Lab # 10



Fall 2024

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

Submitted by: MUHAMMAD SADEEQ

Registration No.: 21PWCSE2028

Section: C

“On my honor, as a student of the University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work”

Submitted to:

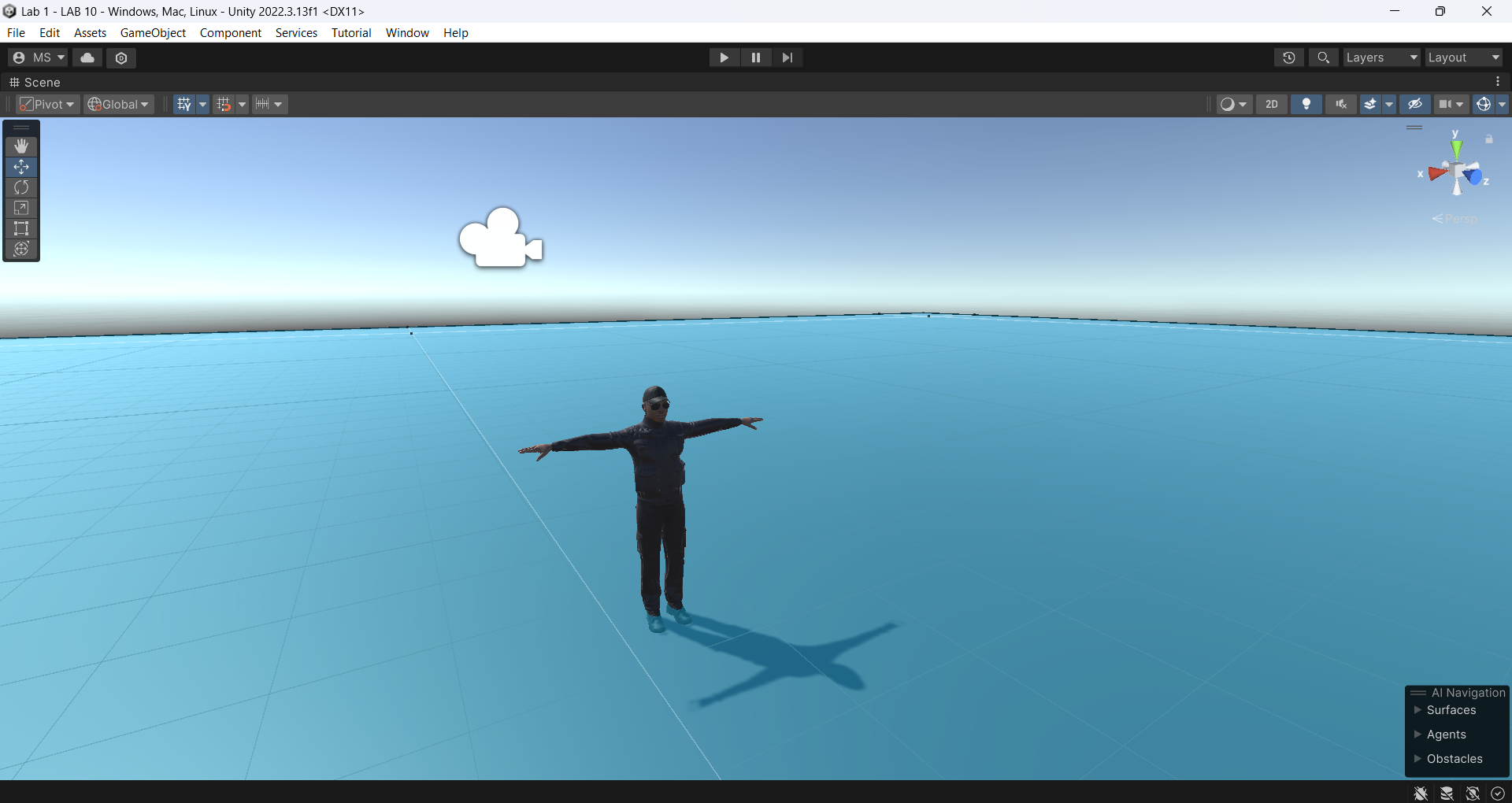
Engr. Abdullah Hamid

(25 Dec 2024)

