Lab 5-6 (LAB Assignment 1)

Lab Manual: Simple Car Racing Game in Unity

Objective

Create a simple car racing game using Unity, where the player controls a car that moves along a track, reduces fuel, and refuels by hitting oil drums. The game ends when the fuel finishes, the car hits another object, or the player finishes the track within 3 minutes.

Prerequisites

- Unity Hub installed on your computer

- Basic knowledge of C# programming language

- Familiarity with Unity interface and basic components

Lab Exercise

Part 1: Setting up the Project

1. Create a new Unity project and name it "CarRacingGame".

2. Set the project type to "3D" and select a suitable template.

3. Create a new scene and name it "RacingTrack".

Part 2: Creating the Track and Car

1. Create a new 3D object (e.g., a cube/plane) and name it "Road".
2. Also add road side

3. Add a collider component to the track to detect collisions.

4. Create a new 3D object (e.g., a cube) and name it "Car".

5. Add a Rigidbody component to the car to enable physics.

6. Add a Collider component to the car to detect collisions.

Part 3: Implementing Fuel System

1. Create a new script (e.g., "CarMovement.cs") and attach it to the car.

2. In the CarMovemen script, declare a public float variable "fueValuel" and initialize it to 100.

3. Use the Update method to reduce the fuel value by 0.1 every second.

4. Create a UI Text component to display the fuel value.

5. ensure the fuel value does not go below 0.

6. if it reaches 0, load GAME OVER SCENE

Part 4: Implementing Low Fuel Warning

1. In the FuelSystem script, check if the fuel value is less than or equal to 20.

2. If true, display a "Low Fuel" text message on the screen using a UI Text component.

Part 5: Implementing Refueling

1. Create a new 3D object (e.g., a cube) and name it "OilDrum".

2. Add a Collider component to the oil drum to detect collisions.

3. Add refueling CODE in "CarMovemen.cs" using the OnCollisionEnter method to detect when the car collides with the oil drum.

