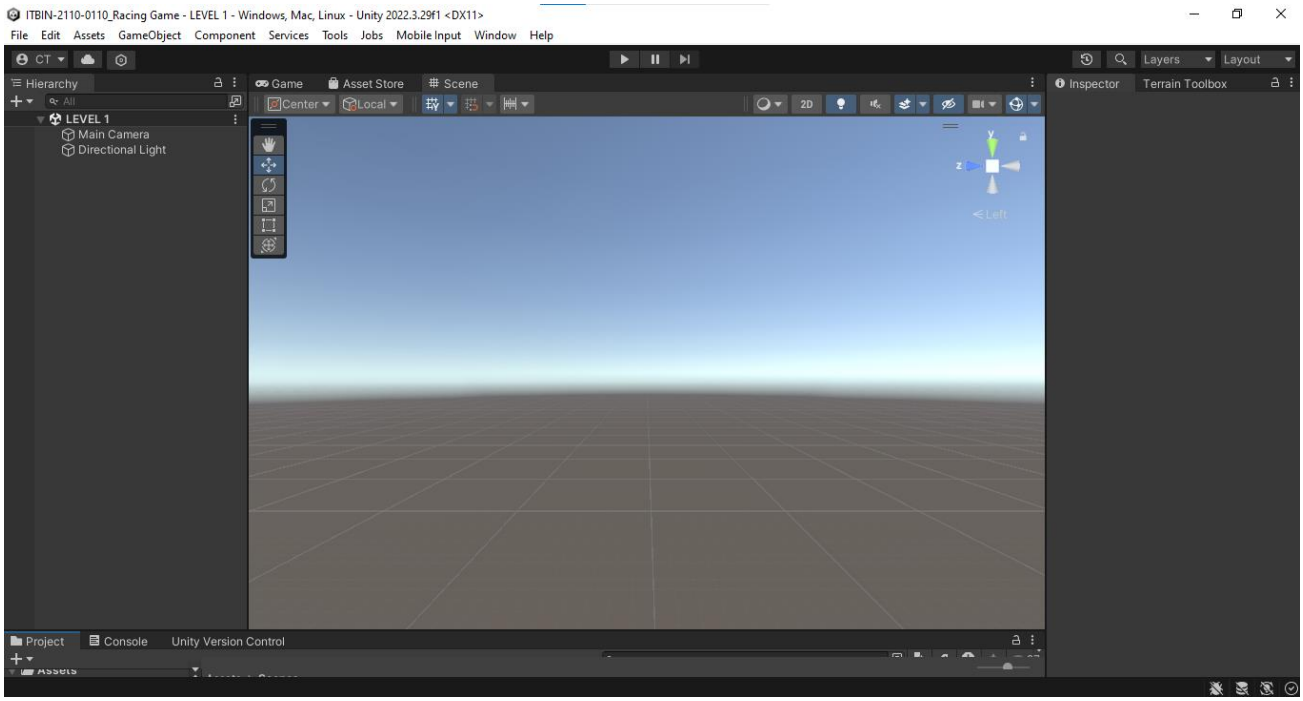
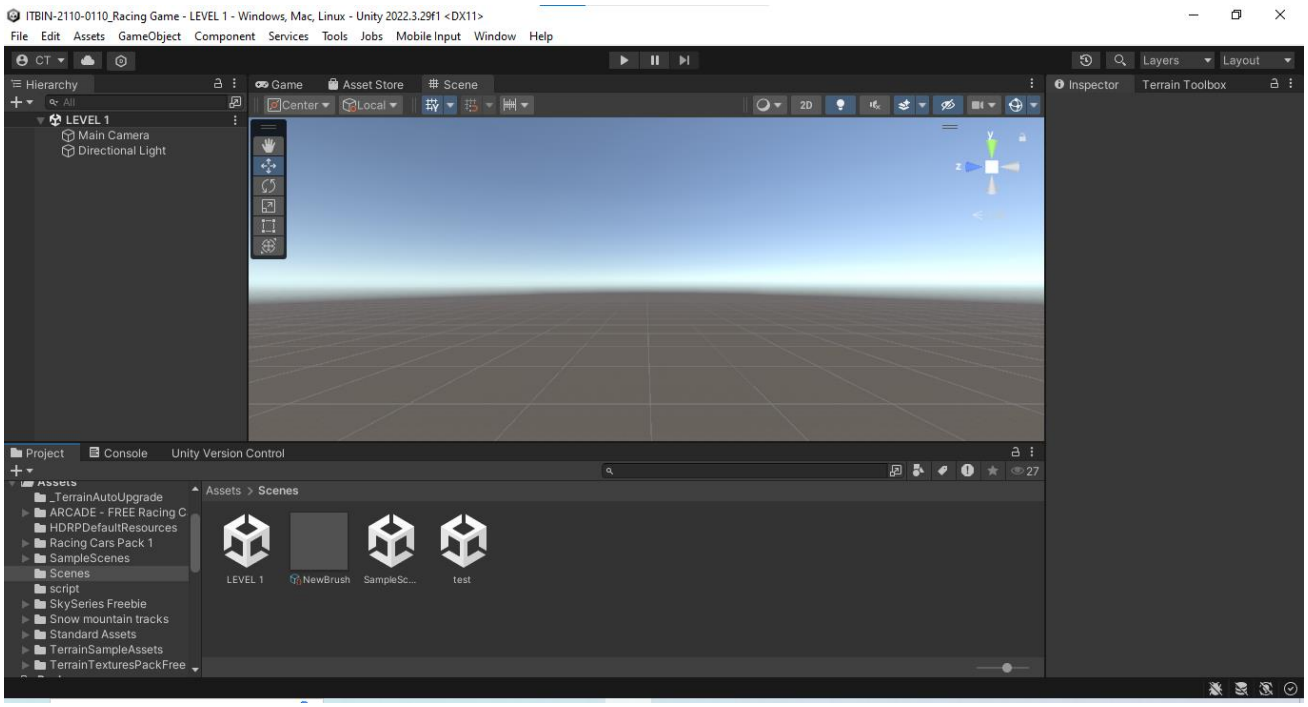
Racing Game

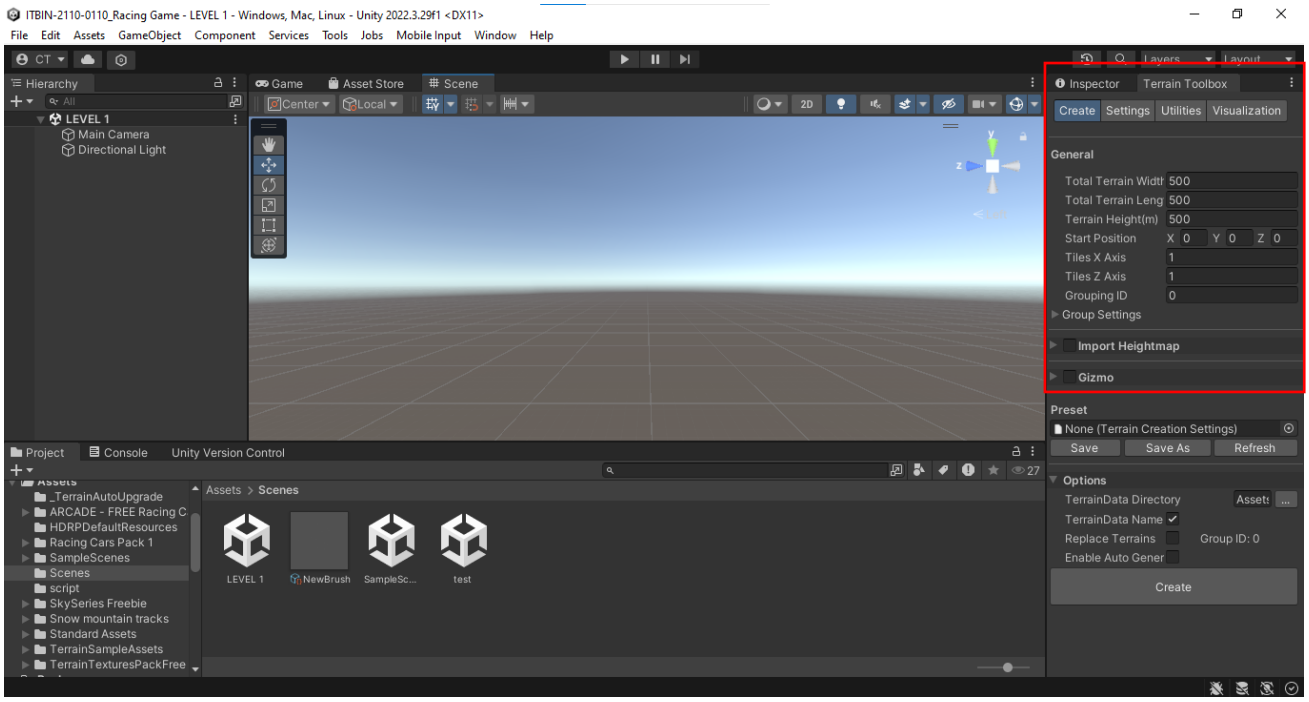
