Unity C# Advanced

LAB # 10

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:

17th January 2025

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Objective:

In this lab we further explored the Unity API.

Tasks:

- Create a new Unity scene.
- Add a simple plane to serve as the ground in the scene.
- Use character models from Mixamo.com to create a player and an enemy.
- Both the player and the enemy characters should have walk, idle, and punch animations.
- When the game starts, the enemy should follow the player if the distance between them is less than 5 units.
- When the enemy gets close enough to the player, it should trigger the punch animation.
- The player should also be able to trigger the punch animation.
- Use delegates in such a way that:
- Pressing **Mouse0** and **Q** triggers the player's punch animation.
- Pressing **Mouse0** and **W** triggers another animation, such as a kick.

Code:

PlayerController class

Enemy.cs | PlayerController.cs

Assets > Labs > Lab9 > Scripts > PlayerController.cs > PlayerController > punchDel

```
using UnityEngine;

namespace Lab9{
    2 references
    public class PlayerController : MonoBehaviour{
        [SerializeField]
        8 references
        private Animator playerAnimator;

        [SerializeField]
        1 reference
        private float speed = 1f;

        [SerializeField]
        1 reference
        private float rotationSpeed = 10f; // Determines how smooth the rotation is

        4 references
        private Vector3 targetDirection;   // The direction the player is moving
        2 references
        private Quaternion targetRotation; // The target rotation for the player
```

```
    public delegate void PunchDelegate();
    0 references
    public delegate void KickDelegate();
    2 references
    PunchDelegate kickDel;
    3 references
    PunchDelegate punchDel;
    3 references
    private bool isMoving = false;

    0 references
    void Start(){
        // Assign the delegate to the Punch method
        punchDel = Punch;
        kickDel = Kick;

        Debug.Log(punchDel.Method);
        if (playerAnimator == null)
            playerAnimator = GetComponent<Animator>();

        targetDirection = Vector3.forward; // Default direction
    }

    0 references
    void Update(){
        //playerAnimator.SetBool("Move", false);
        playerAnimator.SetBool("Punching", false);
        playerAnimator.SetBool("Kicking", false);

        if (Input.GetKey(KeyCode.W))
```

```
        if (Input.GetKey(KeyCode.W))
            MoveCharacter(Vector3.forward);

        if (Input.GetKey(KeyCode.S))
            MoveCharacter(Vector3.back);

        if (Input.GetKey(KeyCode.A))
            MoveCharacter(Vector3.left);

        if (Input.GetKey(KeyCode.D))
            MoveCharacter(Vector3.right);

        //Debug.Log(Input.GetMouseButton(0));
        // Use GetKeyDown to call the delegate once per key press
        if (Input.GetKeyDown(KeyCode.Q) && Input.GetMouseButton(0))
            punchDel?.Invoke();


        // Use GetKeyDown to call the delegate once per key press
        if (Input.GetKeyDown(KeyCode.E) && Input.GetMouseButton(0))
            kickDel?.Invoke();

        // Smoothly rotate the character to the target rotation
        transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, Time.

        if(isMoving)
            // Move the character in the target direction
            transform.Translate(targetDirection * speed * Time.deltaTime, Space.World);
    }
```

```
        public void MoveCharacter(Vector3 direction){
            Debug.Log("MoveCharacter");
            // Set the target direction
            targetDirection = direction;

            // Calculate the target rotation based on the direction
            targetRotation = Quaternion.LookRotation(targetDirection);

            // Move the character in the target direction
            isMoving = true;
            playerAnimator.SetBool("Move", true);
        }

        1 reference
        public void StopCharacter(){
            isMoving = false;
            playerAnimator.SetBool("Move", false);
        }

        1 reference
        public void Punch(){
            playerAnimator.SetBool("Punching", true);
            Debug.Log("Punch"); // Check the Unity Console for this log message
        }
        1 reference
        public void Kick(){
            playerAnimator.SetBool("Kicking", true);
            Debug.Log("Kick"); // Check the Unity Console for this log message
        }
    }
```

