Lab # 09

Unity Animations and Animator



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:

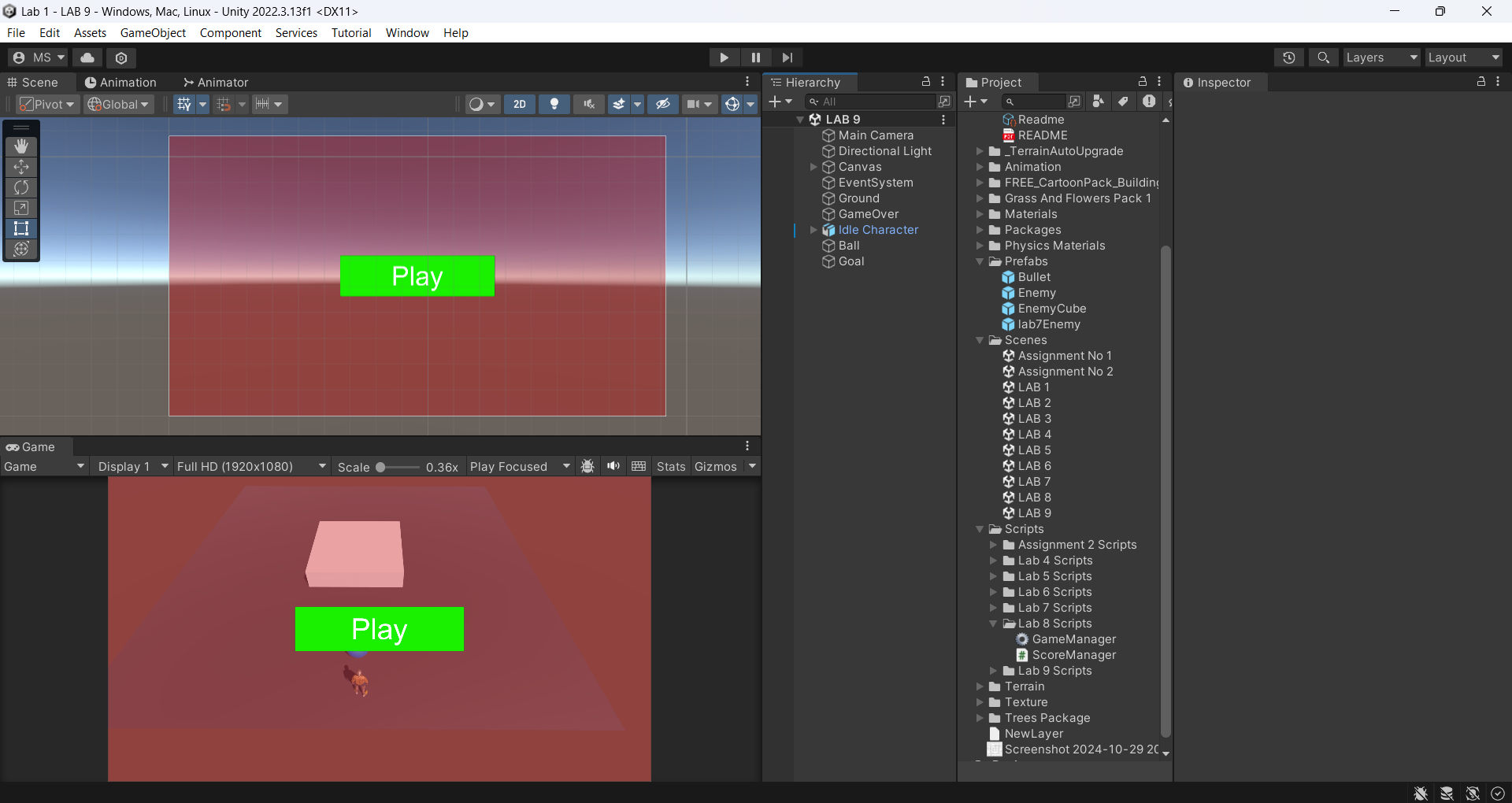
Engr. Abdullah Hamid

(15 Dec 2024)

Department of Computer systems engineering

University of Engineering and Technology, Peshawar

1. **Scene Setup:**
   1. Created a new Unity scene and designed the game layout.
   2. Added a main panel with a Play button as the central UI element.