4. If a collision is detected, increase the fuel value by 30 using the FuelSystem script.

5. Play a fuel filling sound effect using an Audio Source component.

Part 6: Implementing Game Over Conditions

1. In the FuelSystem script, check if the fuel value is less than or equal to 0.

2. If true, load the "GameOver" scene using the SceneManager class.

3. Play a losing sound effect using an Audio Source component.

4. In the Car script, use the OnCollisionEnter method to detect when the car collides with another object.

5. If a collision is detected, load the "GameOver" scene using the SceneManager class.

6. Play a crashing sound effect using an Audio Source component.

Part 7: Implementing Winning Condition

1. In Update method check if car is been traveling for 3 minutes

3. If true, load the "WinningPanel" scene using the SceneManager class.

4. Play a winning sound effect using an Audio Source component.

Part 8: Testing and Debugging

1. Test the game by playing it and checking for any errors or bugs.

2. Use the Unity Debugger to debug any issues that arise.

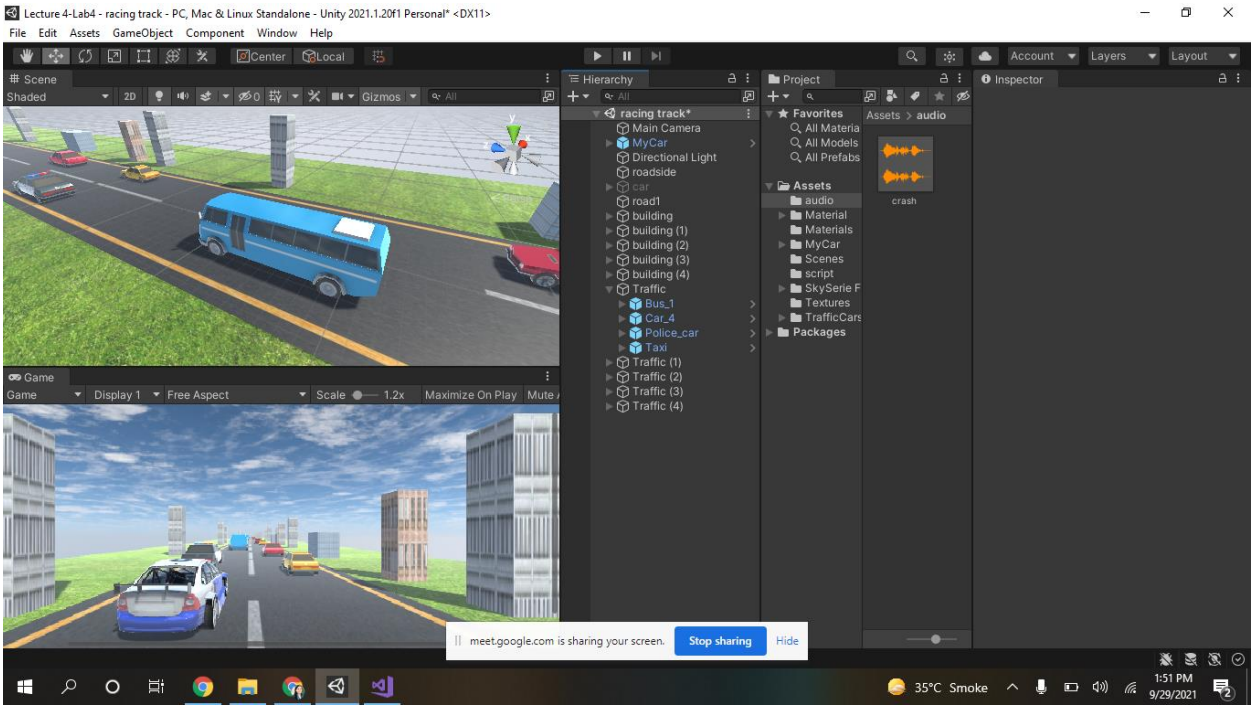
Deliverables

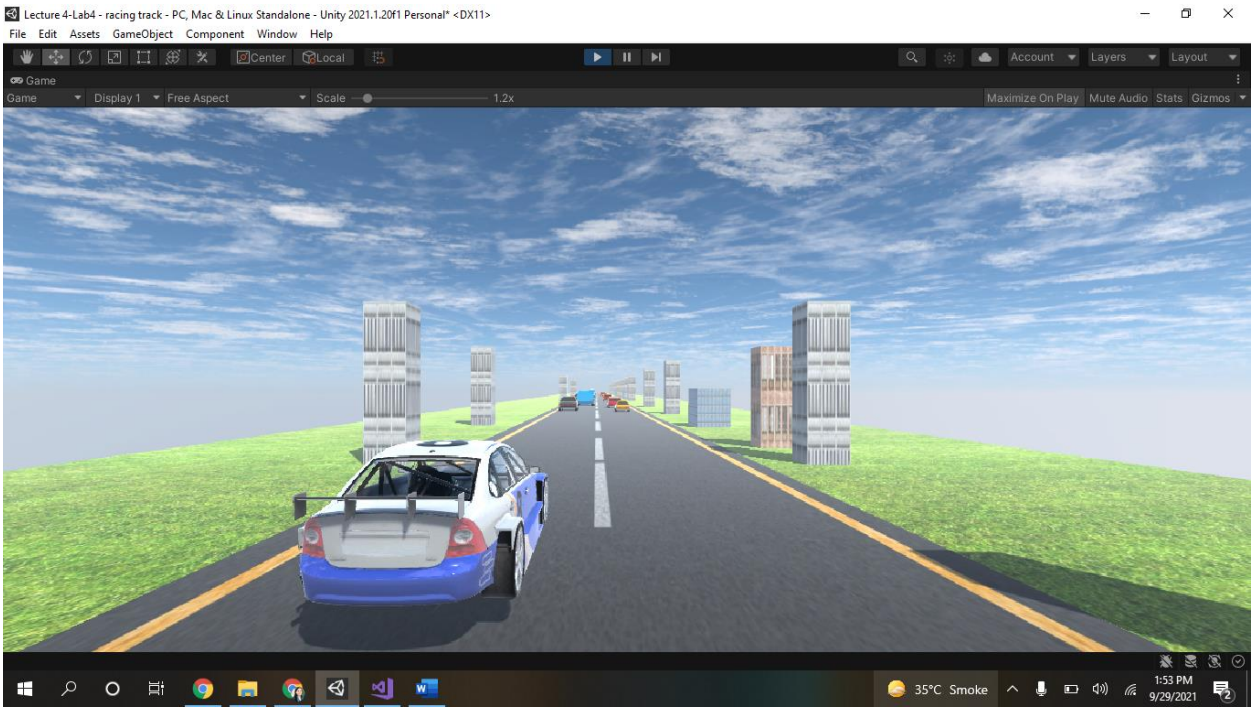
- All scripts

- A 10 seconds video of the game

- A screenshot of the game in action

- Upload each file separately, not in a folder




Lecture 4-Lab4 - racing track - PC, Mac & Linux Standalone - Unity 2021.1.20f1 Personal* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Account Layers Layout
Game
Game Display 1 Free Aspect Scale 1.2x
Maximize On Play Mute Audio Stats Gizmos
35°C Smoke
1:53 PM
9/29/2021