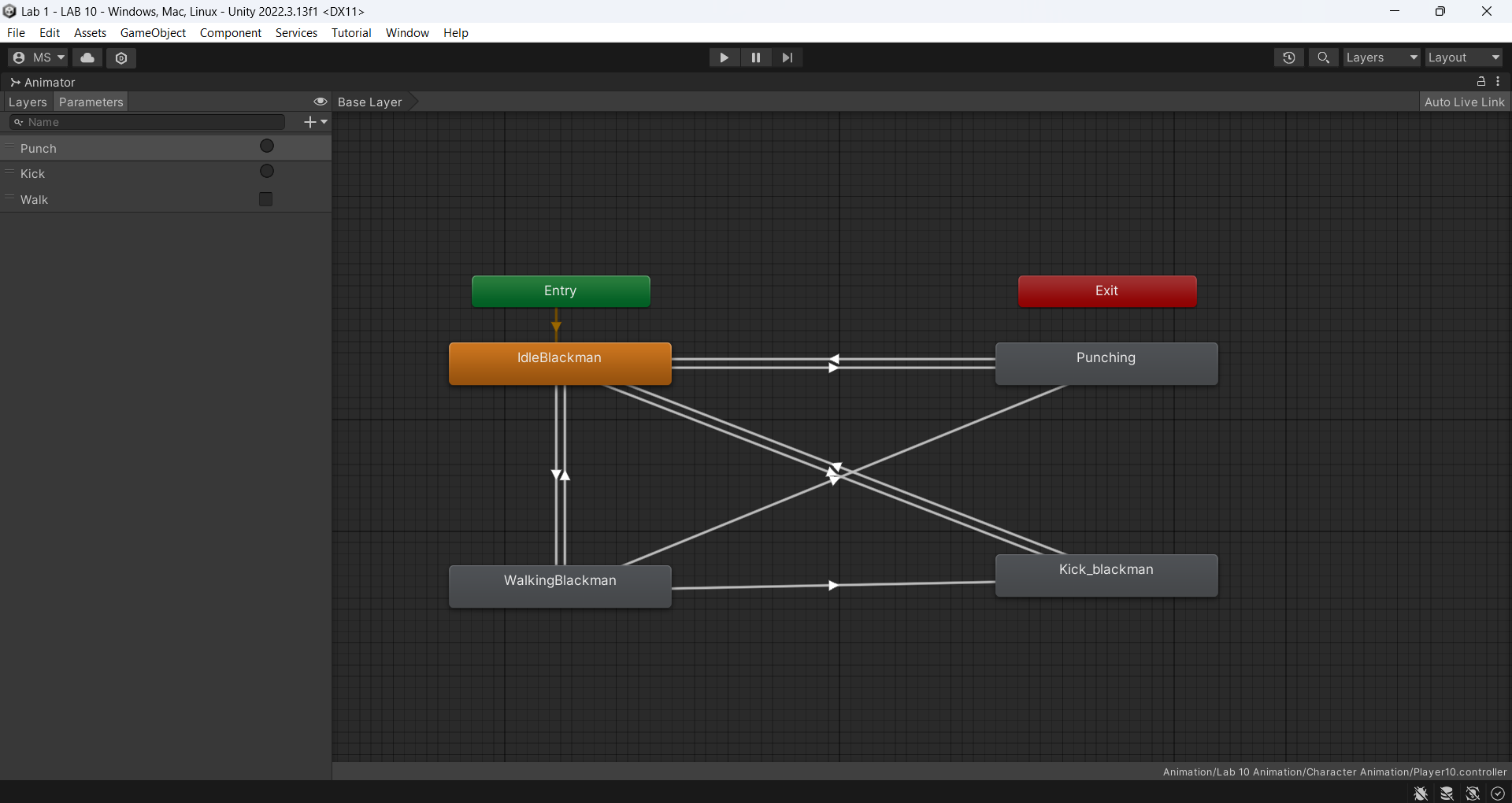
Department of Computer systems engineering