1. **Game Paused on Start:**
   1. Ensured the game starts in a paused state.
   2. Configured the Play button to:
   3. Animate the main panel to slide out of the main camera’s view.
   4. Resume the game upon interaction.

**Panel Script:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.UIElements;

public class Panel : MonoBehaviour

{

Animator animator;

public UnityEngine.UI.Button playButton;

private void Awake()

{

animator = GetComponent<Animator>();

}

// Start is called before the first frame update

void Start()

{

Time.timeScale = 0f;

playButton.onClick.AddListener(StartGame);

}

public void StartGame()

{

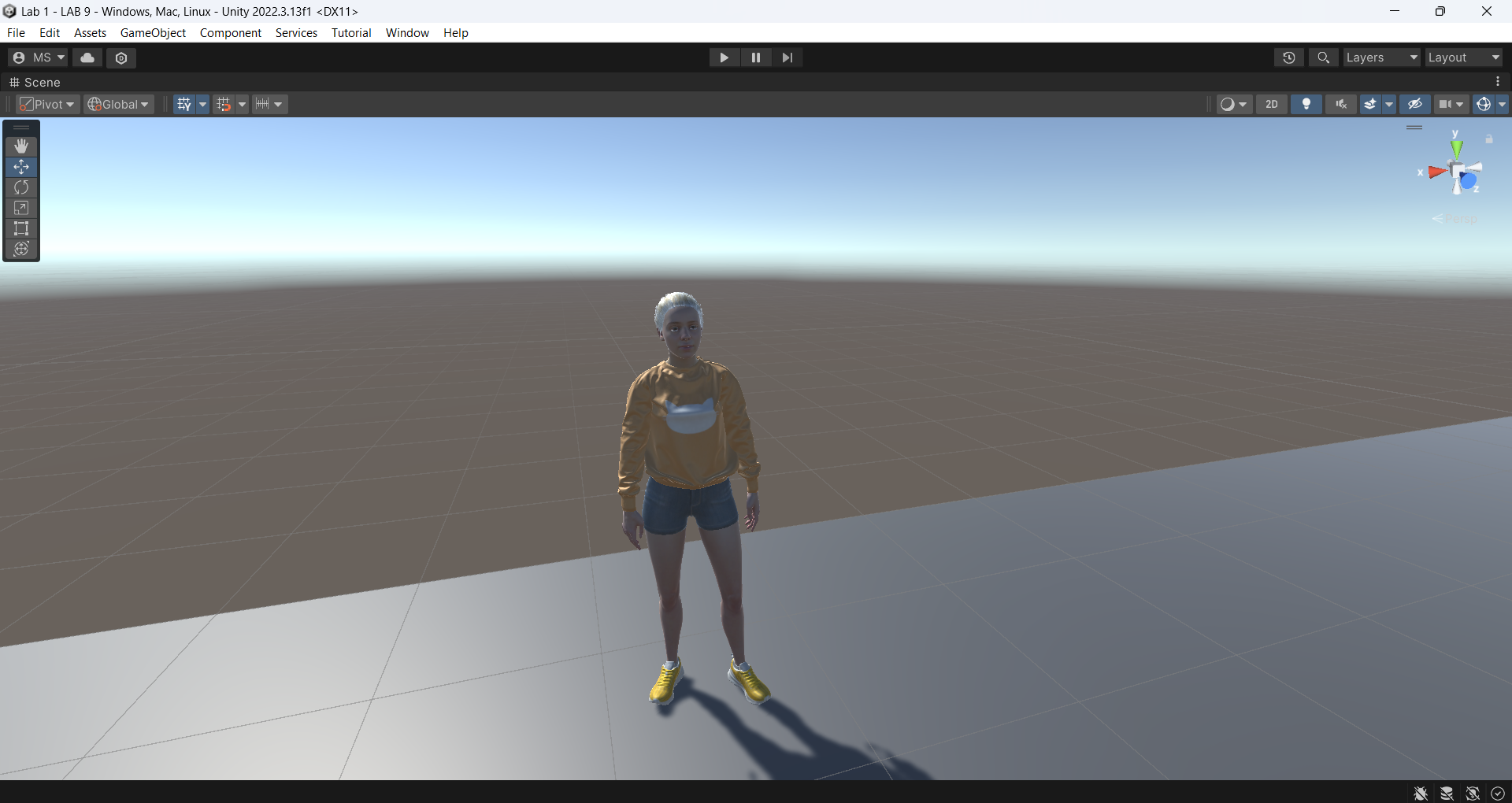
animator.SetBool("isStarted", true);