❖ Firstly, I open the UNITY to make a game.

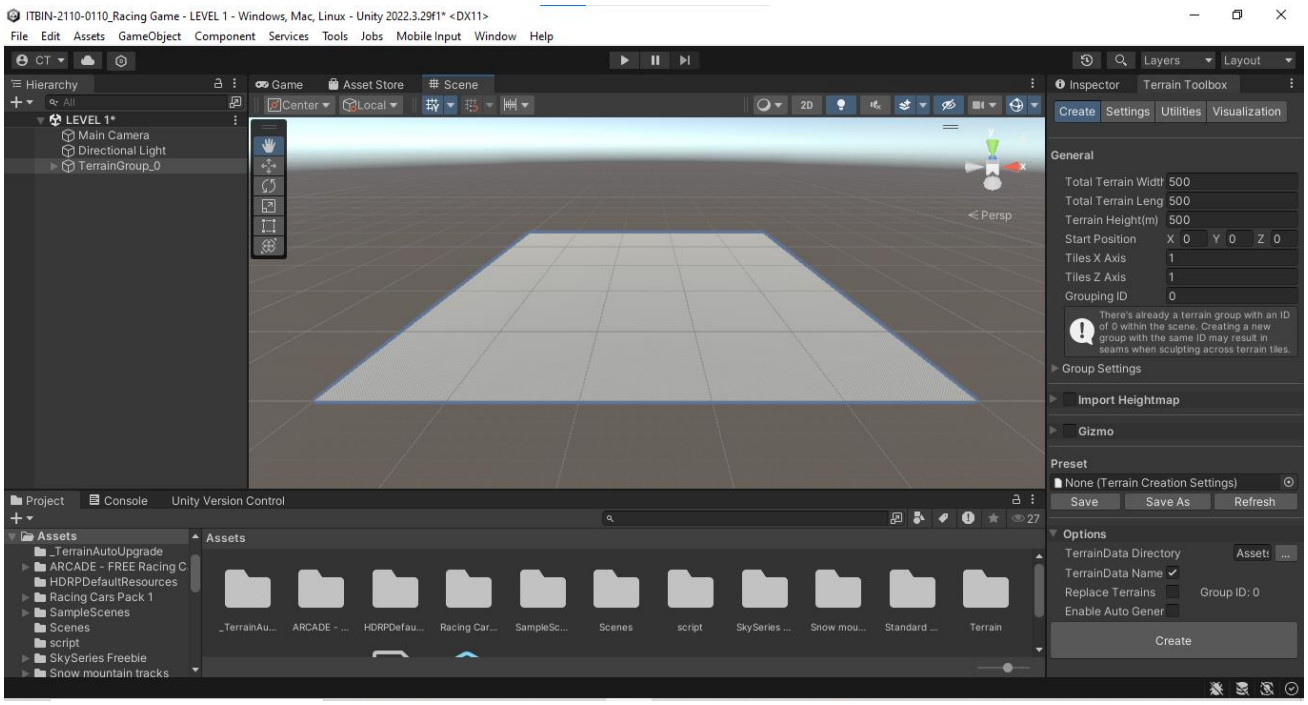


❖ Next, I create a new **Scene,** called **LEVEL 1**

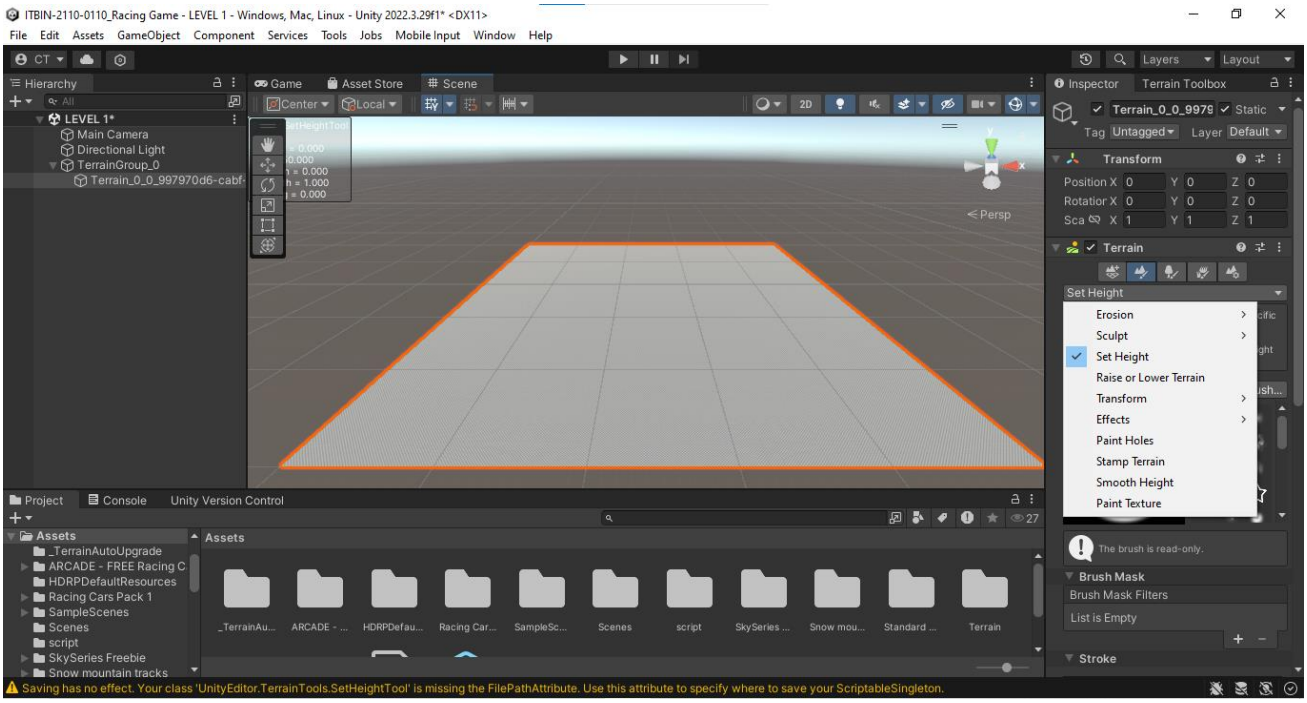


❖ Then I add the Terrain Toolbox to my project, to create my own Racing Track **(Game Environment)** and I create a new Terrain. Its size is 500m * 500m.