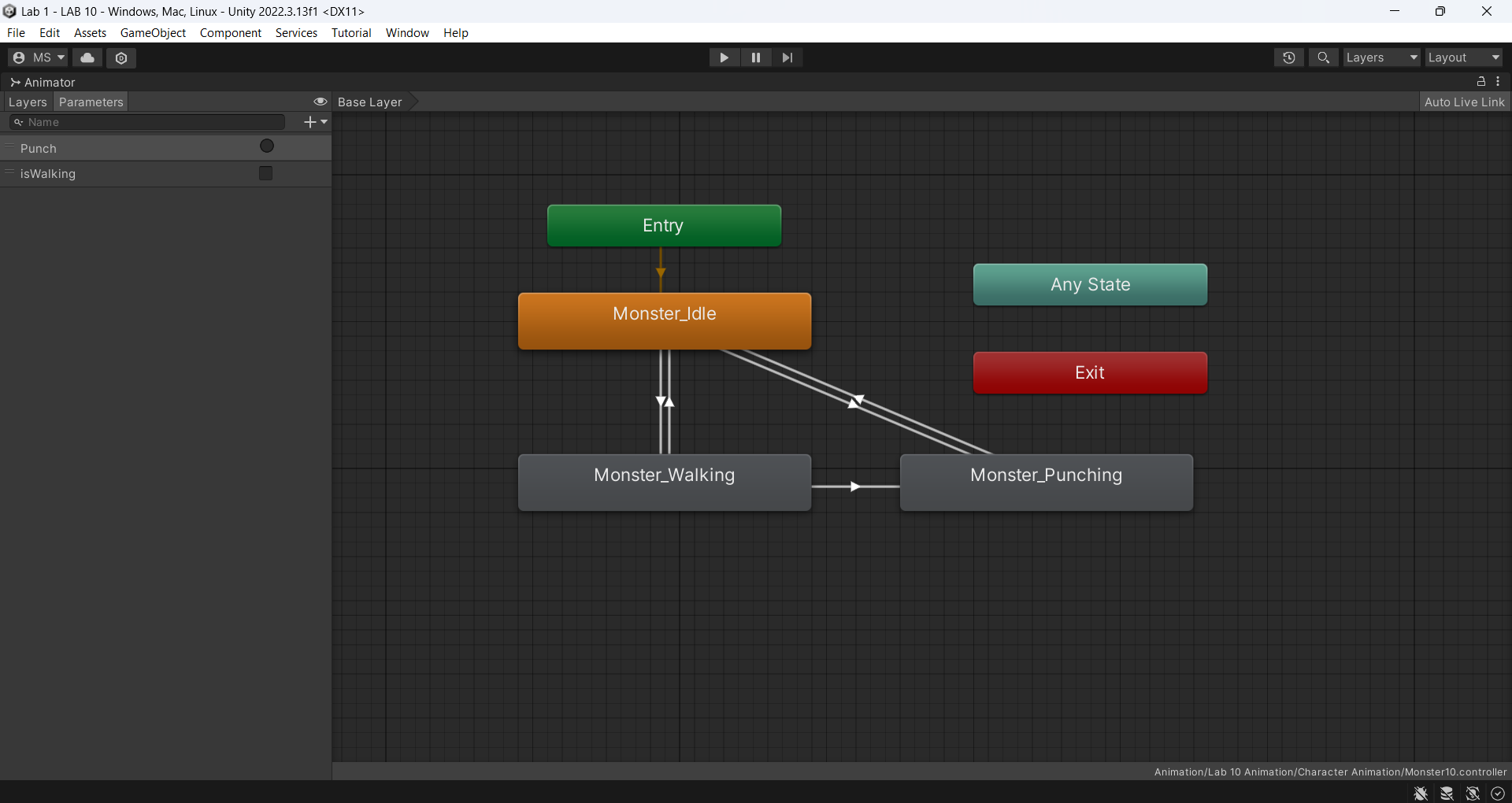
University of Engineering and Technology, Peshawar