Time.timeScale = 1f;

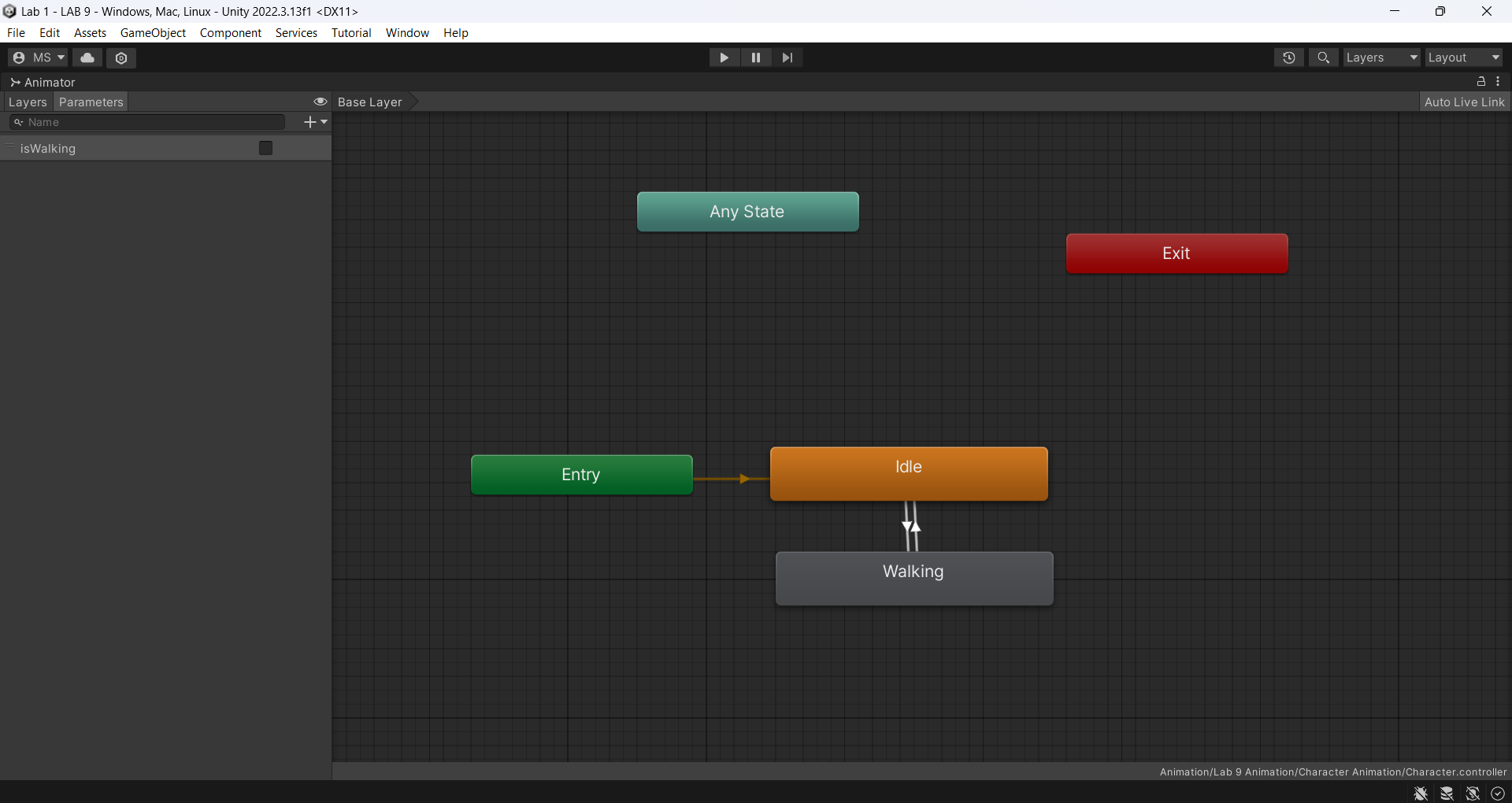
}

}

1. **Character Model and Animations:**
   1. Downloaded a character model and basic animations (Walk and Idle) from Mixamo.
2. **Conversion to Humanoid Rig:**
   1. Converted the character and animations to Unity’s Humanoid rig, ensuring compatibility with Unity’s Animator system.



1. **Animator Controller:**
   1. Created a new Animator Controller for the character to manage animation states.
   2. Configured the controller to use the Idle animation as the default state.



1. **Idle and Walk Animations:**
   1. Set up controls for the character:
   2. Pressing W, A, S, or D triggers the Walk animation and moves the player in the corresponding direction.

**Character Controller Script:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CharacterController : MonoBehaviour

{

Animator animator;

public float speed = 1.0f;

private Quaternion targetRotation;

public float rotationSpeed = 5f;

// Start is called before the first frame update

void Start()

{

animator = GetComponent<Animator>();

}

// Update is called once per frame

void Update()

{

if (Input.GetKey(KeyCode.W))

{

animator.SetBool("isWalking", true);

transform.Translate(Vector3.forward \* speed \* Time.deltaTime);

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

}

if (Input.GetKeyUp(KeyCode.W))

{

animator.SetBool("isWalking", false);

}

if (Input.GetKey(KeyCode.S))

{

animator.SetBool("isWalking", true);

transform.Translate(Vector3.forward \* speed \* Time.deltaTime);

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

}

if (Input.GetKeyUp(KeyCode.S))

{

animator.SetBool("isWalking", false);

}

if (Input.GetKey(KeyCode.A))

{

animator.SetBool("isWalking",true);

transform.Translate(Vector3.forward \* speed \* Time.deltaTime);

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

}

if (Input.GetKeyUp(KeyCode.A))

{

animator.SetBool("isWalking", false);

}

if (Input.GetKey(KeyCode.D))

{

animator.SetBool("isWalking", true);

transform.Translate(Vector3.forward \* speed \* Time.deltaTime);

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

}

if (Input.GetKeyUp(KeyCode.D))

{

animator.SetBool("isWalking", false);

