RACE AND LEARN: AN ANDROID RACING GAME WITH LEARNING

BY

Shomrat Sahabuddin
ID: 151-15-4945

T.M. Ashikur Rahman
ID: 151-15-4971

MD. Rakib Mahmud
ID: 151-15-4832

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science and Engineering

Supervised By

Nazmun Nessa Moon
Assistant Professor
Department of CSE
Daffodil International University

Co-Supervised by

Dr. Fernaz Narin Nur
Assistant Professor
Department of CSE
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

DECEMBAR 2018

APPROVAL

This Project/internship titled "Race and Learn", submitted by ShomratSahabuddin, T.M. Ashikur Rahman and MD. Rakib Mahmud, ID No respectively: 151-15-4945, 151-15-4971, 151-15-4832 to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 9th December 2018.

DECLARATION

We hereby declare that, this project has been done by us under the supervision of **Ms. NazmunNessa Moon, Assistant professor, Department of CSE** Daffodil International University. Wealso declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:

[illegible]

Ms. Nazmun Nessa Moon

Assistant professor

Department of CSE

Daffodil International University

Submitted by:

[illegible]

Shomrat Sahabuddin

ID: 151-15-4945

Department of CSE

Daffodil International University

[illegible]

T.M Ashik

ID: 151-15-4971

Department of CSE

Daffodil International University

[illegible]

Rakib Mahmud Khan

ID: 151-15-4832

Department of CSE

Daffodil International University

ACKNOWLEDGEMENT

In the beginning, we would like to express our heartiest gratitude to the Almighty for enabling us to prepare the final year project successfully.

Now we want to mention our Supervisor **Ms. Nazmun Nessa Moon**, **Assistant professor**, Department of CSE Daffodil International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of “Android” to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, errors and correcting them at all stages have made it possible to complete this project.

Afterwards, we would like to express our heartiest gratitude to **Dr. Syed Akhter Hossain, Professor & Head, Department of CSE**, for his kind help to finish our project and also to other faculty member and the staff of CSE department of Daffodil International University.

Finally, we must acknowledge with due respect the constant support and patients of our parents.

ABSTRACT

The project titled "**Race and Learn: An Android Racing Game with Learning**" is a different type of an android racing game normally we used to play. Many of us are loved playing android game, racing game lovers are also there. Most of the racing games are represented urban areas graphics and environment. But we tried to make a change and came out from this concept. Think differently that, how was that if it could have been related to learning something? From this concept we built a racing game in different way. Changing in outside graphics of racing track and adding some historical places. Instead of urban areas the outside graphics of nature inside road represents our beautiful Bangladesh as well. Very much like a countryside road in our country. In the learning section, before start playing the game, there will be included some information's about the historical places. Gamer needs to read them and answers some basic questions about the place. Afterwards game will be played. If answers went wrong, game won't be played until answered correctly. So our project can help anyone to gather some knowledge about a historical place by an eye catching graphical representation of the place. Therefore, we think it becomes very interesting for every gamer.

TABLE OF CONTENTS

CONTENTS **PAGE**

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

In our daily life we are quite familiar with our smartphones. We are used to play various types of games. The reason is only recreation. Through playing a game if one can learn something that would be a better option rather searching only recreation. In this project we are tried to build a racing game where both recreation and learning meets together. Learning is about that particular historical place. Most of the racing games are foreign graphics and urban area related environments. Here we included some historical places and countryside road map that will give a different feel to the gamer.

1.2 Motivation

Those we played car racing game may be noticed that car, game environment, and other graphics are mostly urban related. In lieu of urban areas we are introducing here natural beauties and places inside racing track that represents our beautiful Bangladesh. Moreover only playing game is also wasted of time. It motivates us to think differently. That’s why we made a change here too. Adding a learning section. Gamer won’t be able to reach next level unless giving correct answers.

1.3 Objectives

- To learn through entertainment.
- To give an overview about a historical place as a short note.
- To represent a historical place graphically.
- To prevent wasting time by adding learning section for gamers.
- To encourage people for visiting that place.

1.4 Expected Outcome

The purpose of playing game is all about entertainment. But our racing game with learning is more than entertainment. We hope one can get entertained and learned as well. Because learning through entertainment is more interesting than general way of learning.

1.5 Report Layout

Chapter 1: Introduction

In this chapter we have discussed about the introduction, motivation, objectives and expected outcome of the project. Later followed by the report layout.

Chapter 2: Background

We discuss about the background circumstances of our project. We also talk about the related works, comparison to other candidate systems, the scope of the problem and challenges of the project.

Chapter 3: Requirement Specification

This chapter is all about the requirements like business process modeling, the requirement collection and analysis, the use case model of the project and their description, the logical data model and the design requirements.

Chapter 4: Design Specification

In this chapter all the designs of the project. Front-end design, back-end design, interaction design and UX and the implementation requirements

Chapter 5: Implementation and Testing

This chapter contains the implementation of the game, front-end designs, interactions, test implementation and the test results of the project.

Chapter 6: Conclusion and Future Scope

We discussed about the conclusion and the scope for further developments which pretty much derive about the project

CHAPTER 02

BACKGROUND

2.1 Introduction

We designed an android racing game which is a bit different from other. Before this, about a racing game we all know that only playing game is the main vision. Only entertainment is the output from those types of games. In this project one can be entertained through playing game with some learning as well. As a 3d racing game, this game is developed in unity 3d. Racing track is made on Blender a 3d software. Game environment inside road is also creating on unity. Photoshop is being used for texture resizing, creating image (speedo meter), and on the purpose of basic editing of some downloaded images. The core languages are C sharp and JavaScript.

2.2 Related Works

Since it's a racing game, naturally racing part is common with lots of games. But they are different in road side environment, as we introduced there some historical places. And compare with learning, there is no racing games actually matches where you can find both of them meets together. Only when you considered racing part except learning section, then some countryside environment graphics racing games are there, but there are some significant difference in car, road and outside graphics. Note that following related works are only considered game environment.

1 Forza Horizon 3 [1]
2 Rush Rally 2 [2]
3 Grid Autosport [3]

Forza Horizon 3

The following figure 2.1 is about a racing game environment titled Forza Horizon 3. The similarity with our project is outside graphics are countryside. It shows a speedy car with motion blur effect in game outside environment.

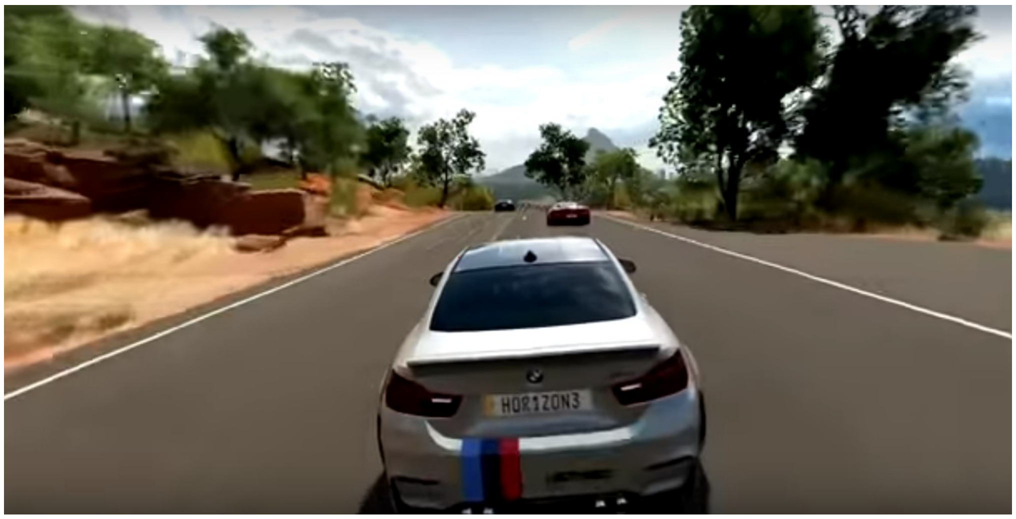


Figure 2.1: Forza Horizon 3

Rush Rally 2

This is another game that there is some sort of similarity with outside game environment. Figure 2.2 describes gaming outside graphics when it’s being played.

Figure 2.2: Rush Rally 2

Grid Autosport

This is another game where the outside environment is bit similarity. Countryside area is main theme here. Figure 2.3 shows the environment when it is played.

Figure 2.3: Grid Autosport

2.3 Comparative Studies

Above mentioned racing games has some sort of gaming environment similarity and some significant difference also existed. The theme of racing part will similar to all racing games, whereas environment and outside graphics varied. So, the differences between “Race and Learn” with competitors are provided below:

Table 2.1: Comparison between Race and Learn and competitors

No	Race and Learn	Competitors
1	Learning part is available	No learning part
2	Outside graphics contains historical places	Only countryside graphics
3	Less storage needed	More storage needed
4	Not a paid version	Free versions are not available

2.4 Scope of the Problem

We are providing this for a free version. Anyone can access this game. They didn’t need any signup or registration. Android has become one of the mostly used operating system for a smart phone. So, there are many developers who are working for making android racing games. After finishing the project most of the cases it seems that it doesn’t fulfill all the requirements of the needs. There will be always a limitation in the technology world.

Most of the current games user interfaces are not user-friendly or have some limitations. That’s why our primary purpose was made an easy user interface. While entering into the game, some information about places will be displayed. Then user needs answering some basic questions. After that game will be resumed.

2.5 Challenges

Making a car racing game despite of not having any previous knowledge about how the process will go. Because game development is not a small sector and as a fresher was never easy to start. We are just confident enough to take this as a challenge that anyhow we are determined to do it. So, there was challenges in every step of my work. Some of the challenges are like,

- Get used to with new user interface of Blender [4] for making racing track. Since we are beginner, after trying 4th time in a row we are able to make the racing track.
- Setup game environment with different textures and Unity tools.
- Then for some basic editing of downloaded textures and creating images with Photoshop [5] we need to learn Photoshop basic as well.
- The main editor among them, Unity 3d [6]. We are previously familiar with Android Studio but compare to Unity there is some significant difference between these two. So, finally we need to deal with Unity user interface.
- We have no previous working experience with C# [7], JavaScript [8] and different Unity build in functions for racing game [9] that really helps reduce coding and must needed for different aspect. Searching them and learning was not easy.
- Working with different coordinates, their values, rotation angle was very confusing.
- Wheel movement, camera movement with car, suspension and motion blur was challenging. Following figure 2.4 shows that a terrible output where wheels and body of the car moves separately in both thee seen and game view.

