CAR RACING GAME

Project Report
Submitted in partial fulfillment of the
Requirements for the award of the Degree of

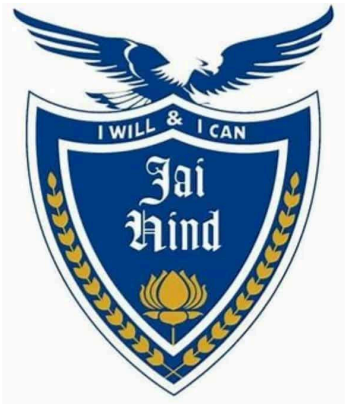
BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

Fiza Shaikh

UID - 20BIT055

Under the esteemed guidance of
Mr. Wilson Rao
Co-ordinator
And
Ms. Aafreen Shaikh
Mrs. Bertilla Fernandes
(Assistant Professors)



DEPARTMENT OF INFORMATION TECHNOLOGY

JAI HIND COLLEGE
(Autonomous)
MUMBAI, 400020

MAHARASHTRA
2020-23
JAI HIND COLLEGE

(Autonomous)

MUMBAI, 400020 MAHARASHTRA

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled, "**Car Racing Game**", is bonafied work of **FIZA SHAMSHUDDIN SHAIKH** bearing UID / Roll No.:(**20BIT055**) submitted in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from Jai Hind College Autonomous (University of Mumbai).

Internal Guide **Coordinator**

External Examiner

Date: **College Seal**

DECLARATION

I hereby declare that the project entitled, **"CAR RACING GAME"** done at Jai Hind College, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY) to be submitted as final semester project as part of our curriculum.

Name and Signature of the Student

ACKNOWLEDGEMENT

I would like to express my thanks to the people who have helped me most throughout my project. I am grateful to my **Prof. Ms. Aafreen Shaikh and Prof**. **Mrs**. **Bertilla Fernandes** for nonstop support for the project. I can't say thank you enough for her tremendous support and help.

I owe my deep gratitude to our HOD of Information Technology Department **Mr**. **Wilson Rao** who took keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

At last, but not the least I want to thank all of my friends who helped/treasured me out in completing the project, where they all exchanged their own interesting ideas, thoughts and made this possible to complete my project with all accurate information. I wish to thank my parents for their personal support or attention who inspired/encouraged me to go my own way.

ABSTRACT

Games have become an integral part of everyday life for many people. A traditional game often presents a situation where “players engage in an artificial conflict, defined by rules and results in a quantifiable outcome” Such artificial conflicts are often represented as a puzzle or a challenge, and having the puzzle solved or the challenge resolved provides a real-world purpose to the game players. This type of games is sometimes referred to as “serious” games. However, this kind of traditional or “serious games” has been increasingly replaced by electronic games, especially for the so-called “game generation”. This generation typically consists of “digital natives,” who, in contrast to the “digital immigrants” of the older generation, grew up playing a lot of games and who are trained in skills such as “dealing with large amounts of information quickly even at the early ages, using alternative ways to get information, and finding solutions to their own problems through new communication paths” .The 3D RACING GAME is a contemporary video game, and its objective is to offer the player a challenging and enjoyable experience in a car race against a game-controlled car. Although the player’s goal within the game is to win the race against the game-controlled car, the AI techniques adopted in the game are primarily designed to give the player an enjoyable time racing his or her car. In other words, the objective of winning by either side is not given the highest priority.

TABLE OF CONTENTS

List of Figures

1. INTRODUCTION

1.1 BACKGROUND:

There has been substantial research work that focuses on developing AI-controlled components of game systems, which can approximate or emulate human game playing styles. These components are often referred to as “bots.” The motivation for developing the “bots” is that a human player’s enjoyment in the gaming experience will be higher if he or she can be led to believe that the opponents in the game are other human players. In addition to decision trees, other AI techniques are also used for building bots so that they would mimic or emulate human behavior. in driving and racing games, human players expect to control their cars by making small adjustments to the car’s direction while driving, just like how they would control a car in real life. Therefore, the race car bot must make continuous adjustments to its car’s direction as opposed to simply driving in a straight line and turning only when a curve in the road approaches. To ensure the race car bot can mimic human behavior in this manner, some AI techniques were used for implementing the bots, which are explained in detail further.

1.2 OBJECTIVES

Games have become an integral part of everyday life for many people. The theme of our game is to compete with the other opponents that are controlled by computer in a racing tournament:

- to offer the player a challenging and enjoyable experience in a car race
- offer to the human player an enjoyable experience through his or her interaction with the game
- to score the highest score possible in the shortest time

1.3 PURPOSE, SCOPE and APPLICABILITY

1.3.1 Purpose

The main purpose of this work is to develop a game to simulate the virtual world. It would be written in C# and developed with UNITY 3D. The project includes a complete level of game documentation.

1.3.2 Scope

The goals of this project is to create an easy to use, pick up and play game that could be played by all ages as long as they have a desktop computer or laptop PC. The game would be single player. The project will be based on creating a racing game with the goal in mind of it being fun.

1.3.3 Applicability

This system is designed for those who want to play car racing games whether it be adults or kids.

1.4 Achievements

The project was successful in meeting all the requirements and fulfilling all the requirements that the solution demanded. The user can play games with ease due to the simple user interface. The objectives are met successfully.

1.5 ORGANIZATION OF REPORT

1.5.1 Requirement and Analysis

Normal requirements consist of objectives and goals that are stated during the meeting with the gamers. Normal requirements of our project are:

- User friendly efficient and lucrative system.
- Minimum maintenance cost (may be graphics definition).
- Availability of expected requirements within the PC/mobile configuration.
- Easy to operate.
- They observe our game as this is built with professional manner.
- The game with measured coding, professional thinking.

1.5.2 System Design

This shows us how our system actually works

- The system design is included in all conceptual schema diagrams which explains the inter-relation between the modules and their dependencies.
- Working, Features and Flow of the system is described in this part.

1.5.3 Implementation and Testing

- Implementation process is described in this chapter.
- Describes which all testing approaches are going to take place in this project.

1.5.4 Results and Discussion

- In the end, we get a full-fledged car racing Game.

1.5.5 Conclusion

In the proposed system, this platform will provide an easy way to play game using artificial intelligence. People can play the game at any time from wherever they want. This system gives an opportunity to improve skills.

2. SURVEY OF TECHNOLOGIES

Unity 3D:

Unity3D is a powerful cross-platform 3D engine and a user friendly development environment. The engine can be used to create three-dimensional (3D) and two-dimensional (2D) games, as well as interactive simulations and other experiences. The engine has been adopted by industries outside video gaming, such as film, automotive, architecture, engineering, construction, and the United States Armed Forces.

Unity is the creator of the world's most widely-used real-time 3D (RT3D) development platform, giving content creators around the world the tools to create rich, interactive 2D, 3D, VR and AR experiences.

The engine is used in games including Pokémon Go, Monument Valley, Call of Duty: Mobile, Beat Saber and Cuphead according to the Financial Times.

C#:

- C# is a programming language developed by Microsoft that runs on the .NET Framework.
- It is used to develop web apps, desktop apps, mobile apps, games and much more.

Visual Studio:

- Visual Studio is an integrated development environment from Microsoft.
- It is used to develop computer programs including websites, web apps, web services and mobile apps
- We will connect Unity 3D to VS to design our car controller and model

3. REQUIREMENT AND ANALYSIS

3.1 Problem Definition

3.1.1 Problem Statement

The computer market for car games is moving towards complex, photo-realistic 3D car simulation games. The problems are long startup times and high learning curves for the player. Furthermore, these games require a large amount of CPU time, which forces the players to upgrade their computer system to the hardware needs for these games.

3.1.2 Existing System

Previously, online games had no advantages and if provided, most of them have signup options. But to create easy gaming options, there was no single button sign up option. There was no provision to change the gaming controls and settings for the game environment for the convenience of the players. Most of the many were random AI features that were not available under existing systems. There was no provision by the administrator to add additional gaming sections for future reference in order to make their website higher in the future. The point that was not available was the decision control by the system. Previously the system was forced to use the default rules but the new system will overcome all these difficulties

3.1.3 Proposed System

In this new gaming environment, playing games requires no registration and no fees. After all, there are two types of games that users can play. They should have the option to switch between these games while playing, and the system will monitor specific modules of their size. To create an interesting gaming environment, these two games will be available, graphical number puzzle games and another is disaster Strike. We have tried to code smart decisions to adopt algorithms to make the system more intelligent and learn tricks from people. At the same time, most of us have problems managing keys and other controls while playing. To overcome all these problems, users are given the freedom to change their convenient gaming controls and change the settings to control them while playing games.

3.2 Requirement Specification

Functional Requirements:

The game should be able to:

- **Start the game**
- **Stop the game**
- **Change the direction of the driven car**
- **Change the speed of the driven car**
- **Control the user driven car**
- **Configure the cars**
- **Select the type of car**
- **Select the type of track**

Non-Functional Requirements:

- **Availability**: This system will be easily available to authorized users
- **Reliability**: The system is highly reliable and secure
- **Maintainability**: The system is easily maintainable and provides an easy access of resources
- **Flexible**: Changes can be made, if any.
- **Integrity:** The information entered by the user into the proposed system will be validated to avoid invalid data entering the system.
- **Usability:** The user interface of the system will be simple and easy to understand.