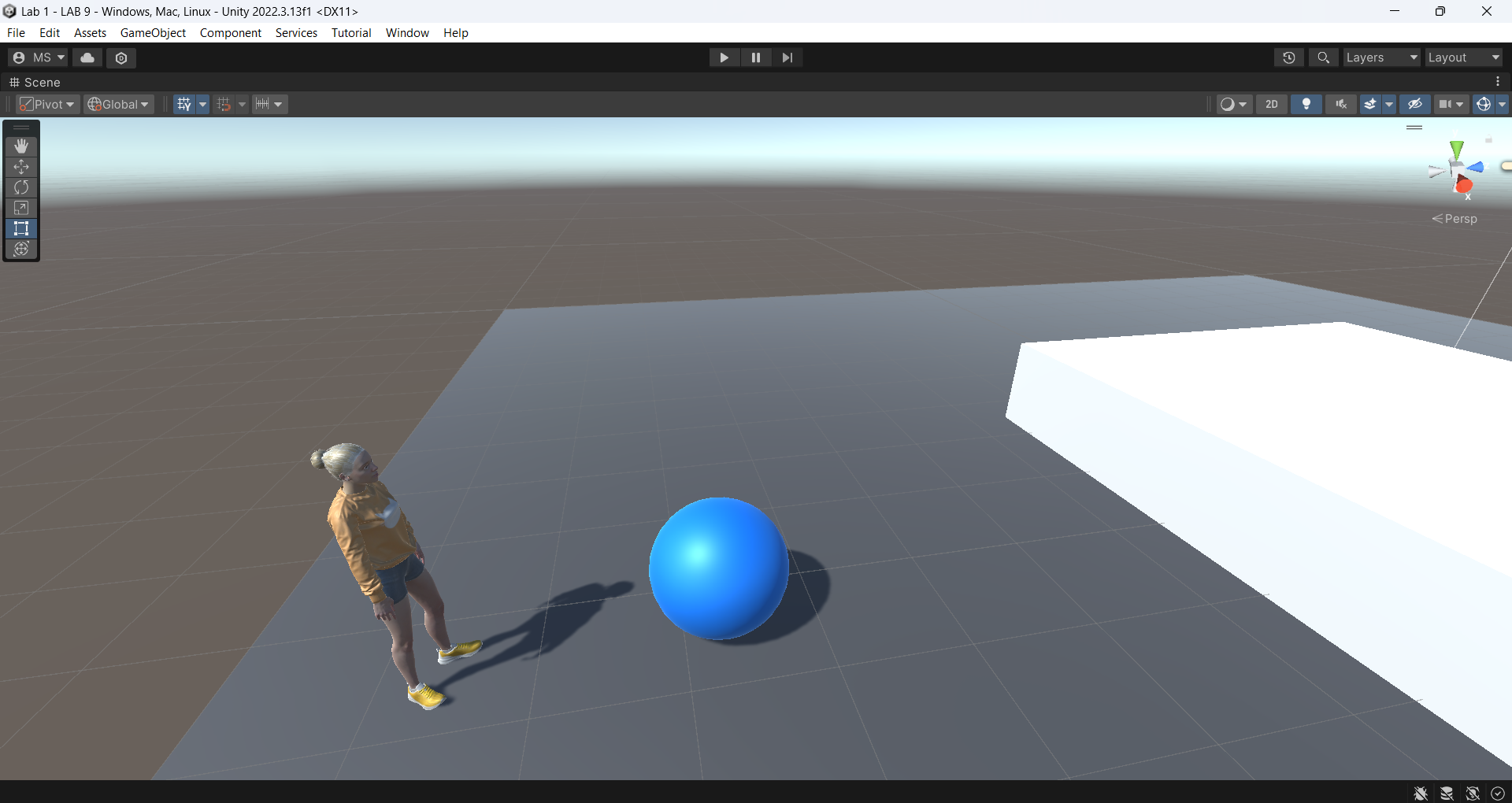
}

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

}

}

1. **Game Scene Design:**
   1. Added a Plane as the ground and placed the player and a Ball on it.
   2. Applied a Rigidbody and Bouncy Physics Material to the ball for realistic interactions.
   3. Configured the player to push the ball toward a designated goal.



1. **Goal Mechanic:**
   1. Created a Goal object with a color-changing animation to signify interactivity.
   2. Implemented functionality so that when the ball reaches the goal:
   3. A UI message, “Game Complete!”, is displayed.
   4. The game ends.

**Goal Script:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class GoalScript : MonoBehaviour

{

public GameObject gameCompletedPanel;

private void OnTriggerEnter(Collider other)

{

if (other.gameObject.CompareTag("Ball"))

{

gameCompletedPanel.SetActive(true);

Time.timeScale = 0.0f;

}

}

}

1. **Game Over Mechanic:**
   1. Added an Invisible Collider underneath the plane to detect if the ball falls off.
   2. Implemented logic such that if the ball collides with this object:
   3. A UI message, “Game Over, you lose!”, is displayed.
   4. The game ends.