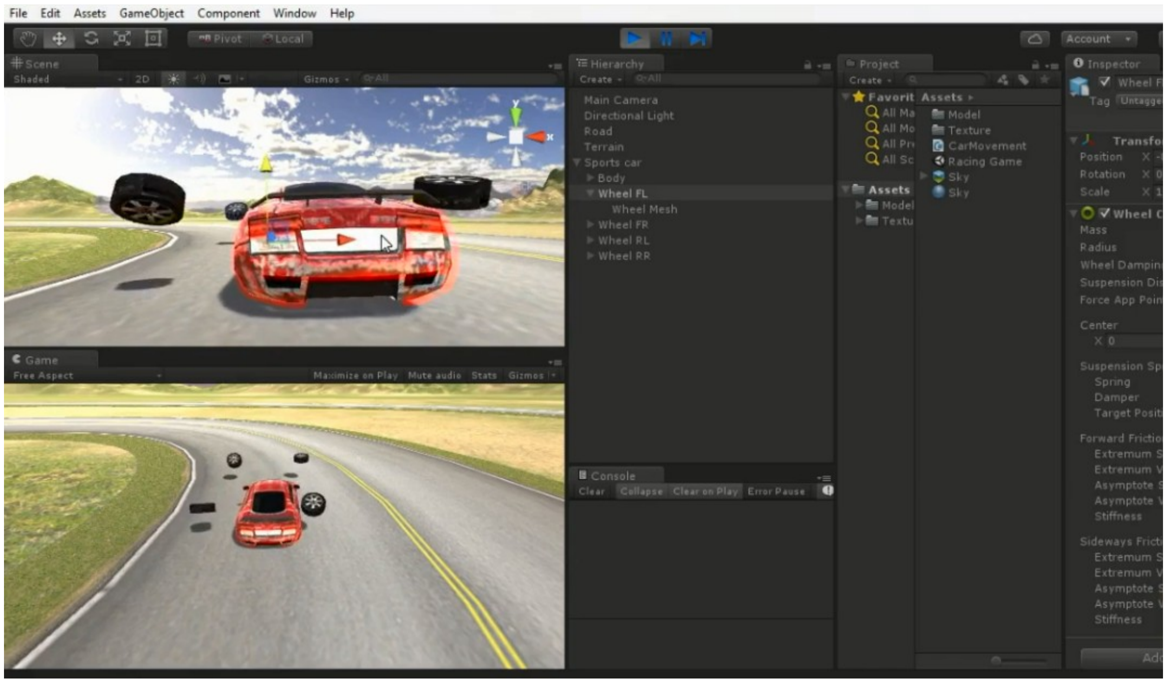


Figure 2.4: Wheel and car movement challenge

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 Business Process Modelling

BPM abbreviated as Business Process Modeling is a process of constructing a structural view of a system or process. It includes some symbols, conditions as like a flow chart. In this process we can see that under game will be started under which conditions to be fulfilled. After clicking a new game user will get some information about then question answer part will come. If everything goes right game will start.

BPM model show in figure 3.1.

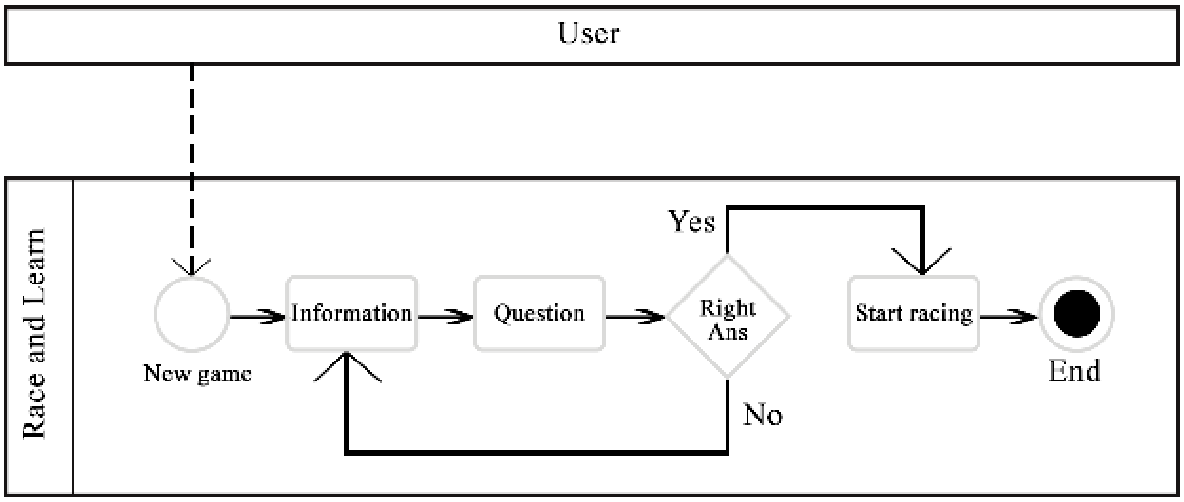


Figure 3.1: BPM of Tutor Map

3.2 Requirements Collection and Analysis

Before starting any project, requirements collection and analysis is a primary condition. An android racing game requires a lot of resources to fulfil the desires of the game. Coding languages we are used C# and JavaScript. Editors are Unity 3d, Blender and Photoshop. One more thing needs for playing the game i.e. any Android phone. There are two types of requirements, one is functional and other is non-functional [10].

Functional requirements are those activities that the system can perform or the functionalities. From the point of view of our system at the beginning of the game it will show pop up box where some information be displayed. Then user have to answer some questions, based on correct answers the game will start.

Non-functional requirement's defines efficiency, security, performance issue of a system. User interface of our game is very user friendly and gorgeous for excellence user experience as well.

3.3 Use Case Modelling and Description

A use case model is graphically described the interactions among the elements of a system in different stages. It is a methodology used in system analysis to identify, clarify and organizing system requirements. Also find out conditions for achieving goal. And conditions for not to reach goal. For our system the use case model is given below in figure 3.2.

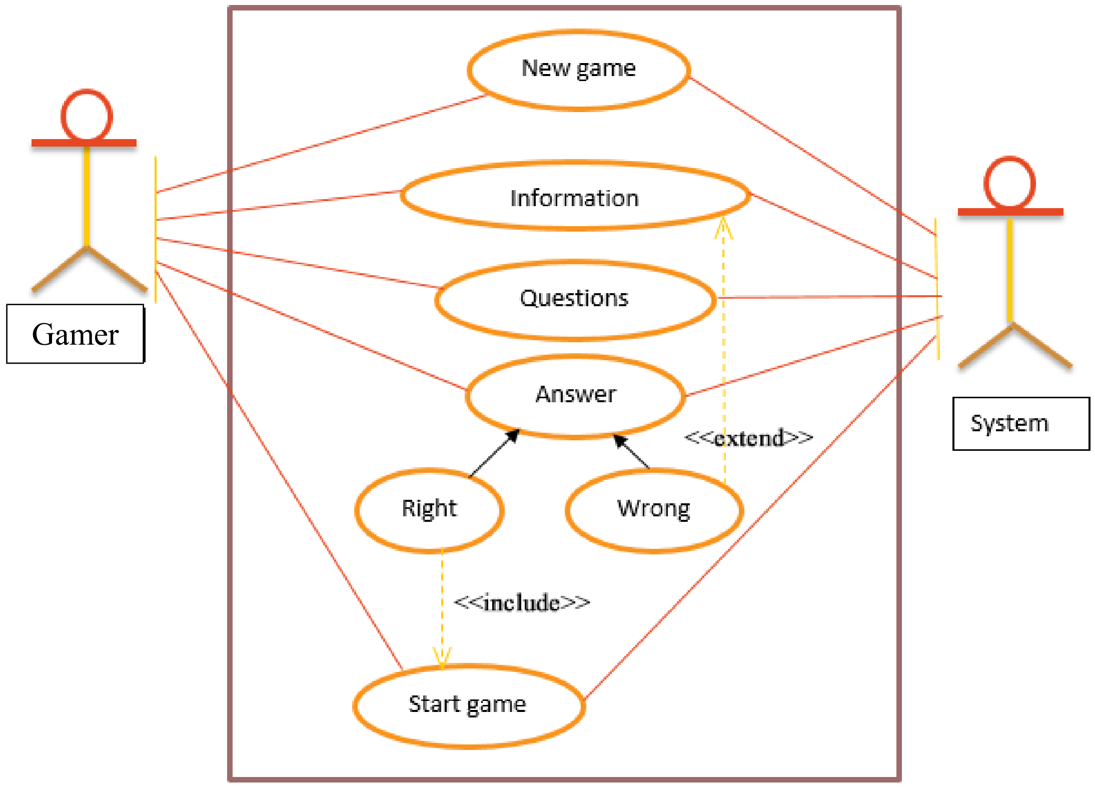


Figure 3.2: Use case diagram

Use Case Description for Race and Learn

Use Case: Race and Learn

Actor: Gamer, system.

Pre-Condition: Reading information that displayed.

Purpose: Start playing game.

Description:

1. At first user needs to enter the game and click on new game option.
2. Then some information will be displayed about a historical place. Gamer should this.
3. Then some questions will be asked based about information previously displayed.
4. If gamer will be able to answer questions correctly, then game will start.
5. If answer went wrong, information will be displayed again.

Post condition:

- If user answers correctly.

3.4 Logical Data Model

The term Logical Data Modelling is a process used to define and analyze requirements needed to support the business processes within the scope of corresponding information systems in organizations. The Entity-Relationship model or Entity-Relationship diagram (ERD) is a logical data models, it includes the entity, attributes and relationships. Level can be varied so it is a multivalued attribute in the diagram.

Logical data model is giving below in figure 3.3.

