PREMIER UNIVERSITY CHITTAGONG

Department of Computer Science & Engineering

Course Code : CSE-451

Course Title : Computer Graphics and Image Processing Laboratory.

Report Name: 2D Street Racing car game .

Date of Submission : 14/03/2023

SUBMITTED BY

MD: Mohiuddin Faysal
ID:1903610201765
Department : CSE
Semester: 7th
Section: C2

SUBMITTED TO

MD. NEAMUL HOQUE
Lecturer
Department of CSE

Abstract

Unity Hub is a software application developed by Unity Technologies that allows users to manage and organize their Unity projects and installations. With Unity Hub, we can easily download and install different versions of Unity, switch between them, and manage your projects for each version.

In addition to managing Unity installations, Unity Hub also provides access to Unity Learn, Unity Asset Store, and Unity Collaborate, which are resources for learning, downloading assets, and collaborating with team members, respectively.

Unity Hub also offers various features such as project templates, project search, and project creation. Overall, Unity Hub is a useful tool for managing and organizing Unity projects and installations, making it easier for developers to work efficiently and productively.

My Game 2D Unity car game developed using Unity Hub is an exciting and engaging game that challenges players to navigate a variety of courses while driving a car. The game includes a range of features, including realistic physics, challenging obstacles, and customizable cars that can be upgraded throughout gameplay. The game is designed with intuitive controls that allow players to easily steer their car and use power-ups to gain an advantage.

Table of Contents

Chapter 01

Introduction

Unity car racing game is an exciting and popular genre of video games that has captivated gamers worldwide. It is a type of game that typically involves players driving virtual cars around different tracks or environments, competing against other players or computer-controlled opponents.

Unity is a powerful game development engine that allows developers to create highly realistic and immersive racing games with customizable cars, challenging tracks, and physics-based gameplay.

The game design involves creating a path that consists of a series of car game blocks that the player must navigate through. The player controls a car, which moves along the path, and must collect coins while avoiding obstacles such as gaps in the path .

The project involves implementing several features such as physics-based movement of the car, user input control, scoring and game over mechanics, and a user interface. The game will be developed for Android mobile devices and will be compatible with a wide range of screen sizes.

Overall, 2D car -Racing-Game-with-C project is an exciting and challenging game development project that provides an opportunity to develop and showcase skills in game design, programming, and mobile app development.

Chapter 02

Objective

The objective of a 2D car racing game with with-C-Sharp project by Unity is to create an engaging and immersive racing game that challenges players to compete against each other or computer-controlled opponents in a variety of tracks and environments.

The game should include realistic physics, intuitive controls, and customizable cars that can be upgraded as players progress through the game. The project aims to create a visually appealing and immersive experience for players, with dynamic sound effects and engaging graphics.

Additionally, the game should offer players multiple levels of increasing difficulty, power-ups, and obstacles to create a challenging and entertaining experience.

Ultimately, the objective of a 2D car racing game project by Unity is to provide players with an exciting and engaging racing game that offers endless hours of entertainment.

Chapter 03

Tools and Environment

To develop a 2D car racing game, several tools and environments are required. Here are some of the most common tools and environments used to develop a 2D car racing game:

1. Unity: Unity is a powerful game development engine that allows developers to create high-quality games for various platforms, including PC, consoles, and mobile devices. It provides a range of tools and features to develop 2D car racing games, such as physics engines, audio and graphics tools, and scripting tools. I used Unity version-2018.2.11f1.

2. Photoshop: Photoshop is an essential tool for creating game graphics, including character and environment designs, textures, and user interface elements.

3. Tiled: Tiled is an open-source software that allows developers to create tile maps for 2D games. It is a useful tool for creating racing tracks and environments for 2D car racing games.

4. Asperity: Asperity is a pixel art tool used for creating game graphics, such as characters, cars, and background elements. It allows developers to create high-quality game graphics and animations.

5. Audacity: Audacity is a free audio editing tool that allows developers to create and edit sound effects, music, and voiceovers for their games.

6. Visual Studio Code: Visual Studio Code is an integrated development environment (IDE) used for coding game logic and scripts. It provides a range of tools and features to improve code quality and productivity.

The above tools and environments are commonly used to develop 2D car racing games.