1. **Scene Setup**
2. Created a new Unity scene named CharacterInteractionScene.
3. Added a plane to serve as the ground in the scene and scaled it to fit the playable area.
4. **Character Import and Setup**
5. Downloaded character models for the Player and Enemy from Mixamo.com.
6. Included Idle, Walk, and Punch animations for both characters.
7. Imported the models and animations into Unity and set up their rigs as Humanoid.



1. **Animator Configuration**
2. Created two Animator Controllers:
3. One for the Player
4. One for the Enemy
5. Added animation clips for Idle, Walk, and Punch to each controller.
6. Configured triggers for animations:
7. Punch
8. Kick (for the player only).





1. **Enemy Follow Logic**
2. Added a NavMesh to the plane by marking it as a Walkable Surface and baking the NavMesh.
3. Attached a NavMeshAgent to the Enemy character.
4. Wrote an **EnemyController10**.cs script to make the enemy follow the player if the distance between them is less than 5 units. If the enemy is close enough, it triggers the punch animation.

**Enemy Controller Script:**

using System.Collections;

using UnityEngine;

using UnityEngine.AI;

public class EnemyController10 : MonoBehaviour

{

public NavMeshAgent Agent;

public GameObject Player;

Animator animator;

private int walkHash;

private int punchHash;

// Start is called before the first frame update

void Start()

{

animator = GetComponent<Animator>();

walkHash = Animator.StringToHash("isWalking");

punchHash = Animator.StringToHash("Punch");

Agent = gameObject.GetComponent<NavMeshAgent>();

Player = GameObject.FindGameObjectWithTag("Player");

Agent.updateRotation = true;

}

// Update is called once per frame

void Update()

{

if (Vector3.Distance(transform.position, Player.transform.position) < 5f)

{

Agent.SetDestination(Player.transform.position);

animator.SetBool(walkHash, true);

if(Vector3.Distance(transform.position, Player.transform.position) < 1.5f)

{

animator.SetTrigger(punchHash);

}

}

else

{

Agent.ResetPath();