PlayerInputs class

```
using UnityEngine;
using UnityEngine.EventSystems;

namespace Lab10{

0 references
public class PlayerInputs : MonoBehaviour{

    7 references
    public Lab9.PlayerController playerController;
    1 reference | 1 reference | 1 reference | 1 reference
    public GameObject upBtn, downBtn, leftBtn, rightBtn;
    1 reference
    public GameObject stopBtn;
    5 references
    public delegate void MovementAction(Vector3 direction);
    2 references
    public delegate void StopCharacter();

    // Start is called before the first frame update
0 references
void Start(){
    if (playerController == null)
        playerController = GetComponent<Lab9.PlayerController>();
```

```
        // Define movement actions for each button
        MovementAction moveForward = playerController.MoveCharacter;
        MovementAction moveBackward = playerController.MoveCharacter;
        MovementAction moveLeft = playerController.MoveCharacter;
        MovementAction moveRight = playerController.MoveCharacter;
        StopCharacter stop = playerController.StopCharacter;

        // Set up the buttons with their respective movement directions
        SetupButtonMovement(upBtn, moveForward, Vector3.forward);
        SetupButtonMovement(downBtn, moveBackward, Vector3.back);
        SetupButtonMovement(leftBtn, moveLeft, Vector3.left);
        SetupButtonMovement(rightBtn, moveRight, Vector3.right);

        // Set up the stop button
        SetupStopButton(stopBtn, stop);
    }

    4 references
    private void SetupButtonMovement(GameObject button, MovementAction action, Vector3 directio
        EventTrigger trigger = button.AddComponent<EventTrigger>();
        EventTrigger.Entry entry = new EventTrigger.Entry();
        entry.eventID = EventTriggerType.PointerDown;

        // Use the delegate with a lambda to pass parameters
        entry.callback.AddListener((data) => action(direction));

        trigger.triggers.Add(entry);
    }
```

```

    1 reference
    private void SetupStopButton(GameObject button, StopCharacter stopAction){
        EventTrigger trigger = button.AddComponent<EventTrigger>();
        EventTrigger.Entry entry = new EventTrigger.Entry();
        entry.eventID = EventTriggerType.PointerDown;

        // Add the stopAction delegate directly
        entry.callback.AddListener((data) => stopAction());
        trigger.triggers.Add(entry);
    }
    }
    }
```

