.
6. Advanced Animation Features:
    a. Add smooth animation transitions using Unity's Animator Controller (e.g., transitioning between idle, walk, and attack states).
    b. 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:
    a. Add a health system to both the player and the enemy:
    b. Display health bars above each character or on the UI.
    c. When the player or enemy lands a punch or kick, reduce the opponent's health.
    d. Trigger a defeat animation when health reaches 0, and display a "Game Over" or "Victory" message.

Code:

PlayerController class

```
8   public class PlayerController : MonoBehaviour
9   {
        9 references
10      [SerializeField] private Animator playerAnimator;
        4 references
11      [SerializeField] private HealthSystem playerHealth;
        1 reference
12      [SerializeField] private float speed = 1f;
        1 reference
13      [SerializeField] private float rotationSpeed = 10f; // Smooth rotation factor
14
15      // UI Buttons
        1 reference | 1 reference | 1 reference | 1 reference | 1 reference
16      public Button upBtn, downBtn, leftBtn, rightBtn, stopBtn;
        3 references
17      private Vector3 targetDirection = Vector3.forward; // Default movement direction
        2 references
18      private Quaternion targetRotation;
        3 references
19      private bool isMoving = false;
20      // Delegates for punch & kick actions
        1 reference
21      public delegate void PunchDelegate();
        1 reference
22      public delegate void KickDelegate();
        2 references
23      private PunchDelegate punchDel;
        2 references
24      private KickDelegate kickDel;
        3 references
25      public Collider punchCollider;
```

```
26      public Collider kickCollider;
        1 reference
27      public float punchActiveTime = 0.2f;
        0 references
28      private void Start()
29      {
30          // Assign punch and kick actions to delegates
31          punchDel = Punch;
32          kickDel = Kick;
33
34          if (playerAnimator == null)
35              playerAnimator = GetComponent<Animator>();
36
37          if (playerHealth == null)
38              playerHealth = GetComponent<HealthSystem>();
39
40          // Setup UI Button event listeners
41          SetupButton(upBtn, Vector3.forward);
42          SetupButton(downBtn, Vector3.back);
43          SetupButton(leftBtn, Vector3.left);
44          SetupButton(rightBtn, Vector3.right);
45          stopBtn.onClick.AddListener(StopCharacter);
46      }
47
        0 references
48      private void Update()
49      {
50          if (playerHealth.isDead() )
51          {
```

```
52              playerAnimator.SetBool("isDead", true);
53              return;
54          }
55          // Reset animations
56          playerAnimator.SetBool("Punching", false);
57          playerAnimator.SetBool("Kicking", false);
58
59          // Keyboard movement (optional)
60          if (Input.GetKey(KeyCode.W)) MoveCharacter(Vector3.forward);
61          if (Input.GetKey(KeyCode.S)) MoveCharacter(Vector3.back);
62          if (Input.GetKey(KeyCode.A)) MoveCharacter(Vector3.left);
63          if (Input.GetKey(KeyCode.D)) MoveCharacter(Vector3.right);
64
65          // Punch & Kick actions
66          if (Input.GetKeyDown(KeyCode.Q) && Input.GetMouseButton(0)) punchDel?.Invoke();
