Computer Graphics and Game Programming – 508837
3rd Assignment
Advisor: Assoc. Dr. Murat KURT

3rd Assignment Report

04.28.2023

By:
Mehmet ISIKHAN, 91220001906

Table of Contents

Introduction

This report is about a car race game's developed features and processes with using Unity game engine. The game includes five different cars and race maps, and the players can choose these maps and cars with a menu. Also, the program can run on both desktop and android based mobile devices. On the android mobile devices, the digital joystick includes to the left of the screen and the player can move the car with this digital joystick. On the desktop devices, cars can move with the “W”, “A”, “S”, “D” and the arrow keys on the keyboard. The maps elements and tracks change according to the maps, but the roads type does not change because of the keeping standard of the ways in the different maps and areas. In this way, players play on the same roads with different tracks on the different maps.

Car Race Game

1. Cars

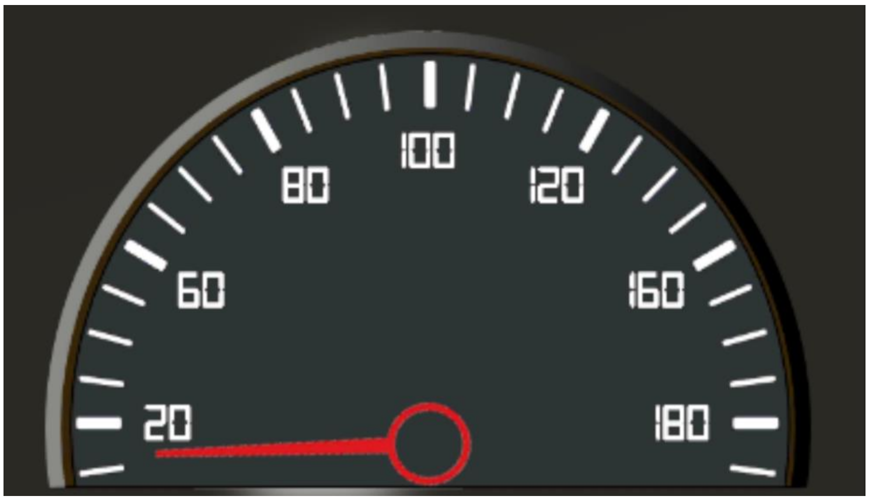
All the cars included in the Unity game engine have three main parts. Firstly, I found three-dimensional car models on the "cgtrader" web page *[1]*. Some 3D models did not have the necessary parts, or there were unified components. However, the car models should consist of a body and wheels. Additionally, each wheel should have a different layer for movement. For instance, a car should have one body part (it does not matter in the components of the body), and one wheels component. But the wheels component should include the four different wheel components (with rim etc.) I redesigned the cars according to this necessary consisting on the Blender software, and I am got exports with ".fbx" formatted. Also, I modified some cars as a race car for compatibility of the environment and the car. When importing the cars into Unity, the material settings had to be fixed according to the Blender export. I adjusted the material settings for each car and saved each of them as a prefab. Then, I created another prefab for all cars and their scripts and user interface components, and I saved it with the cars' prefabs in this prefab. When I was saving the cars on the prefab, I put each of them in the same position and I added "Cinemachine Virtual Camera" to each car with a follow point.

The "Cinmemachine" asset *[2]* helps to improve camera moving for each car. In all maps have a main camera with the "Cinemachine Brain" component and each car has "Cinemachine Virtual Camera" and a follow point. This virtual camera looks and follows to the "followpoint" element. In this way, the car moving and the showing of this moving makes it more realistic and intuitive.

For the movement of the cars, I could not find the ready-made asset (Standard Asset) in the Unity Asset Store as defined on the assignment paper *[3]*, so I searched for another ready-made script *[10]* on the Unity developer web page, which I used for the movement of my cars. This script requires a four-wheel collider for the wheels, as well as rigid body and box collider components for the body of the vehicle. I positioned the wheel colliders for each wheel and each car based on the cars' specific rim sizes. Then, I assigned these wheel colliders with the wheel game object components with the corresponding wheel locations. In this way, the cars can move with the keyboard keys (“W”, “A”, “S”, “D” and the arrow keys) but the game should be playable on the mobile devices. So, I added a digital joystick in the cars prefab’s canvas parts (for UI) with the car’s dial.

I created the “CarController1.cs” script file for car’s dial features, manage joystick activity and the cars activity according to player’s preferences. In this script the “activeCarSpeed” function runs for the car’s dial cursor movement. According to the selected car object’s speed *[3]* the car’s dial cursor’s rotation changes by multiplying to double. In this way, the dial cursor moves on the dial with the car’s speed value and the “activeCarSpeed” calling in the “Update” function for updating the cursor when the car’s speed changes.

```
    private float carSpeed = 0f;
    public GameObject dialArrow;

    //Dial Cursor movement//
    public void activeCarSpeed(){
        carSpeed = 
selectedCarObject.GetComponent<Rigidbody>().velocity.magnitude;
        carSpeed = -carSpeed*2;
        dialArrow.transform.rotation = 
Quaternion.Euler(0,0,carSpeed);
    }
```