Chapter 04

Source code

1. BanditCarbehaviour

```
•   public class BanditCarBehaviour : MonoBehaviour
•   {
•       public GameObject bomb;
•       public int bombsAmount;
•       public int banditCarVerticalSpeed;
•       public int banditCarHorizontalSpeed;
•       public float bombDelay;
•
        private float _delay;
•       private GameObject _playerCar;
•       private Vector3 _banditCarPosition;
•
      private void Start()
•       {
•           _playerCar = GameObject.FindWithTag("Player");
•           _delay = bombDelay;
•       }
•
        private void Update()
•       {
•           if (_playerCar == null)
•           {
•               _playerCar = GameObject.FindWithTag("Player");
•           }
•           else
•           {
•               if (gameObject.transform.position.y > 3.8f && bombsAmount > 0)
•                   this.gameObject.transform.Translate(new Vector3(0, -1, 0) * banditCarVerticalSpeed *
    Time.deltaTime);
•
                else if (bombsAmount <= 0)
•               {
•                   this.gameObject.transform.Translate(new Vector3(0, 1, 0) * banditCarVerticalSpeed *
    Time.deltaTime);
•                   if (gameObject.transform.position.y > 6.2f)
•                       Destroy(this.gameObject);
•               }
•
                else
•               {
•                   _banditCarPosition = Vector3.Lerp(transform.position, _playerCar.transform.position,
    Time.fixedDeltaTime * banditCarHorizontalSpeed);
•                 //our position, object to follow position, time.
•
                  transform.position = new Vector3(_banditCarPosition.x, transform.position.y, 0);
•
                  _delay -= Time.deltaTime;
•                 if (_delay <= 0 && bombsAmount > 0)
•                 {
•                     _delay = bombDelay;
•                     bombsAmount--;
•                     Instantiate(bomb, transform.position, Quaternion.identity);
•                 }
•                 else if (_delay <= 0 && bombsAmount <= 5 && bombsAmount > 0)
•                 {
•                     _delay = bombDelay / 2;
•                     bombsAmount--;
•                     Instantiate(bomb, transform.position, Quaternion.identity);
•                 }
•             }
•         }
•     }
•
    }
•
```

2. Bonuses

```
•   public class Bomb : MonoBehaviour
•   {
•       public int bombDmg;
•       public float bombSpeed;
•
        private void Update()
•       {
•           this.gameObject.transform.Translate(new Vector3(0, -1, 0) * bombSpeed * Time.deltaTime);
•       }
•
        private void OnTriggerEnter2D(Collider2D obj)
•       {
•           if (obj.gameObject.tag == "Player")
•           {
•               obj.gameObject.GetComponent<CarControll>().durability -= bombDmg;
•               Destroy(this.gameObject);
•           }
•           else if (obj.gameObject.tag == "Shield")
•           {
•               Destroy(this.gameObject);
•           }
•           else if (obj.gameObject.tag == "EndOfRoad")
•           {Destroy(this.gameObject); } }
```

3. BonusSpawner

```
•   public class Bonuses : MonoBehaviour
•   {
•       [Header("Type of bonus")]
•       public bool isDurability;
•       public bool isShield;
•       public bool isSpeed;
•
        [Header("Bonuses settings")]
•       public float bonusSpeed = 5f;
•
        [Header("Durability settings")]
•       public float repairPoints;
•
        [Header("Shield settings")]
•       public GameObject shield;
•       private GameObject _playerCar;
•       private Vector3 _playerCarPosition;
•
        [Header("Speed settings")]
•       public float speedBoost;
•       public float duration;
•       private bool _isActive = false;

•   private void Update()
•       {
•           this.gameObject.transform.Translate(new Vector3(0, -1, 0) * bonusSpeed * Time.deltaTime);
    //go down with road.
•       }
•
        private void OnTriggerEnter2D(Collider2D obj)
•       {
•           if (obj.gameObject.tag == "Player" || obj.gameObject.tag == "Shield")
•           {
•               if (isDurability == true)
•               {
•                   obj.gameObject.GetComponent<CarControll>().durability += repairPoints;
•                   Destroy(this.gameObject);
•               }
•               else if (isShield == true)
•               {
•                   _playerCar = GameObject.FindWithTag("Player");
•                   obj.gameObject.tag = "Shield";
•                   _playerCarPosition = _playerCar.transform.position;
•                   _playerCarPosition.z = -0.1f;
•                           GameObject shieldObj = (GameObject)Instantiate(shield, _playerCarPosition,
    Quaternion.identity);
•                   shieldObj.transform.parent = _playerCar.transform;
•                   Destroy(this.gameObject);
•               }
•               else if (isSpeed == true)
•               {
•                   gameObject.GetComponent<SpriteRenderer>().enabled = false;
•                   _isActive = true;
•                   StartCoroutine("SpeedBoostActivated");
•               }
•               else if (obj.gameObject.tag == "EndOfRoad" && _isActive == false)
•                   Destroy(this.gameObject);
•           }
•       }
•
        IEnumerator SpeedBoostActivated()
•       {
•           while(duration > 0)
•           {
•               duration -= Time.deltaTime / speedBoost;
•               Time.timeScale = speedBoost;
•               yield return null;
•           }
•
            Time.timeScale = 1f;
•           Destroy(this.gameObject);
•       }
•   }
•
```