67          if (Input.GetKeyDown(KeyCode.E) && Input.GetMouseButton(0)) kickDel?.Invoke();
68
```

```
68
69          // Smoothly rotate the player towards movement direction
70          transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, Time.
71
72          // Move player if moving
73          if (isMoving)
74              transform.Translate(targetDirection * speed * Time.deltaTime, Space.World);
75      }
76
            4 references
77      private void SetupButton(Button button, Vector3 direction)
78      {
79          button.onClick.AddListener(() => MoveCharacter(direction));
80      }
81
            5 references
82      public void MoveCharacter(Vector3 direction)
83      {
84          Debug.Log("MoveCharacter");
85          targetDirection = direction;
86          targetRotation = Quaternion.LookRotation(targetDirection);
87          isMoving = true;
88          playerAnimator.SetBool("Move", true);
89      }
90
            1 reference
91      public void StopCharacter()
92      {
93          isMoving = false;
94          playerAnimator.SetBool("Move", false);
```

```
94          playerAnimator.SetBool("Move", false);
95          Debug.Log("StopCharacter");
96      }
97
            1 reference
98      public void Punch()
99      {
100         playerAnimator.SetBool("Punching", true);
101         Debug.Log("Punch");
102     }
103
            1 reference
104     public void Kick()
105     {
106         playerAnimator.SetBool("Kicking", true);
107         Debug.Log("Kick");
108     }
            0 references
109     void OnTriggerEnter(Collider other){
110             playerHealth.SubtractHealth(10);
111     }
```

```
115         // This method will be called by the animation event to enable the punch collider
            0 references
116         public void EnablePunchCollider()
117         {
118             if (punchCollider != null)
119             {
120                 punchCollider.enabled = true; // Enable the collider during the punch
121             }
122             Debug.Log("punchCollider");
123 
124             StartCoroutine(DisableColliderAfterDelay(0));
125         }
126         // This method will be called by the animation event to enable the punch collider
            0 references
127         public void EnableKickCollider()
128         {
129             if (kickCollider != null)
130             {
131                 kickCollider.enabled = true; // Enable the collider during the punch
132             }
133             Debug.Log("kickCollider");
134 
135             StartCoroutine(DisableColliderAfterDelay(1));
136         }
137 
138        // This method will be called by the animation event to disable the punch collider a
139         // Coroutine to disable the collider after a delay
            2 references
140         private IEnumerator DisableColliderAfterDelay(int colNo)
```

```
140         private IEnumerator DisableColliderAfterDelay(int colNo)
141         {
142             yield return new WaitForSeconds(punchActiveTime); // Wait for the specified tim
143             if (colNo == 0)
144             {
145                 punchCollider.enabled = false; // Disable the collider after the delay
146             }
147             else{
148                 kickCollider.enabled = false; // Disable the collider after the delay
149 
150             }
151         }
152     }
153 }
```