3.3 Planning and Scheduling

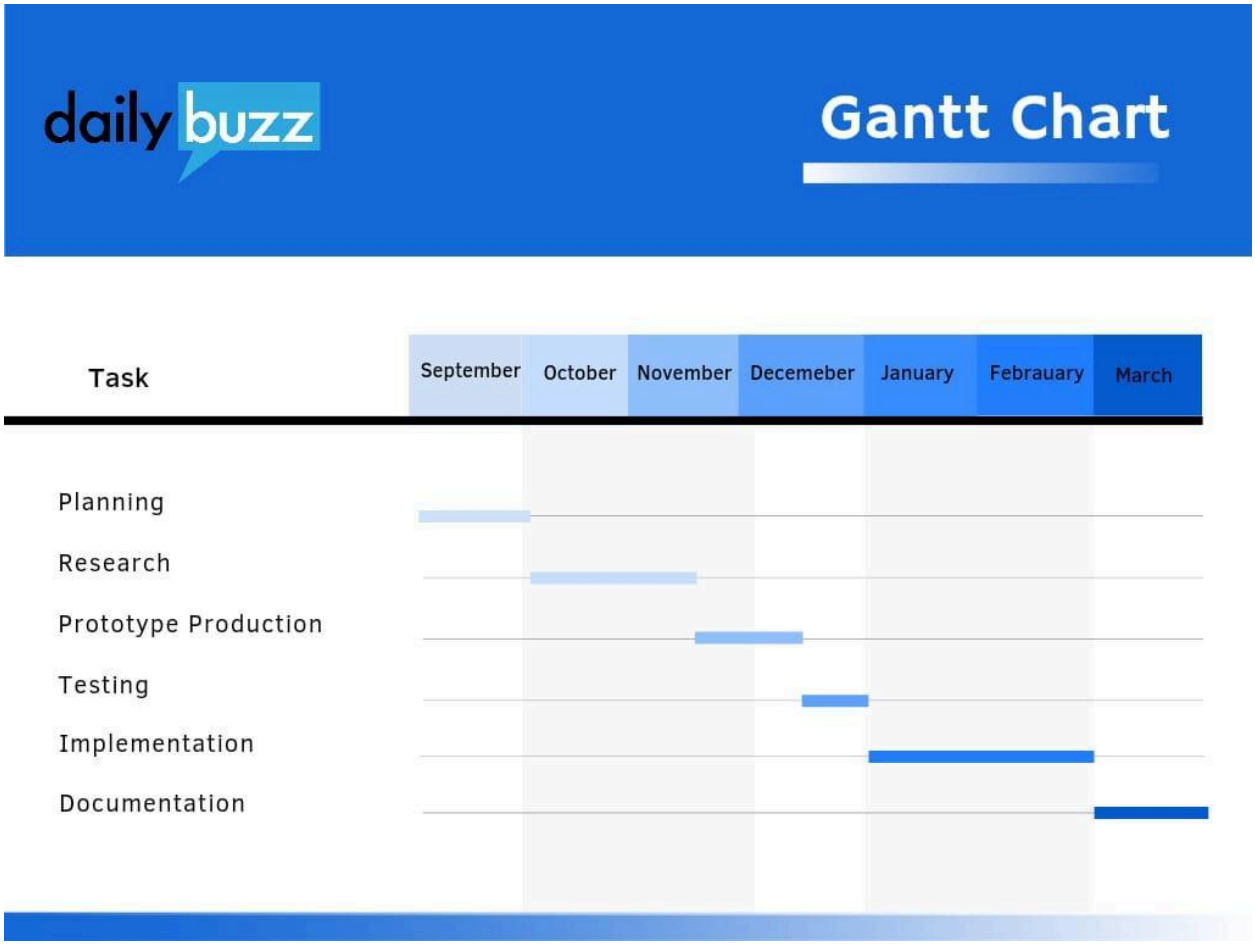


Figure 1. Gantt Chart

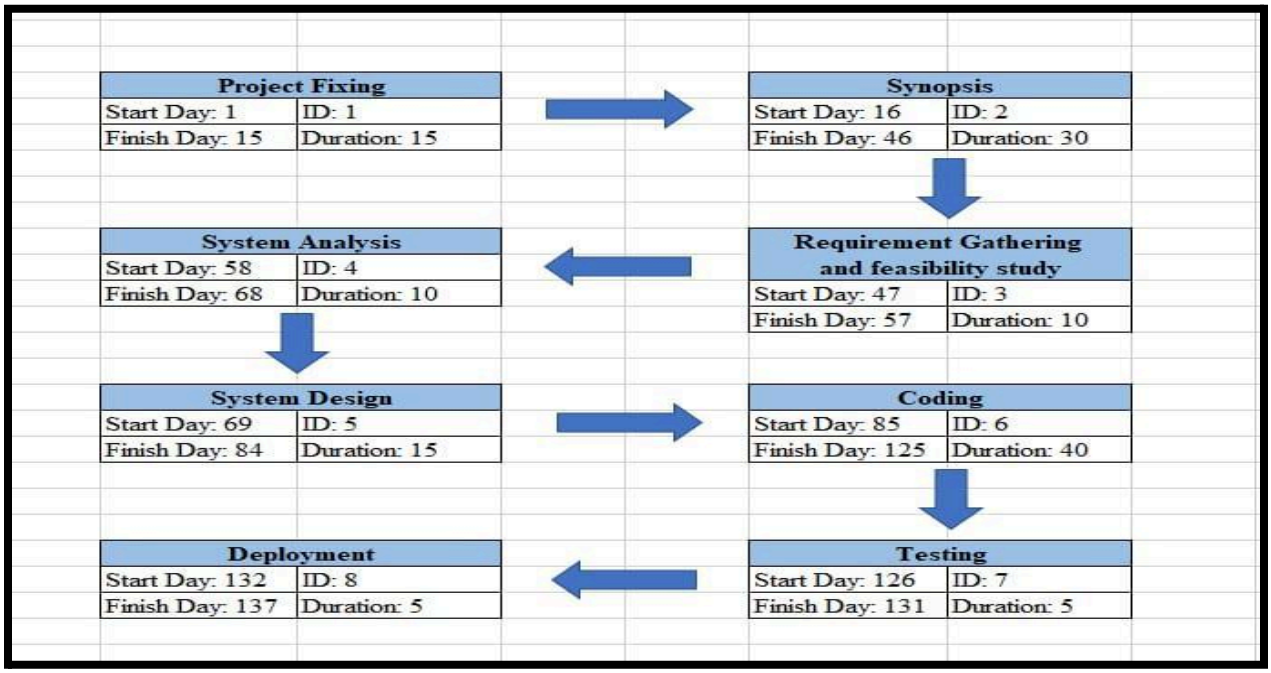


Figure 2: PERT Chart

3.4 Software and Hardware Requirements

Software Requirements

Operating System: Windows

Technology: C#

IDE: Visual Code/Notepad++

Hardware Requirements:

Laptop/PC

RAM: 8GB and above

System type: 64-bit operating system, x64-based processor

3.5 Preliminary Product Description

Module 1: Car Design

This module contains the main element of our game “A CAR”. We will design the car with waypoints using AI. Also the wheels to work properly on the track. Designing will also include the car model, color and overall look.

Module 2: Track and Terrain Design

This module will contain the design of the terrain. All the bumps, mountains and patches will be added to the terrain. The track will contain the race track designed for the car to run on.

Module 3: Main Menu

This module will enable the player to start the game. It will also include options to change and select the car and track

Module 4: Time tracker and Lap tracker

This module keeps a track of the number of laps the car takes and also tracks the time it took to complete one lap. This way a track of the high score can be kept by the player.

Module 5 Car Controller

This is the main module which will control how the car will run on the tracks. This includes functions such as Applying Brakes, Controlling the car motor, etc.

Module 6: Audio/Visual effects

This module includes audio for every time the car dashes or while its running on the track or when the car completes one lap. Also at every dash or drift the car will emit smoke

Module 7: Mini Map

This module will include a mini map of the entire terrain and track and will also show the position of the car.