4. CarControll

```
•   public class CarControll : MonoBehaviour
•   {
•       public float carHorizontalSpeed;
•       public float maxDurability = 100f;
•       //[HideInInspector]
•       public float durability;
•
        private Vector3 _carPosition;
•
        public void Start()
•       {
•           _carPosition = this.gameObject.transform.position;
•           durability = 100f;
•       }
•
        private void Update()
•           {  carPosition.x  +=  Input.GetAxis("Horizontal")  *  carHorizontalSpeed  *  Time.deltaTime;
    //deltaTime real game time.
      _carPosition.x = Mathf.Clamp(_carPosition.x, -2.40f, 2.40f);
      this.gameObject.transform.position = _carPosition; } }
```

5. InfiniteRoad

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

public class InfiniteRoad : MonoBehaviour {

    public float scrollSpeed;

    private Vector2 offset;

    private void Update()
    {
        offset = new Vector2(0, Time.time * scrollSpeed); //params OsX, OsY

        GetComponent<Renderer>().material.mainTextureOffset = offset;
    }
}

```

6. Shield

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

public class Shield : MonoBehaviour
{
    public float shieldDuration;

    private void Update()
    {
        shieldDuration -= Time.deltaTime;

        if (shieldDuration <= 0)
        {
            this.gameObject.transform.parent.tag = "Player";
            Destroy(this.gameObject);
        } }
```