**Fallzone Script:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Fallzone : MonoBehaviour

{

public GameObject gameoverPanel;

private void OnTriggerEnter(Collider other)

{

if (other.gameObject.CompareTag("Ball"))

{

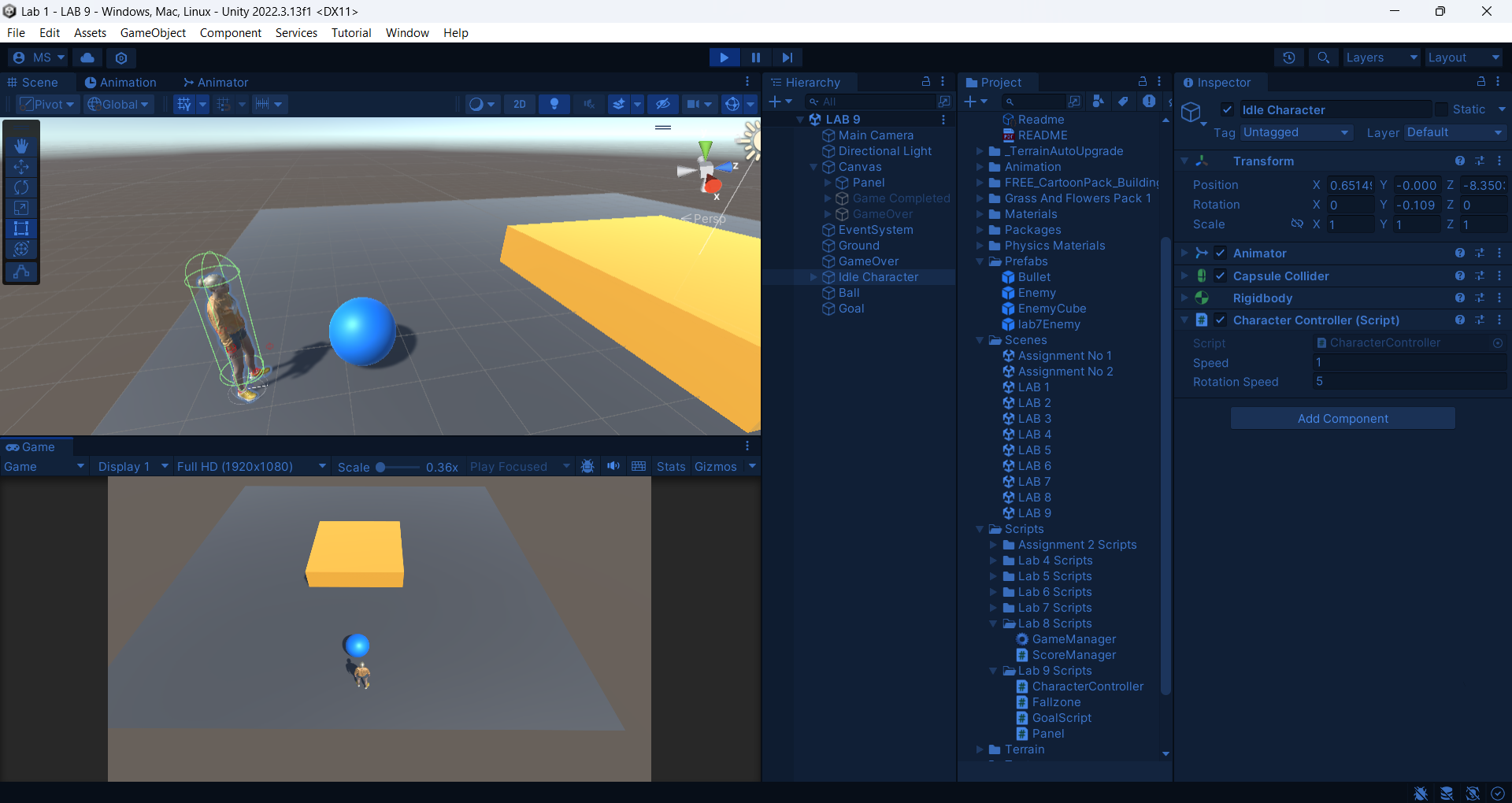
gameoverPanel.SetActive(true);