3.6 Conceptual Models:

Figure 3: Handbook Diagram

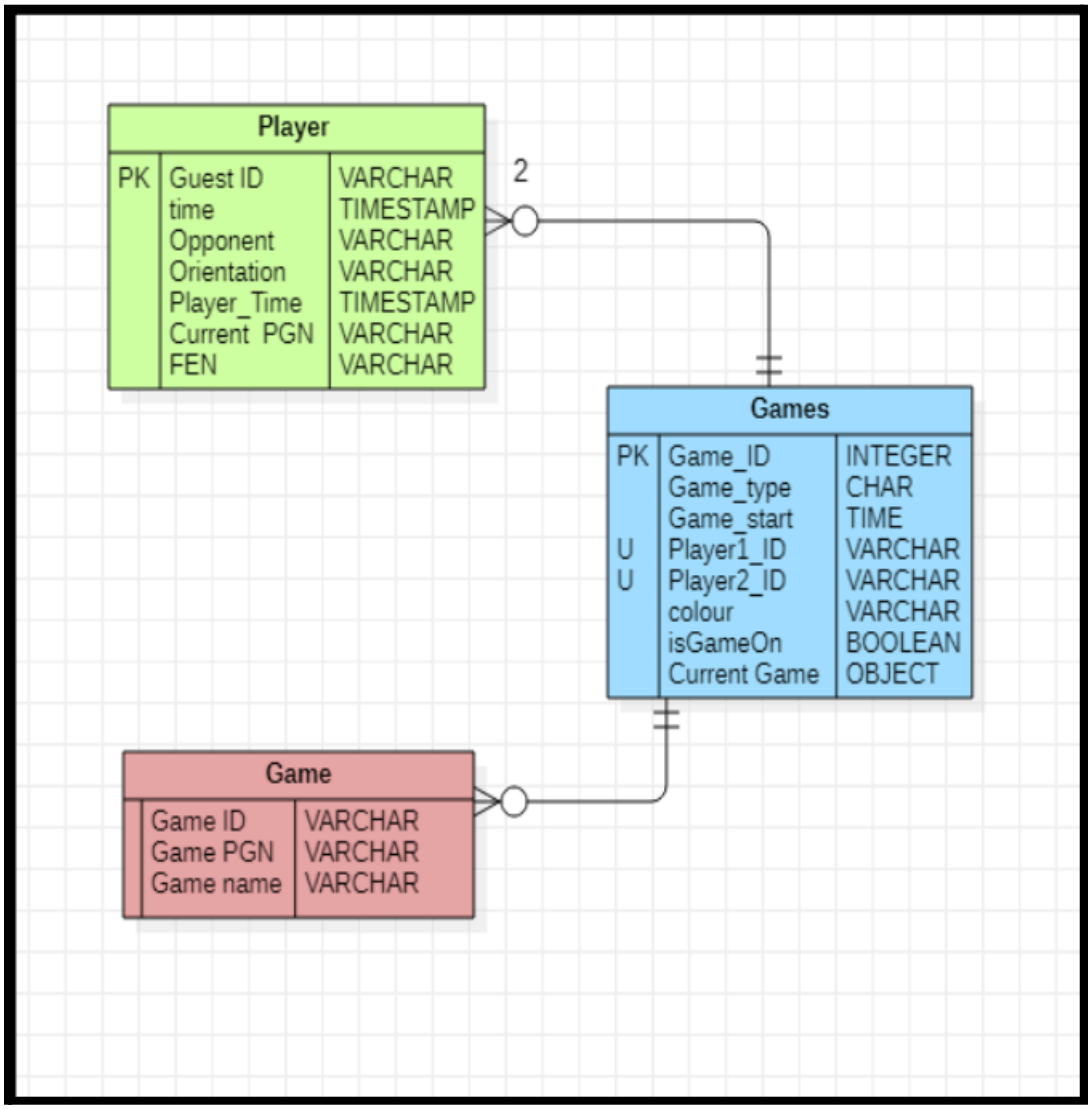


Figure 4: ER Diagram

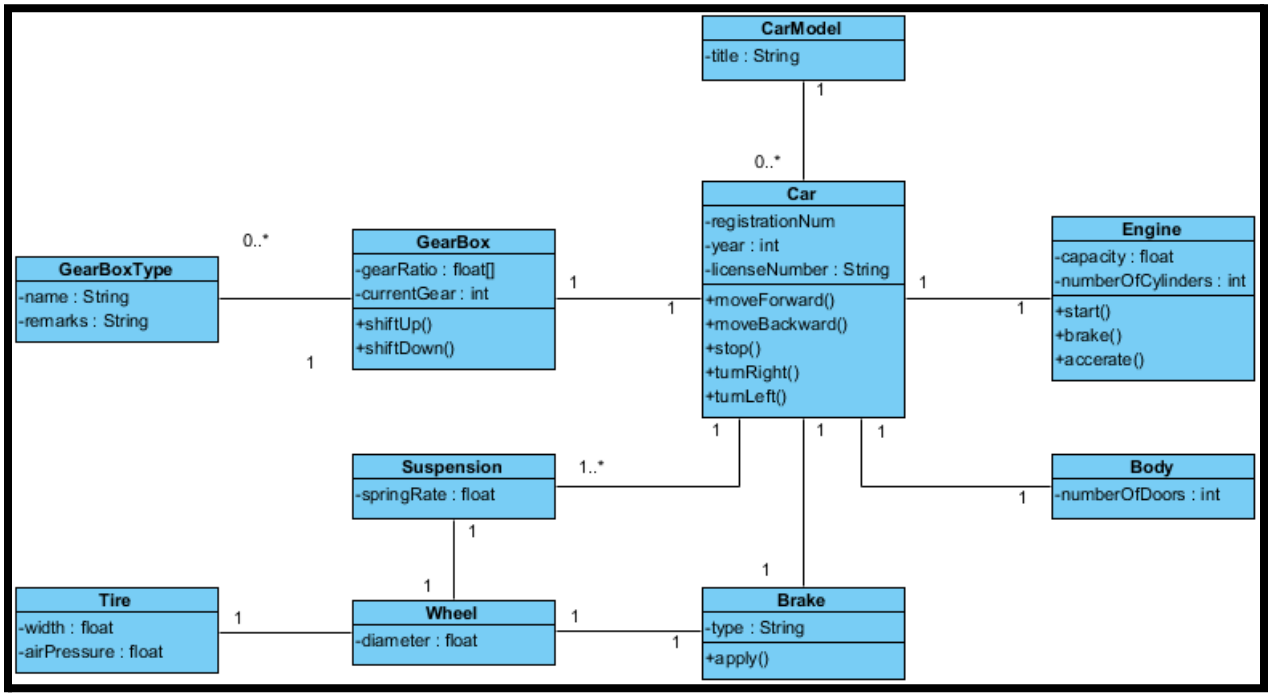


Figure 5: CLASS Diagram

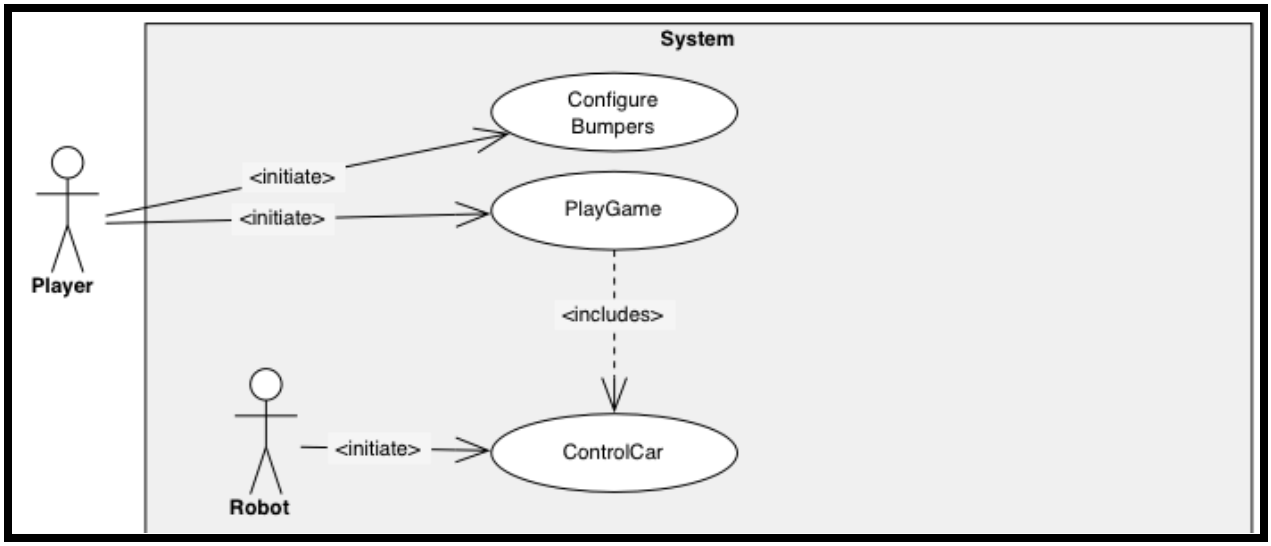


Figure 6: USE CASE Diagram

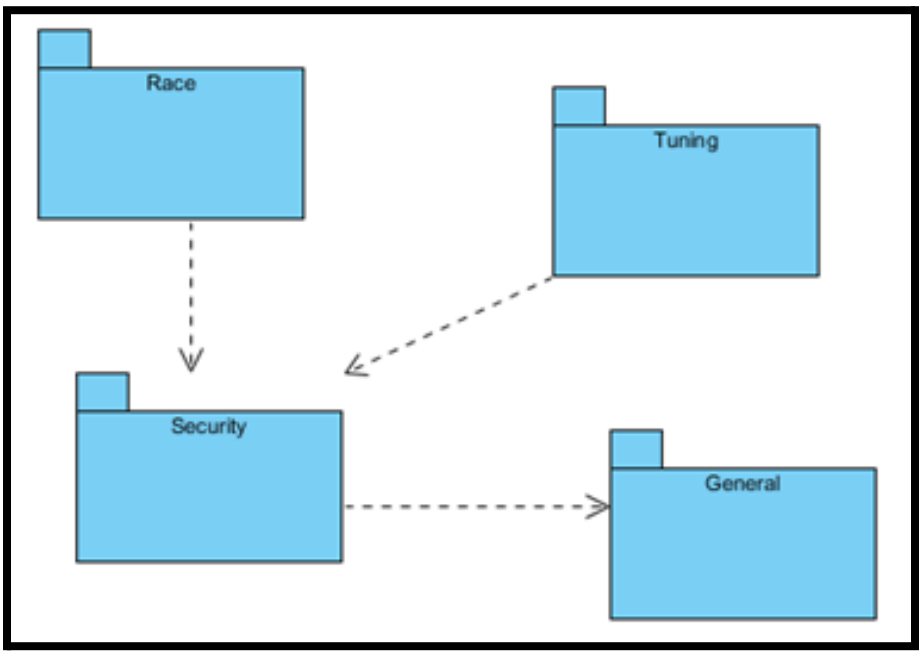
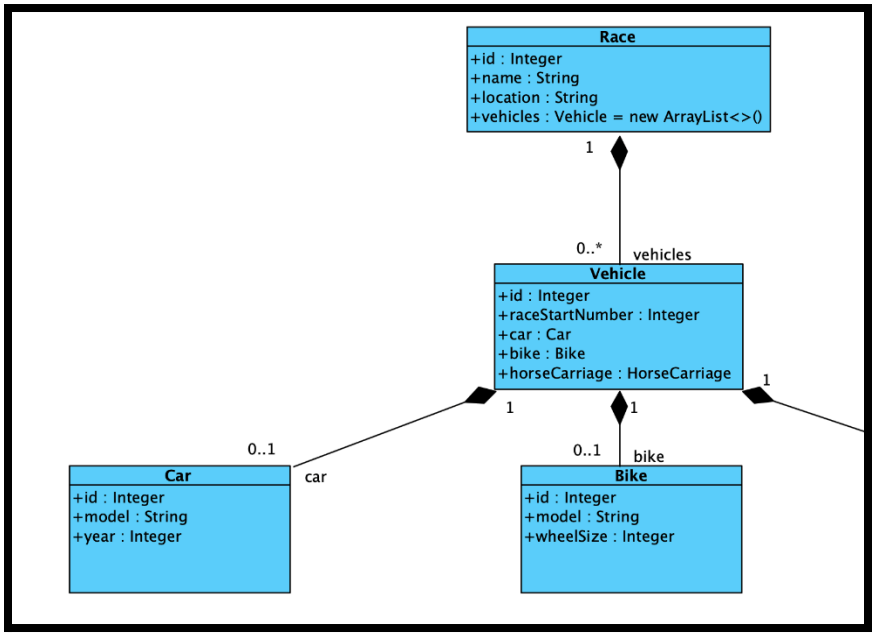