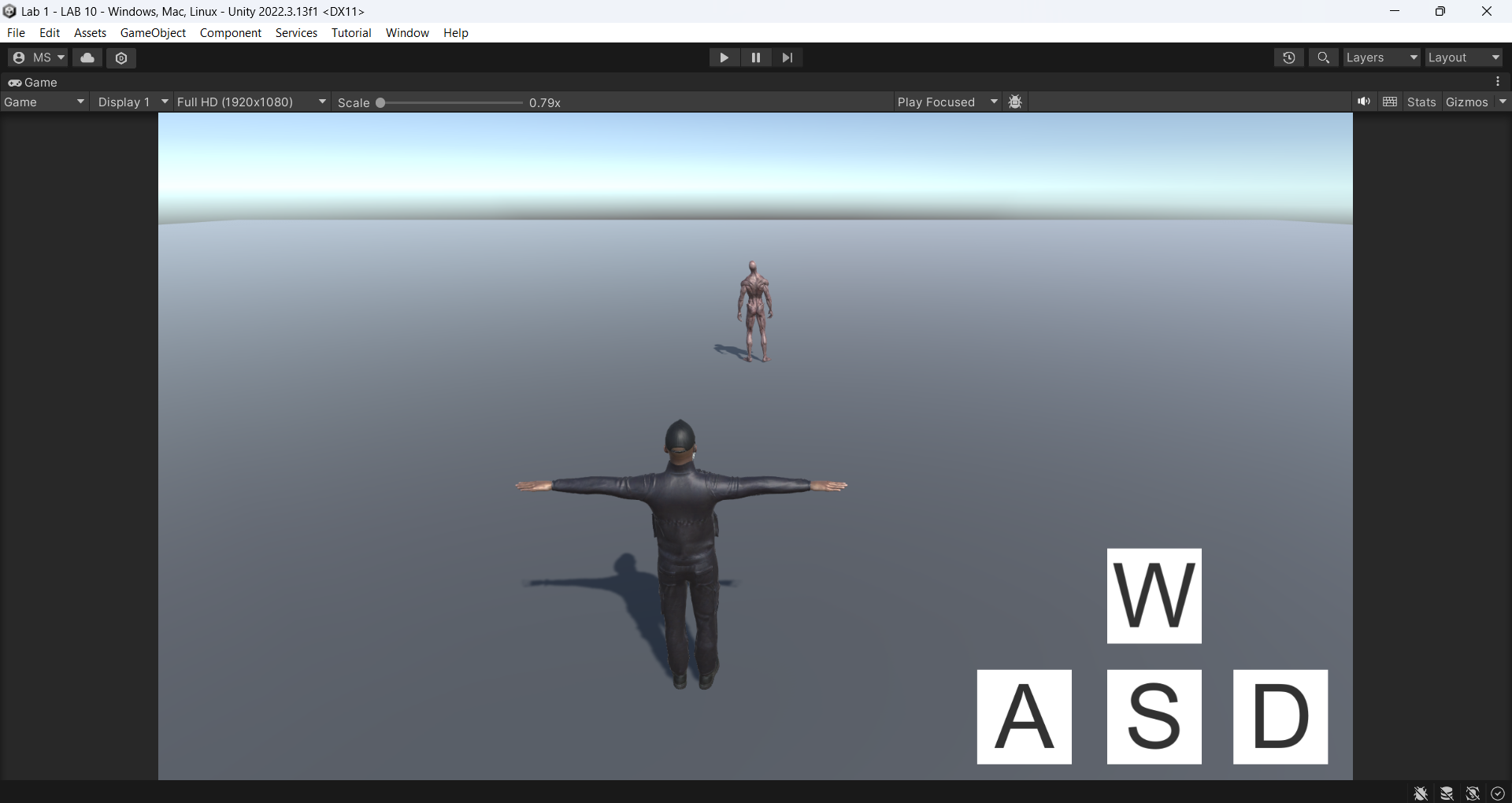
animator.SetBool(walkHash, false);

}

}

}

1. **Player Movement with On-Screen Buttons**
2. Added a Canvas to the scene and created four UI buttons labeled Forward, Backward, Left, and Right.
3. Wrote a **PlayerController10**.cs script to handle movement logic based on button presses.



**Player Controller Script:**

using Unity.VisualScripting;

using UnityEngine;

using UnityEngine.EventSystems;

public class PlayerController10 : MonoBehaviour

{

Animator animator;

Rigidbody rb;

float speed = 1.0f;

public float rotationSpeed = 5f;

public GameObject W, A, S, D;

private Vector3 movementDirection = Vector3.zero;

int walkHash, punchHash;

Quaternion targetRotation;

public delegate void PlayerActionHandler();

public static event PlayerActionHandler OnPunch;

public static event PlayerActionHandler OnKick;

// Start is called before the first frame update

void Start()

{

OnPunch += TriggerPunchAnimation;

OnKick += TriggerKickAnimation;

animator = GetComponent<Animator>();

walkHash = Animator.StringToHash("Walk");

punchHash = Animator.StringToHash("Punch");

rb = GetComponent<Rigidbody>();

EventTrigger W\_ET = W.AddComponent<EventTrigger>();

EventTrigger S\_ET = S.AddComponent<EventTrigger>();

EventTrigger A\_ET = A.AddComponent<EventTrigger>();

EventTrigger D\_ET = D.AddComponent<EventTrigger>();

EventTrigger.Entry Forwardwalk = new EventTrigger.Entry();