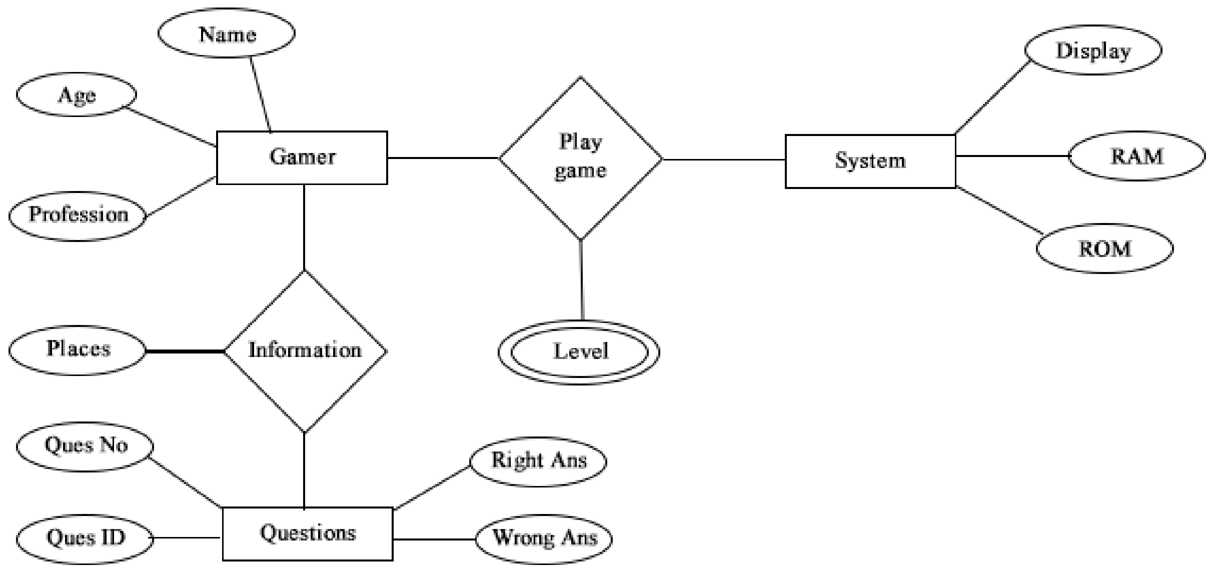


Figure 3.3: ER Diagram

3.5 Design Requirements

The following goals were kept in mind while designing the system:

Make System Simple and Flexible for Users

This is our first goal. The system users are able to have a great amount of control over their purpose in achieving objectives.

Make the System Compatible

It should be fit in the total system, future maintenance and enhancement must less.

Efficiency

It is the most important. The system should run and work minimum hardware of software resource available without delaying. Trying to make it light for a faster response for all devices.

CHAPTER 4

DESIGN SPECIFICATION

4.1 Front-end Design

The meaning of front-end design is the visual part of a system. By which the user can interact with the system. While developing the game we tried to make it user-friendly. Our intension was kept it as simple and as possible. The point of view from our project front-end refers that game environment design, racing track making, racing car design and all other visual graphics. That means we considered non coding part as our front-end design.

4.1.1 Racing Track in Blender

We start our journey towards making a racing game from here. Before going to make it, we first designed it in Paint, that how the racing track would be designed on Blender. Figure 4.1 shows our draft demonstration. Because directly making the final output is not easy as we are in beginner level.

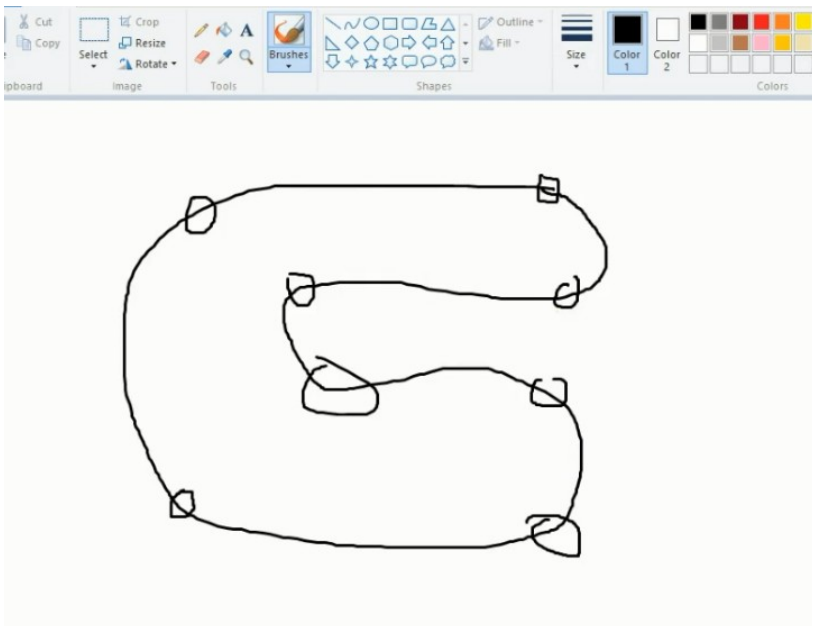


Figure 4.1 Draft design of racing track

After that we was going to Blender, with the help of Bezier curve edges are being draw. After performing some basic operation on Bezier curve like scaling, rotating, extrude curve, changing ground position we are able to draw it like our draft demonstration. Figure 4.2 shows the software demonstration of racing track.

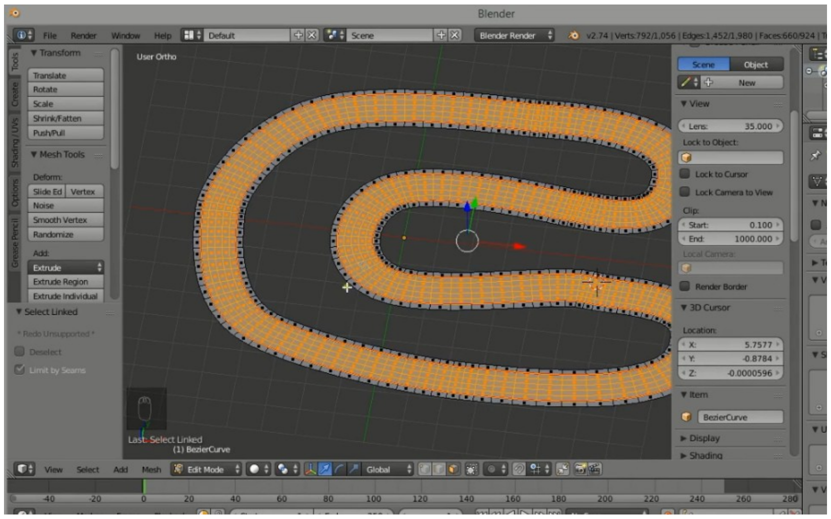


Figure 4.2: Racing track on Blender

We are almost done our racing track, afterwards for giving this track a genuine racing track view, some downloaded texture [11] needs to be placed on the road. And also outer part of the road. Some editing was done in Photoshop before using on Blender. From UV editing [12] and selecting face option of Blender, we imported two texture for road one for road and other for road side. Again scaling, ground position are performed for fixing up images on tracing track. Initial extension of a blender file is .blend, we saved our file also in that format. After that we exported it as .fbx format because we need to work with this one on Unity. Finally we are able to obtain the following output that shows in figure 4.3.

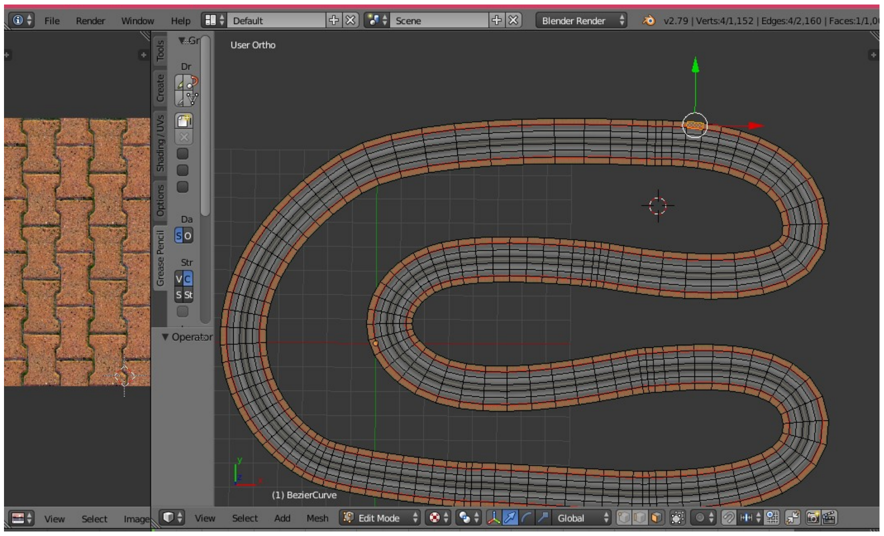


Figure 4.3: Final output of racing track

4.1.2 Making Game Environment

Since we made our environment based on road and inside road, first we imported the .fbx file on unity. Then we fixed our road position with as needed. With the help of different types of brushes on unity we designed the environment in whole terrain. Figure 4.4 shows the initial environment where the upper one is seen view and the lower one is game view. After that we updated this by adding some textures of trees and other environment related materials.

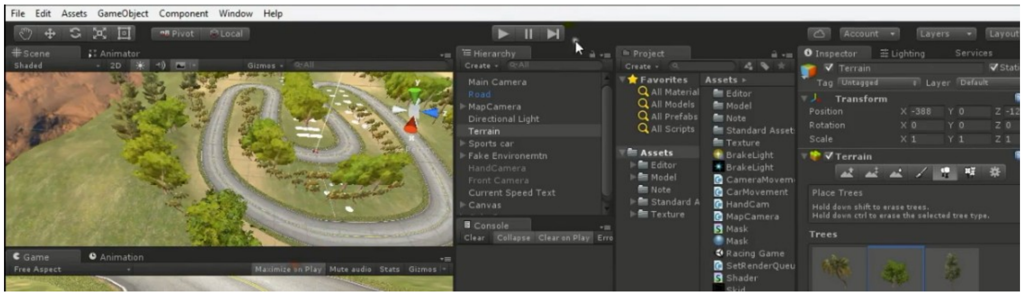


Figure 4.4: Initial game environment.