HealthSystem class

```
5 references
public class HealthSystem : MonoBehaviour
{
    [SerializeField]
    2 references
    private int MaxHealth = 100;
    [SerializeField]
    6 references
    public int currentHealth = 0;
    4 references
    public Slider healthSlider;

    0 references
    void Awake(){
        currentHealth = MaxHealth;
        healthSlider.maxValue = MaxHealth;
        healthSlider.value = currentHealth;
    }

    2 references
    public void SubtractHealth(int amount){
        currentHealth -= amount;
        if (currentHealth <= 0){
            Die();
            healthSlider.value = 0;
        }
        healthSlider.value = currentHealth;
    }
```

```
    1 reference
    void Die(){
        GameControllerOEL.Instance.GameOver();
        Debug.Log("DIED");
    }

    3 references
    public bool isDead(){
        if (currentHealth <= 0){
            return true;
        }
        else{
            return false;
        }
    }
}
```

Enemy class

```
0 references
public class Enemy : MonoBehaviour
{
    [SerializeField]
    6 references
    private Transform target;
    4 references
    NavMeshAgent agent;
    7 references
    Animator animator;
    4 references
    HealthSystem enemyHealth;
    4 references
    bool reached = false;
    3 references
    bool actionExecuted = false; // Flag to ensure "reached" logic executes only onc

    [SerializeField]
    0 references
    private int punchDamage = 10;

    // Reference to the hand collider
    [SerializeField]
    3 references
    private Collider leftHandCollider; // Assign this in the Inspector to the hand's
    [SerializeField]
    3 references
    private Collider rightHandCollider; // Assign this in the Inspector to the hand'
    [SerializeField]
    1 reference
    private float punchActiveTime = 0.2f;
```

```
    private float punchActiveTime = 0.2f;
    // Start is called before the first frame update
    0 references
    void Start()
    {
        animator = GetComponent<Animator>();
        agent = GetComponent<NavMeshAgent>();
        if (target == null)
            target = GameObject.FindGameObjectWithTag("Player").transform;

        if (enemyHealth == null)
            enemyHealth = GetComponent<HealthSystem>();

    }

    // Update is called once per frame
    0 references
```

```
0 references
void Update()
{
    if (target == null)
        return;

    if (enemyHealth.isDead() )
    {
        animator.SetBool("isDead", true);
        return;
    }
    float distance = Vector3.Distance(transform.position, target.position);

    if (distance < 5 && distance > 0.75f)
    {
        animator.SetBool("Walk", true);
        animator.SetBool("Punch", false);
        agent.SetDestination(target.position);
        reached = false; // Reset reached if player moves away
        actionExecuted = false; // Reset flag to allow re-execution
    }
    else if (distance < 0.75f && !reached)
    {
        reached = true; // Mark as reached
    }

    if (reached && !actionExecuted)
    {
```

```
    {
        actionExecuted = true;
        animator.SetBool("Walk", false);
        if (target.GetComponent<HealthSystem>().isDead())
        {
            animator.SetBool("Punch", false);
            agent.SetDestination(transform.position);

        }
        else
        {
            animator.SetBool("Punch", true);
            agent.SetDestination(transform.position);

        }
    }
}
0 references
void OnTriggerEnter(Collider other){
    if (other.gameObject.CompareTag("PlayerCollider"))
    {
        enemyHealth.SubtractHealth(100);
        Debug.Log("ENEMY HURT");

    }
}
```

```
        0 references
94      public void EnableLeftPunchCollider()
95      {
96          if (leftHandCollider != null)
97          {
98              leftHandCollider.enabled = true; // Enable the collider during the punch
99          }
100         Debug.Log("EnableLeftPunchCollider");
101
102         StartCoroutine(DisablePunchColliderAfterDelay(0));
103     }
104
105
106     // This method will be called by the animation event to enable the punch collider
        0 references
107     public void EnableRightPunchCollider()
108     {
109         if (rightHandCollider != null)
110         {
111             rightHandCollider.enabled = true; // Enable the collider during the punch
112         }
113         Debug.Log("EnableRightPunchCollider");
114         StartCoroutine(DisablePunchColliderAfterDelay(1));
115     }
116
117     // This method will be called by the animation event to disable the punch collide
118     // Coroutine to disable the collider after a delay
        2 references
```

```
        2 references
119     private IEnumerator DisablePunchColliderAfterDelay(int handCollider)
120     {
121         yield return new WaitForSeconds(punchActiveTime); // Wait for the specified tim
122         if (handCollider == 0)
123         {
124             leftHandCollider.enabled = false; // Disable the collider after the delay
125         }
126         else{
127             rightHandCollider.enabled = false; // Disable the collider after the delay
128
129         }
130     }
131 }
132 }
133
```