Picture.1: Car Dial

35 MEHMET 35

Picture.2: Audi A7

Picture.3: Nissan Micra

Picture.4: Ford Mustang

16
MAHLE
Kaspersky

Picture.5: Ferrari F1

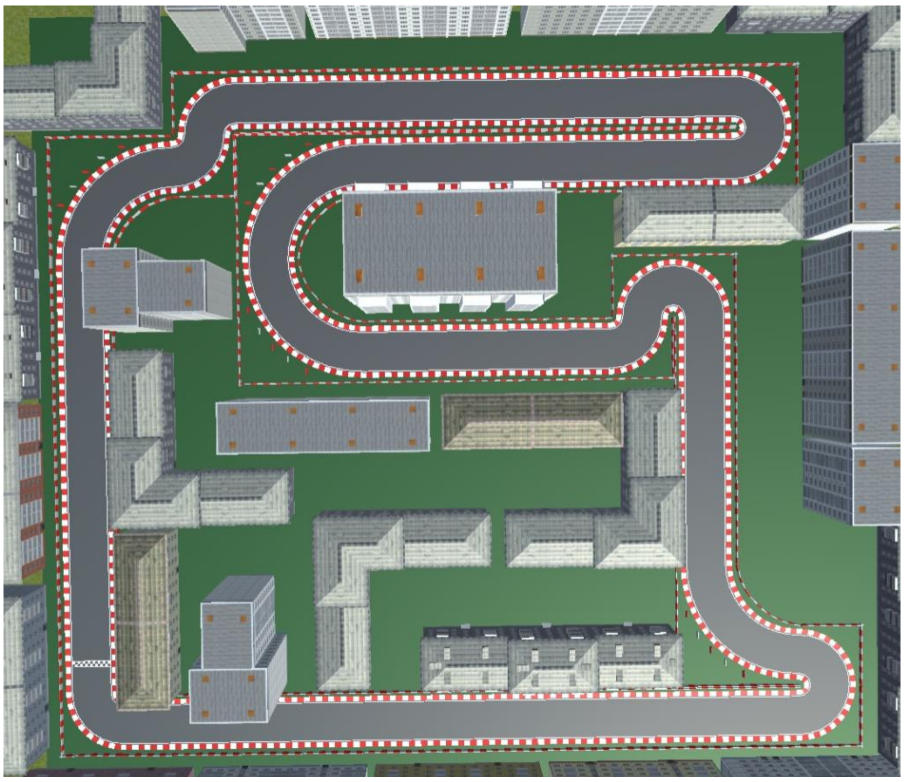
Picture.6: Audi R8

2. Maps

Firstly, before starting the maps, I added some necessary tools to my project. These tools include the car race 3b objects like roads, turbines, and barriers from the Unity asset store *[4][5]*. Also, for making terrain, I added the terrain tools again from the Unity asset store *[6]*. I used these two components in all maps in the game, also I used different assets for each map for the building or the other special 3d objects. I am created some prefabs again for the roads with barriers, and I am used these prefabs in my base road type.

a. City

For this map, I added low poly Russian buildings asset *[7]* to my project after the located the road comes from my created roads prefabs. I cover the roads with the buildings, and I am giving shape to the map's terrain with the terrain tools for the long-distance visualization. This map has less curved roads and shorter track the other maps, so we can say for beginners for this map.

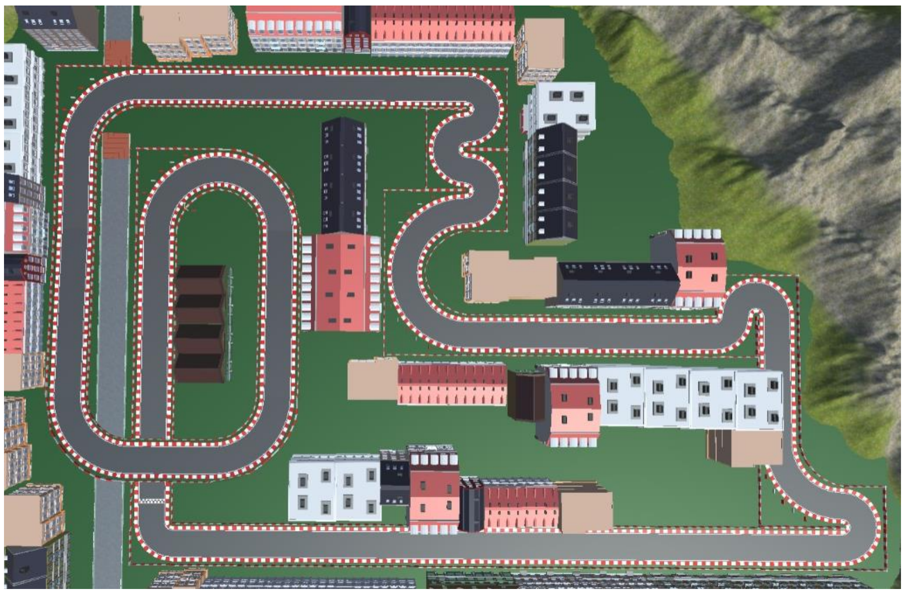


Picture.7: Map City

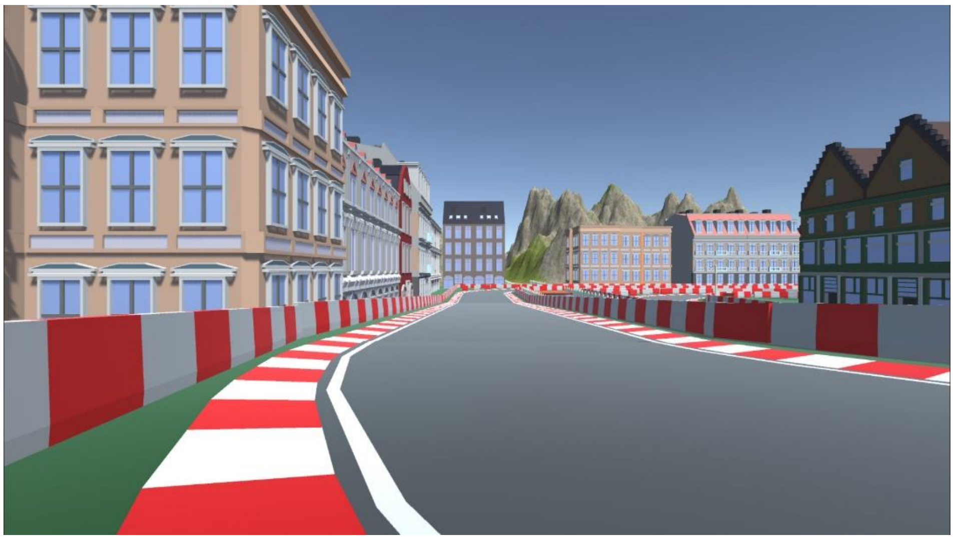
Picture.8: Map City

b. City-2

Specific feature of this map, it has a river and a bridge on this river. Like the first city map, I used road prefabs to create the roads on this map. For the European buildings, I used another low poly European buildings asset from the Unity store *[8]*. Additionally, this map is larger than the first city map, but it still has fewer curved roads, making it a good choice for beginners.