4.1.3 Setup the Car

First we thought that simply making a car on Photoshop then import it on unity and that would be the way for set up car. But it's a 3d project and we needed a 3d car as well. Not only needed a 3d car, with the car we needed unity assets for the car in order to work on it. So a car has been downloaded [13] for this project with unity assets and other materials. Then we changed its body shape and color in such a way that matches with our project on Photoshop. Then some textures of the car like car body, car parts and wheel martial were assigned. Afterwards the car has been imported in on unity and figure 4.5 shows that. With that operation our front-end part was done.

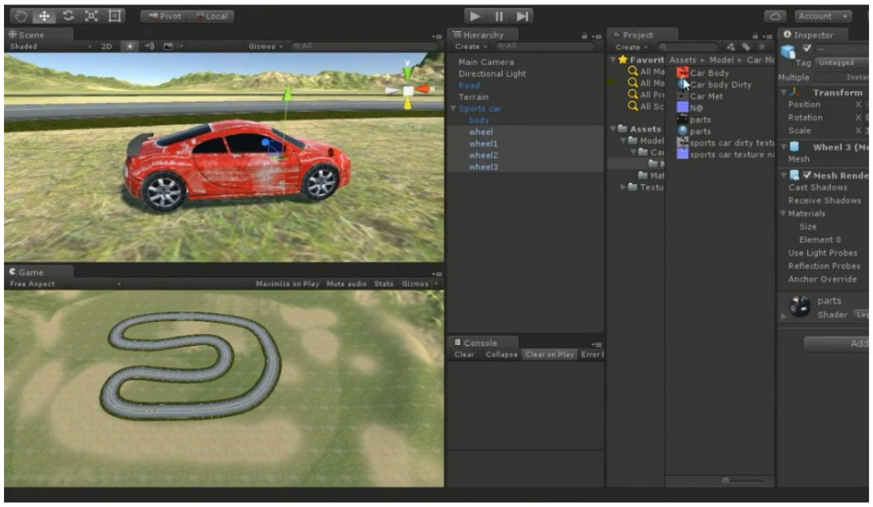


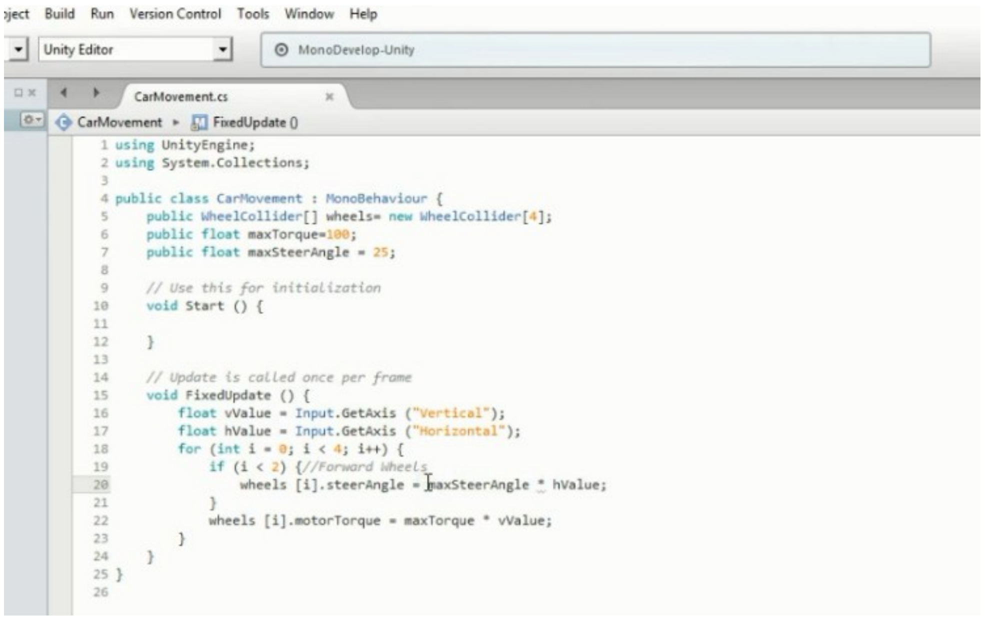
Figure 4.5: Importing Car on Unity

4.2 Back-end Design

In this part we will discuss about coding. Most of the cases we used C# and JavaScript. Back-end is such a part where all the logics are worked behind the system. The user cannot interact with this part or anything. The back-end design is the part where the actual work happens. This part is the most crucial part for a developing a system.

4.2.1 Wheel and Car movement

First we made four public variables for four wheels as we are assigned them from outside in unity MonoDevelop, they are wheelFL, wheelFR, wheelRL, wheelRR. Here wheelFL is the wheel of forward left side of the car. Similarly wheelFR is forward right, wheelRL is rear left and wheelRR indicates rear right side wheel. For moving wheels we picked up input axis from project setting panel. When it start moving to ensure that it follows real physics with surface we used motorTorque [14] function in all four wheels. Then it moved slowly, for faster movement we assigned maxTorque value as 10. Figure 4.6 shows the back-end code of wheel and car movement.



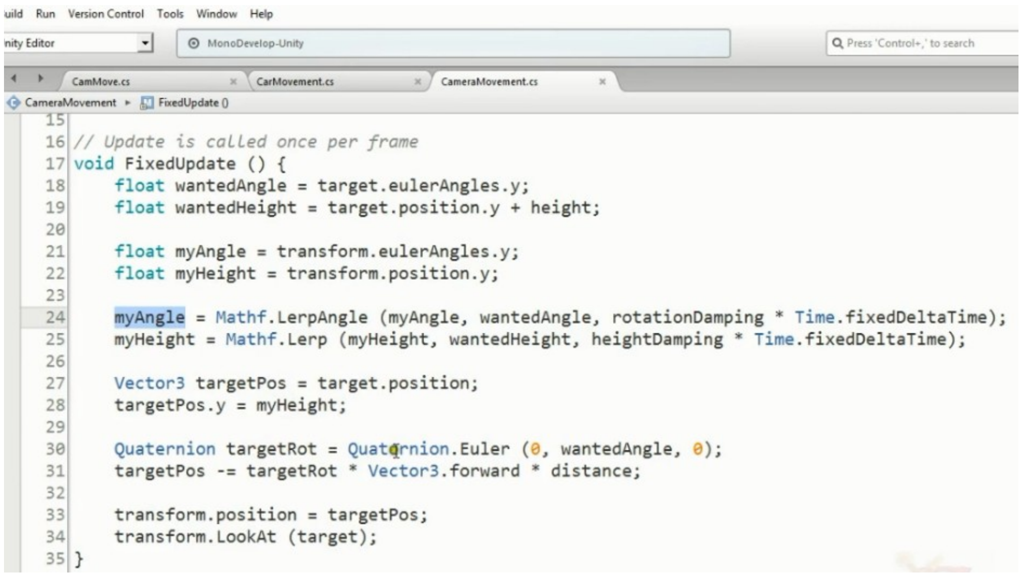
ject Build Run Version Control Tools Window Help
Unity Editor
MonoDevelop-Unity
CarMovement.cs
CarMovement ▸ FixedUpdate ()

```
using UnityEngine;
using System.Collections;

public class CarMovement : MonoBehaviour {
    public WheelCollider[] wheels= new WheelCollider[4];
    public float maxTorque=100;
    public float maxSteerAngle = 25;

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void FixedUpdate () {
        float vValue = Input.GetAxis ("Vertical");
        float hValue = Input.GetAxis ("Horizontal");
        for (int i = 0; i < 4; i++) {
            if (i < 2) {//Forward Wheels
                wheels [i].steerAngle = maxSteerAngle * hValue;
            }
            wheels [i].motorTorque = maxTorque * vValue;
        }
    }
}
```

Figure 4.6: Wheel Movement with Car

4.2.2 Camera Movement

Initially when car start moving, camera didn't move with car because we didn't write any code for this. It remains still and car is moving as it coded. For working we need to define camera its focusing area, with car position changing how it will moves etc. for its focusing area we used lookAt function. Other thing camera's position, distance from car and height form surface. After fixed this, we realized that camera movement was working but it wasn't move smoothly. For a smooth movement we use Mathf.LerpAngle function [15]. In addition, whenever car moves faster camera will zoom out so that it looks like a bit faster to the user, and when speed decreases camera will zoom in towards car. While pressing beak zoom in operation works faster as speed decreases very fast. Figure 4.7 shows the necessary codes for above description.

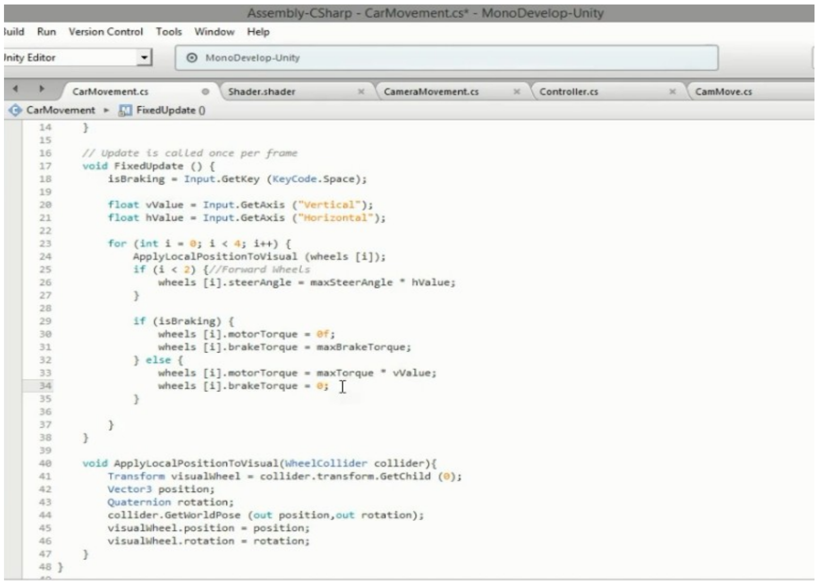

uild Run Version Control Tools Window Help
nity Editor
MonoDevelop-Unity
Press 'Control+,' to search
CamMove.cs
CarMovement.cs
CameraMovement.cs
CameraMovement ▸ FixedUpdate ()

```
// Update is called once per frame
void FixedUpdate () {
    float wantedAngle = target.eulerAngles.y;
    float wantedHeight = target.position.y + height;

    float myAngle = transform.eulerAngles.y;
    float myHeight = transform.position.y;

    myAngle = Mathf.LerpAngle (myAngle, wantedAngle, rotationDamping * Time.fixedDeltaTime);
    myHeight = Mathf.Lerp (myHeight, wantedHeight, heightDamping * Time.fixedDeltaTime);

    Vector3 targetPos = target.position;
    targetPos.y = myHeight;

    Quaternion targetRot = Quaternion.Euler (0, wantedAngle, 0);
    targetPos -= targetRot * Vector3.forward * distance;

    transform.position = targetPos;
    transform.LookAt (target);
}
```

Figure 4.7: Camera Movement

4.2.3 Apply Break

In previous session we applied motorTorque for moving wheels. Now for applying break we used another function named breakTorque [16] which will gradually decrease the speed of the car based on our given value. This operation will be performed when user pressed a key, for controlling from pc we used space bar. In order to do that we made a Boolean function variable named isBraking. Then assigned it with space bar by using isBraking = input.Getkey(KeyCode.Space); command. Figure 4.8 shows back-end code for applying break.

