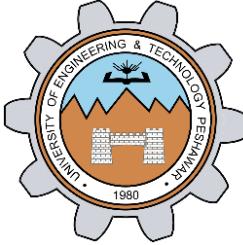


Unity Animations and Animator

LAB # 9



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:

25th December 2024

**Department of Computer Systems Engineering
University of Engineering and Technology, Peshawar**

Objective:

In this lab we further explored the Unity API.

Tasks:

- 1) **Set Up the Scene:**
 - a) Open or create a new scene in Unity.
 - b) In this scene, create your game layout and add a main panel with a “Play” button.
- 2) **Pause the Game on Start:**
 - a) Ensure the game is paused when it starts.
 - b) When the “Play” button is clicked:
 - c) Animate the main panel to slide up and out of the main camera’s view.
 - d) Resume the game.
- 3) **Download Assets:**
 - a) Go to Mixamo and download a character model along with basic animations like **walk** and **idle**.
- 4) **Convert to Humanoid:**
 - a) Convert the character and animations to **Humanoid** rig in Unity.
 - b) (See conversion method on the next page).
- 5) **Set Up Animator Controller:**
 - a) Create a new **Animator Controller** for the character.
 - b) This character will act as the main player in your game.
- 6) **Idle and Walk Animation:**
 - a) Set the default animation state of the player to **Idle**.
 - b) Configure the animator so that pressing W, A, S, or D will:
 - c) Trigger the **Walk** animation.
 - d) Move the player in the corresponding direction.
- 7) **Design the Game Scene:**
 - a) Add a **Plane** as the ground.
 - b) Place the player and a **Ball** on the plane.
 - c) Apply a **Rigidbody** and **Bouncy Physics Material** to the ball.
 - d) Ensure the player can push the ball towards a goal on the plane.
- 8) **Goal Mechanic:**
 - a) Add a **Goal** object to the scene.
 - b) The goal should have a **color-changing animation**.
 - c) When the ball reaches the goal:
 - d) Display the UI message: “**Game Complete!**”
 - e) End the game.
- 9) **Game Over Mechanic:**
 - a) Add an **Invisible Collider** underneath the plane.
 - b) If the ball falls off the plane and collides with this object:
 - c) Display the UI message: “**Game Over, you lose!**”
 - d) End the game.

Code:

GameController class



The screenshot shows the Unity code editor with the file `GameController.cs` open. The code defines a `GameController` class in the `Lab9` namespace. The class contains methods for handling button presses, starting the game, and managing game completion and over states.

```
5
6
7  namespace Lab9
8  {
9
10     public class GameController : MonoBehaviour
11     {
12         public Button playButton;
13         public GameObject mainPanel, gameCompletePanel, gameOverPanel;
14         public Animator mainMenuAnimator;
15
16         void Awake()
17         {
18             playButton.onClick.AddListener(()=> OnPlayPressed());
19             mainMenuAnimator = mainPanel.GetComponent<Animator>();
20         }
21
22         void Start()
23         {
24             Time.timeScale = 0;
25             mainMenuAnimator.SetBool("CloseMenu", false);
26         }
27
28         void OnPlayPressed()
29         {
30             Time.timeScale = 1;
31             mainMenuAnimator.SetBool("CloseMenu", true);
32         }
33
34         public void GameComplete()
35         {
36             Time.timeScale = 0;
37             gameCompletePanel.SetActive(true);
38         }
39     }
}
```