Figure 7: Package Diagram

Figure 8:



ACTIVITY Diagram

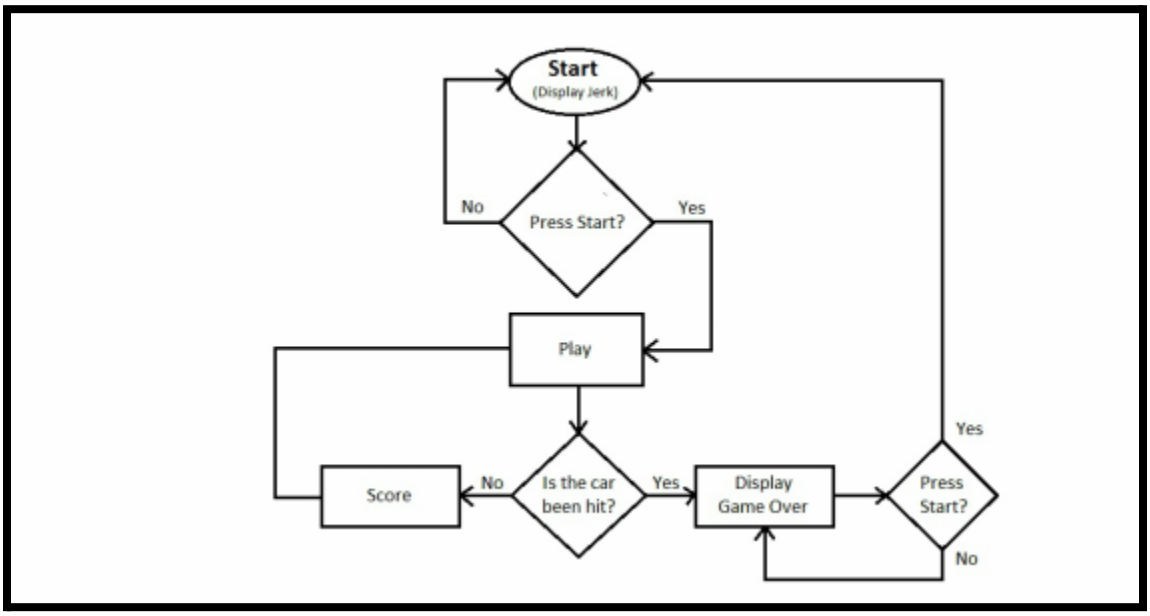


Figure 9: SEQUENCE Diagram

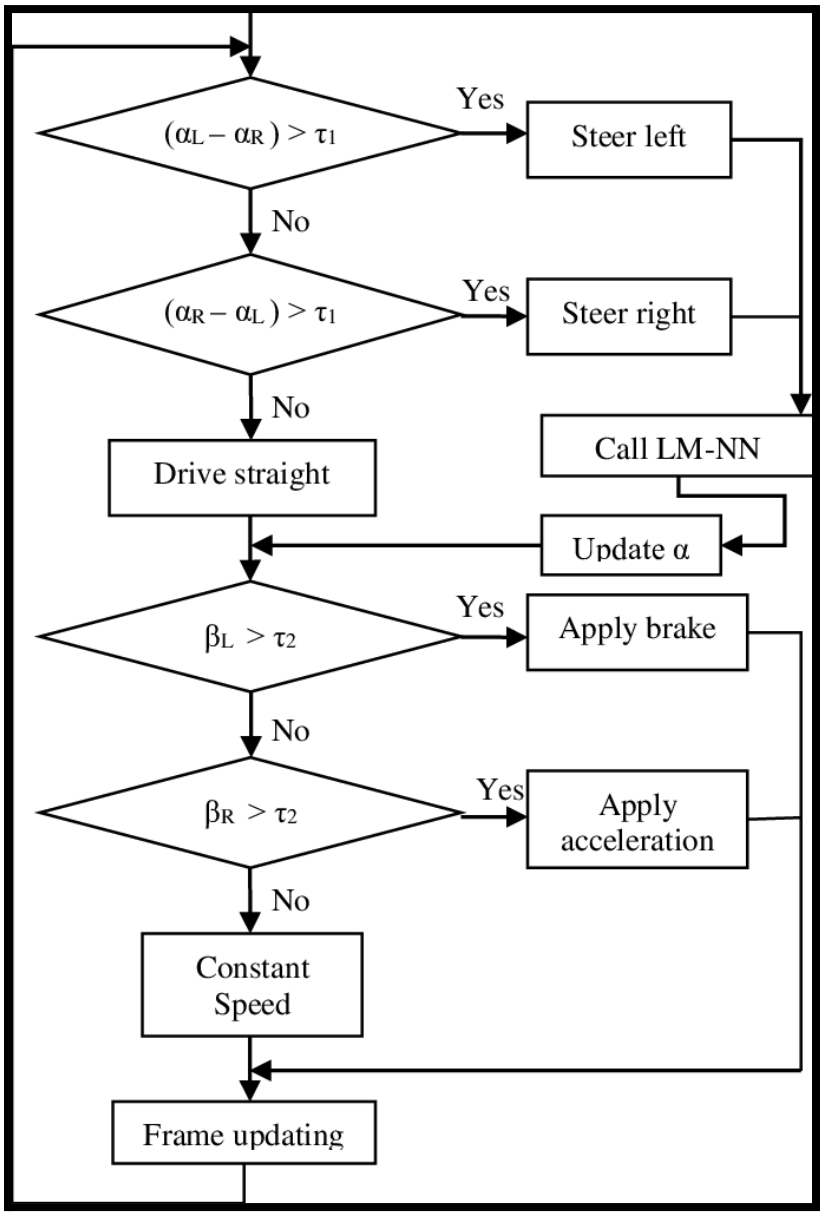


Figure 10: State Chart Diagram

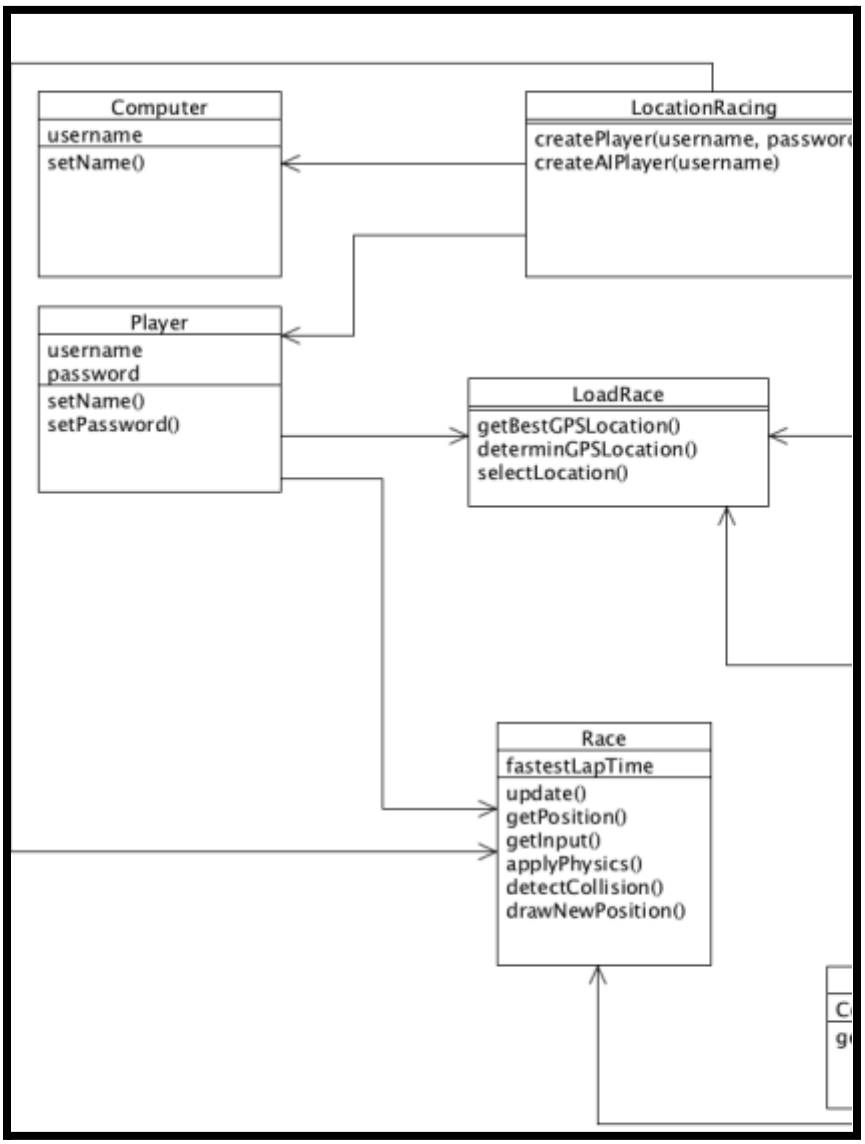
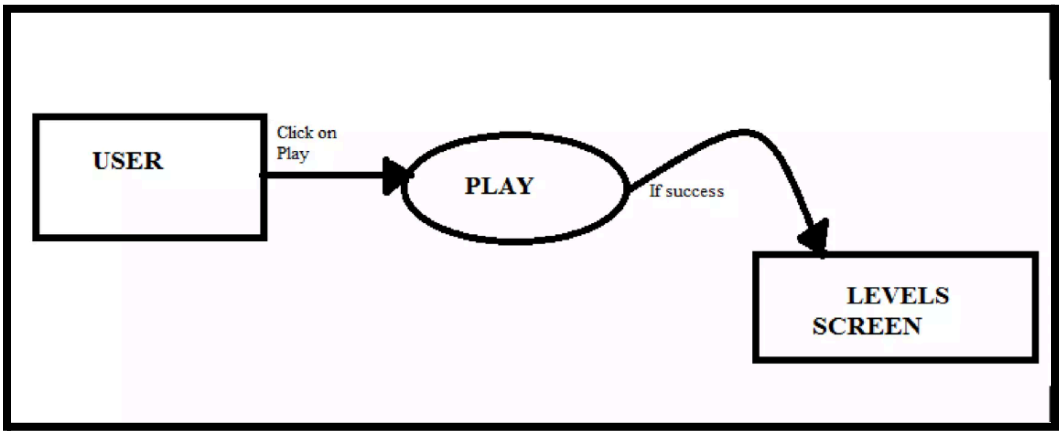
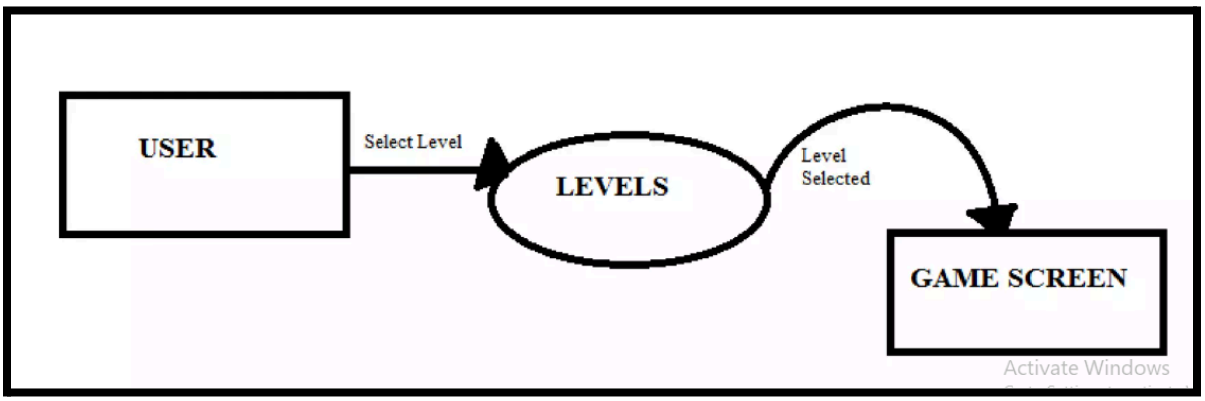