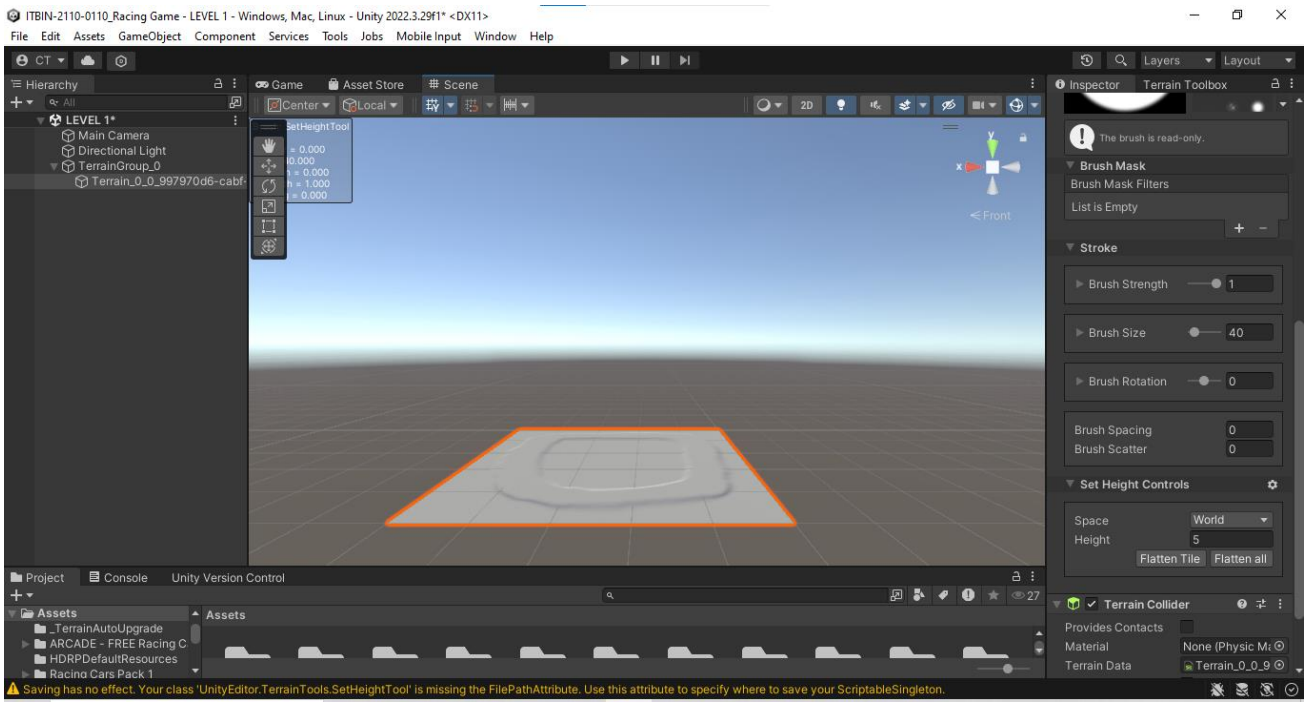




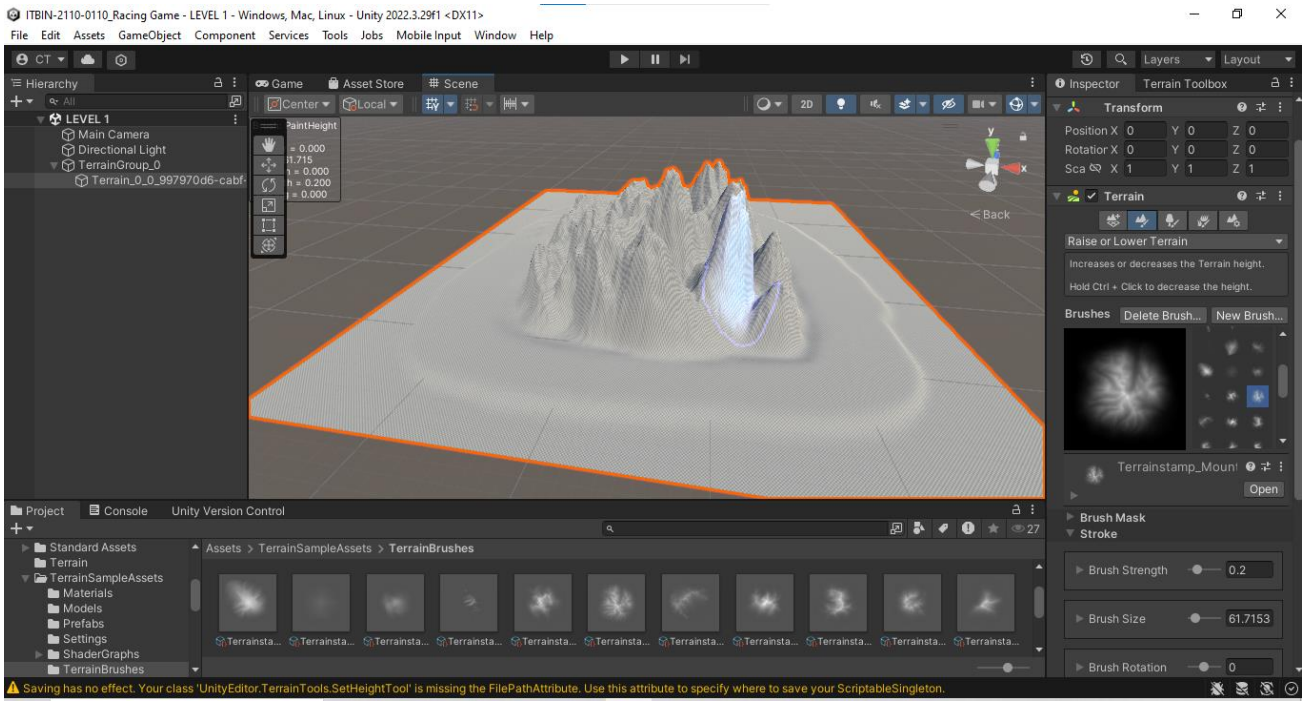
❖ After that I select the terrain and