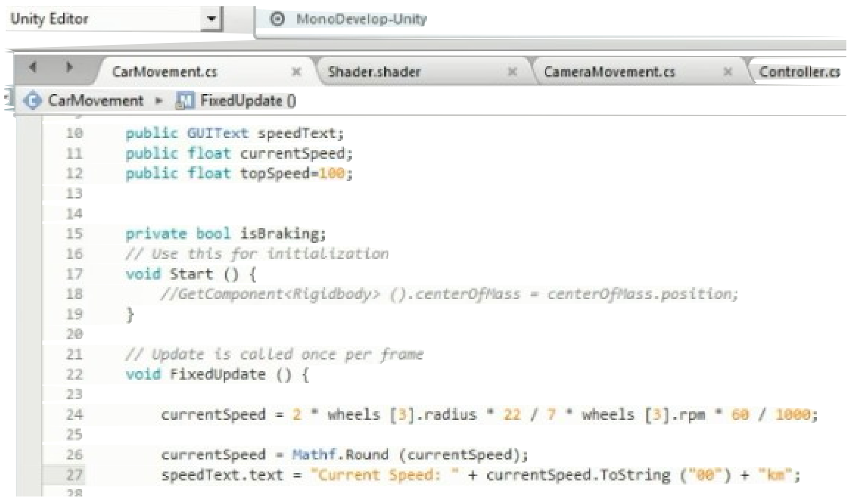

Assembly-CSharp - CarMovement.cs* - MonoDevelop-Unity
Build Run Version Control Tools Window Help
Unity Editor
MonoDevelop-Unity
CarMovement.cs
Shader.shader
CameraMovement.cs
Controller.cs
CamMove.cs
CarMovement ▸ FixedUpdate ()

```
    }

    // Update is called once per frame
    void FixedUpdate () {
        isBraking = Input.GetKey (KeyCode.Space);

        float vValue = Input.GetAxis ("Vertical");
        float hValue = Input.GetAxis ("Horizontal");

        for (int i = 0; i < 4; i++) {
            ApplyLocalPositionToVisual (wheels [i]);
            if (i < 2) {//Forward Wheels
                wheels [i].steerAngle = maxSteerAngle * hValue;
            }

            if (isBraking) {
                wheels [i].motorTorque = 0f;
                wheels [i].brakeTorque = maxBrakeTorque;
            } else {
                wheels [i].motorTorque = maxTorque * vValue;
                wheels [i].brakeTorque = 0;
            }

        }
    }

    void ApplyLocalPositionToVisual(WheelCollider collider){
        Transform visualWheel = collider.transform.GetChild (0);
        Vector3 position;
        Quaternion rotation;
        collider.GetWorldPose (out position,out rotation);
        visualWheel.position = position;
        visualWheel.rotation = rotation;
    }
}
```

Figure 4.8: Back-end code for applying break

4.2.4 Limit Top Speed

In order to limit top speed the primary condition is to know the current speed. Then set a value for the top speed. When the current speed is crossed the limit of top speed then break will be applied automatically, that prevents speed up. And if the speed is less than top speed the current speed will increase towards the top speed in case user continuously increasing up current speed. In addition the current speed of the car has been also displayed in the game view. Figure 4.9 has shown back-end code for those operation.



```
    public GUIText speedText;
    public float currentSpeed;
    public float topSpeed=100;


    private bool isBraking;
    // Use this for initialization
    void Start () {
        //GetComponent<Rigidbody> ().centerOfMass = centerOfMass.position;
    }

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

        currentSpeed = 2 * wheels [3].radius * 22 / 7 * wheels [3].rpm * 60 / 1000;

        currentSpeed = Mathf.Round (currentSpeed);
        speedText.text = "Current Speed: " + currentSpeed.ToString ("00") + "km";
```

Figure 4.9: Limit Top Speed and Display Current Speed

4.2.5 Speedometer in Photoshop

The purpose of using speedometer is to display the current speed. We have to create this texture on Photoshop with the help of ellipse, rounded rectangle and type tools. Stick of speedometer was made in another PSD. Figure 4.10 shows creating speedometer.

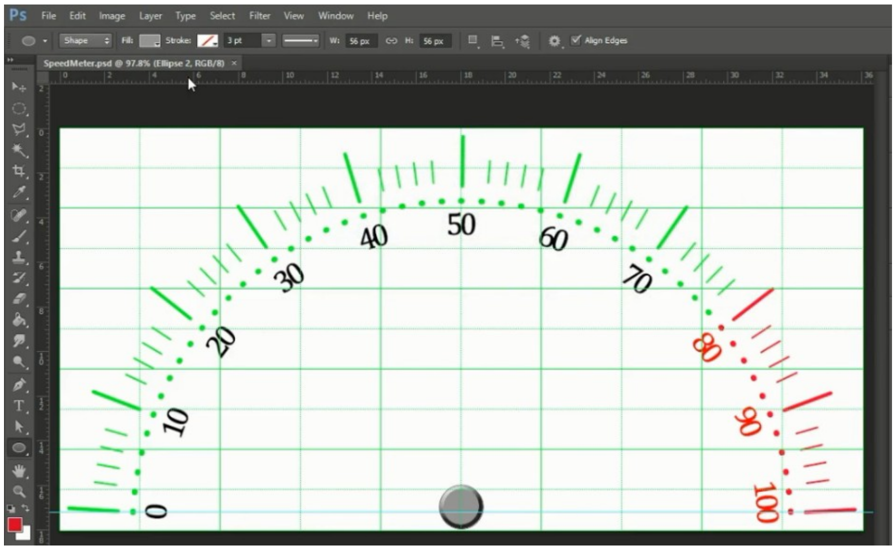
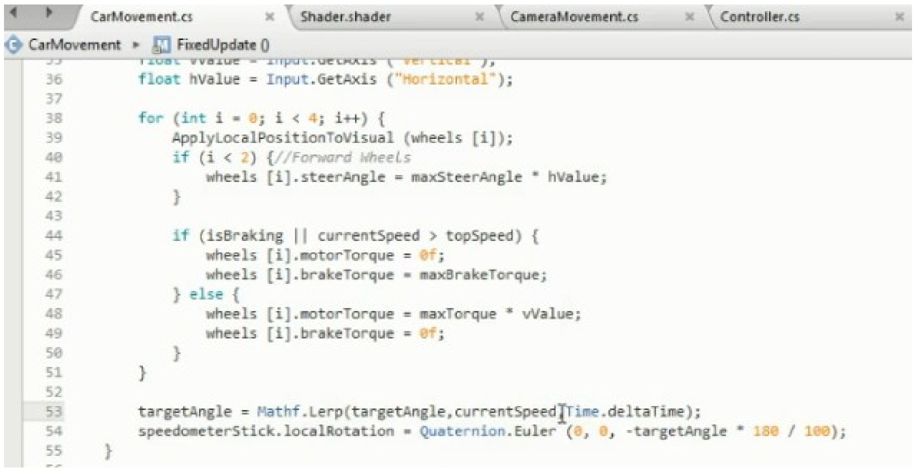


Figure 4.10: Creating Speedometer

4.2.6 Speedometer in Unity

After hiding grid from the document and making background transparent, we imported stick and speedometer in unity project panel. Figure 4.11 shows back-end code of working speedometer with changing speed.

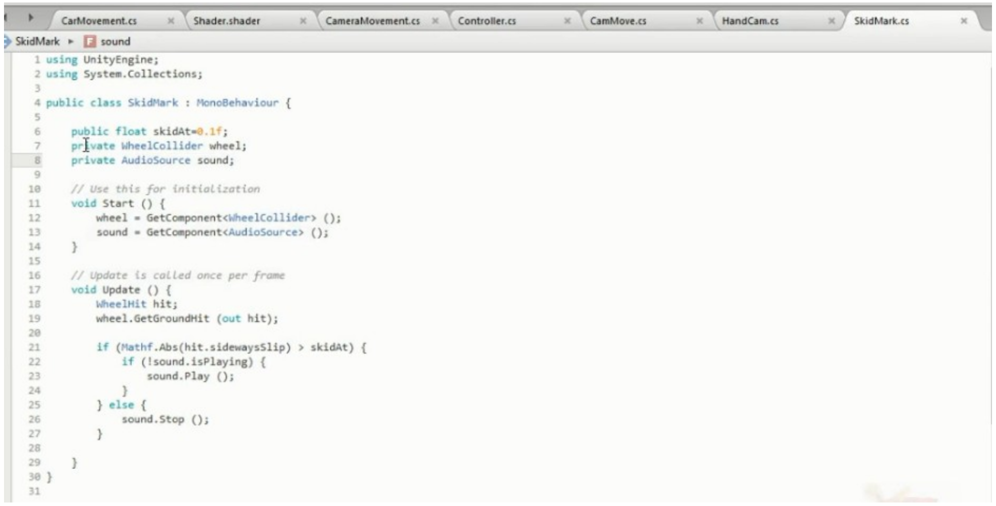


```
        float vValue = Input.GetAxis ("Vertical");
        float hValue = Input.GetAxis ("Horizontal");

        for (int i = 0; i < 4; i++) {
            ApplyLocalPositionToVisual (wheels [i]);
            if (i < 2) {//Forward Wheels
                wheels [i].steerAngle = maxSteerAngle * hValue;
            }

            if (isBraking || currentSpeed > topSpeed) {
                wheels [i].motorTorque = 0f;
                wheels [i].brakeTorque = maxBrakeTorque;
            } else {
                wheels [i].motorTorque = maxTorque * vValue;
                wheels [i].brakeTorque = 0f;
            }
        }

        targetAngle = Mathf.Lerp(targetAngle,currentSpeed*Time.deltaTime);
        speedometerStick.localRotation = Quaternion.Euler (0, 0, -targetAngle * 180 / 100);
    }
```

Figure 4.11: Speedometer Back-end code

4.2.7 Car Sound Setting

Some audios are already included in Unity for racing games. To explore those files we need to visit import package form project panel and there is a vehicle category where audios are stored. From the whole packages we just import car category where included accelerate, decelerate, skid sound etc. Figure 4.12 shows car sound playing back-end code.