Figure 12: COMPONENT Diagram

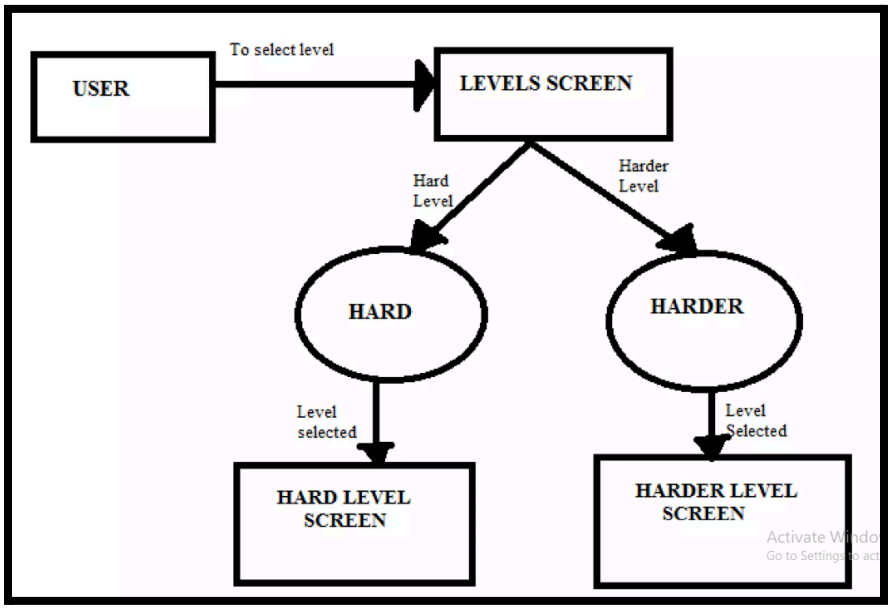
Level 0 DFD:



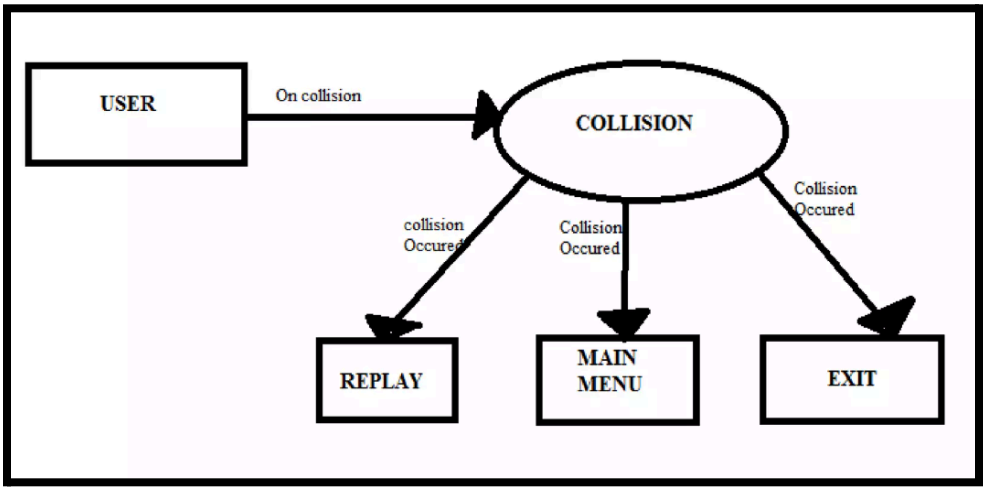
Level 1 DFD:



Level 2 DFD:



Level 3 DFD:



Level 4 DFD:

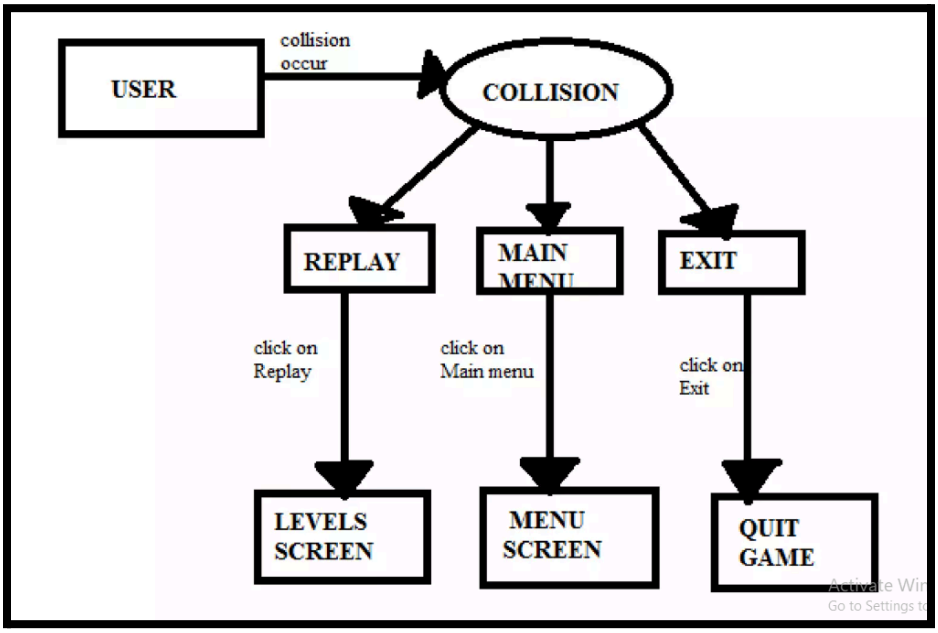


Figure 14: DATA FLOW Diagram

4. SYSTEM DESIGN

4.1 Basic Modules:

Car Design

This module contains the main element of our game “A CAR”. We will design the car with waypoints using AI. Also the wheels to work properly on the track. Designing will also include the car model, color and overall look.

Track and Terrain Design

This module will contain the design of the terrain. All the bumps, mountains and patches will be added to the terrain. The track will contain the race track designed for the car to run on.

Main Menu

This module will enable the player to start the game. It will also include options to change and select the car and track

Time tracker and Lap tracker

This module keeps a track of the number of laps the car takes and also tracks the time it took to complete one lap. This way a track of the high score can be kept by the player.

Audio/Visual effects

This module includes audio for every time the car dashes or while its running on the track or when the car completes one lap. Also at every dash or drift the car will emit smoke

Mini Map

This module will include a mini map of the entire terrain and track and will also show the position of the car.

4.2 Data Design:

4.2.1 Data Schema:

- **CAR:**

CAR_ID INT
MODEL STRING
NAVIGATION INT
CAR_TYPE STRING

- **ACTION:**

Id INT
round_id INT
action_type INT
Action_value FLOAT
Modification_time INT

- **ROUNDS:**

game_id INT
lap_id INT

best_time ID INT
lap_tracker INT

4.2.2 Data Integrity and Constraints:

1) The space can be limited if not properly handled.

2) The project will only work within a particular platform.

3) Shortcomings in software design.