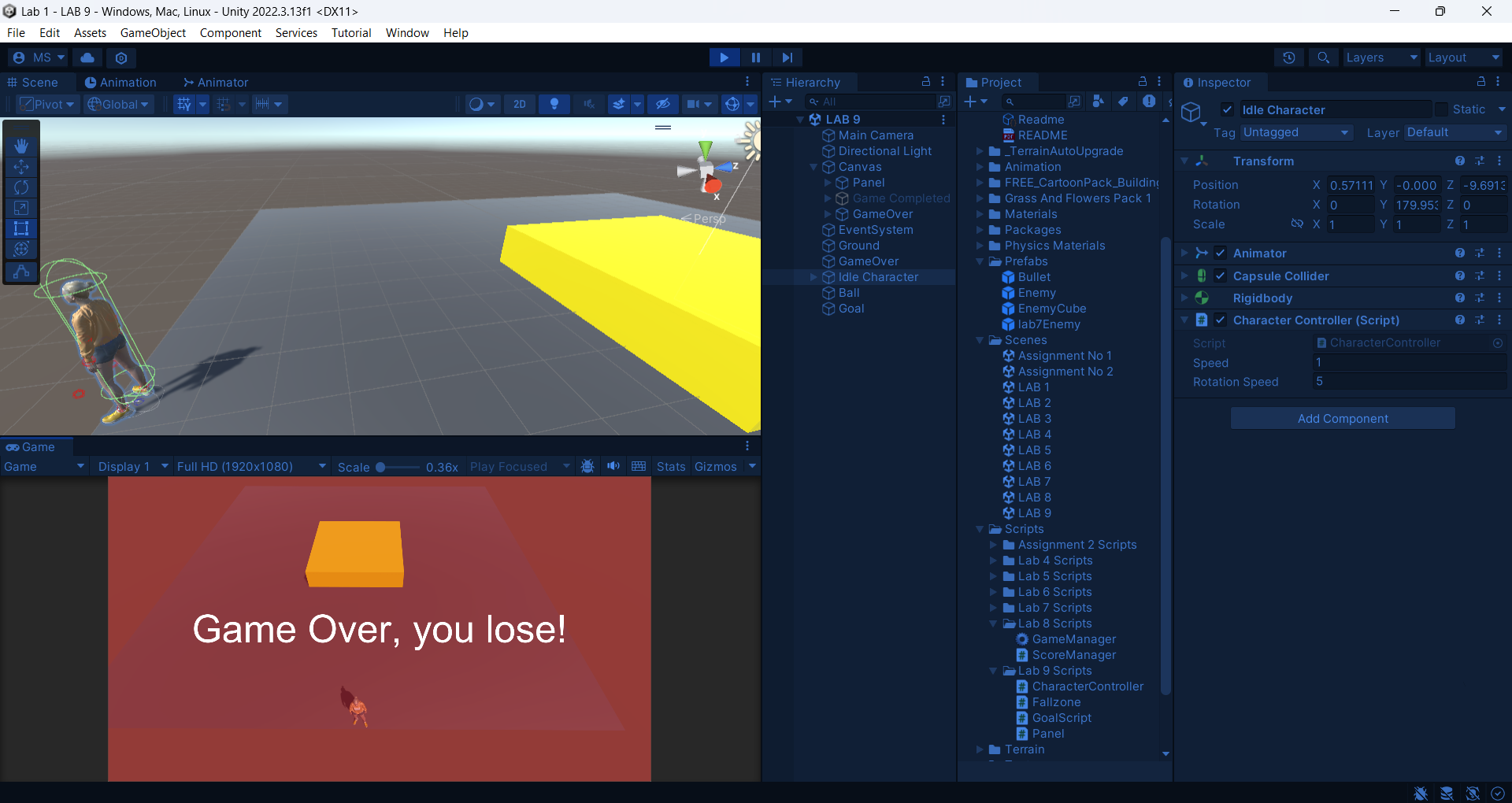
Time.timeScale = 0.0f;

}

}

}

****

****