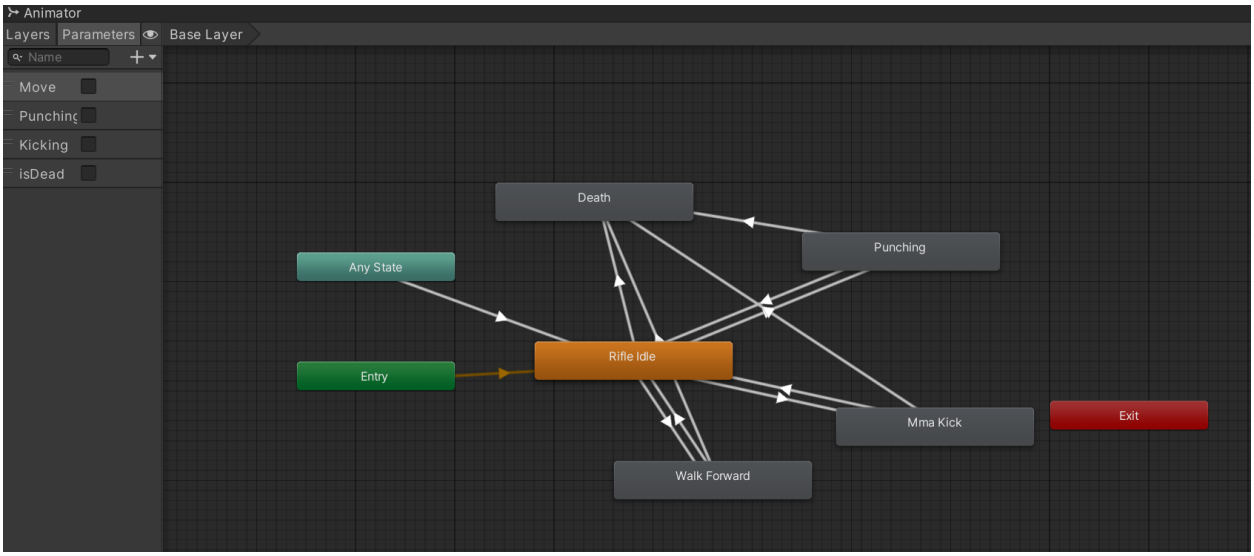
GameController

PlayerControllerOELNew.cs | HealthSystem.cs | Enemy.cs | GameControllerOEL.cs

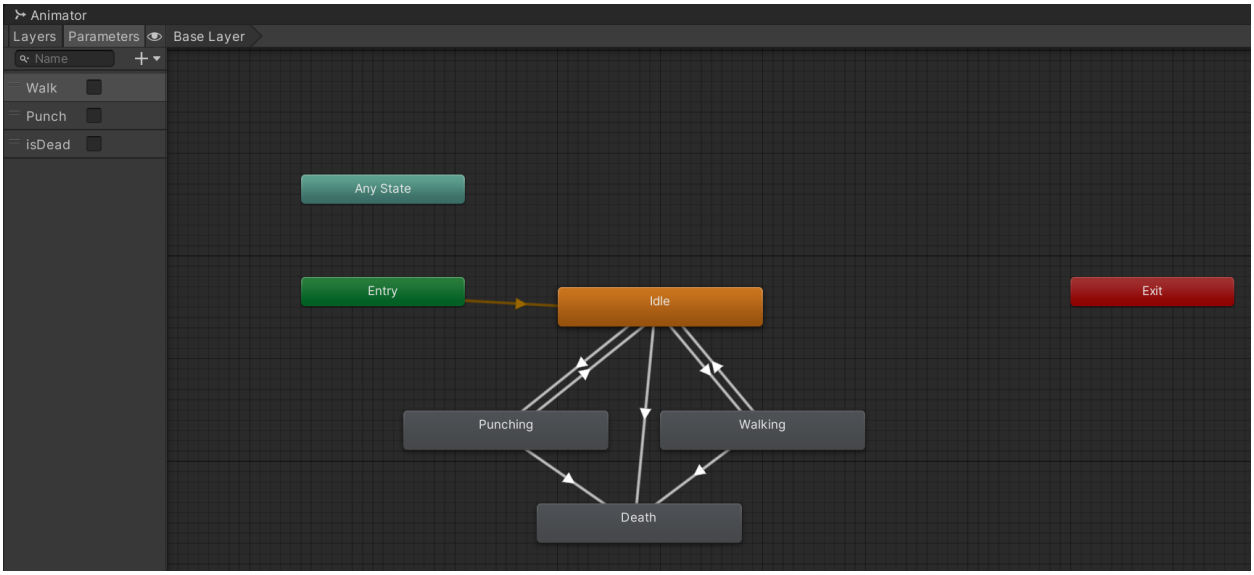
ssets > Labs > OpenEndedLab > GameControllerOEL.cs > GameControllerOEL > Awake

```
1   using UnityEngine;
2   using UnityEngine.SceneManagement;
    2 references
3   public class GameControllerOEL : MonoBehaviour
4   {
        3 references
5       public static GameControllerOEL Instance { get; private set; } // Singleton insta
        2 references
6       public GameObject gameOverUI; // Assign a UI panel for Game Over
        0 references
7       private void Awake(){
8           if (Instance == null){
9               Instance = this;
10          }
11          else{
12              Destroy(gameObject);
13              return;
14          }
15      }
16
        1 reference
17      public void GameOver(){
18          if (gameOverUI != null)
19              gameOverUI.SetActive(true); // Show Game Over UI
20      }
21
        0 references
22      public void Restart(){
23          SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
24      }
25  }
```