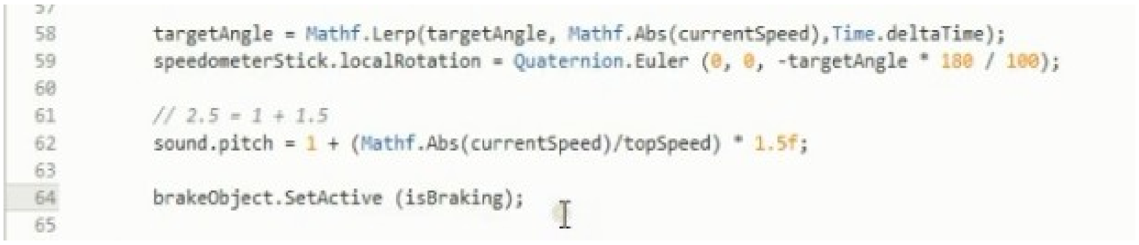


```
using UnityEngine;
using System.Collections;

public class SkidMark : MonoBehaviour {

    public float skidAt=0.1f;
    private WheelCollider wheel;
    private AudioSource sound;

    // Use this for initialization
    void Start () {
        wheel = GetComponent<WheelCollider> ();
        sound = GetComponent<AudioSource> ();
    }

    // Update is called once per frame
    void Update () {
        WheelHit hit;
        wheel.GetGroundHit (out hit);

        if (Mathf.Abs(hit.sidewaysSlip) > skidAt) {
            if (!sound.isPlaying) {
                sound.Play ();
            }
        } else {
            sound.Stop ();
        }

    }
}

```

Figure 4.12: Working of Car Sound

4.2.8 Break Light

In applying break section we are talking about break system. Now our purpose is at the time of pressing break red lights are on at the back side of car. For this we need a new game object named as breakObject that was declared and lens flare texture that has been downloaded. Figure 4.13 shows the operation.

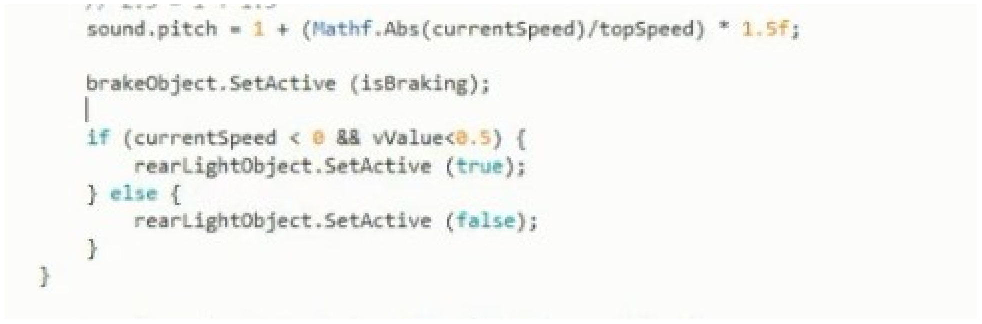


```
        targetAngle = Mathf.Lerp(targetAngle, Mathf.Abs(currentSpeed),Time.deltaTime);
        speedometerStick.localRotation = Quaternion.Euler (0, 0, -targetAngle * 180 / 100);

        // 2.5 = 1 + 1.5
        sound.pitch = 1 + (Mathf.Abs(currentSpeed)/topSpeed) * 1.5f;

        brakeObject.SetActive (isBraking);

```

Figure 4.13: Activate Red Light on Break

4.2.9 Rear Light

We set up rear lights and they will be turned on while backtracking is happened. Processes are almost same as break light, we need another game object named as rearLightOblect. When current speed is less than zero rear lights are turned on following figure 4.14 shows back-end code for rear light.



```
    sound.pitch = 1 + (Mathf.Abs(currentSpeed)/topSpeed) * 1.5f;

    brakeObject.SetActive (isBraking);

    if (currentSpeed < 0 && vValue<0.5) {
        rearLightObject.SetActive (true);
    } else {
        rearLightObject.SetActive (false);
    }
}
```

Figure 4.14: Rear Light on Backtracking

4.2.10 Skidding Mark

It will start working when wheels have a strong friction with surface. Specially turning with speed or changing its direction while running it marks wheel movement. For this we created a skidding mark by transparent background by brush tool on Photoshop. Figure 4.15 shows that.

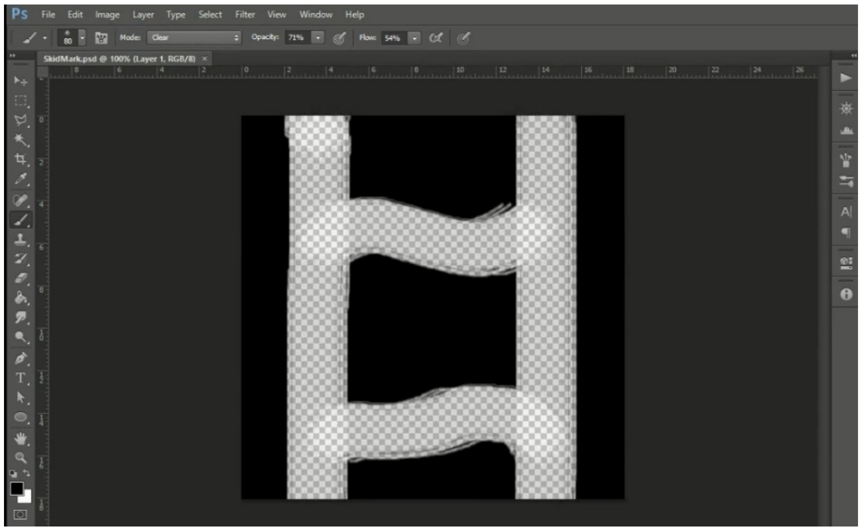
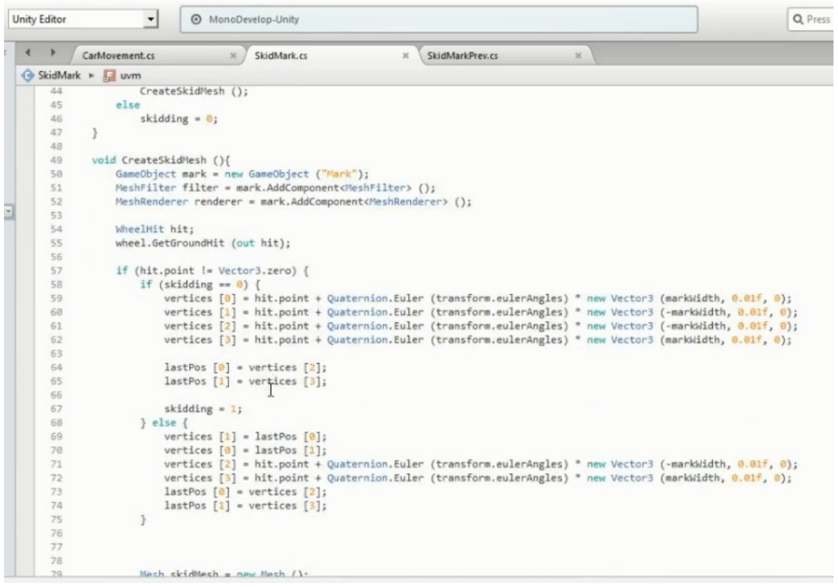


Figure 4.15: Skidding Mark on Photoshop

Then import it on Unity. Replacing color with a shade of black, repeating is on and other functionalities are set there. After five seconds skidding mark will disappear to make road clean again. Figure 4.16 shows back-end code for skidding mark.