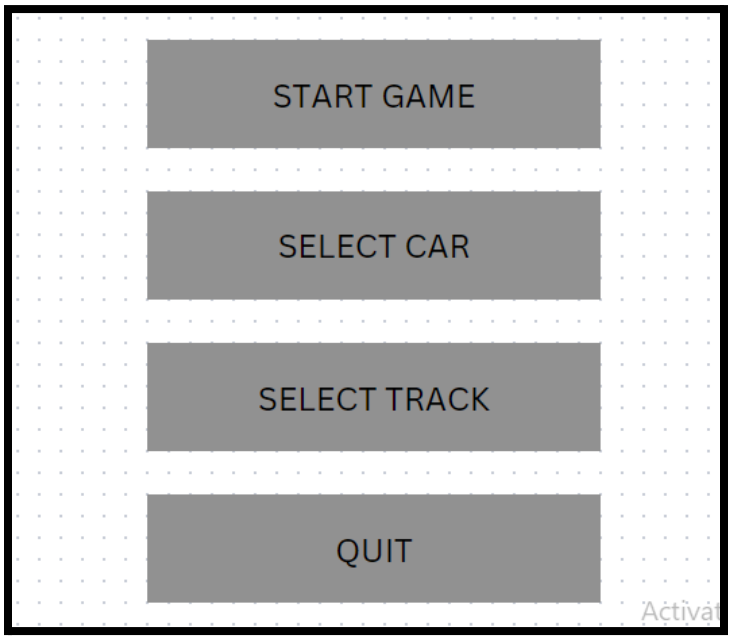
4) GUI is only in English.

5) UI can be confusing

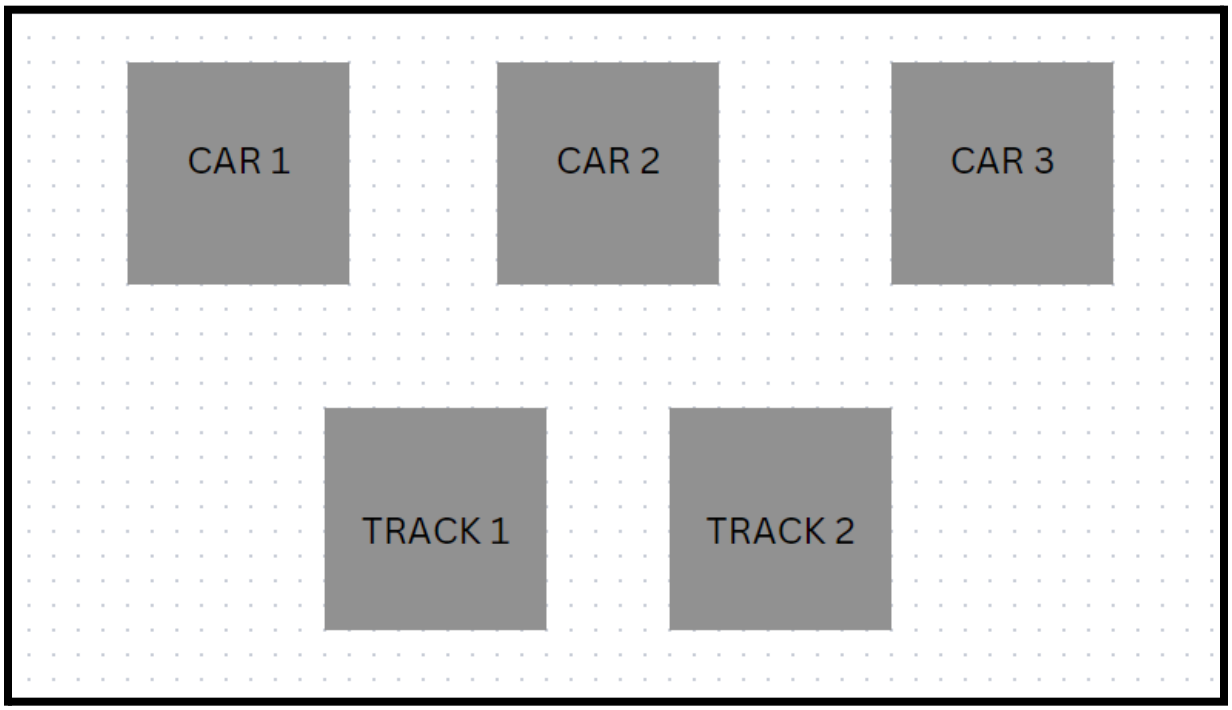
6) No retry option

4.3 User Interface:

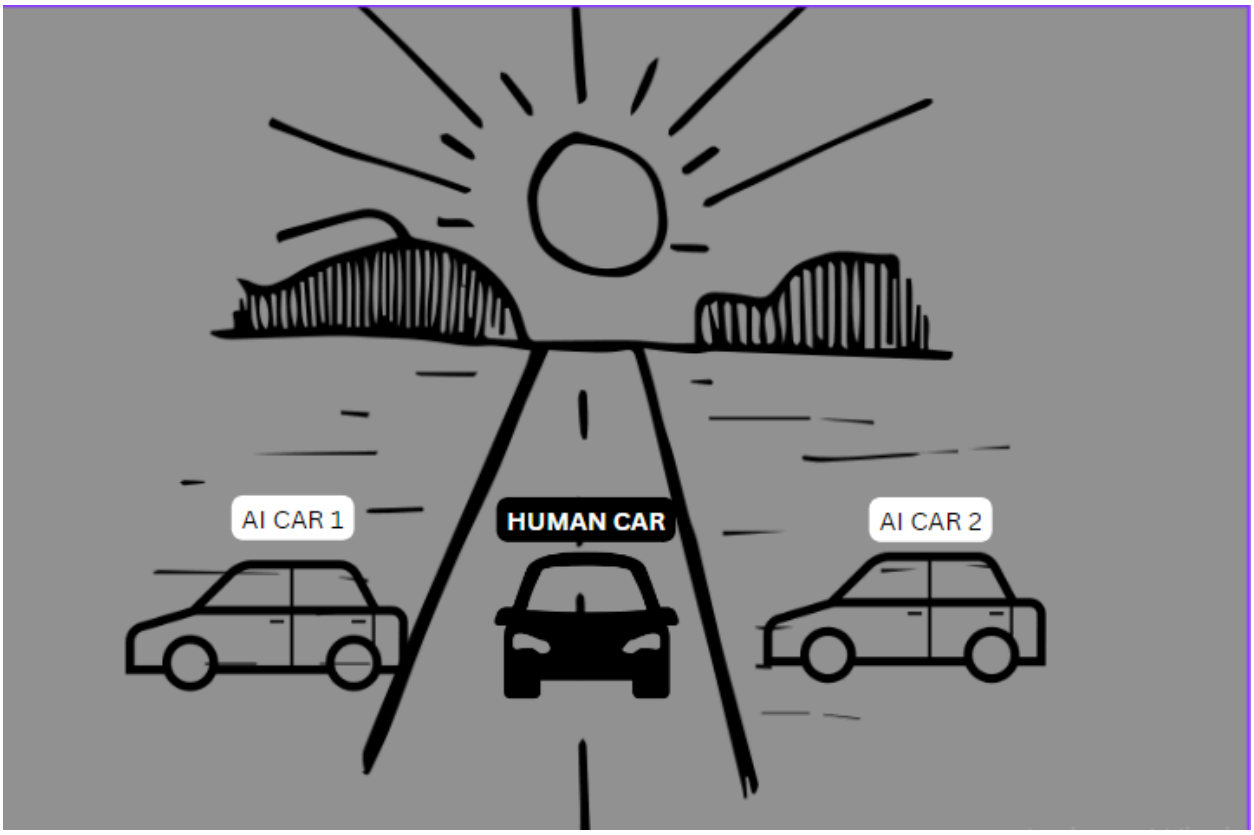
- **MAIN MENU**



- **CAR AND TRACK SELECTION:**



 GAME PAGE:



4.4 Security Issues:

- **Authorization:**

  The system is authorized to give permission to register users to the system and login access to registered users

- **Authentication:**

  This approach is being used because every user has to authenticate himself/herself to operate the system.

- **Using a web application firewall:**

  This feature can monitor the network traffic and determine unexpected traffic and hence can block possible attacks like denial of service. Web application firewalls can also be used to detect cross-site request forgery attacks**.**

- **Using computer backups and boosting physical security:**

  These help an institution to retrieve data in case of any network attack, natural disasters and physical attacks.

5. IMPLEMENTATION AND TESTING

5.1 Implementation Approach:

a. Introduction:

The **CAR RACING GAME** starts with a main menu where the user (player) has options to select the type of car he wants to drive from 3 different options. The player also gets an option to select the track he wants to drive on. He has 2 options for that. After the player clicks on the start game button, the select track and car will be displayed and the race against the player and 2 AI cars will start. After the completion of 2 laps whoever wins will be displayed on the screen. The game also contains a lap counter, best lap time counter and a mini map.

b. Input and Output Design Implementation:

This aspect discusses the input and output design of the system so as to give a clear view of what is meant by the terms input/output design.

Input: It should be understood that input is the data that should be fed into the system to serve as basis for the desired output. The input to the new system is designed to capture data in the subsystem.

Output: With the identification of the input design, it is logical to state the output design too. Output is the data that should go out of the system. The output design of this system includes reports that can be viewed by the game developer

c. System Implementation

The system consists of multiple modules. The player also gets an option to select the track he wants to drive on. He has 2 options for that. After the player clicks on the start game button, the select track and car will be displayed and the race against the player and 2 AI cars will start. After the completion of 2 laps whoever wins will be displayed on the screen. The game also contains a lap counter, best lap time counter and a mini map.

d. Code Module:

Unity is a native C++-based game engine. You write code in C#, JavaScript (UnityScript) . Your code, not the Unity engine code, runs on Mono or the Microsoft .NET Framework, which is Just-in-Time (JIT) compiled (except for iOS, which doesn't allow JIT code and is compiled by Mono to native code using Ahead-of-Time [AOT] compilation). Unity lets you test your game in the IDE without having to perform any kind of export or build. When you run code in Unity, you're using Mono version 3.5, which has API compatibility roughly on par with that of the .NET Framework 3.5/CLR 2.0.

The game I have developed has been entirely coded using C#. C# (C-Sharp) is a programming language developed by Microsoft that runs on the .NET Framework. It is used to develop web apps, desktop apps, mobile apps, games and much more.