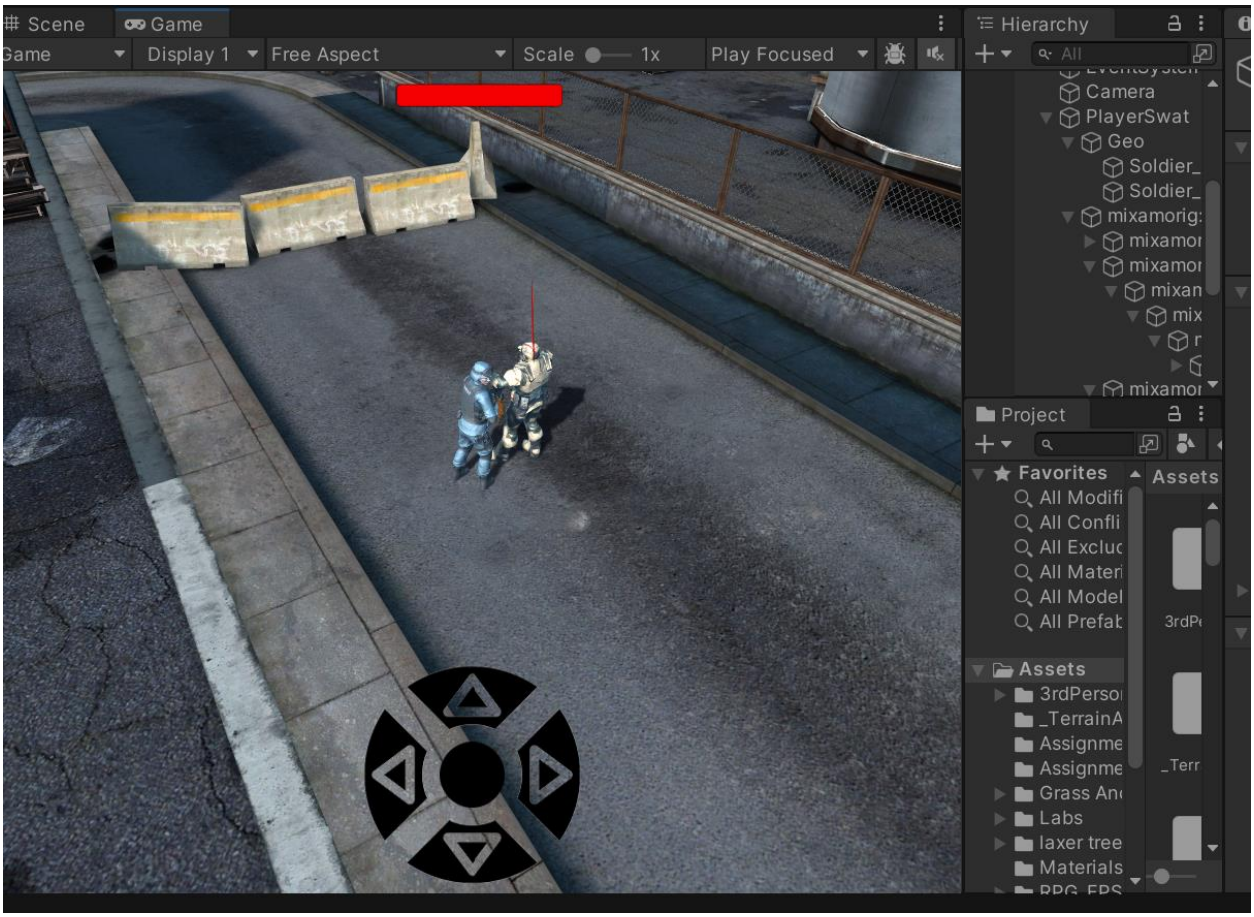
Player Animator



Enemy Animator



Output:


Scene
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Hierarchy
Camera
PlayerSwat
Geo
Soldier_
Soldier_
Project
Favorites
Assets
Assets
Labs
Materials