go to the inspector panel and choose the brush called **set height**. I used that brush to create my road.

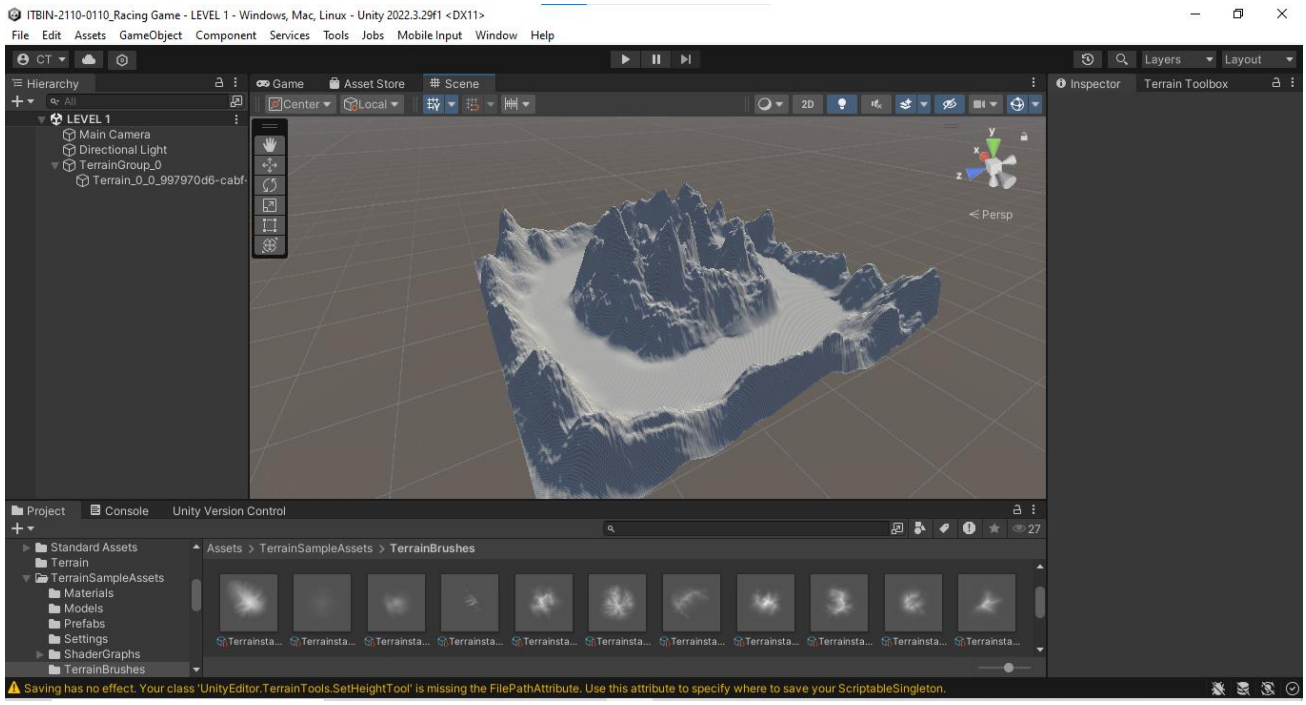


❖ By using this brush I created **Rode**.

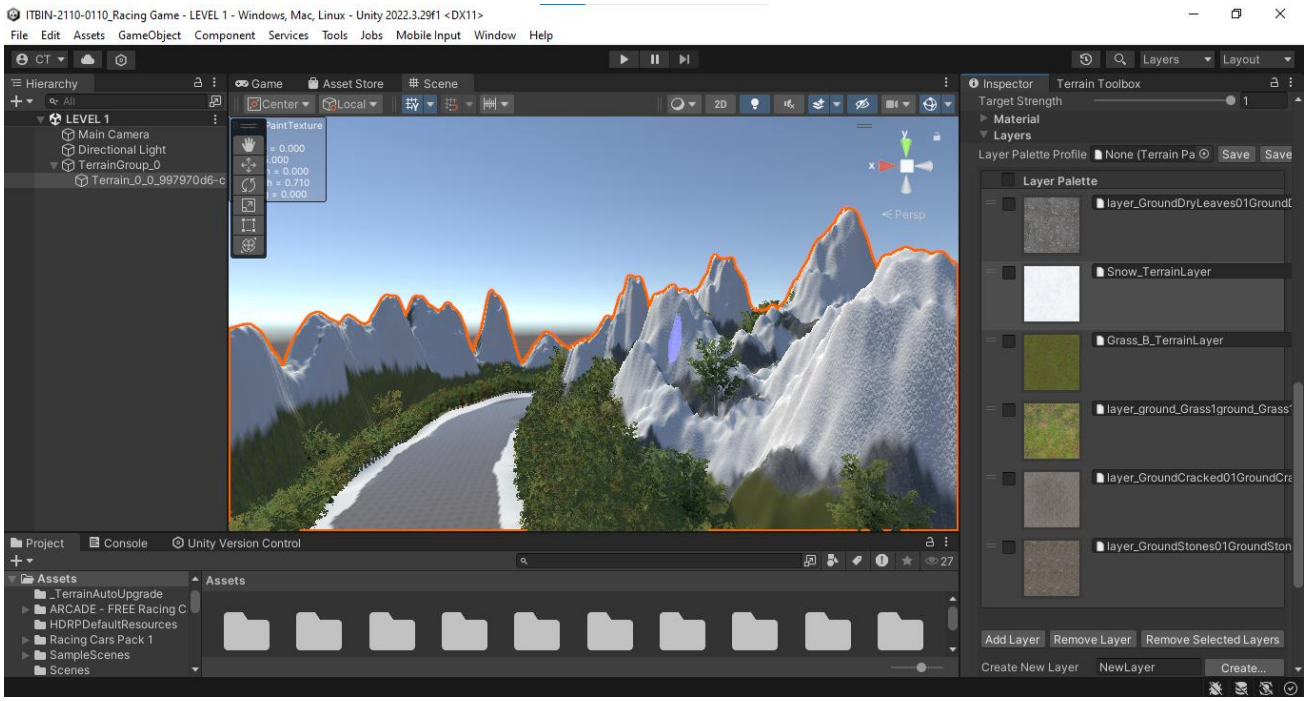