Apart from that I've connected my unity 3D software to visual studio professional IDE for easier coding.

e. Project Summary:

The game starts with a main menu where the user (player) has options to select the type of car he wants to drive from 3 different options. They player also gets an option to select the track he wants to drive on. He has 2 options for that. After the player clicks on the start game button, the select track and car will be displayed and the race against the player and 2 AI cars will start. After the completion of 2 laps whoever wins will be displayed on the screen. The game also contains a lap counter, best lap time counter and a mini map.

5.2 Coding Details and Code Efficiency:

Coding Details:

Game Options:

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

public class GameOption : MonoBehaviour {


// Update is called once per frame
void Update () {
        if (Input.GetKeyDown(KeyCode.KeypadEnter))
        {
            if (Time.timeScale == 1)
            {
                Time.timeScale = 0;
            }
            else
            {
                Time.timeScale = 1;
            }
        }
    }
}
```

Car:

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

public class GlobalCar : MonoBehaviour {
    public static int CarTyp;
    public GameObject TrackWindow;
    public void redCar()
    {
        CarTyp = 1;
        TrackWindow.SetActive(true);
    }
    public void blueCar()
    {
        CarTyp = 2;
        TrackWindow.SetActive(true);
    }
    public void policeCae()
    {
        CarTyp = 3;
        TrackWindow.SetActive(true);
    }
}
```

Point Trigger:

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

public class HalfPointTrigger : MonoBehaviour {

    public GameObject LapCompleteTrig;
    public GameObject HalfLapTrig;

    void OnTriggerEnter()
    {
        LapCompleteTrig.SetActive(true);
        HalfLapTrig.SetActive(false);
    }
}
```

Lap Time Complete:

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

public class LapComplete : MonoBehaviour
{

    public GameObject LapCompleteTrig;
    public GameObject HalfLapTrig;

    public GameObject MinuteDisplay;
    public GameObject SecondDisplay;
    public GameObject MilliDisplay;

    public GameObject LapTimeBox;
    // count the laps that was compeleted
    public GameObject LapCounter;
    public int LapsDone;
    void OnTriggerEnter()
    {
        LapsDone += 1;
        if (LapTimeManager.SecondCount <= 9)
        {
            SecondDisplay.GetComponent<Text>().text = "0" + LapTimeManager.SecondCount +
". ";
        }
        else
        {
            SecondDisplay.GetComponent<Text>().text = "" + LapTimeManager.SecondCount +
". ";
        }

        if (LapTimeManager.MinuteCount <= 9)
        {
            MinuteDisplay.GetComponent<Text>().text = "0" + LapTimeManager.MinuteCount +
". ";
        }
        else
        {
            MinuteDisplay.GetComponent<Text>().text = "" + LapTimeManager.MinuteCount +
". ";
        }
        MilliDisplay.GetComponent<Text>().text = "" + LapTimeManager.MilliCount;
```

```
        PlayerPrefs.SetInt("MinSave",LapTimeManager.MinuteCount);
        PlayerPrefs.SetInt("SecSave",LapTimeManager.SecondCount);
        PlayerPrefs.SetFloat("MilliSave",LapTimeManager.MilliCount);

        LapTimeManager.MinuteCount = 0;
        LapTimeManager.SecondCount = 0;
        LapTimeManager.MilliCount = 0;
        ///
        LapCounter.GetComponent<Text>().text = "" + LapsDone;
        HalfLapTrig.SetActive(true);
        LapCompleteTrig.SetActive(false);
    }
}
```

LapTime Manager:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class LapTimeManager : MonoBehaviour {

    public static int MinuteCount;
    public static int SecondCount;
    public static float MilliCount;
    public static string MilliDisplay;

    public GameObject MinuteBox;
    public GameObject SecondBox;
    public GameObject MilliBox;

    // Update is called once per frame
    void Update () {

        MilliCount += Time.deltaTime * 10;
        MilliDisplay = MilliCount.ToString("F0");
        MilliBox.GetComponent<Text>().text = "" + MilliDisplay;

        if (MilliCount >= 10)
        {
            MilliCount = 0;
            SecondCount += 1;
        }

        if (SecondCount <= 9)
        {
            SecondBox.GetComponent<Text>().text = "0" + SecondCount + ".";
        }
        else
        {
            SecondBox.GetComponent<Text>().text = "" + SecondCount + ".";
        }

        if (SecondCount >= 60)
        {
            SecondCount = 0;
            MinuteCount += 1;
        }

        if (MinuteCount <= 9)
```

```
        {
            MinuteBox.GetComponent<Text>().text = "0" + MinuteCount + ":";
        }
        else
        {
            MinuteBox.GetComponent<Text>().text = "" + MinuteCount + ":";
        }

    }

}
```

LoadLap Time:

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

public class LoadLapTime : MonoBehaviour {
    public int MinCount;
    public int SecCount;
    public float MilliConut;
    public GameObject MinDisplay;
    public GameObject SecDisplay;
    public GameObject MilliDisplay;
    // to show user left time in ( miniuts , seconds and milliseconds)
    void Start () {

        MinCount = PlayerPrefs.GetInt("MinSave");
        SecCount = PlayerPrefs.GetInt("SecSave");
        MilliConut = PlayerPrefs.GetFloat("MilliSave");

        MinDisplay.GetComponent<Text>().text = "" +MinCount+":";
        SecDisplay.GetComponent<Text>().text = "" +SecCount+".";
        MilliDisplay.GetComponent<Text>().text = ""+MilliConut;
    }

}
```

Tracker:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
 public class Tracker : MonoBehaviour
{
    public GameObject TheMarker;
    public GameObject Mark01;
    public GameObject Mark02;
    public GameObject Mark03;
    public GameObject Mark04;
    public GameObject Mark05;
    public GameObject Mark06;
    public GameObject Mark07;
    public GameObject Mark08;
    public GameObject Mark09;
    public GameObject Mark10;
    public GameObject Mark11;
    public GameObject Mark12;
    public GameObject Mark13;
    public int MarkTracker;
    void Update()
    {
        if (MarkTracker == 0)
        {
            TheMarker.transform.position = Mark01.transform.position;
        }
        if (MarkTracker == 1)
        {
            TheMarker.transform.position = Mark02.transform.position;
        }
        if (MarkTracker == 2)
        {
            TheMarker.transform.position = Mark03.transform.position;
        }
        if (MarkTracker == 3)
        {
            TheMarker.transform.position = Mark04.transform.position;
        }
        if (MarkTracker == 4)
        {
            TheMarker.transform.position = Mark05.transform.position;
        }
        if (MarkTracker == 5)
        {
```