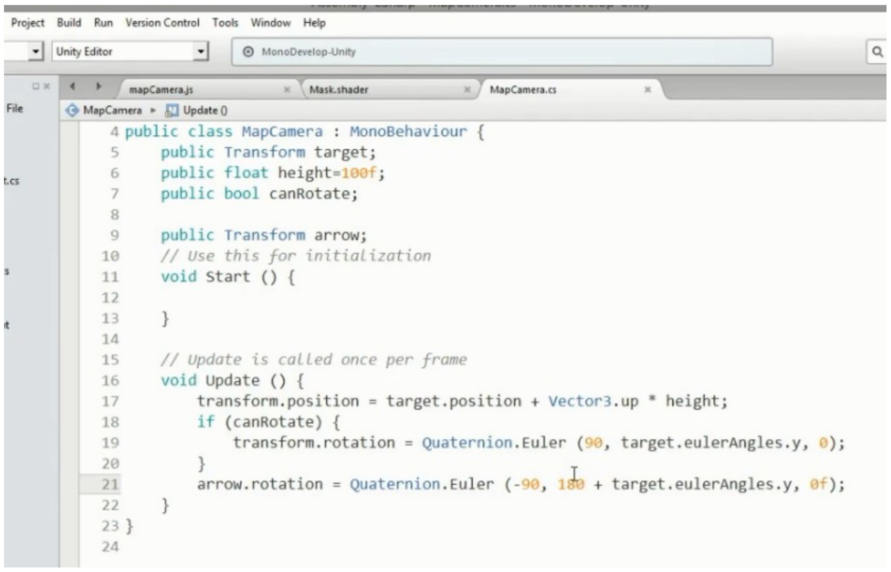


```
            CreateSkidMesh ();
        else
            skidding = 0;
    }

    void CreateSkidMesh (){
        GameObject mark = new GameObject ("Mark");
        MeshFilter filter = mark.AddComponent<MeshFilter> ();
        MeshRenderer renderer = mark.AddComponent<MeshRenderer> ();

        WheelHit hit;
        wheel.GetGroundHit (out hit);

        if (hit.point != Vector3.zero) {
            if (skidding == 0) {
                vertices [0] = hit.point + Quaternion.Euler (transform.eulerAngles) * new Vector3 (markWidth, 0.01f, 0);
                vertices [1] = hit.point + Quaternion.Euler (transform.eulerAngles) * new Vector3 (-markWidth, 0.01f, 0);
                vertices [2] = hit.point + Quaternion.Euler (transform.eulerAngles) * new Vector3 (-markWidth, 0.01f, 0);
                vertices [3] = hit.point + Quaternion.Euler (transform.eulerAngles) * new Vector3 (markWidth, 0.01f, 0);

                lastPos [0] = vertices [2];
                lastPos [1] = vertices [3];

                skidding = 1;
            } else {
                vertices [1] = lastPos [0];
                vertices [0] = lastPos [1];
                vertices [2] = hit.point + Quaternion.Euler (transform.eulerAngles) * new Vector3 (-markWidth, 0.01f, 0);
                vertices [3] = hit.point + Quaternion.Euler (transform.eulerAngles) * new Vector3 (markWidth, 0.01f, 0);
                lastPos [0] = vertices [2];
                lastPos [1] = vertices [3];
            }



            Mesh skidMesh = new Mesh ();
```

Figure 4.16: Skidding Mark

4.2.11 Road Preview Camera

What to next and where be your next destination? For answering those questions we added another camera where you can find those direction. Figure 4.17 shows back-end code.

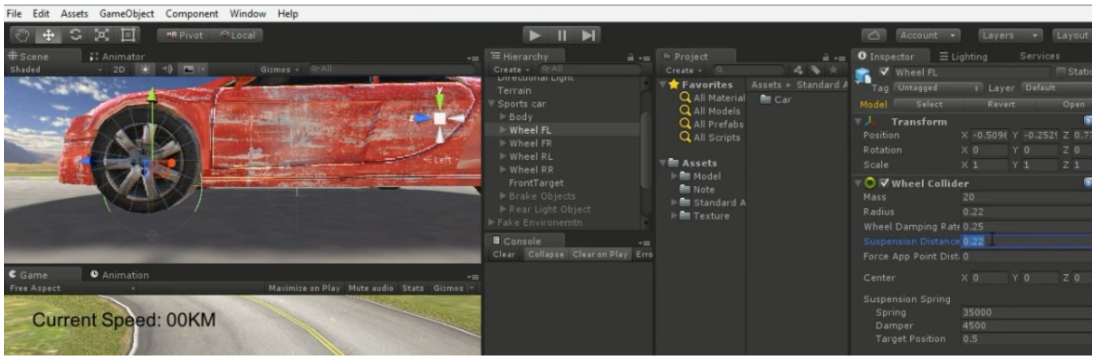


```
public class MapCamera : MonoBehaviour {
    public Transform target;
    public float height=100f;
    public bool canRotate;

    public Transform arrow;
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        transform.position = target.position + Vector3.up * height;
        if (canRotate) {
            transform.rotation = Quaternion.Euler (90, target.eulerAngles.y, 0);
        }
        arrow.rotation = Quaternion.Euler (-90, 180 + target.eulerAngles.y, 0f);
    }
}
```

Figure 4.17: Map Camera

4.2.12 Suspension and Motion Blur

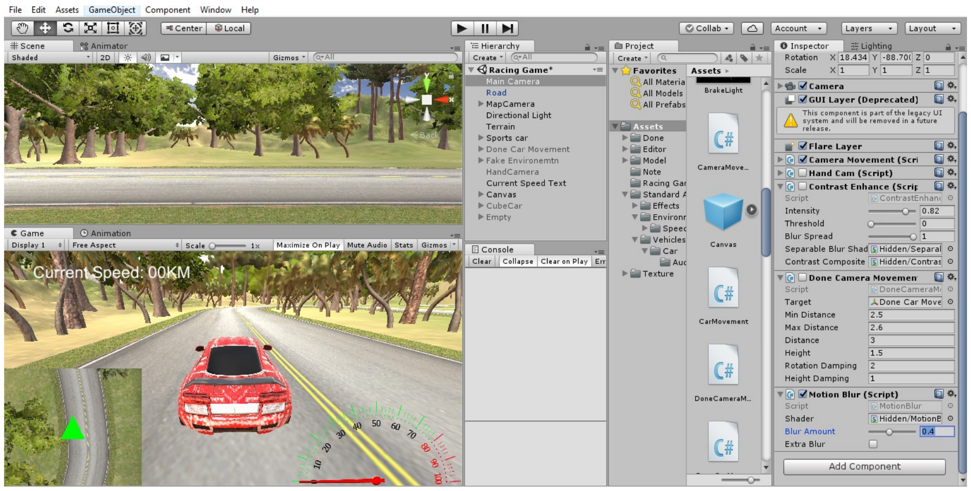
For making the car more realistic rather than a static object while driving we used suspension. So that it behaves like a real car followed by real physics and gravity with velocity changing. It will work when break is pressed or decelerating mode. To do this we didn’t need to write any script, it’s already there. We can control or change this as needed. Figure 4.18 shows the controlling suspension.



```
Wheel FL
Tag Untagged    Layer Default
Transform
Position   X -0.509( Y -0.252( Z 0.7
Rotation   X 0  Y 0  Z 0
Scale      X 1  Y 1  Z 1
Wheel Collider
Mass                    20
Radius                  0.22
Wheel Damping Rate      0.25
Suspension Distance     0.22
Force App Point Dist.   0
Center     X 0  Y 0  Z 0
Suspension Spring
  Spring                35000
  Damper                4500
  Target Position       0.5
```

Figure 4.18: Suspension Controlling

For motion blur effect we also didn’t need to write script. Unity has already this script. From project panel we just import them to use. This effect will be added on main camera, from there add component as motion blur from image effect category. This will give a fell to the user that it runs very fast. Figure 4.18 shows controlling motion blue effect.

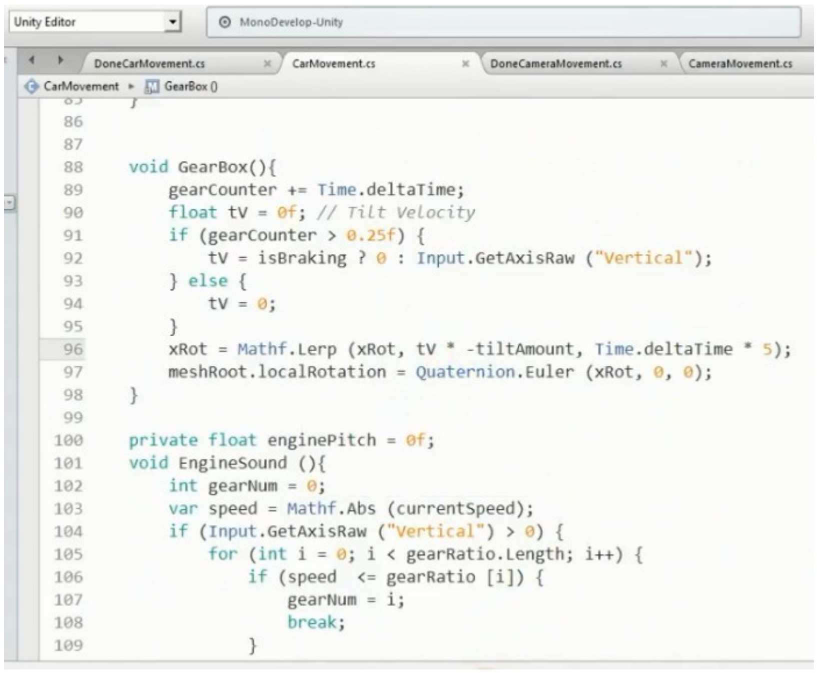


```
Rotation  X 18.434  Y -88.70(  Z 0
Scale     X 1  Y 1  Z 1
Camera
GUI Layer (Deprecated)
This component is part of the legacy UI system and will be removed in a future release.
Flare Layer
Camera Movement (Scri
Hand Cam (Script)
Contrast Enhance (Scrip
  Script                  ContrastEnhanc
  Intensity               0.82
  Threshold               0
  Blur Spread             1
  Separable Blur Shad     Hidden/Separal
  Contrast Composite      Hidden/Contras
Done Camera Movemen
  Script                  DoneCameraM
  Target                  Done Car Move
  Min Distance            2.5
  Max Distance            2.6
  Distance                3
  Height                  1.5
  Rotation Damping        2
  Height Damping          1
Motion Blur (Script)
  Script                  MotionBlur
  Shader                  Hidden/MotionB
  Blur Amount             0.4
  Extra Blur
Add Component
```

Figure 4.19: Motion Blur Setup

4.2.13 Gear Box

At the time of running when speeds are up or down changing sound according to the change of speed is important. This script will perform so, and shake car body with movement along with changing velocity. Hard break, shake on gear changing is also possible here. Following figure 4.19 shows backend code for above mentioned operation.



```
    }


    void GearBox(){
        gearCounter += Time.deltaTime;
        float tV = 0f; // Tilt Velocity
        if (gearCounter > 0.25f) {
            tV = isBraking ? 0 : Input.GetAxisRaw ("Vertical");
        } else {
            tV = 0;
        }
        xRot = Mathf.Lerp (xRot, tV * -tiltAmount, Time.deltaTime * 5);
        meshRoot.localRotation = Quaternion.Euler (xRot, 0, 0);
    }

    private float enginePitch = 0f;
    void EngineSound (){
        int gearNum = 0;
        var speed = Mathf.Abs (currentSpeed);
        if (Input.GetAxisRaw ("Vertical") > 0) {
            for (int i = 0; i < gearRatio.Length; i++) {
                if (speed  <= gearRatio [i]) {
                    gearNum = i;
                    break;
                }
```

Figure 4.20: Gear Box Setup

4.3 Interaction Design and UX

It actually represent the interaction of the application through using it. Interaction design is all about the overall procedure of the system while running by users. Better interaction design depends on the overall outcome of the system.