Chapter 05

Result

1. **Game interface**

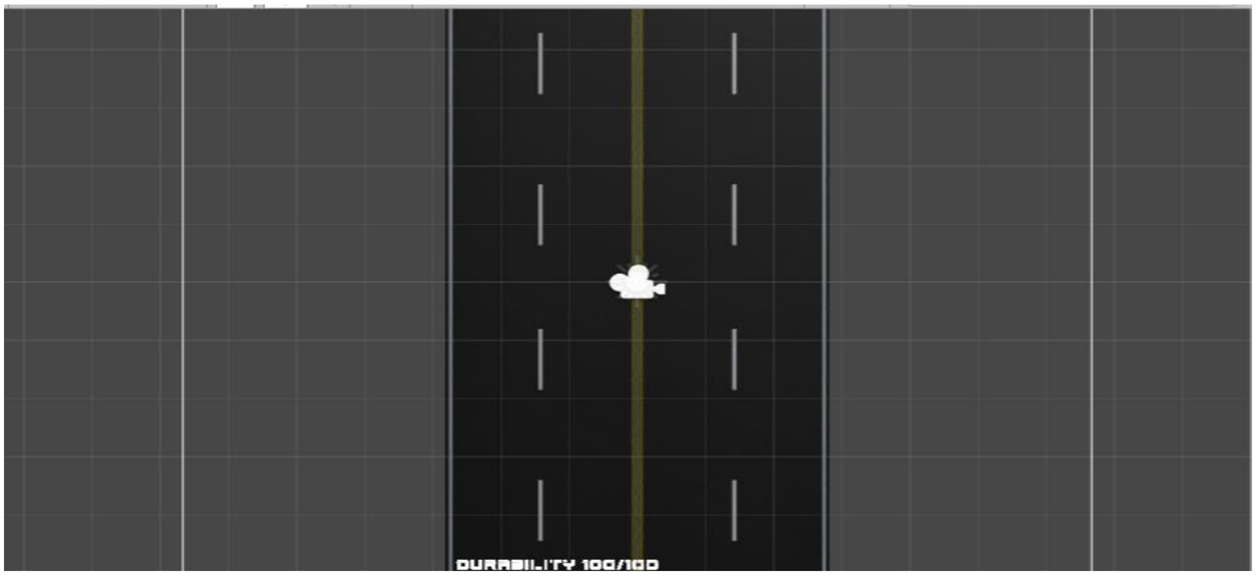

Fig 5.1 : Game interface

2. **Game play**

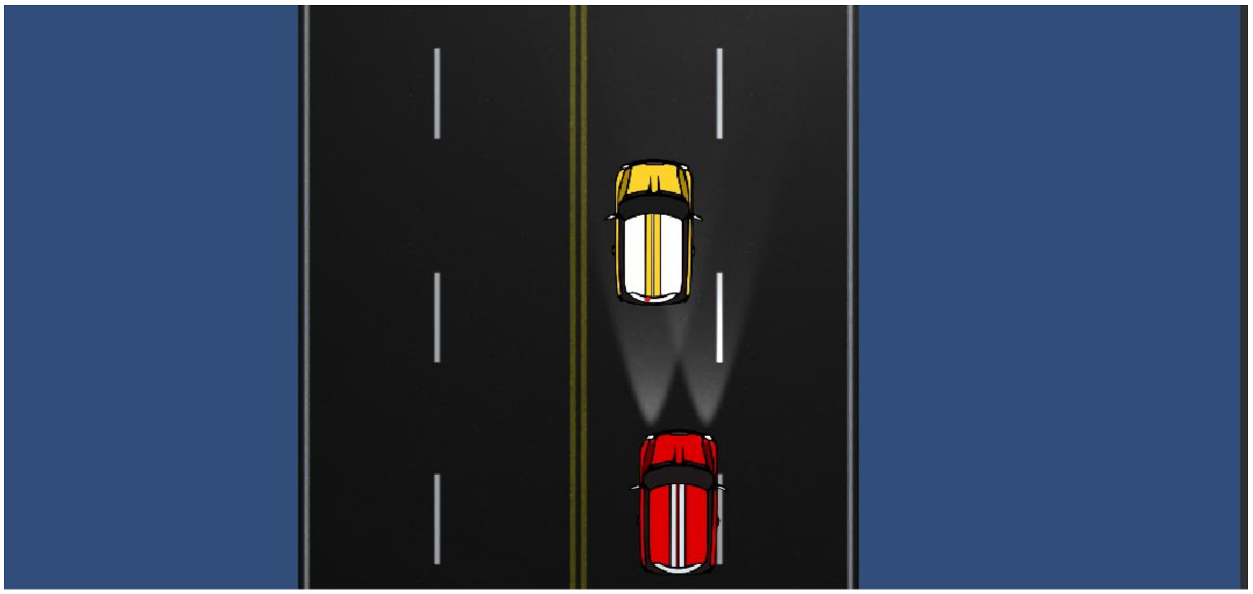

Fig 5.2: Game playing condition

Chapter 06

Conclusion and limitation

There could be various limitations for 2D racing car game developed in Unity. Some of the possible limitations could be:

■ Performance: The performance of the game might suffer if the game is not optimized properly. For instance, if the game has too many obstacles or too many particle effects, it might lead to frame rate drops and affect the gameplay.

■ Device compatibility: The game might not run smoothly on low-end devices or devices with outdated hardware specifications. This could lead to a poor user experience, and the game might not be enjoyable to play.

■ Design limitations: Depending on the design of the game, there could be limitations on the types of obstacles or environments that can be created. For example, if the game is set in a futuristic city, it might not be possible to add natural environments such as forests or mountains.

■ Lack of replayability: The game might lack replayability if there are not enough levels or if the gameplay is too repetitive. This could lead to players losing interest in the game over time.

■ Limited audience: The game might only appeal to a niche audience, which could limit its reach and popularity.

■ Lack of innovation: If the game doesn't offer anything unique or innovative, players may lose interest in it quickly. It's important to keep the game fresh and exciting by adding new features and gameplay mechanics over time.

In conclusion, the 2D racing car game created by Unity provides an exciting and immersive experience for players. With its smooth controls, challenging levels, and visually appealing graphics, the game is sure to keep players engaged for hours. this game offers something for everyone. Overall, the game is a great example of the capabilities of Unity as a game development platform and the potential for 2D games to deliver a thrilling gaming experience**.**

Chapter 07

Reference

Street 2D racing car game   - Unity 2018 Course Youtube.
https://www.youtube.com/watch?v=vF9R7dhHJCw

https://github.com/CosmicYogi/2D-Car-Racing-Game-with C-Sharp