```
            TheMarker.transform.position = Mark06.transform.position;
        }
        if (MarkTracker == 6)
        {
            TheMarker.transform.position = Mark07.transform.position;
        }
        if (MarkTracker == 7)
        {
            TheMarker.transform.position = Mark08.transform.position;
        }
        if (MarkTracker == 8)
        {
            TheMarker.transform.position = Mark09.transform.position;
        }
        if (MarkTracker == 9)
        {
            TheMarker.transform.position = Mark10.transform.position;
        }
        if (MarkTracker == 10)
        {
            TheMarker.transform.position = Mark11.transform.position;
        }
        if (MarkTracker == 11)
        {
            TheMarker.transform.position = Mark12.transform.position;
        }
        if (MarkTracker == 12)
        {
            TheMarker.transform.position = Mark13.transform.position;
        }
    }
    IEnumerator OnTriggerEnter(Collider collision)
    {
        if (collision.gameObject.tag == "PoliceCar01")
        {
            this.GetComponent<BoxCollider>().enabled = false;
            MarkTracker += 1;
            if (MarkTracker == 13)
            {
                MarkTracker = 0;
            }
            yield return new WaitForSeconds(1);
            this.GetComponent<BoxCollider>().enabled = true;
        }
    }
}
```

5.2.2 Code efficiency:

The gaming app is developed completely using C# language

Reasons:

- C# executes faster and it has some language features which provide significant advantages over Unity Script.
- Developing with C# has advantage of Visual studio. It provides better code compilation features than Mono develop.
- The majority of Assets on stores are developed using C# or offer both JS and C#.
- C# is popular, mature, native to .NET and JS.

5.3 Testing Approach:

Functional Testing:

Functional testing can be performed on survival of the fastest to check that the appropriate output is coming from the input provided. Each Activity in this game include validation in which the next activity is depend upon the user choice.

5.3.1 Unit Testing:

This testing is done by the developers on the individual units of source code assigned area. The developers use test data that is separate from the test data of the quality assurance team. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality

.

5.3.2 Integration Testing:

The testing of combined parts of an application to determine if they function correctly together is Integration testing In this phase of testing we are emphasizing on how two or more applications or functions are working with each other. like how are they activities are depend upon each other, which activity is come under which module.

5.3.3 System Testing:

This is the next level in the testing and tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards. This type of testing is performed by a specialized testing team

5.4 Modifications and Improvements:

Many more areas can be included in the proposed system such that it can be have more levels and activities.

Areas which can be included in the system are:

- Settings activity that can be occurred after Menu in which user can choose the controls and other options like audio settings or also vehicle settings.
- Accelerator and brakes can be added to the game for better controls. It can make our project more general and attractive.

6. RESULTS AND DISCUSSIONS:

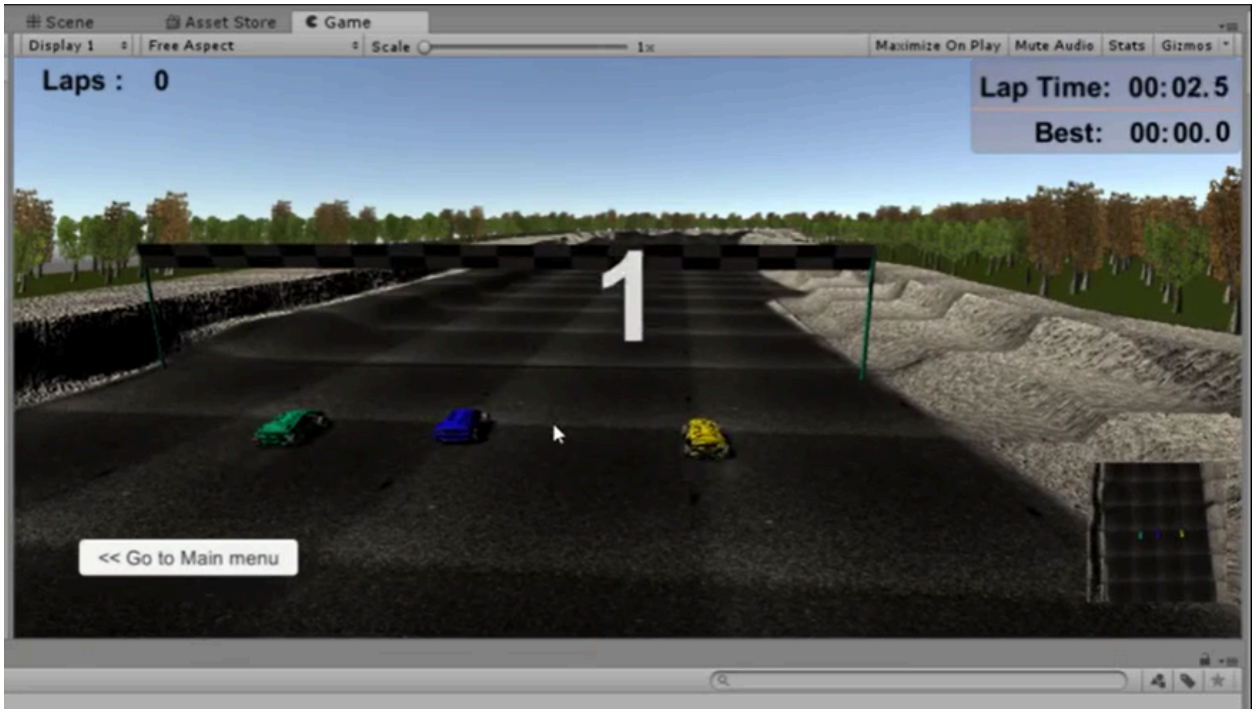
6.1 User Documentation:

The car racing game starts with a main menu where the user (player) has options to select the type of car he wants to drive from 3 different options. They player also gets an option to select the track he wants to drive on. He has 2 options for that. After the player clicks on the start game button, the select track and car will be displayed and the race against the player and 2 AI cars will start. After the completion of 2 laps whoever wins will be displayed on the screen. The game also contains a lap counter, best lap time counter and a mini map.

- **MAIN MENU**

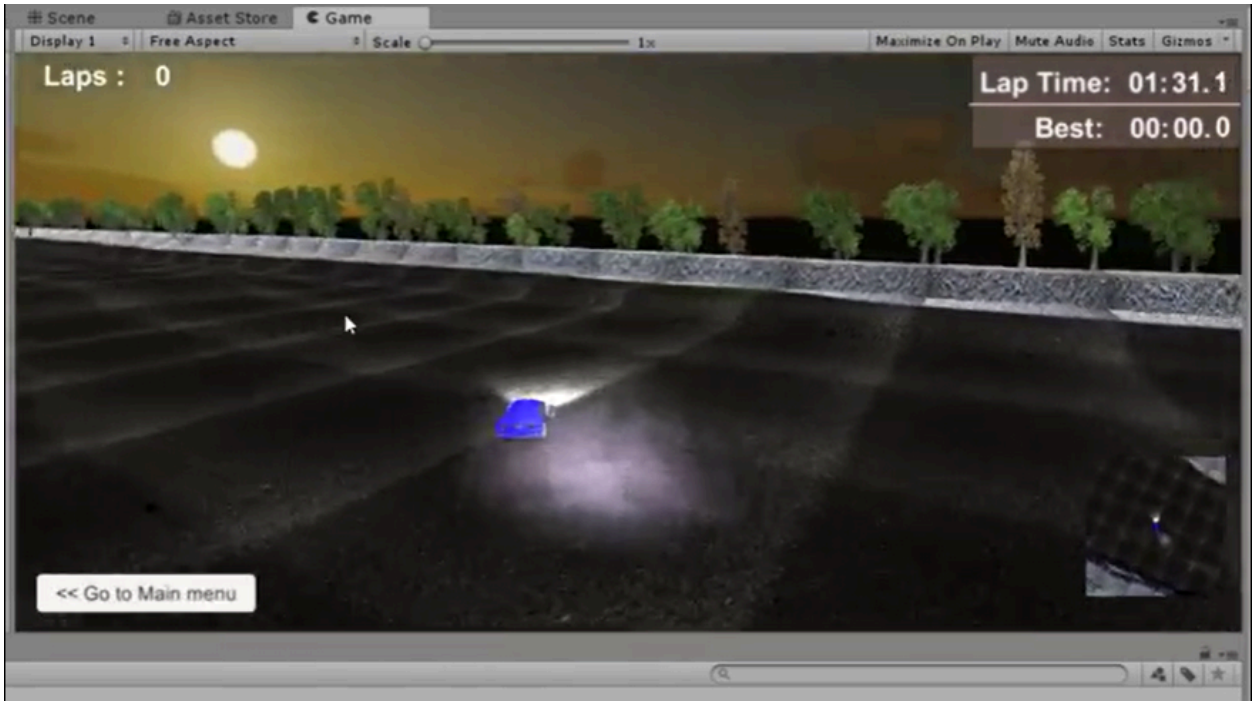
- **CAR AND TRACK SELECTION:**

- **GAME START**

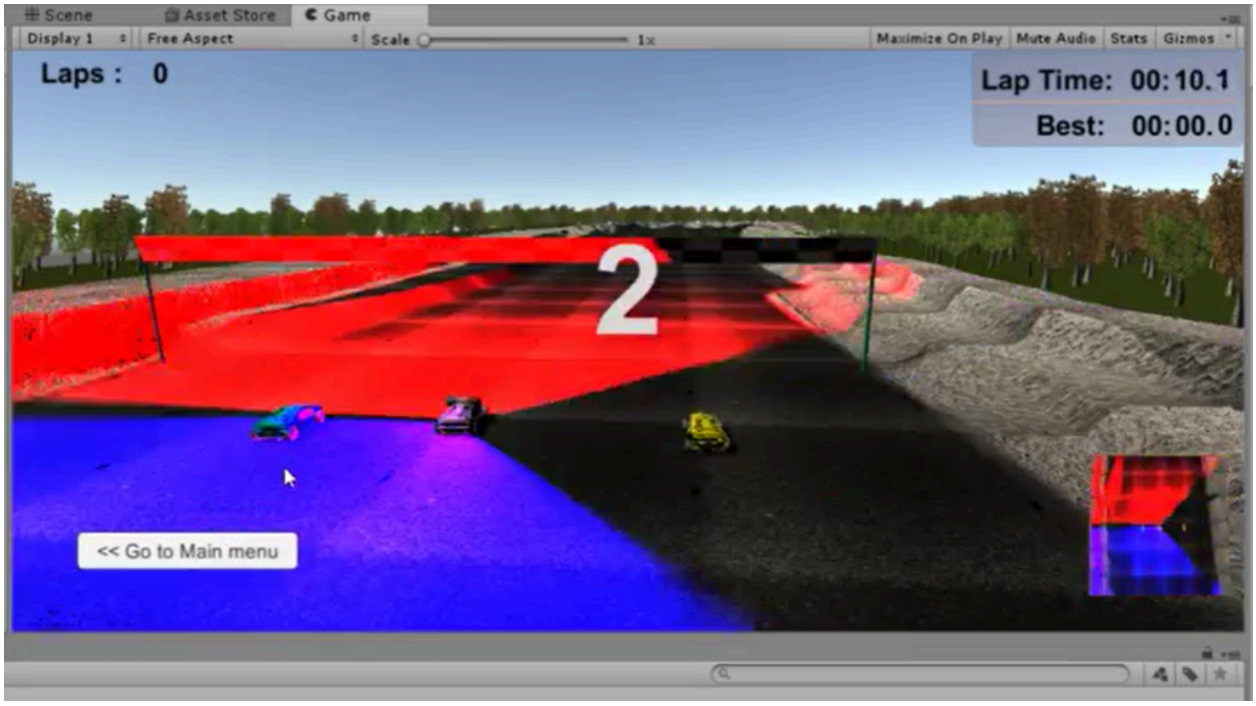


 TRACK 2

 AUDIO/VISUAL EFFECTS (DRIFT SMOKE)



- **CAR 2**


Scene
Asset Store
Game
Display 1
Free Aspect
Scale
1x
Maximize On Play
Mute Audio
Stats
Gizmos
Laps : 0
Lap Time: 00:10.1
Best: 00:00.0
2
<< Go to Main menu

7.1 CONCLUSION

Research and development are continuous processes; this is same in computer and game development. However, the effectiveness and efficiency of this game for further improvement. As early mentioned some of the scope of this project were actualized. The objectives could be improved upon, the car racing game developed will offer greater opportunity in game development.

7.2 Limitations:

1) The space can be limited if not properly handled.

2) The project will only work within a particular platform.

3) Shortcomings in software design.

4) UI can be confusing

7) No retry option

7.3 Future Scope:

When implement all those basic functions stated above, we can focus on improving the other aspects of this Car Racing Game. For example, I would like to add music and some audio effect to make this game feel more real. Also I can improve the playability by implementing some extra score calculation or bonus, for instance, a bonus is given when the player drives the car to a certain distance, and then the bonus icon will be displayed on top right of the screen. Moreover, I can make a score ranking to attract players to compete with each other.

BIBLIOGRAPHY:

Google scholar.com

www.youtube.com/user/unity3D

https://learn.unity.com

https://unity3D.com

www.Stackoverflow.com

www.wikipedia.org