Picture.9: Map City-2

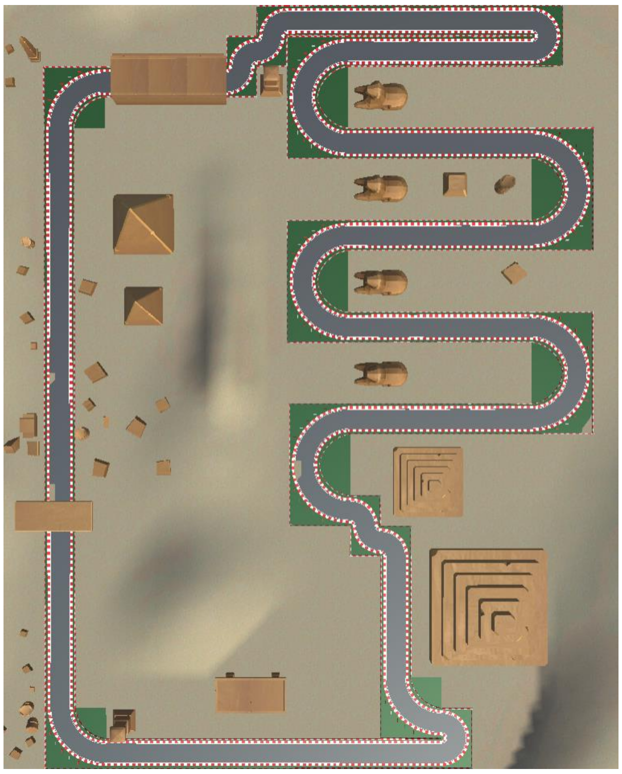


Picture.10: Map City-2

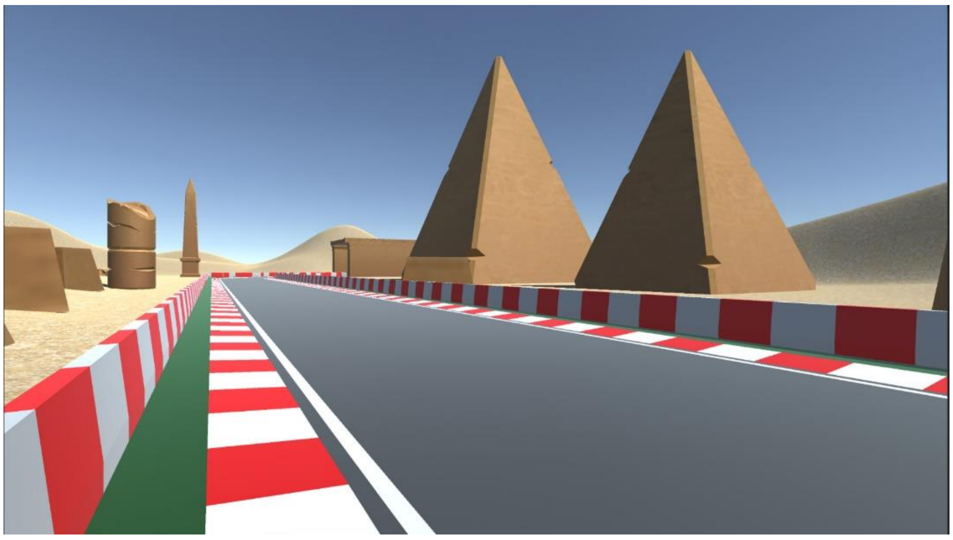
c. Desert

This map has the biggest track between the maps also it has long straight road for the test of the top speed. Also, the concept of this map is different from the first two maps. It includes the ancient Egyptian city ruins as the decor. The terrain texture is the sand mostly and this map has the most graphical performance of all the maps.

Note: I could take the lighting render only on this map.

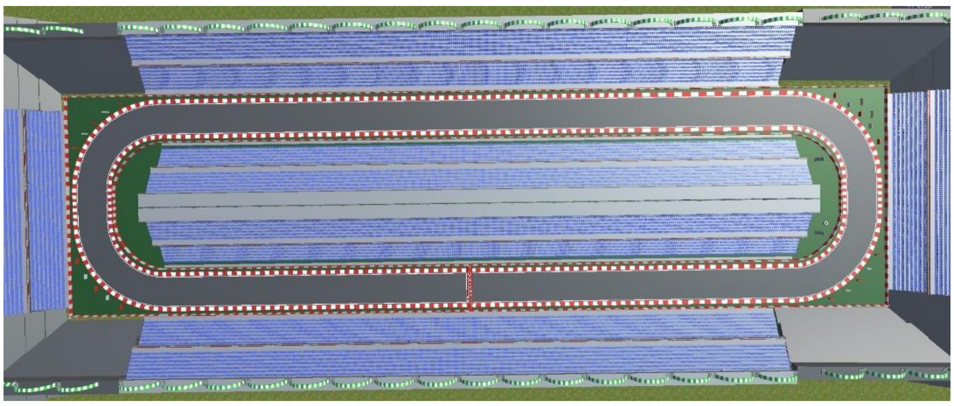


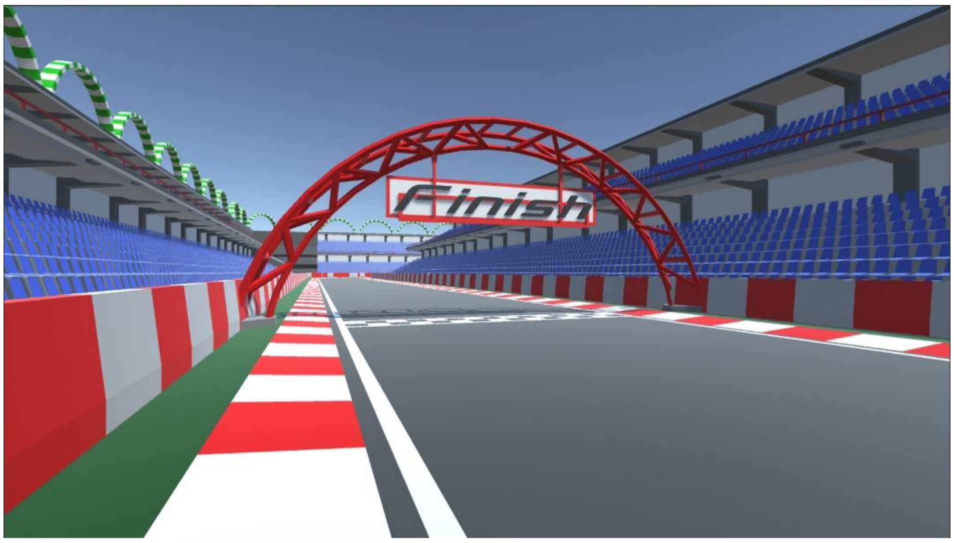
Picture.11: Map Desert


Picture.12: Map Desert

d. Racetrack

This map is designed to simulate Formula-1 racing with turbines and resembles the racetrack in the first Cars movie. The track on this map is simple and short, but the most important feature is the turning time of the cars. To create this map, I used only the racetrack turbine asset and the road prefabs.

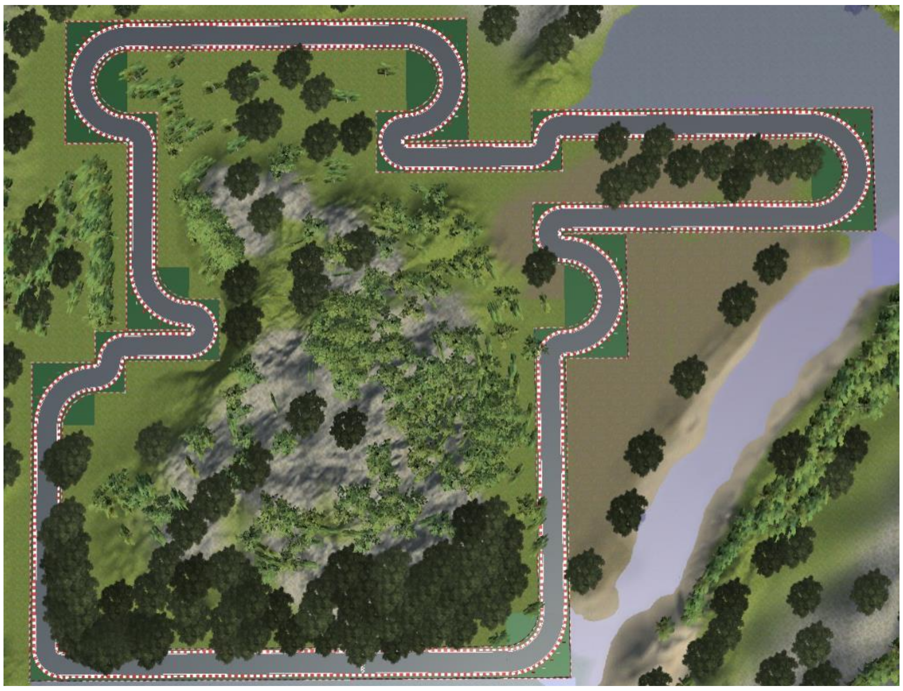

Picture.13: Map RaceTrack



Picture.14: Map RaceTrack

e. Forest

The most graphically demanding map for devices is this one, as it includes numerous trees and grass objects. It includes a river that flows into a lake at the end. I created the river with blue colored transparent material. For this map, I used the terrain tools to create all the components such as trees, grass, and mountains, which makes it one of the most graphically demanding maps for devices. This map features a large number of curved roads, making it more challenging and suitable for expert players.



Picture.15: Map Forest

Picture.16: Map Forest

I could not take the lighting render for all maps because of my computer rendering capabilities, so there are some lighting and shadowing problems in the maps, except for the desert map. I was able to render lighting for only the desert map.

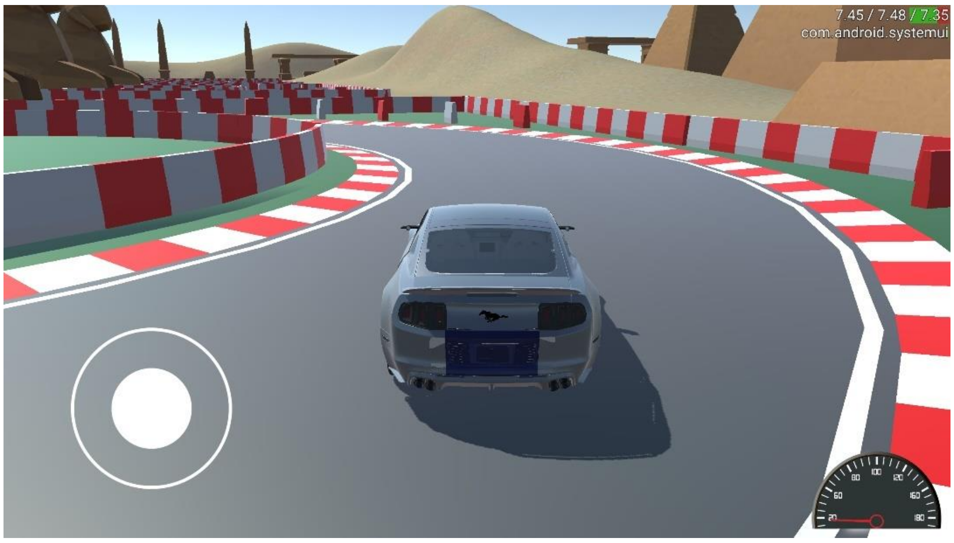
3. User Interface

For the selection process of the maps and cars, I created two different menu scenes in Unity. Also, there is a starter page in the game with a device setup button. Firstly, I want to talk about the device setup button feature. The device determining is working automatically but the players can change the platform with this button when they want. When the device is setup on the "Desktop", players can play with the determined keyboard keys ("W", "A", "S", "D" and the arrow keys). If the device is set to "Mobile", the system puts the digital joystick *[8]* which I added from Unity asset store on the screen for movement of the cars and players can play with this joystick with touch. The control of this setting manages by "PlatformType.cs" script. In this script system checks the system type of touching input or not *[9]* and it defined to "gameType" according to this checking process. At the end of the starting function of the script, the system assigns to "gameType" variable player preferences with "platform" tag. So, the system can keep the platform type with this player preferences value. If the player touches the button the device type changes, and the system keeps it in this time with the "gameMode" function.

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

public class PlatformType : MonoBehaviour
{

    private string gameType;
    public GameObject gameModeButton;


    void Start()
    {
        if(SystemInfo.deviceType == DeviceType.Handheld ){
            gameType = "Mobile";

            gameModeButton.GetComponent<TMPro.TextMeshProUGUI>().t
            ext = "Mobile";
        }else{
            gameType = "Desktop";

            gameModeButton.GetComponent<TMPro.TextMeshProUGUI>().t
            ext = "Desktop";
        }
        PlayerPrefs.SetString("platform", gameType);
    }

    public void gameMode(){
        if(gameType == "Mobile"){

            gameModeButton.GetComponent<TMPro.TextMeshProUGUI>().t
            ext = "Desktop";
            gameType = "Desktop";
        }else{

            gameModeButton.GetComponent<TMPro.TextMeshProUGUI>().t
            ext = "Mobile";
            gameType = "Mobile";
        }
        PlayerPrefs.SetString("platform", gameType);
    }
}
```

In the cars prefabs according to player preferences with “platform” tag, the joystick active or inactive with the “joysStickActivity” function.

```
public void joyStickActivity(){
    if(PlayerPrefs.GetString("platform") == "Mobile"){
        joystickObject.SetActive(true);
    }else{
        joystickObject.SetActive(false);
    }
}
```

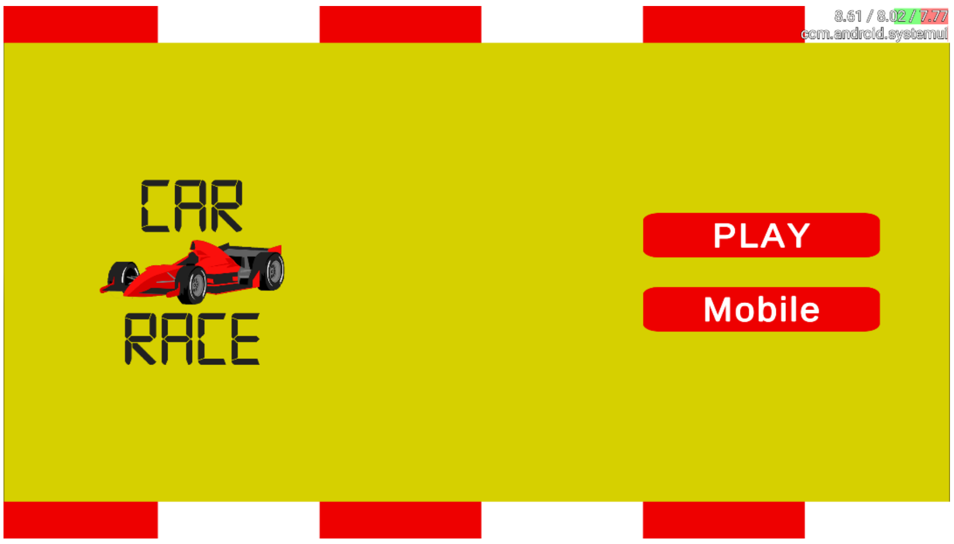


Picture.17: Digital Joystick

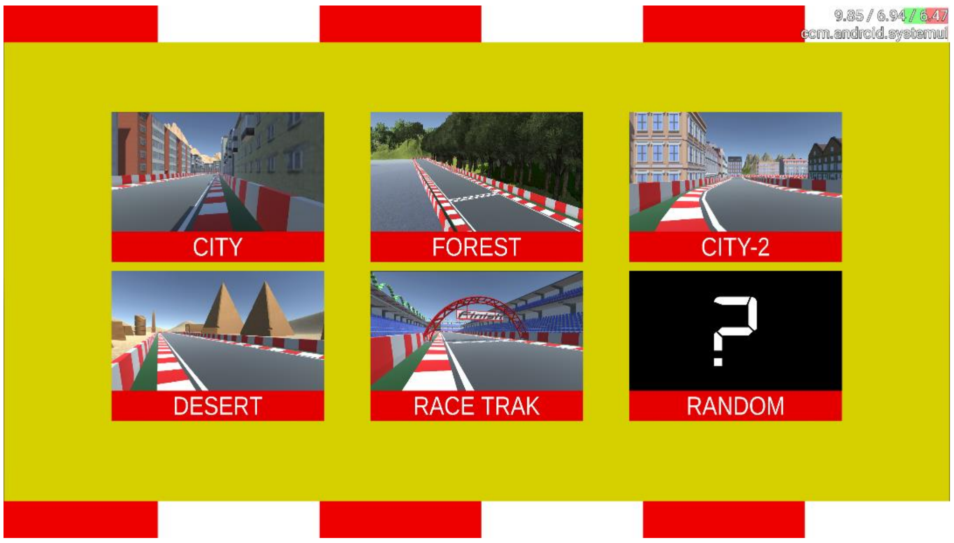
Before the game starts, players need to select a car and a map. To facilitate this, two menus have been created: one for selecting the map and the other for choosing the car. The script for selecting the map (SceneController.cs) runs on the second scene, and based on the buttons clicked by the player, the “mapSelection” function assigns an integer value to the “selectedMap” variable, which corresponds to the scene index for the selected map, as specified in the player's “selectedMap” tag. Once the car is selected, the same process is repeated, and the “selectedCar” variable is assigned the integer value corresponding to the player's “selectedCar” tag. The system also includes a random button for each selection menu, which generates a random scene index between 3 and 8 for the map and a random integer value between 1 and 6 for the car selection.

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

public class SceneController : MonoBehaviour
{

    static int selectedMap = 0;
    static int selectedCar = 0;

    public void playButton(){
        SceneManager.LoadScene(1);
    }

    public void mapSelection(int map){
        switch(map){
            case 1:
                    selectedMap = 3;
                    SceneManager.LoadScene(2);
                    break;
            case 2:
                    selectedMap = 5;
                    SceneManager.LoadScene(2);
                    break;
            case 3:
                    selectedMap = 6;
                    SceneManager.LoadScene(2);
                    break;
            case 4:
                    selectedMap = 7;
                    SceneManager.LoadScene(2);
                    break;
            case 5:
                    selectedMap = 4;
                    SceneManager.LoadScene(2);
                    break;
            case 6:
                    selectedMap = Random.Range(3, 8);
                    SceneManager.LoadScene(2);
                    break;
            case 0:
            default:
                    break;
        }
        PlayerPrefs.SetInt("selectedMap", selectedMap);
    }
```

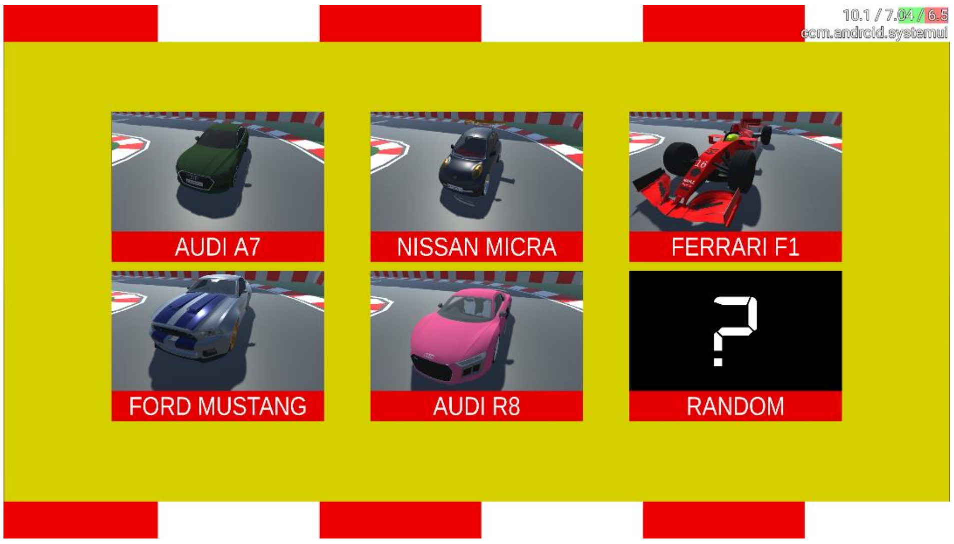
```
        public void carSelection(int car){
            if(car != 0){
                if(car == 6){
                    selectedCar = Random.Range(1, 6);
                }else{
                    selectedCar = car;
                }
                PlayerPrefs.SetInt("selectedCar", selectedCar);
                SceneManager.LoadScene(selectedMap);
            }
        }
    }
```



Picture.18: Main Menu



Picture.19: Map Selection Menu



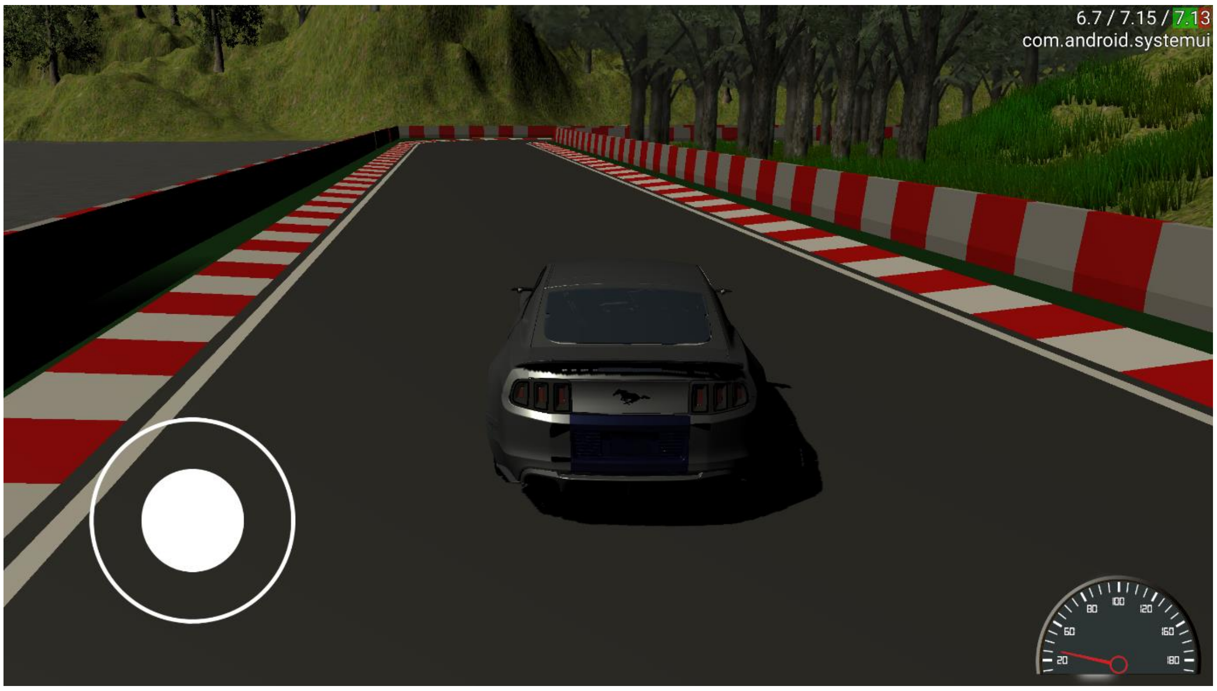
Picture.20: Car Selection Menu

In the cars prefab according to the player preference with “selectdCar” value, the car game objects be active or inactive in the “CarController1.cs” script’s “carActive” function. This function calls beginning of the prefabs with takin of the player preference value. Then the selected car assigns to “selectedCarGameObject” variable for the speed dial features.

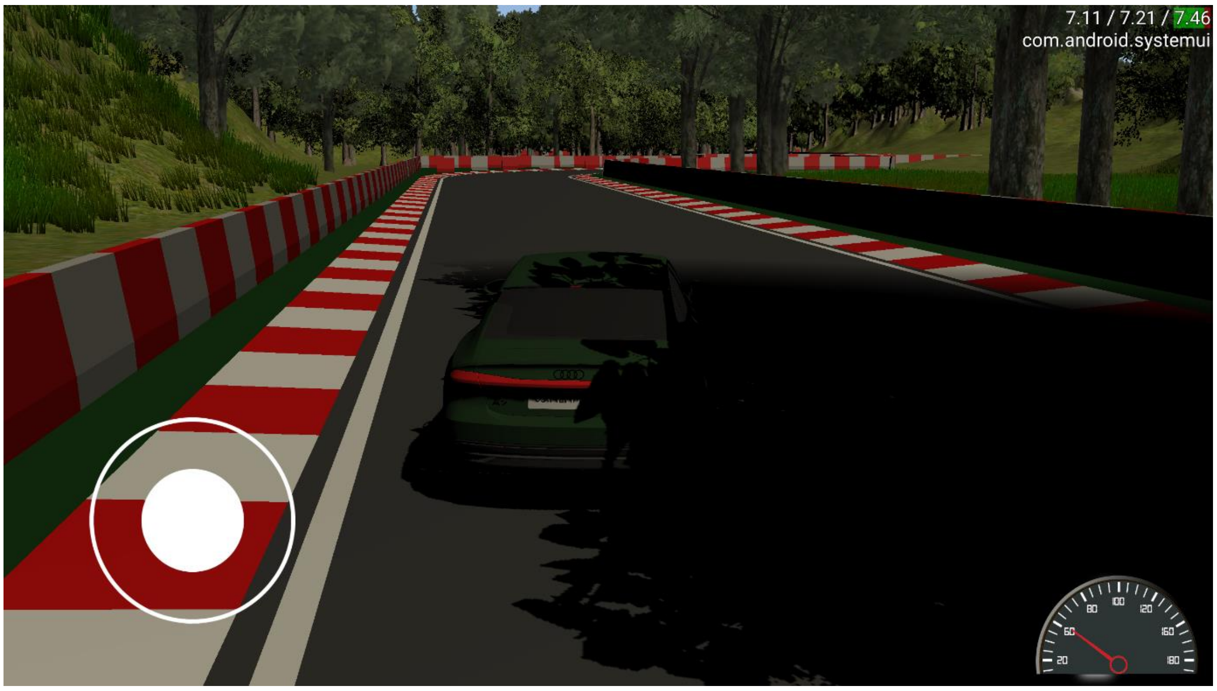
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CarController1 : MonoBehaviour
{
    int selectedCar;
    public GameObject car1;
    public GameObject car2;
    public GameObject car3;
    public GameObject car4;
    public GameObject car5;
    private GameObject selectedCarObject;
    private float carSpeed = 0f;
    public GameObject dialArrow;
    public GameObject joystickObject;

    void Start()
    {
       selectedCar = PlayerPrefs.GetInt("selectedCar");
       joyStickActivity();
       carActive();
    }
```

```
public void carActive(){
    switch(selectedCar){
        case 1:
            car1.SetActive(true);
            car2.SetActive(false);
            car3.SetActive(false);
            car4.SetActive(false);
            car5.SetActive(false);
            selectedCarObject = car1;
            break;
        case 2:
            car1.SetActive(false);
            car2.SetActive(true);
            car3.SetActive(false);
            car4.SetActive(false);
            car5.SetActive(false);
            selectedCarObject = car2;
            break;
        case 3:
            car1.SetActive(false);
            car2.SetActive(false);
            car3.SetActive(true);
            car4.SetActive(false);
            car5.SetActive(false);
            selectedCarObject = car3;
            break;
        case 4:
            car1.SetActive(false);
            car2.SetActive(false);
            car3.SetActive(false);
            car4.SetActive(true);
            car5.SetActive(false);
            selectedCarObject = car4;
            break;
        case 5:
            car1.SetActive(false);
            car2.SetActive(false);
            car3.SetActive(false);
            car4.SetActive(false);
            car5.SetActive(true);
            selectedCarObject = car5;
            break;
        default:
            break;
    }
}
```

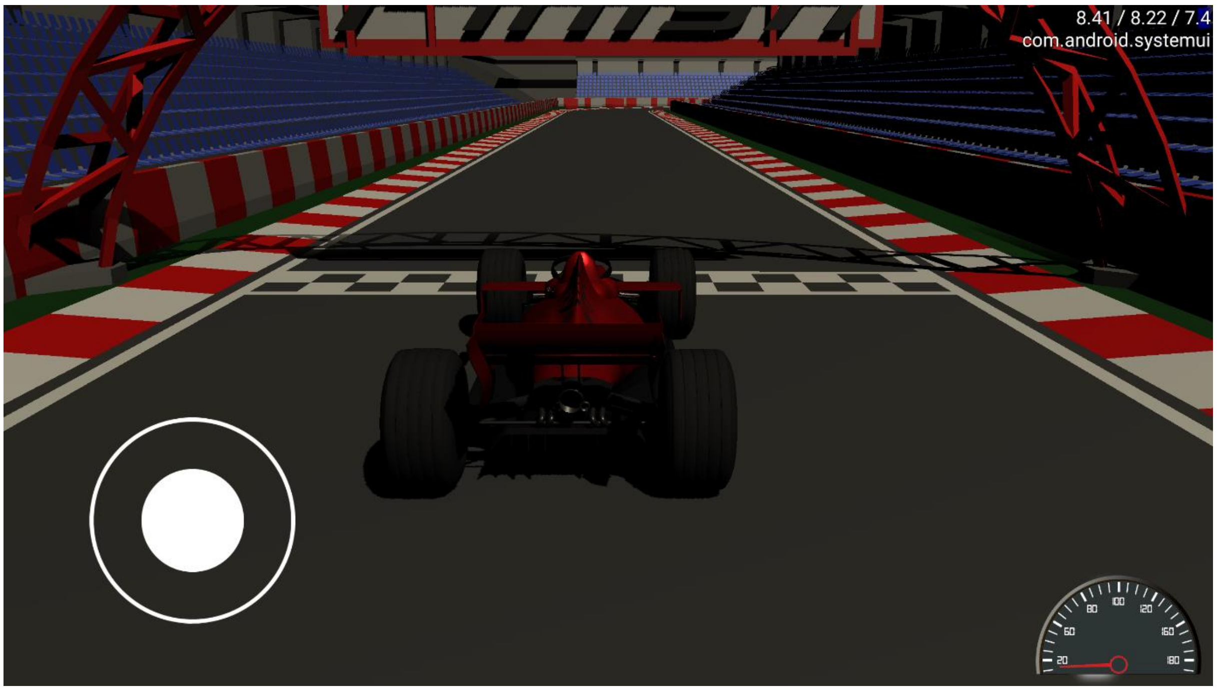
Screen Shots



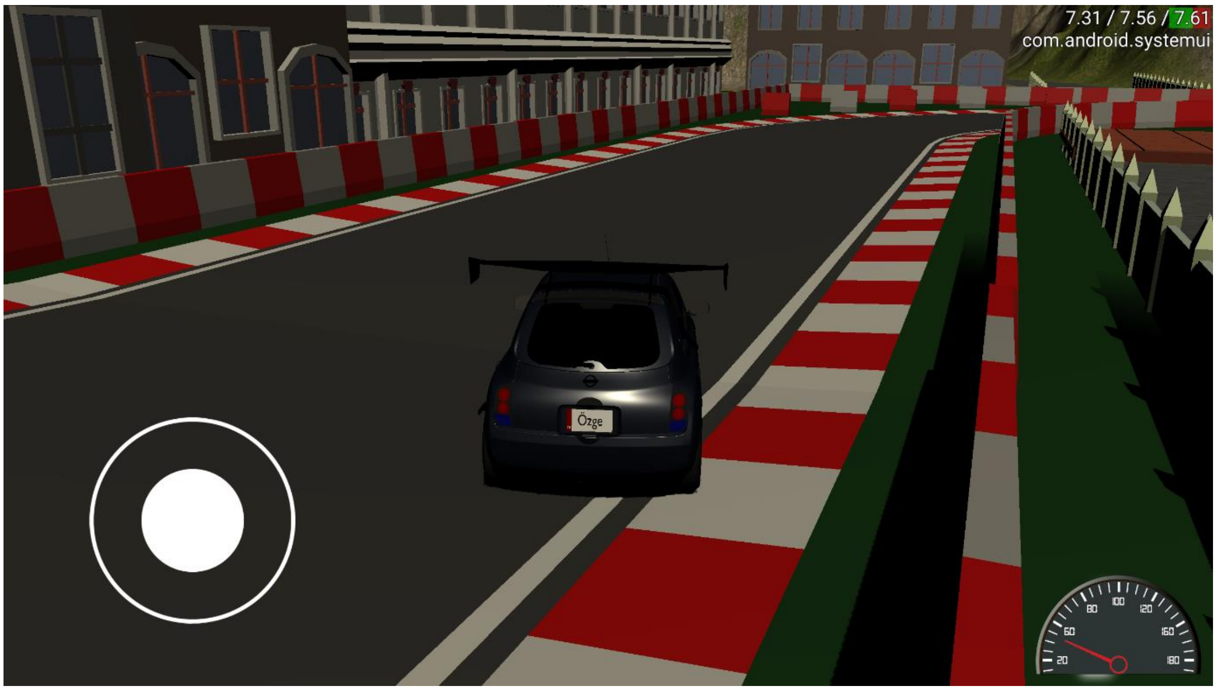
Picture.21: Android Game Play – Forest Map – Ford Mustang



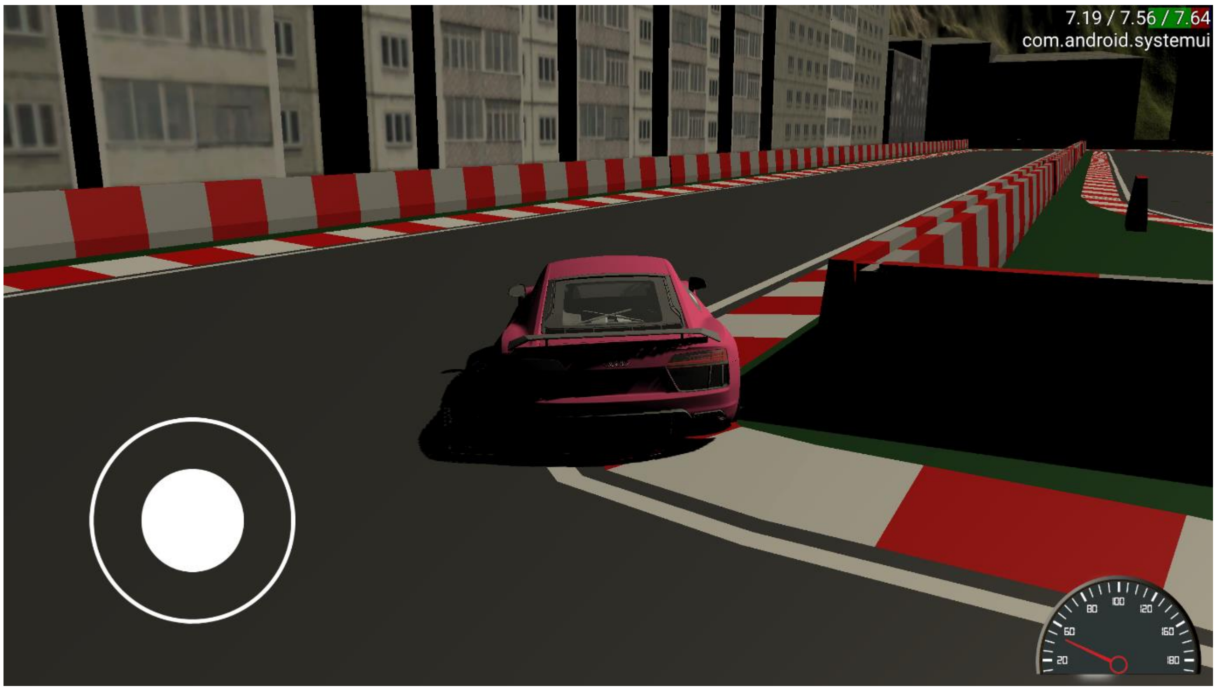
Picture.22: Android Game Play – Forest Map – Audi A7



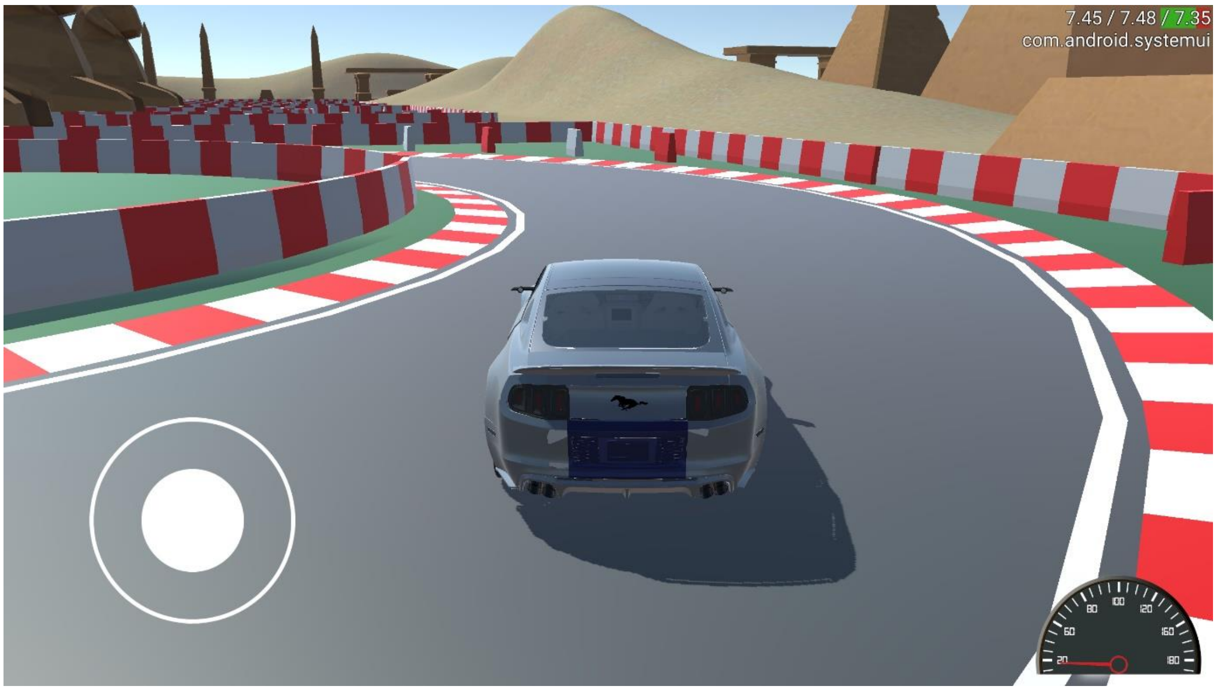
Picture.23: Android Game Play – TrackRace Map – Ferrari F1



Picture.24: Android Game Play –City-2 Map –Nissan Micra



Picture.25: Android Game Play – City Map – Audi R8



Picture.25: Android Game Play – Desert Map – Ford Mustang

References

1. https://www.cgtrader.com/3d-models
2. https://unity.com/unity/features/editor/art-and-design/cinemachine
3. https://answers.unity.com/questions/867706/how-to-get-the-accurate-speed-of-a-car-with-wheelc.html
4. https://assetstore.unity.com/packages/3d/environments/roadways/modular-lowpoly-track-roads-free-205188
5. https://assetstore.unity.com/packages/3d/props/props-for-track-environment-lowpoly-free-211494
6. https://assetstore.unity.com/packages/tools/terrain/tom-s-terrain-tools-527
7. https://assetstore.unity.com/packages/3d/environments/urban/low-poly-european-city-pack-71042
8. https://assetstore.unity.com/packages/tools/input-management/joystick-pack-107631
9. https://www.youtube.com/watch?v=Sot6B-EybIs
10. https://docs.unity3d.com/Manual/WheelColliderTutorial.html