Enemy class

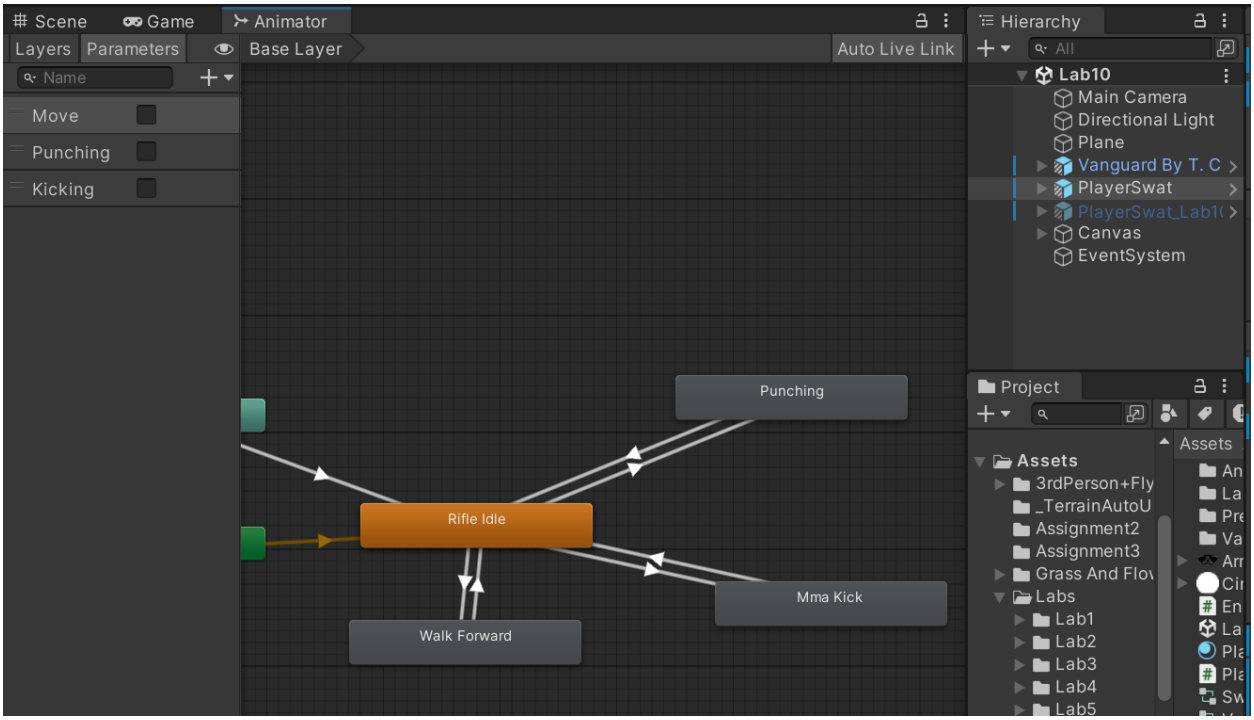
Enemy.cs

Assets > Labs > Lab10 > Enemy.cs > Enemy

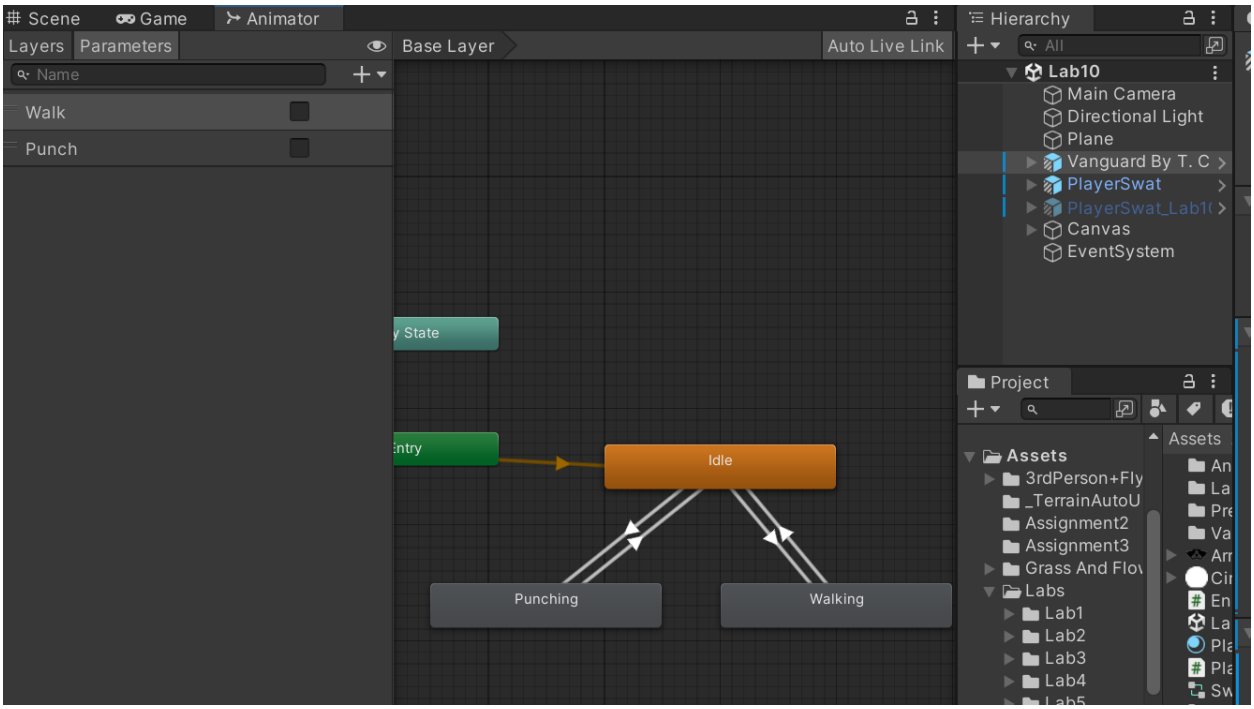
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;

namespace Lab10{
    0 references
    public class Enemy : MonoBehaviour{
        [SerializeField]
        5 references
        private Transform target;
        3 references
        NavMeshAgent agent;
        5 references
        Animator animator;