UX stands for user experience. This term is important for designing any system. Because it varies time after time. Old design or user interface makes monotonous their mind.

To get rid of form this, a change on user interface may be applied. In our game we are tried to maintain these factors that will give satisfaction to the users.

4.4 Implementation Requirements

To make this racing game we used Unity 3d, Blender and Photoshop. All these are not easy for someone at beginner level. For making our racing track we used Blender. Then convert blend file into fbx. Afterwards game environment and backend coding was implemented on Unity 3d. Photoshop was used for creating images, editing textures for environment and making speedometer image. We are implemented this project part by part with these software's. Browsing asset store for different racing assets while it was needed.

CHAPTER 5

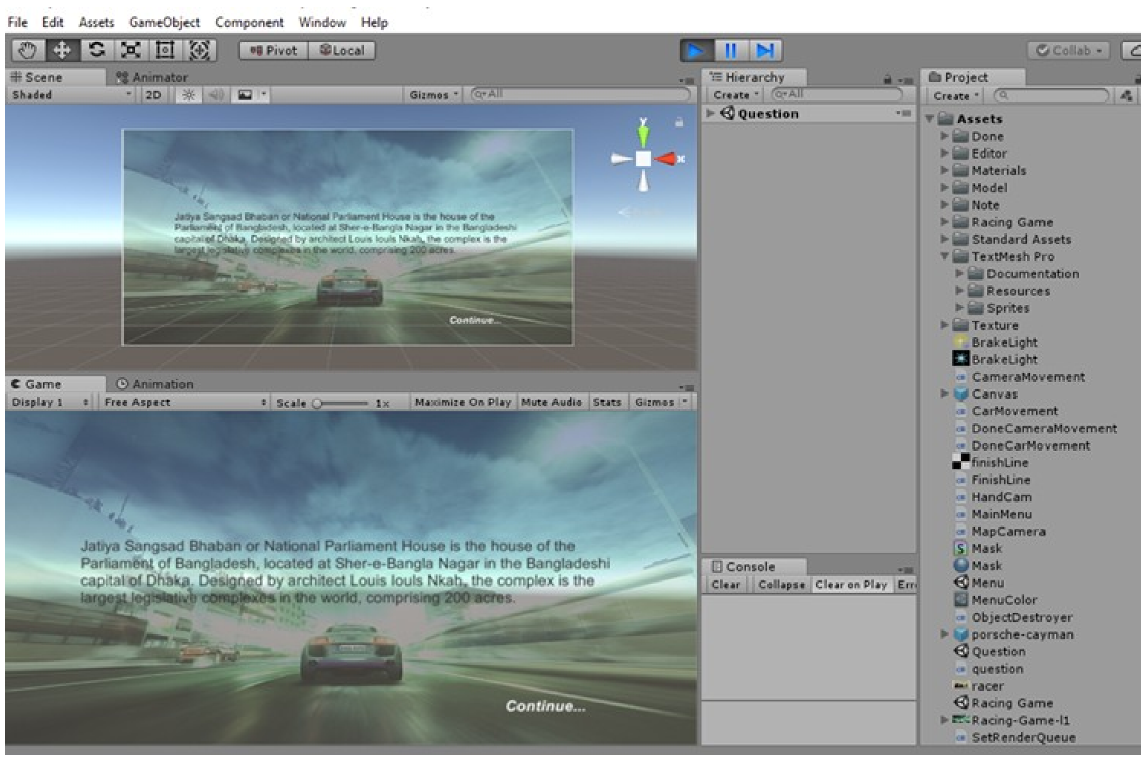
IMPLEMENTATION AND TESTING

5.1 Implementation of Database

Since our project is not about storing information we did not use any database here. The theme of our project is made racing game. In lieu of that, our little information about places was stored as game object.

5.2 Implementation of Front-End Design

As we already discussed about front-end design of our game, here we discus about information's and questions part. Figure 5.1 shows that part.



Jatiya Sangsad Bhaban or National Parliament House is the house of the Parliament of Bangladesh, located at Sher-e-Bangla Nagar in the Bangladeshi capital of Dhaka. Designed by architect Louis louis Nkah, the complex is the largest legislative complexes in the world, comprising 200 acres.

Continue...

Figure 5.1: Questions part

5.3 Implementation of Interactions

User interaction of a system depends on some factors. The more an application is interactive the more it is used. We tried to make our game more interactive to the users. In this project user first need to read information about places, then will ask for some questions and if it goes right the game will start. Otherwise user will need to read again the script to answers correctly.

5.4 Testing Implementation

Here we are going to see some cases where tester will see cases and specification. Under which conditions game would be resumed or interrupted. Table 5.1 shows the test implementation.

Table 5.1: Testing Implementation

Test Case	**Test Input**	**Expected Outcome**	**Actual Output**	**Result**	**Tasted on**
1. **Open game**	Information Displayed	Game to be started	Not started	Failed	06-11-18
2. **Play game**	Wrong answer	Game to be started	Not started	Failed	06-11-18
3. **Play game**	Correct answer	Game to be started	Started	Successful	06-11-18

5.5 Test Results and Reports

We tasted our system as under which condition it would start or not. Whenever the user can be able to give answers correctly game would be started otherwise wont. Then user will need to read information one more time to play the game.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Discussion and Conclusion

This racing game has two platforms one is entertainment and other education. We hope that might be able to give a different feel to the gamer while playing. Developing a racing is not an easy task especially for the fresher's. From our point of view through this project we have learned Unity 3d, Photoshop and using Blender. A word must be said about Unity that is this IDE is very much advanced than many other IDE we have learned so far. Though all three are not easy to learn at the beginning. Time after time we faced problems and by recovering them we become more compact.

The game could also prove to be a great platform for all its users to learn through entertainment in their leisure time. At last we can say we have learned some important topic things those will help us in our professional life.

6.2 Scope for Further Developments

We would like to continue developing this game, some of our plans are given below:

- Adding more historical places.
- Increasing levels.
- Adding database from where questions would be asked randomly on higher level.
- Our final target is to make it professional from all sides and keep it on Google Play Store.

REFERENCES

1 “Forza Horizon 3”. Available: https://www.forzamotorsport.net/en-US/games/fh3 [Last accessed on Oct. 15, 2018 at 11.31 pm].

[2] “RushRally2”. [Online]. Available: https://play.google.com/store/apps/details?id=brownmonster.app.game.rushrally2 [Last accessed on Oct. 15, 2018 at 11.40 pm].

3 “Grid Autosport”. [Online]. Available: https://www.androidcentral.com/grid-autosport-android. [Last accessed on Oct. 15, 2018 at 11.53pm].

4 “Blender”. [Online]. Available: https://www.blender.org/. [Last accessed on Feb. 18, 2018 at 10.06 am].

5 “Photoshop”. [Online]. Available: https://www.adobe.com/products/photoshop.html. [Last accessed on Mar. 06, 2018 at 8.30 pm].

6 “Unity 3d”. [Online]. Available: https://unity3d.com. [Last accessed on Mar. 25, 2018 at 7.00 pm].

7 “C#”. [Online]. Available: https://www.learncs.org. [Last accessed on Nov. 02, 2018 at 12.10 am].

8 “JavaScript”. [Online]. Available: https://www.w3schools.com/js. [Last accessed on Nov. 15, 2018 at 9.15 pm].

9 “Unity function for racing”. [Online]. Available: https://answers.unity.com/questions/46630/how-to-create-race-lap-timer-start-to-finish-displ.html. [Last accessed on Nov. 08, 2018 at 11.32 pm].

[10] “Functional vs. nonfunctional requirements”. [Online]. Available: https://stackoverflow.com/questions/16475979/what-is-the-difference-between-functional-and-non-functional-requirement. [Last accessed on Oct. 05, 2018 at 10.20 pm].

[11] “Seamless road texture”. [Online]. Available: https://thumbs.dreamstime.com/t/seamless-two-lane-road-texture-image-yellow-strip-high-resolution-83052836.jpg. [Last accessed on Feb. 03, 2018 at 2.00 pm].

[12] “UV editing”. [Online]. Available: https://www.youtube.com/watch?v=V6OXSR5Ynyc. [Last accessed on Feb. 04, 2018 at 11.00 am].

[13] “Car”. [Online]. Available: https://assetstore.unity.com/packages/3d/vehicles/land/lowpoly-sports-car-2-in-1-52995. [Last accessed on Jan. 20, 2018 at 11.30 am].

[14] “motorTorque Unity 3d”. [Online]. Available: https://forum.unity.com/threads/wheelcollider-motortorque.260915/. [Last accessed on May. 22, 2018 at 2.00 am].

[15] “Mathf.LerpAngle” [Online]. Available: https://docs.unity3d.com/ScriptReference/Mathf.LerpAngle.html. [Last accessed on Jun. 10, 2018 at 3.00 pm].

[16] “breakTorque”. [Online]. Available https://docs.unity3d.com/Manual/Coroutines.html. [Last accessed on Mar. 15, 2018 at 12.30 am].

APPENDIX

Appendix A: Project Reflection

The purpose of this appendix is about project reflection. From Fall 2017 semester we have started our journey to make this racing game. Our vision is making a different type of a racing game that we are normally used to play.

One can learn and playing game at a time. There are a lot of struggles we faced while making this game as a beginner. We are going forward step by step. After many hard work and spending a lot of time finally we were able to reach our goal.

The project "Race and Learn" will be very helpful for all types of Bangladeshi game lovers. Hopefully they can spend their leisure by learning and entertaining themselves. So, I believe that our game “Race and Learn” will be a positive racing game and will be appreciated and accepted by all age’s people.