GameOverCube class

The screenshot shows the Unity Editor's code editor with the GameOverCube.cs script selected. The script is part of the Lab9 project. It contains a Start() method that finds a GameController object and calls its GameOver() method. It also contains an OnTriggerEnter() method that checks if a colliding object has the "Ball" tag and, if so, calls the GameController's GameOver() method.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  namespace Lab9
6  {
7      public class GameOverCube : MonoBehaviour
8      {
9
10         public GameController gameController;
11
12         void Start(){
13             gameController = FindObjectOfType<GameController>();
14         }
15
16         void OnTriggerEnter(Collider other){
17             if (other.gameObject.CompareTag("Ball")){
18                 gameController.GameOver();
19             }
20         }
21     }
22 }
```

Goal class

The screenshot shows the Unity Editor's code editor with the Goal.cs script selected. The script is part of the Lab9 project. It contains a Start() method that finds a GameController object and calls its GameComplete() method. It also contains an OnTriggerEnter() method that checks if a colliding object has the "Ball" tag and, if so, calls the GameController's GameComplete() method.

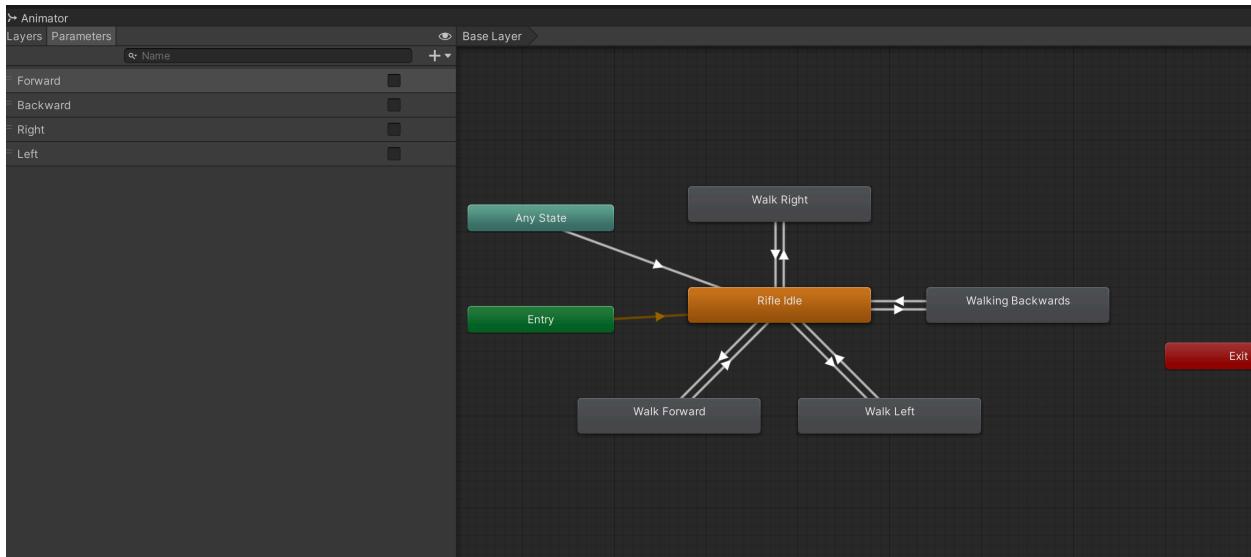
```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  namespace Lab9{
6      public class Goal : MonoBehaviour{
7          public GameController gameController;
8
9          void Start(){
10             gameController = FindObjectOfType<GameController>();
11         }
12
13         void OnTriggerEnter(Collider other){
14             if (other.gameObject.CompareTag("Ball")){
15                 gameController.GameComplete();
16             }
17         }
18     }
19 }
```

PlayerController class

The screenshot shows a Unity code editor with the PlayerController.cs script selected. The tab bar at the top includes Controller.cs, Player.cs, Goal.cs, PlayerController.cs (which is the active tab), and Main. The file path in the title bar is Assets > Labs > Lab9 > Scripts > PlayerController.cs. The code itself is as follows:

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  namespace Lab9
6  {
7
8      public class PlayerController : MonoBehaviour
9      {
10
11         [SerializeField]
12         private Animator playerAnimator;
13
14         [SerializeField]
15         private float speed = 1f;
16
17         // Start is called before the first frame update
18         void Start()
19         {
20             if (playerAnimator == null){
21                 playerAnimator = GetComponent<Animator>();
22             }
23         }
24
25         // Update is called once per frame
26         void Update()
27         {
28             playerAnimator.SetBool("Forward", false);
29             playerAnimator.SetBool("Backward", false);
30             playerAnimator.SetBool("Left", false);
31             playerAnimator.SetBool("Right", false);
32
33             if (Input.GetKey(KeyCode.W)){
34                 transform.Translate(Vector3.forward * Time.deltaTime * speed);
35                 playerAnimator.SetBool("Forward", true);
36             }
37             if (Input.GetKey(KeyCode.S)){
38                 transform.Translate(Vector3.back * Time.deltaTime * speed);
39                 playerAnimator.SetBool("Backward", true);
40             }
41             if (Input.GetKey(KeyCode.A)){
42                 transform.Translate(Vector3.left * Time.deltaTime * speed);
43                 playerAnimator.SetBool("Left", true);
44             }
45             if (Input.GetKey(KeyCode.D)){
46                 transform.Translate(Vector3.right * Time.deltaTime * speed);
47                 playerAnimator.SetBool("Right", true);
48             }
49         }
50     }
```

Animator FSM:



Output:

