CAR RACE GAMING PROJECT FILE

COURSE CODE: CSE3029

COURSE TITLE: GAME PROGRAMMING

PREPARED BY:

ADITYA NARAYAN – 19BCE2172

KISHAN KUMAR SHARMA – 19BCE2569

TITLE: COUNTRYSIDE CAR RACER

AIM:-

Car racing games have always been a topic of keen interest since the evolution of video games.Fastpaced ludicrious cars competing against obstacles such as time or other npc components giev a sense of refreshing thrill.Typically car race games consist of a set up background environment which helps the setting of the mood,a player car which is kept in action is kept which has to overcome the given set of obstacles which would help the player win the given set of race.

Henceforth,the above mentioned game proscrabes traditional setting of populous car tracks and and instead a prestoned-era countryside outlook has been provided.The player has to pass through a given set of obstacles (in forms of npc.levels).For the development of this game usage of unity code gaming development environment has been used and moreoverusage of other deployed software has been done.

TOOLS USED:

:Unity Game Development Software

: Mixamo

:Unity Asset Store

:Visual Studios

:Adobe Photoshop

DESIGN PROCEDURE:

The following game is a single car racing game .Ithas been provided with a countryside background with a rustic appproach.The player would be provided with the oppurtunity to select car race tracks and the type of cars.The player would have to cover a cewrtain set of distance to finish a lap(number of laps provided has been 5).

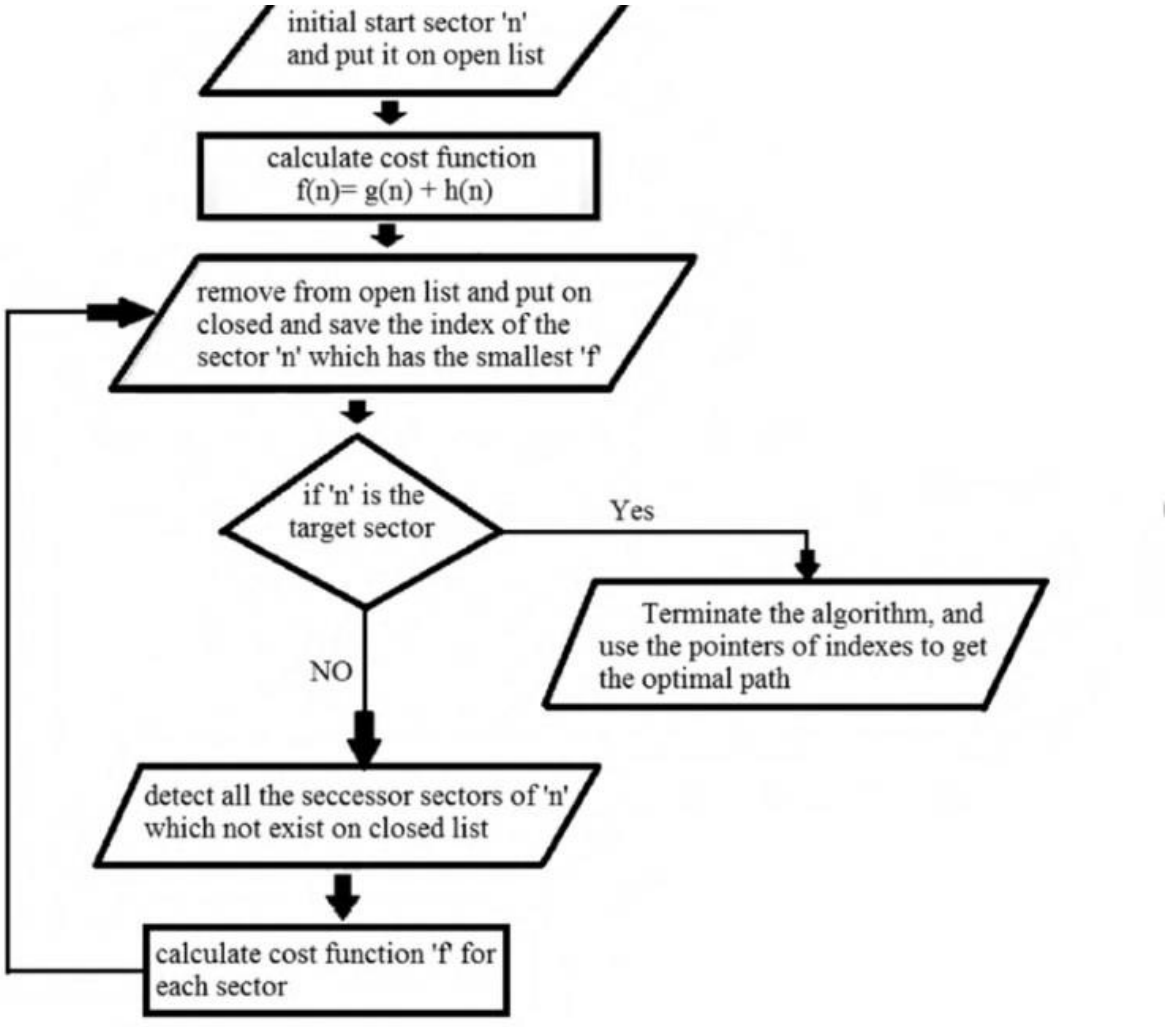
After the completion of a given set of laps the player can move to the next level.Techniques of collision detection has been implemented to detect the colliding movement between the cars and the terrainous obstacles presented by the user.Usage of A* algorithm has been deployed for the racing simulation of the game development .Talking further about A* algorithm we observe that the following set of algorithms have been talked about in detail.

A* algorithm:

When we take A* algorithm ‘s procedural working into account ,we observe that Informally speaking, A* Search algorithms, unlike other traversal techniques, it has “brains”. What it means is

that it is really a smart algorithm which separates it from the other conventional algorithms.
And it is also worth mentioning that many games and web-based maps use this algorithm to find the shortest path very efficiently (approximation).

FLOWCHART OF THE WORKING OF A* ALGORITHM

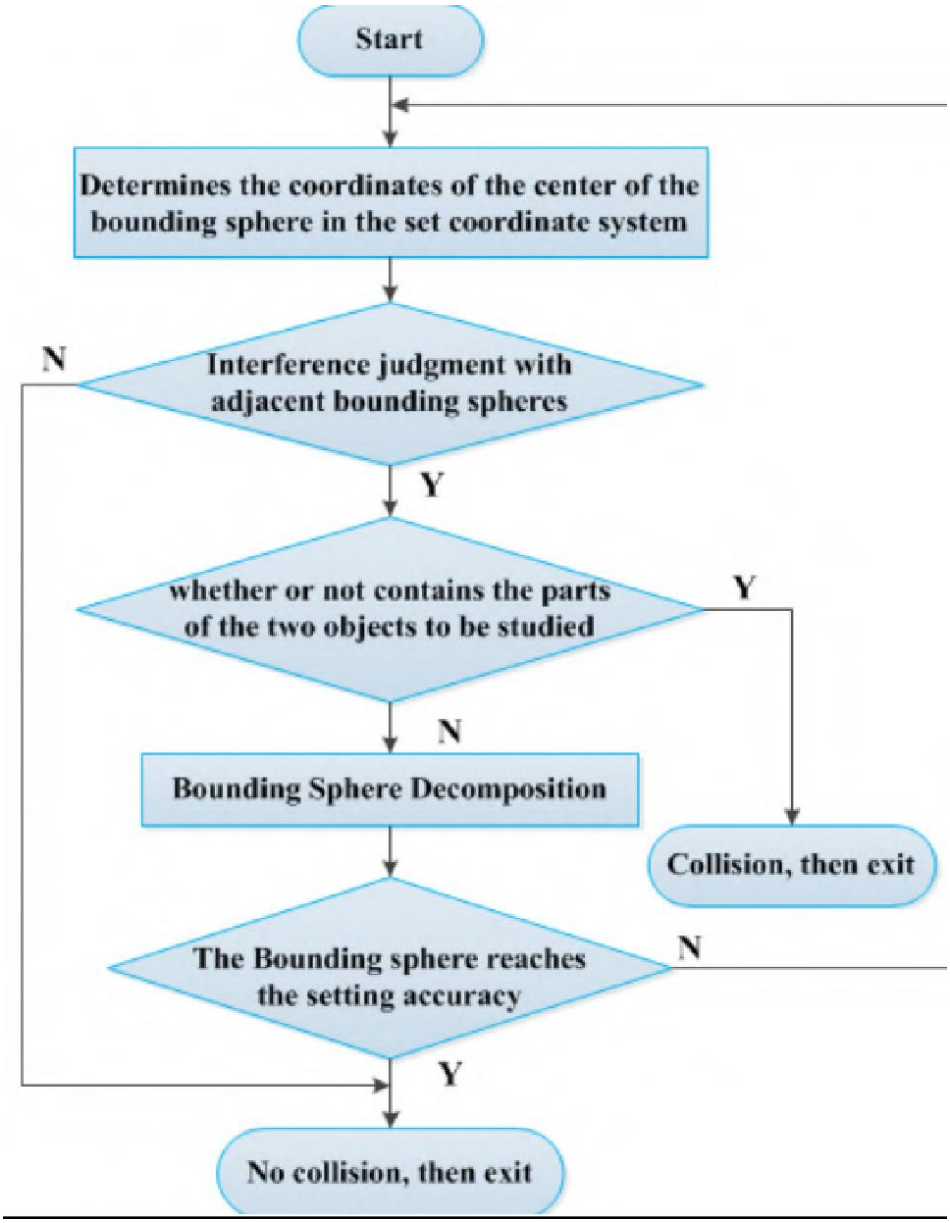


Collision Detection Approach:

Observing the above given collision detection approach we find that Collision detection technology is widely used in computer animation, robotics and computer aided design. It is mainly used to detect whether the objects collide and react accordingly. This technology is essential for the authenticity of the virtual scene. This technology has been concerned by scholars at home and abroad for many years. At present, there are two kinds of classical algorithms: hierarchical bounding box and spatial

partitioning . Hierarchical bounding box algorithm is larger and the geometric characteristics of a simple bounding box to replace the complex geometric objects, and by constructing a hierarchical bounding box tree approximation of real objects, collision detection by traversing the two bounding boxes to roughly determine the intersection condition of objects. The typical hierarchical tree consists of the sphere bounding hierarchy tree , the axial bounding box (AABB) hierarchical tree , the Oriented Bounding Box (OBB) hierarchical tree , the discrete direction bounding box (K-Dop) hierarchical tree etc. The method of spatial segmentation is to divide the virtual space into different regions, and only detect the objects in the same region. The classical algorithm includes the uniform grid, octree, BSP tree.

FLOWCHART OFTHE WORKING OF COLLISION DETEC TION ALGORITHM:



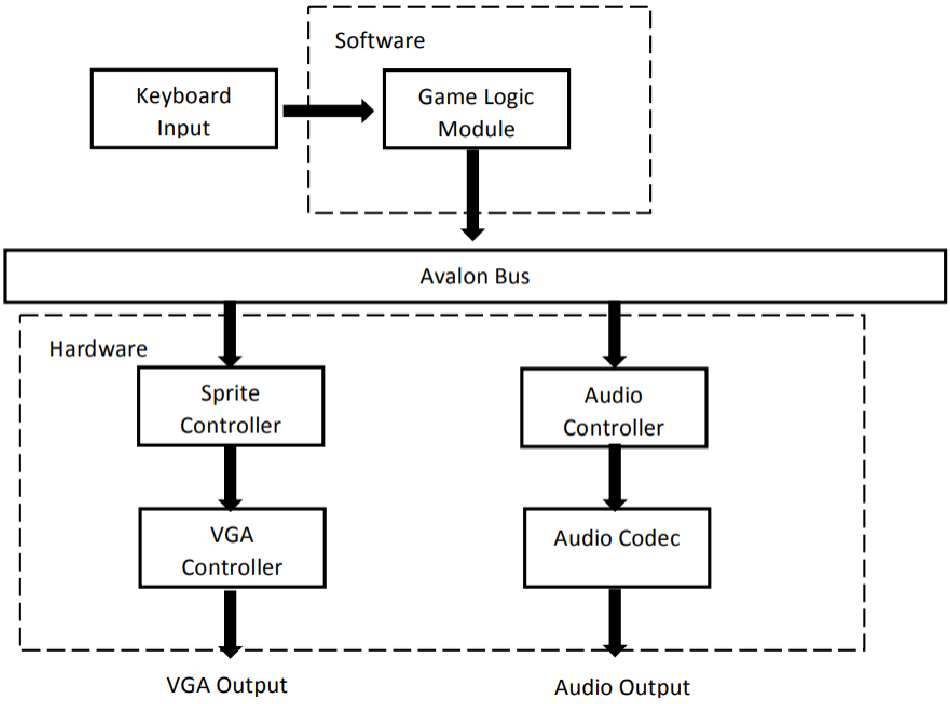


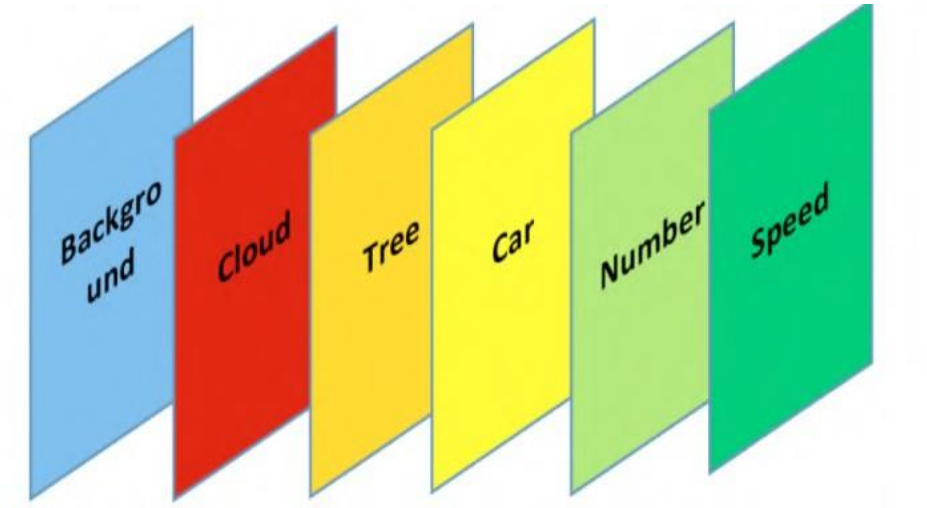
Figure representing the overall design module approach

DESIGN MODULES AND THEIR DESCRIPTION

Game Logic Module: Game Logic Module has been completed in C# language. The main function of the Game Logic is list as following table :

Generate the position (coordinate) of object which is the control input of Sprite Controller, according to the game logic	Road
	Trees
	Clouds
	Cars
	Speed
	Position
	Score
	Timer
Generate the Audio control signal	Sounds like bump, break, accelerate etc.

The Background Generation: In our design, the background containing cloud, land and sky is generated. We have designed a special module for Background Generation.



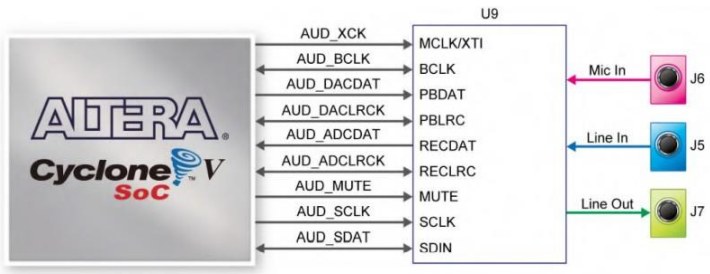
Audio Design: Audio Generator Module is designed to send instructions to the Audio Controller Module in order to play appropriate sound effect according to the game processes.

GAME ASSESTS:

Materials: Colours and pictures applied to different game objects like blue, yellow and green for the car design, green for the trees and grass and black for road.Moreover different sets of colour combination have implemented for the colouring of the car.Different sets of cotouring have been deployed for the obstacles arrangement.

Audio: Audio is played when the user runs the car and also during the collision against obstacles. Terrain: It consists of green pastures ,conifers and lucid grassy fields in the background. And special sound effects such as vehicle acceleration, braking, and collisions are integrated when the specific situations happen. At the beginning of the game, the software extracts the background sound sample previously stored in the SDRAM and load it into the circular queue in the Audio Controller Module, which will continuously send signals to the SSM2603 Audio Codec to play the background sound sample until the player exit the game. During the car racing, the software continuously checks whether the player hits the acceleration or braking button, or if there is any collision happens. Whenever a specific situation is detected, the software will instruct the DRAM to load the corresponding special sound effect into FIFO, and instruct the SSM2603 Audio Codec to play the special sound effect. The Sockit board we are using in the lab provides the Analog Devices SSM2603 audio CODEC (Encoder/Decoder). This chip supports microphone-in, line-in, and line-out ports, with a sample rate adjustable from 8 kHz to 96 kHz. The SSM2603 is controlled via a serial I2C bus interface, which is connected to pins on the CycloneV SoC FPGA.

Representation of the audio circuitry board



- **Models**: 3D art for the racing track course including roads, flags, etc. has been imported.
- **Textures**: Different types of textures have been imported from unity asset store.These include greeny ,shady pastures and moreover segmented car race tracks.

HOW TO PLAY:

- The player has to select a track and a vehicle of his own choice.
- Henceforth he has to click on the start game button
- The game has been divided into a series of 5 laps at each level.
- The position of the player would be displayed in the top above corner and as soon as a particular lap is completed the lap status is updated.
- As soon as the player completes all the lap and he is first among his competitors ,he would be declared winner.
- If the player fails to fulfil the above given task, he would be declared a looser.

DEPLOYMENT OF THE CODE SCRIPTS:

```
public class ButtonOption : MonoBehaviour {
    public void playGame()
    {
        SceneManager.LoadScene(2);
    }
    public void TrackSelect()
    {
        SceneManager.LoadScene(1);
    }
    public void MainMenu()
    {
        SceneManager.LoadScene(0);
    }
    public void quitGame()
    {
        // if (Input.GetKey("escape"))
        Application.Quit();

    }

    ///
    public void track01()
    {
        SceneManager.LoadScene(2);
    }
    public void track02()
    {
        SceneManager.LoadScene(4);
    }
}
```

Fig: *Scripting for the functionality operation of buttons*

```
public class CarChoice : MonoBehaviour {

    public GameObject RedBody;
    public GameObject BlueBody;
    public GameObject normalbody;
    public GameObject police;
    public int CarImport;
    void Start () {
        CarImport = GlobalCar.CarTyp;
        if(CarImport == 1)
        {
            normalbody.SetActive(false);
            police.SetActive(false);
            RedBody.SetActive(true);
        }
        if (CarImport == 2)
        {
            police.SetActive(false);
            normalbody.SetActive(false);
            RedBody.SetActive(false);
            BlueBody.SetActive(true);
        }
        if (CarImport == 3)
        {
            normalbody.SetActive(false);
            RedBody.SetActive(false);
            BlueBody.SetActive(false);
            police.SetActive(true);
        }
    }

}
```

Fig:Car selection script

```
public class CameraMovement : MonoBehaviour {

    public GameObject TheCar;
    public float carx;
    public float cary;
    public float carz;

    void Update () {
        carx = TheCar.transform.eulerAngles.x;
        cary = TheCar.transform.eulerAngles.y;
        carz = TheCar.transform.eulerAngles.z;

        transform.eulerAngles = new Vector3(carx - carx, cary, carz-carz);
    }
}
```

Fig:Scripting of the camera movemement

```
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;
    }
```

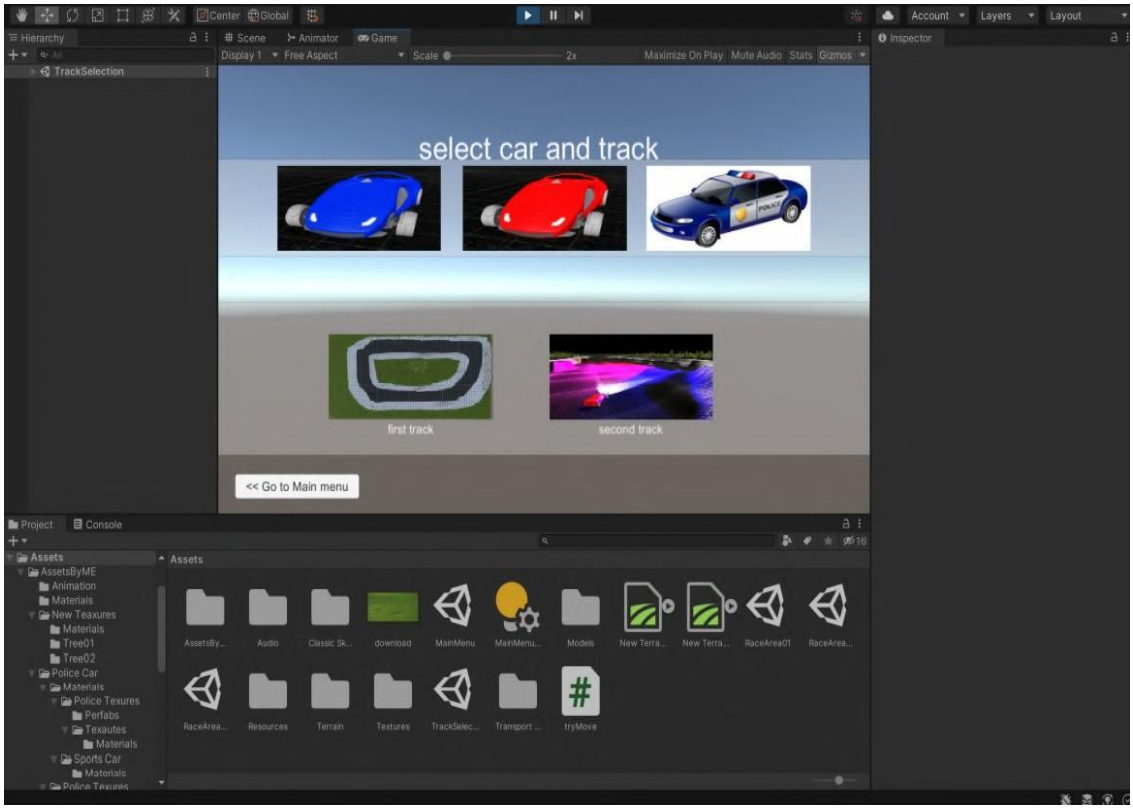
Fig: Scripting of the laptime

IMPLEMENTATION OF THE GAMEPLAY SECTION:

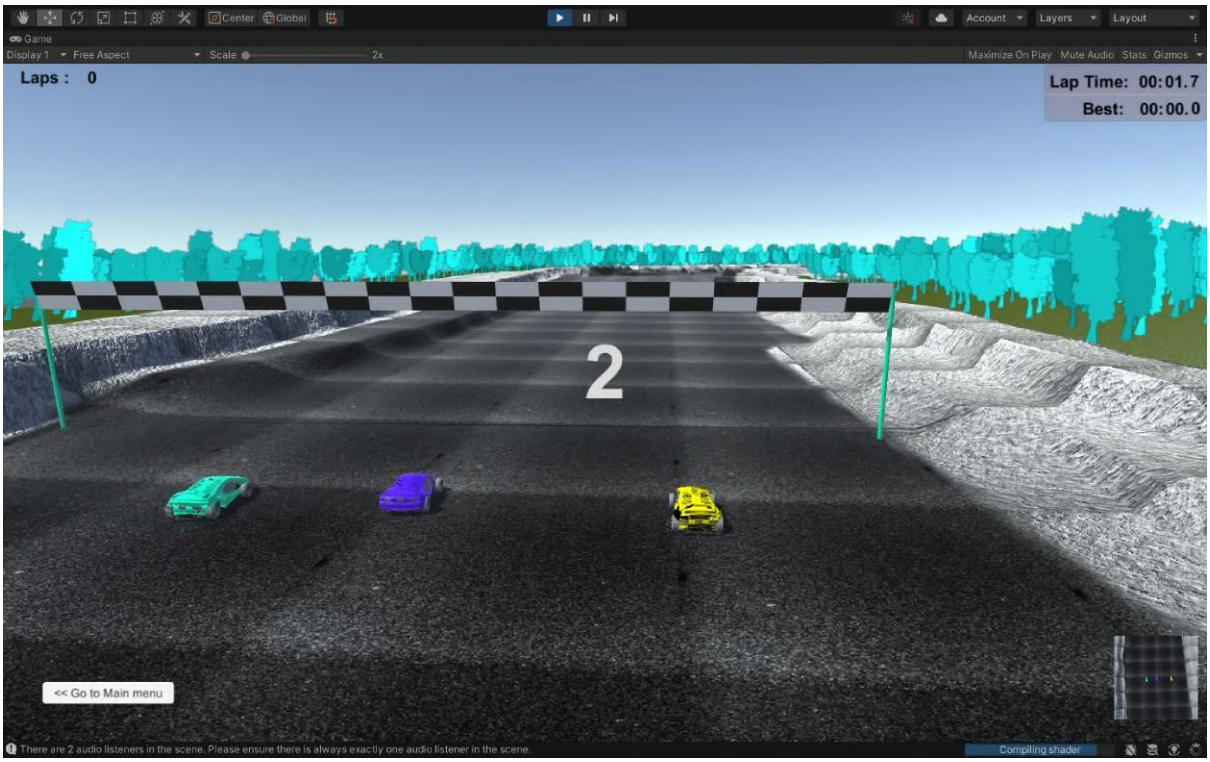
- The player is provided with the front-menu section where he is provided with the option of selecting whether he has to play the game instantly or select a car racing track.

- The player selects the select track option.

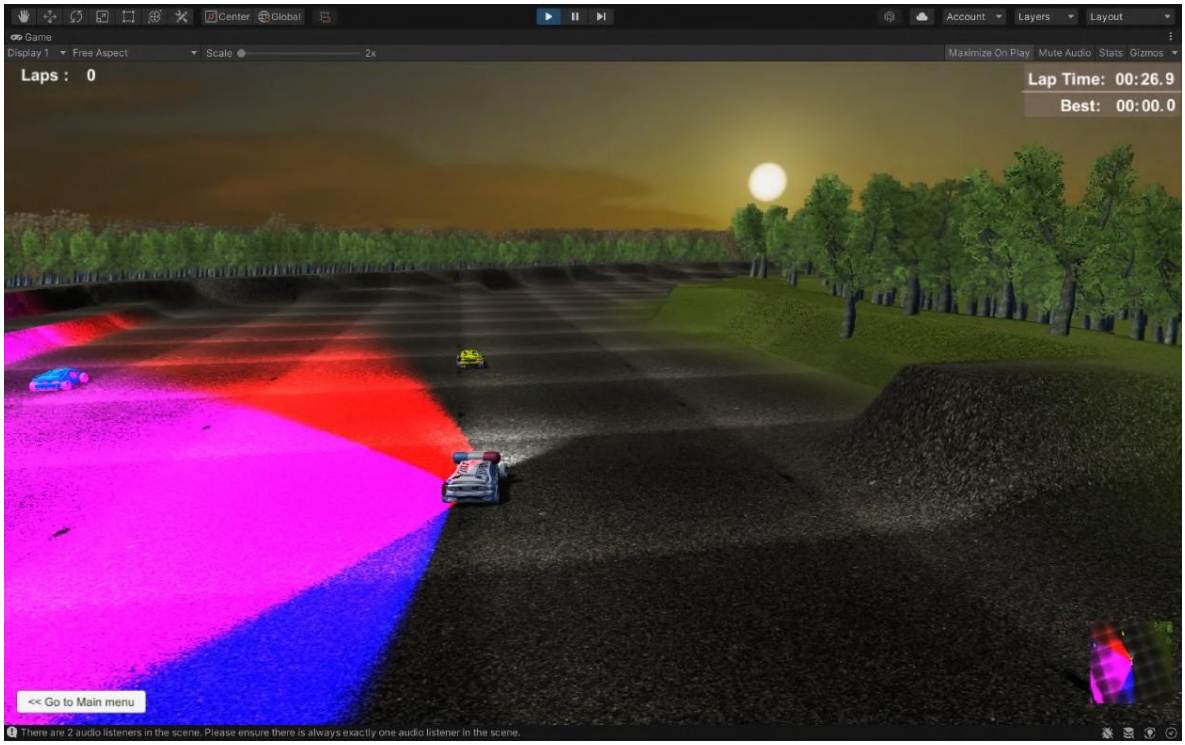
- Selection of the first track by the player.



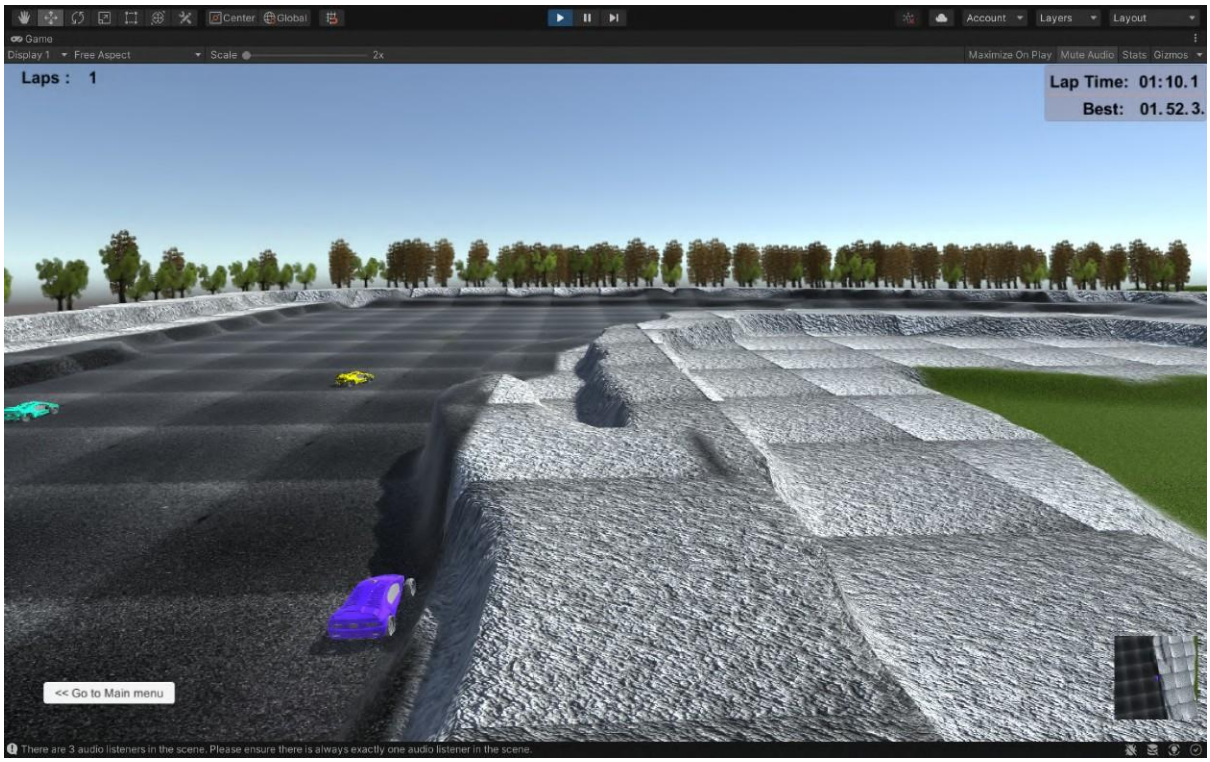
❖ Scene of the gameplay before the start of the game.



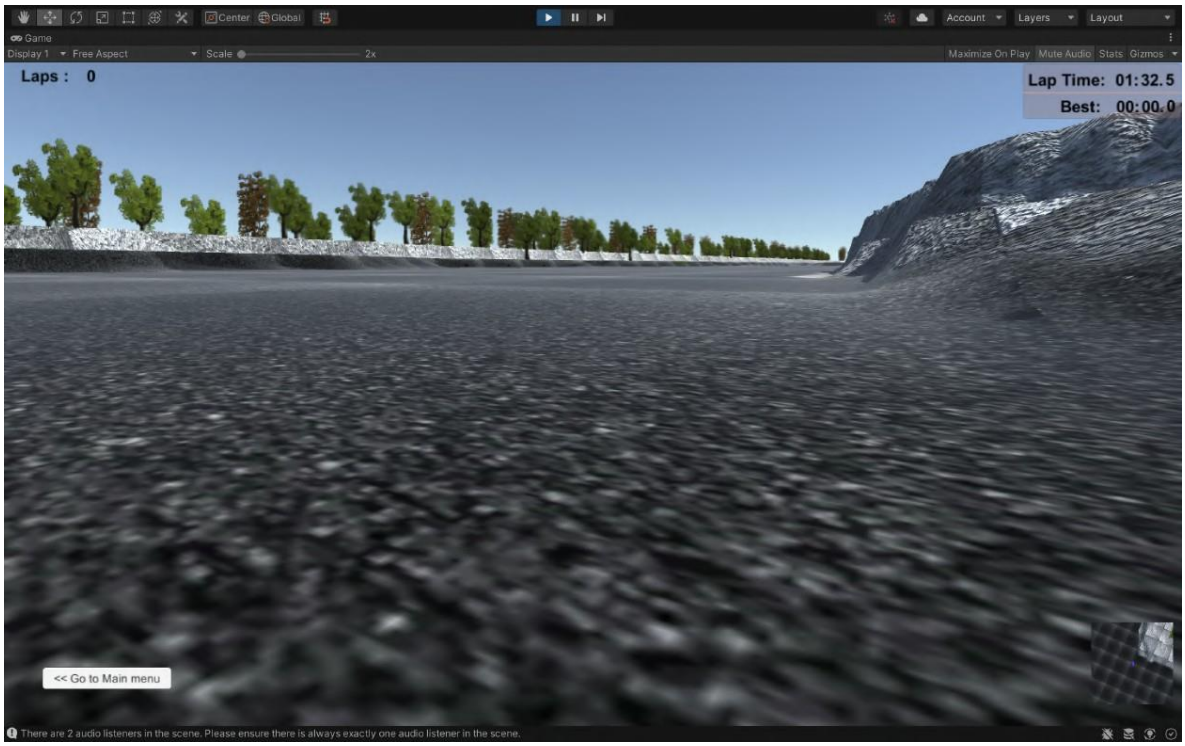
❖ Display of the lap time and display of the best time display.



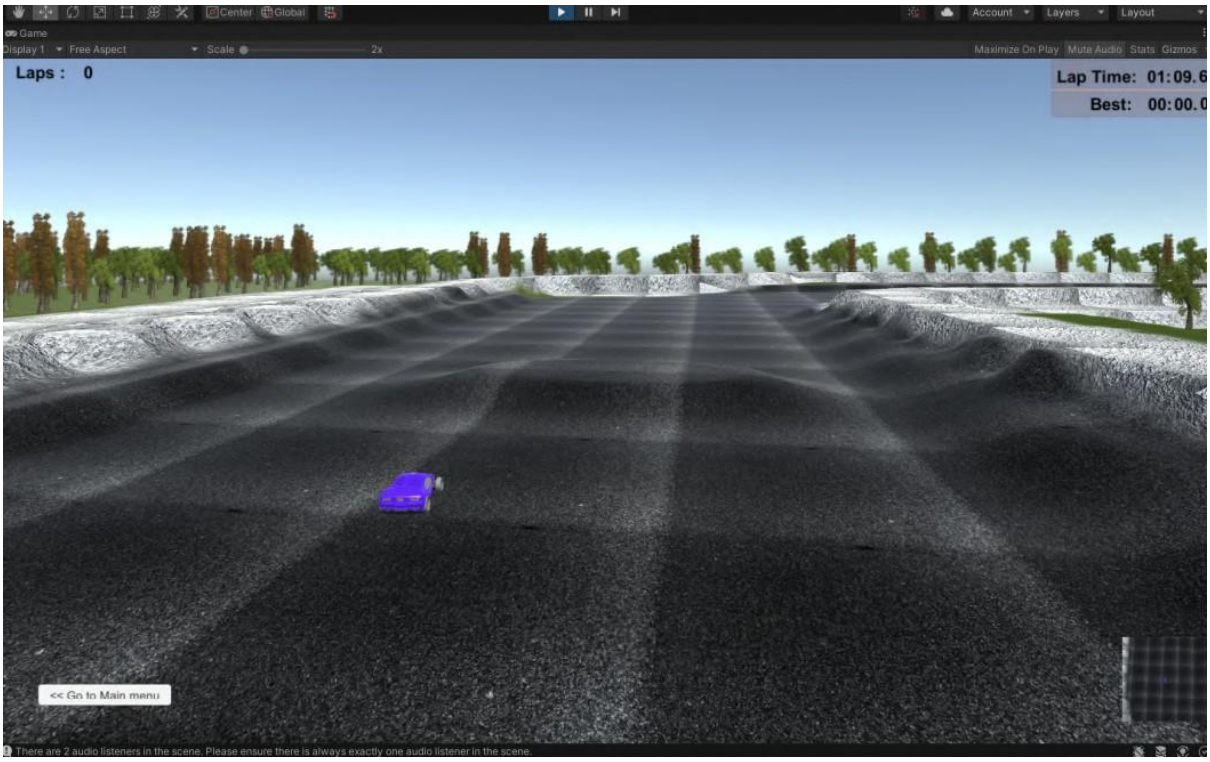
❖ Display of the car in the sidetrack user for the better view of the user.



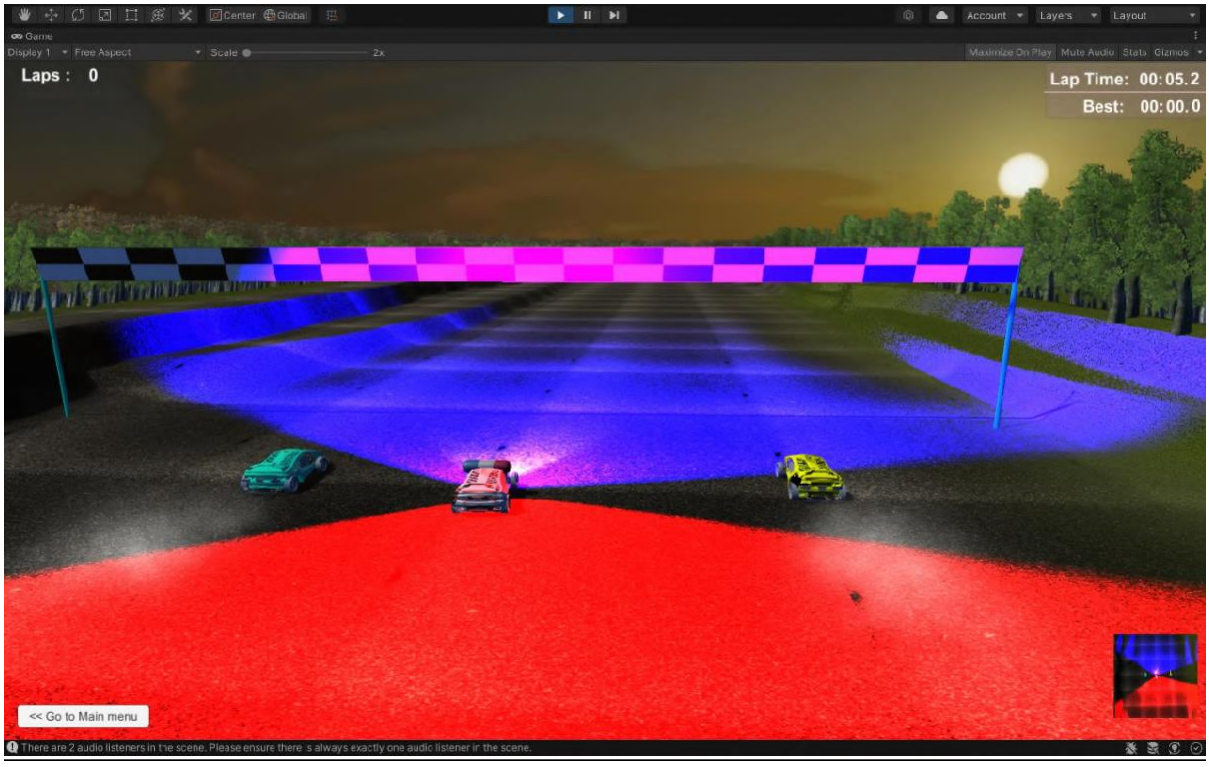
Screenshot of the top view of the given race course



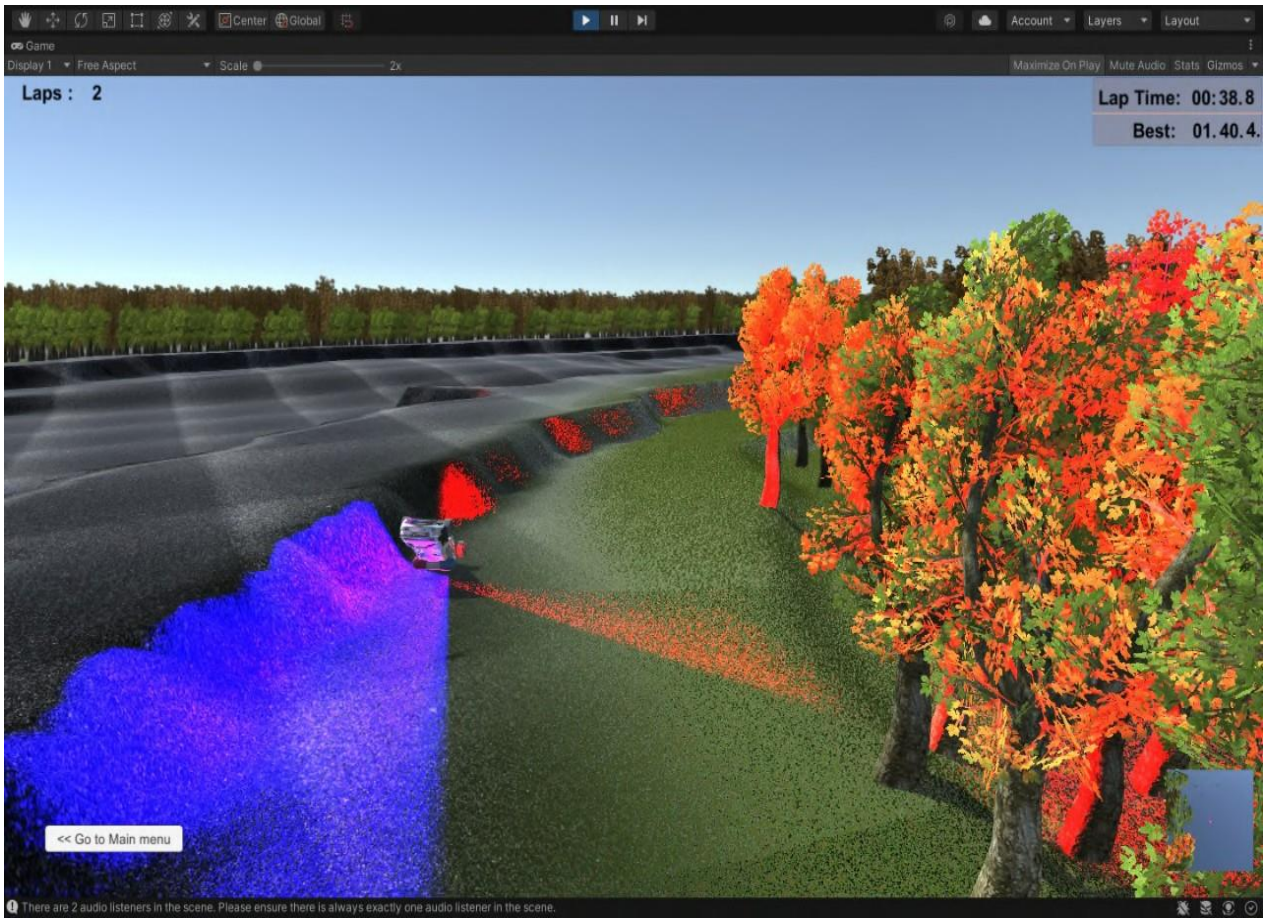
Screenshot of the alternate race track selected



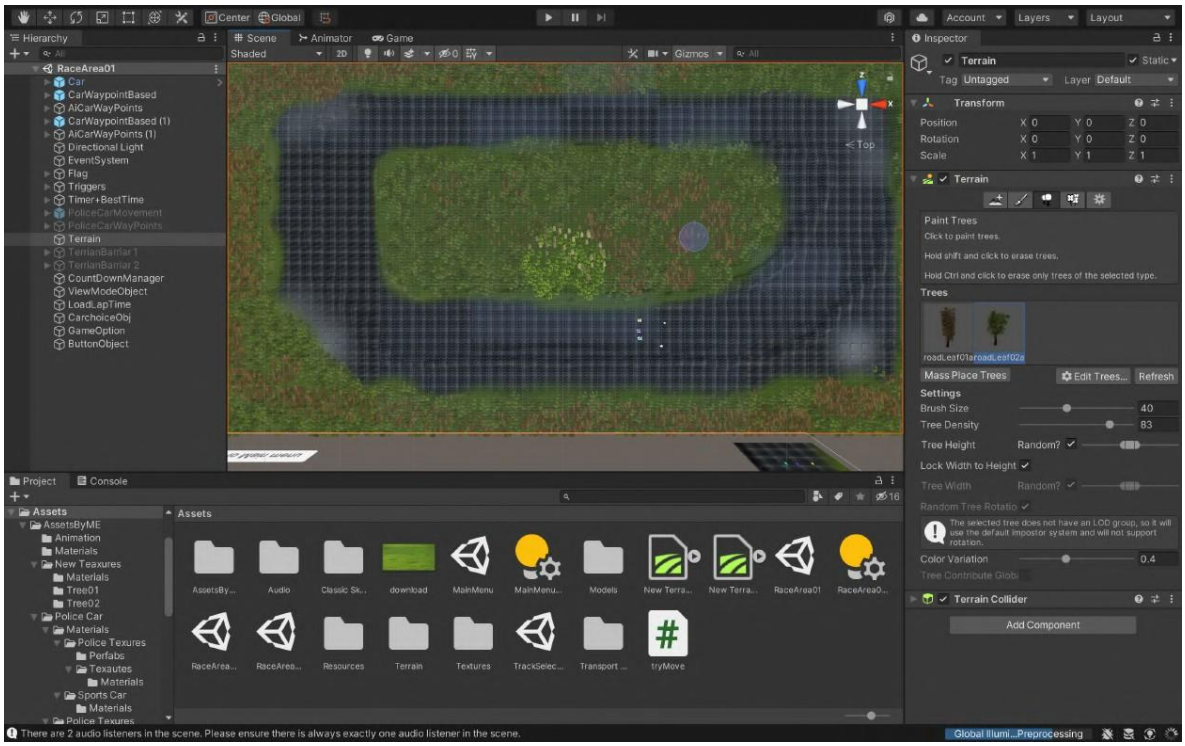
Screenshot of the selection of the police vehicle(alternate vehicle)



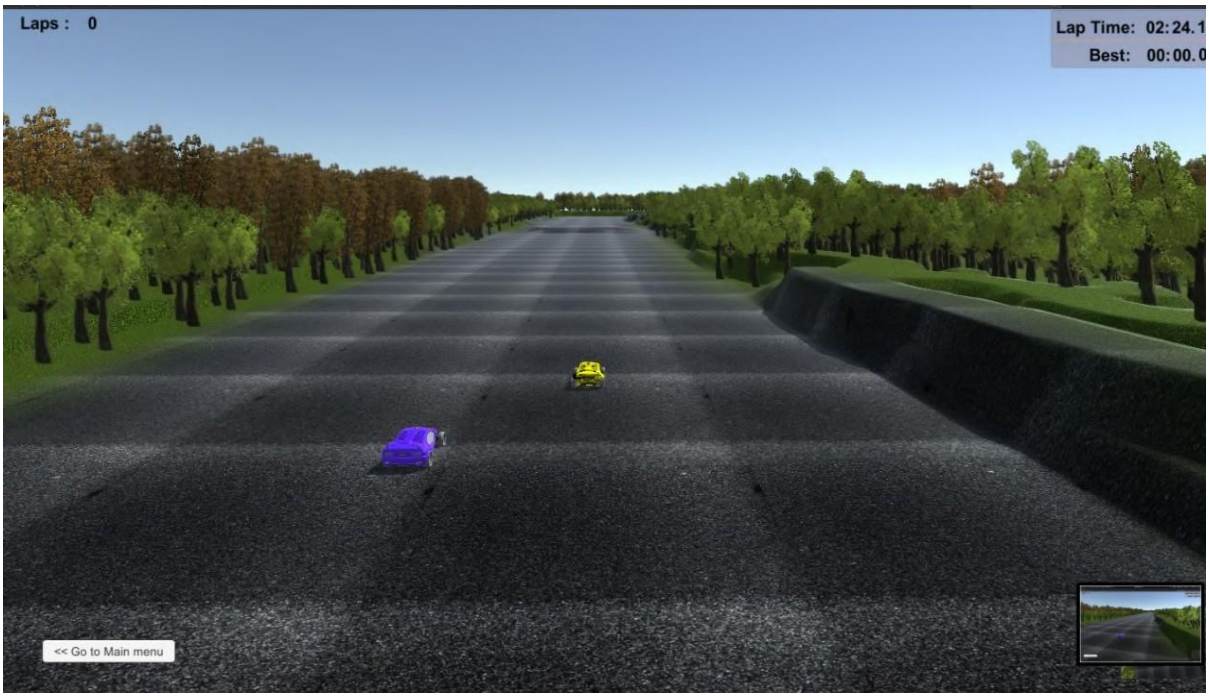
Screenshot of the collision between the vehicle and side-track



Screenshot of Track Design:



Screenshot of the completion of a level



RESULT:

The above mentioned game has been tested by over twenty users, who all reported that they thoroughly enjoyed the game.

◦ It was observed that there was approximately equal number of players who were able to win the race against the game- controlled cars.

◦ This suggests that not only did the game provide an entertaining gaming experience, it also provided a reasonably engaging and challenging game play.

◦ In general, it can be concluded that the Unity platform supported efficient development of the race car game.

◦ The Unity platform supports implementing the race car's search for a path on the racetrack with its components of the waypoint system, the physics engine, and vector calculation functions, all of which are not available if the implementation was done using traditional AI search techniques.

◦ With these Unity components, the developer was able to implement the race car's search for a path on the track with less effort and more efficiently, and the developed race car can successfully mimic human driving behavior.

CONCLUSION:

Development of the game system was made easier because of the implementation tools.

◦ The Unity game engine supports effective development of the game system of racer with its high-level abstraction programming tools and intuitive user interface.

◦ These features support developers in implementation of AI concepts so that they can focus on the game logic and ignore lower level development details such as graphics rendering and physics calculations.

◦ The combined tools of MonoDevelop and Unity support the implementation processes because MonoDevelop consists of auto correction features for many libraries and SDKs used in Unity.

◦ In the Unity platform enables the car racing system of the game to be more efficiently developed. If the Unity platform was not used, the racetrack would be mapped to a set of coordinates or nodes, which represent the 3D search space that covers the track.

◦ Then the path that the race car follows on the racetrack can be determined using either a blind or heuristic search algorithm, which identifies the nodes to be included in the path in the 3D space of the racetrack.

The Unity engine has built-in highlevel abstractions for trigger detection, which also reduced implementation efforts. Also Unity has drag-drop to keep simple to work.

◦ Thus it will easier for us to develop car road fighter using unity which gives effective results as point of view by user. The game environment and effects would be entertaining and statically defined for player.

REFFERENCES:

◦ [1] Ryan Henson Creighton. —Unity 3D Game Development by Example Beginner's Guide‖, p.42-45, 2010.

◦ [2] —Unity Game Development Essentials‖, by Will Goldstone.

◦ [3] James Sugrue. IOS 5 game development, August 2012.

◦ [4] Cheng Ming-zhi. Unity game development technology, p.75-79, June 2012.

◦ [5]XuanYu-Song.Unity3Dgamedevelopment,p.101-103,June2012.

◦ [6] Shi Xiao-ming,Michelle Menard. Unity game development practice, p.10-13, April 2012.

◦ [7] Zhao Ke-xin. Several optimization Suggestions of using Unity to rapidly develop high quality games, p.5-8, 2011.

Appendix:

```
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;
        }
    }

}

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;
    }
```

```
}
```

```
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 + ":";
        }

    }

}


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);
    }

}

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);
    }
}
```

```
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;
            }
        }
    }
}
```

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

using UnityEngine.UI;

public class Countdown : MonoBehaviour
{

    public GameObject CountDown;
    public AudioSource GetReady;
    public AudioSource GoAudio;
    public GameObject LapTimer;
   // public GameObject CarControls;

    // trun on sound sabek and la7ek in the start of the gaem
    public AudioSource LevelMusic;
    void Start()
    {
        StartCoroutine(CountStart());
    }


    IEnumerator CountStart()
    {
```

```
        //yield return new WaitForSeconds(0.5f);
        CountDown.GetComponent<Text>().text = "3";
        GetReady.Play();
        CountDown.SetActive(true);
        yield return new WaitForSeconds(0.5f);
        CountDown.SetActive(false);

        CountDown.GetComponent<Text>().text = "2";
        GetReady.Play();
        CountDown.SetActive(true);
        yield return new WaitForSeconds(0.5f);
        CountDown.SetActive(false);
        CountDown.GetComponent<Text>().text = "1";
        GetReady.Play();
        CountDown.SetActive(true);
        yield return new WaitForSeconds(0.2f);
        CountDown.SetActive(false);
        GoAudio.Play();
        LevelMusic.Play();
        LapTimer.SetActive(true);

    }

}

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

public class CarChoice : MonoBehaviour {

    public GameObject RedBody;
    public GameObject BlueBody;
    public GameObject normalbody;
    public GameObject police;
    public int CarImport;
    void Start () {
        CarImport = GlobalCar.CarTyp;
        if(CarImport == 1)
        {
            normalbody.SetActive(false);
            police.SetActive(false);
            RedBody.SetActive(true);
        }
        if (CarImport == 2)
        {
            police.SetActive(false);
            normalbody.SetActive(false);
            RedBody.SetActive(false);
```

```
            BlueBody.SetActive(true);
        }
        if (CarImport == 3)
        {
            normalbody.SetActive(false);
            RedBody.SetActive(false);
            BlueBody.SetActive(false);
            police.SetActive(true);
        }
    }
}
```

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

public class CancelChange : MonoBehaviour {
    // to cnange Camera Perspective
    public GameObject NoramlCam;
    public GameObject FarCam;
    public GameObject FPCam;
    public int CamMode;

    // Update is called once per frame
    void Update () {
        // if the user click on space the Perspective would be
changed
        if (Input.GetKey(KeyCode.Space))
        {
            if(CamMode == 3)
            {
                CamMode = 0;
            }
            else
            {
                CamMode += 1;
            } StartCoroutine(ModeChange());

        }
    }
    IEnumerator ModeChange()
    {
        yield return new WaitForSeconds(0.01f);
        if(CamMode == 0)
        {
            NoramlCam.SetActive(true);
            FPCam.SetActive(false);
            FarCam.SetActive(true);
```

```
        }
        if (CamMode == 1)
        {
            FarCam.SetActive(true);
            NoramlCam.SetActive(false);
            FPCam.SetActive(false);
        }
        if (CamMode == 2)
        {
            FPCam.SetActive(true);
            FarCam.SetActive(false);
            NoramlCam.SetActive(false);
        }
    }
}
```

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

public class CameraMovement : MonoBehaviour {

    public GameObject TheCar;
    public float carx;
    public float cary;
    public float carz;

    void Update () {
        carx = TheCar.transform.eulerAngles.x;
        cary = TheCar.transform.eulerAngles.y;
        carz = TheCar.transform.eulerAngles.z;

        transform.eulerAngles = new Vector3(carx - carx, cary,
carz-carz);
    }
}
```

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

public class ButtonOption : MonoBehaviour {
    public void playGame()
    {
        SceneManager.LoadScene(2);
    }
    public void TrackSelect()
```

```
    {
        SceneManager.LoadScene(1);
    }
    public void MainMenu()
    {
        SceneManager.LoadScene(0);
    }
    public void quitGame()
    {
        // if (Input.GetKey("escape"))
        Application.Quit();

    }

    ///
    public void track01()
    {
        SceneManager.LoadScene(2);
    }
    public void track02()
    {
        SceneManager.LoadScene(4);
    }
}
```