EventTrigger.Entry Backwardwalk = new EventTrigger.Entry();

EventTrigger.Entry Leftwalk = new EventTrigger.Entry();

EventTrigger.Entry Rightwalk = new EventTrigger.Entry();

EventTrigger.Entry release = new EventTrigger.Entry();

Forwardwalk.eventID = EventTriggerType.PointerDown;

Backwardwalk.eventID = EventTriggerType.PointerDown;

Leftwalk.eventID = EventTriggerType.PointerDown;

Rightwalk.eventID = EventTriggerType.PointerDown;

release.eventID = EventTriggerType.PointerUp;

release.callback.AddListener(upPointer);

Forwardwalk.callback.AddListener(forward);

Backwardwalk.callback.AddListener(backward);

Leftwalk.callback.AddListener(left);

Rightwalk.callback.AddListener(right);

W\_ET.triggers.Add(Forwardwalk);

W\_ET.triggers.Add(release);

S\_ET.triggers.Add(Backwardwalk);

S\_ET.triggers.Add(release);

A\_ET.triggers.Add(Leftwalk);

A\_ET.triggers.Add(release);

D\_ET.triggers.Add(Rightwalk);

D\_ET.triggers.Add(release);

}

// Update is called once per frame

void Update()

{

transform.Translate(movementDirection \* speed \* Time.deltaTime);

if (Input.GetMouseButton(0) && Input.GetKey(KeyCode.Q))

{

OnPunch?.Invoke();

}

// Trigger kick animation (Mouse0 + W)

if (Input.GetMouseButton(0) && Input.GetKey(KeyCode.W))

{

OnKick?.Invoke();

}

transform.rotation = Quaternion.Lerp(transform.rotation, targetRotation, Time.deltaTime \* rotationSpeed);

}

private void TriggerPunchAnimation()

{

animator.SetTrigger("Punch");

}

private void TriggerKickAnimation()

{

animator.SetTrigger("Kick");

}

public void forward(BaseEventData x)

{

animator.SetTrigger(walkHash);

movementDirection = Vector3.forward;

targetRotation = Quaternion.Euler(0, 0, 0);

}

public void backward(BaseEventData x)

{

animator.SetBool(walkHash, true);

movementDirection = Vector3.forward;

targetRotation = Quaternion.Euler(0, 180, 0);

}

public void left(BaseEventData x)

{

animator.SetBool(walkHash, true);

movementDirection = Vector3.forward;

targetRotation = Quaternion.Euler(0, -90, 0);

}

public void right(BaseEventData x)

{

animator.SetBool(walkHash, true);

movementDirection = Vector3.forward;

targetRotation = Quaternion.Euler(0, 90, 0);

}

private void OnDestroy()

{

// Unsubscribe methods to avoid memory leaks

OnPunch -= TriggerPunchAnimation;

OnKick -= TriggerKickAnimation;

}

public void upPointer(BaseEventData x)

{

animator.SetBool(walkHash, false);

movementDirection = Vector3.zero;

}

}

1. **Player Animation with Delegates**
2. Created a PlayerController.cs script to handle animations using delegates.
3. Configured the following triggers:
4. Mouse0 + Q: Triggers the player’s punch animation.
5. Mouse0 + W: Triggers the player’s kick animation.
6. **Testing the Scene**
7. Verified the following functionalities:
8. Enemy follows the player when within 5 units.
9. Enemy punches the player when close enough.
10. Player punches using Mouse0 + Q and kicks using Mouse0 + W.
11. Player moves using the on-screen buttons.
12. **Observations**
13. The NavMeshAgent efficiently handles enemy navigation.
14. Delegates provided a clean way to manage animation triggers.
15. The Unity Event System simplified implementing on-screen controls

#### **Conclusion**

#### This lab successfully demonstrated character interaction in Unity, including movement, animations, and event handling using delegates and the Unity Event System. The project can be further enhanced by adding health systems, sound effects, and more complex AI behaviors.