        4 references
        bool reached = false;
        3 references
        bool actionExecuted = false; // Flag to ensure "reached" logic executes only once

        // 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;
        }
```

```
        0 references
        void Update(){
            if (target == null)
                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);
                animator.SetBool("Punch", true);
                agent.SetDestination(transform.position);
            }
        }
    }
}
```

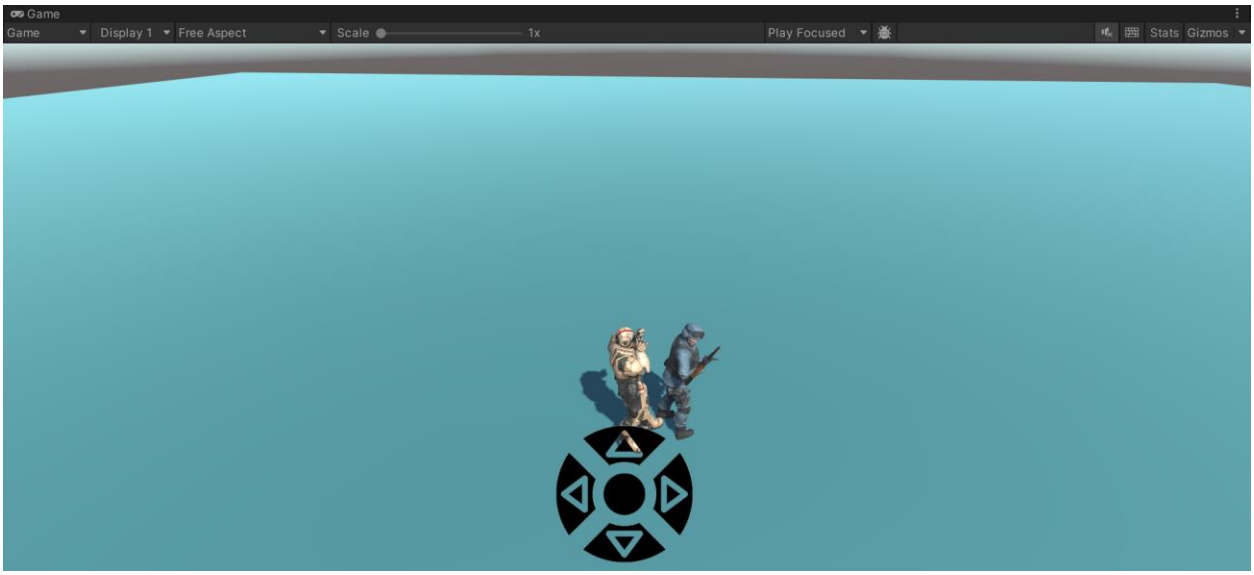
Player Animator FSM:

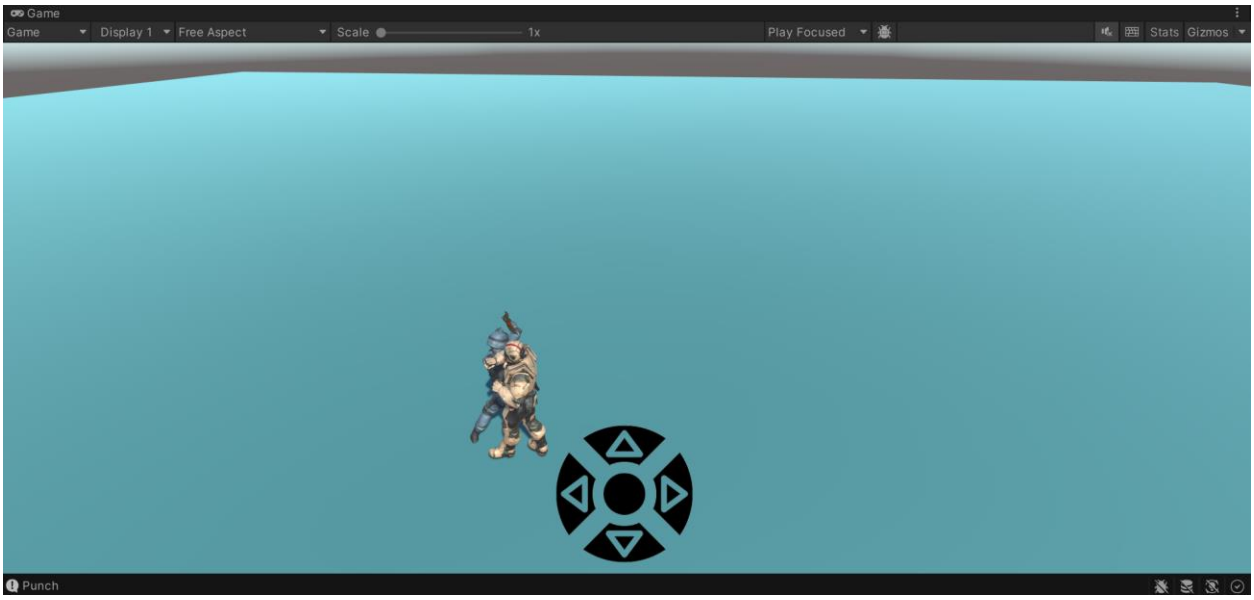


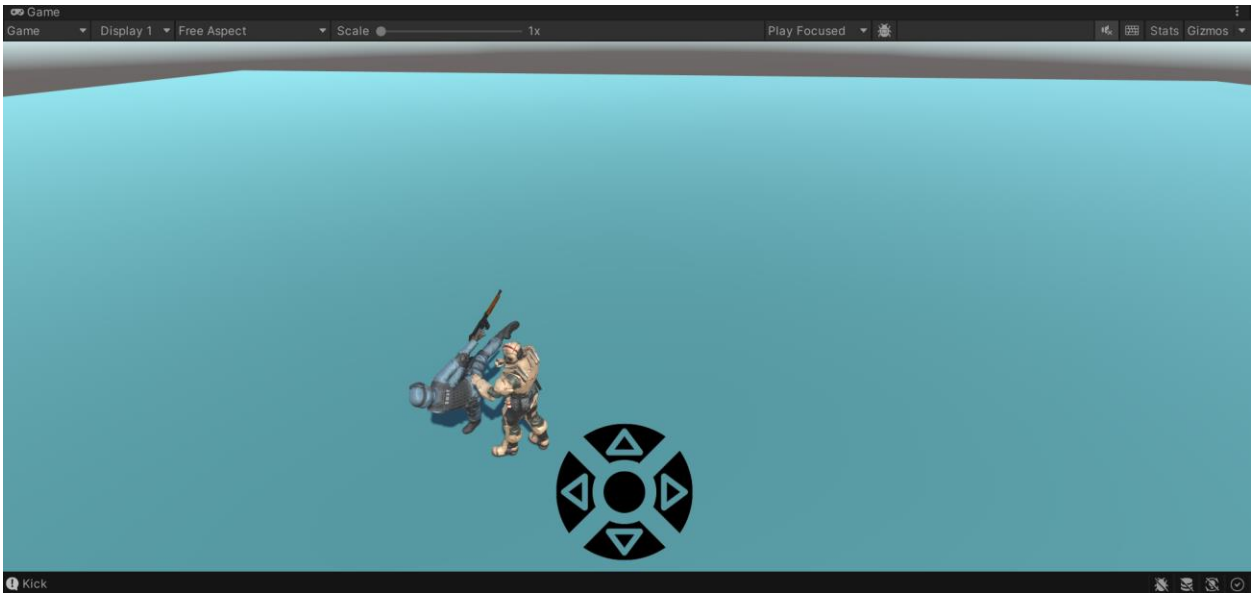
Enemy Animator FSM:



Output:


Game
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Stats
Gizmos


Game
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Stats
Gizmos
Punch


Game
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Stats
Gizmos
Kick