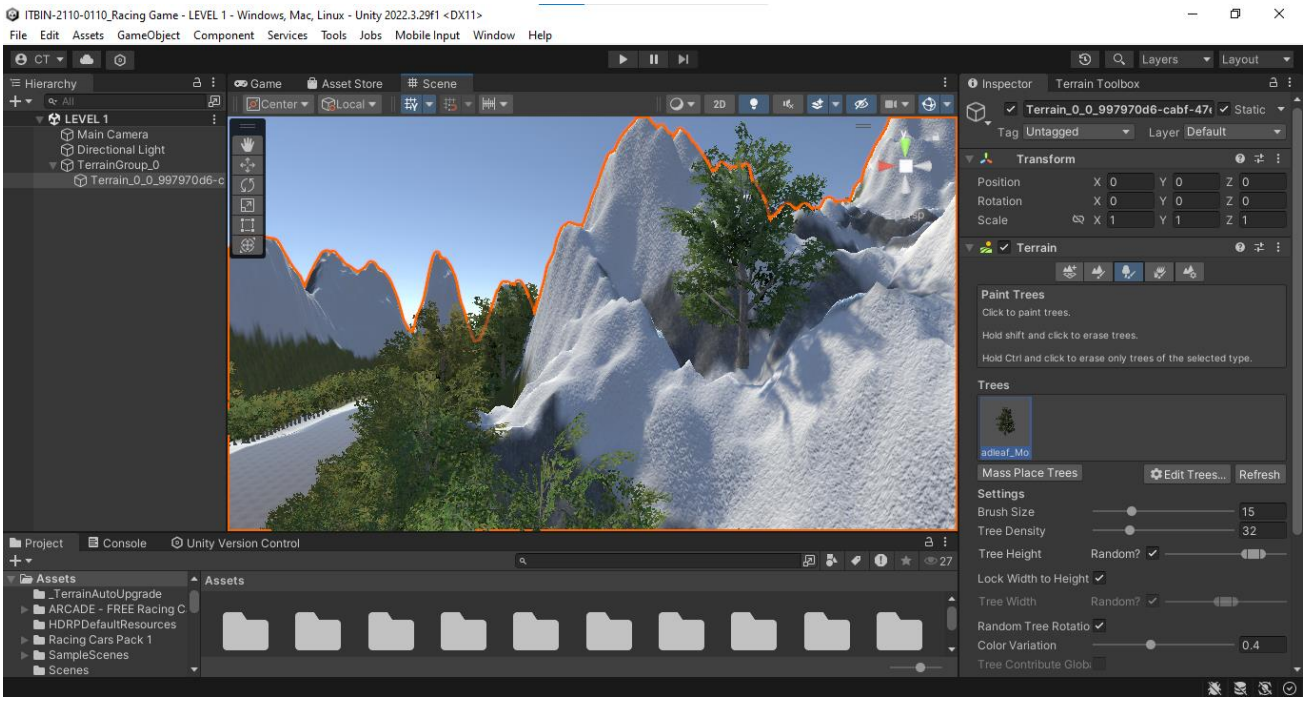
❖ After creating the road I create the mountains by using **Raise or Lower terrain brush.**

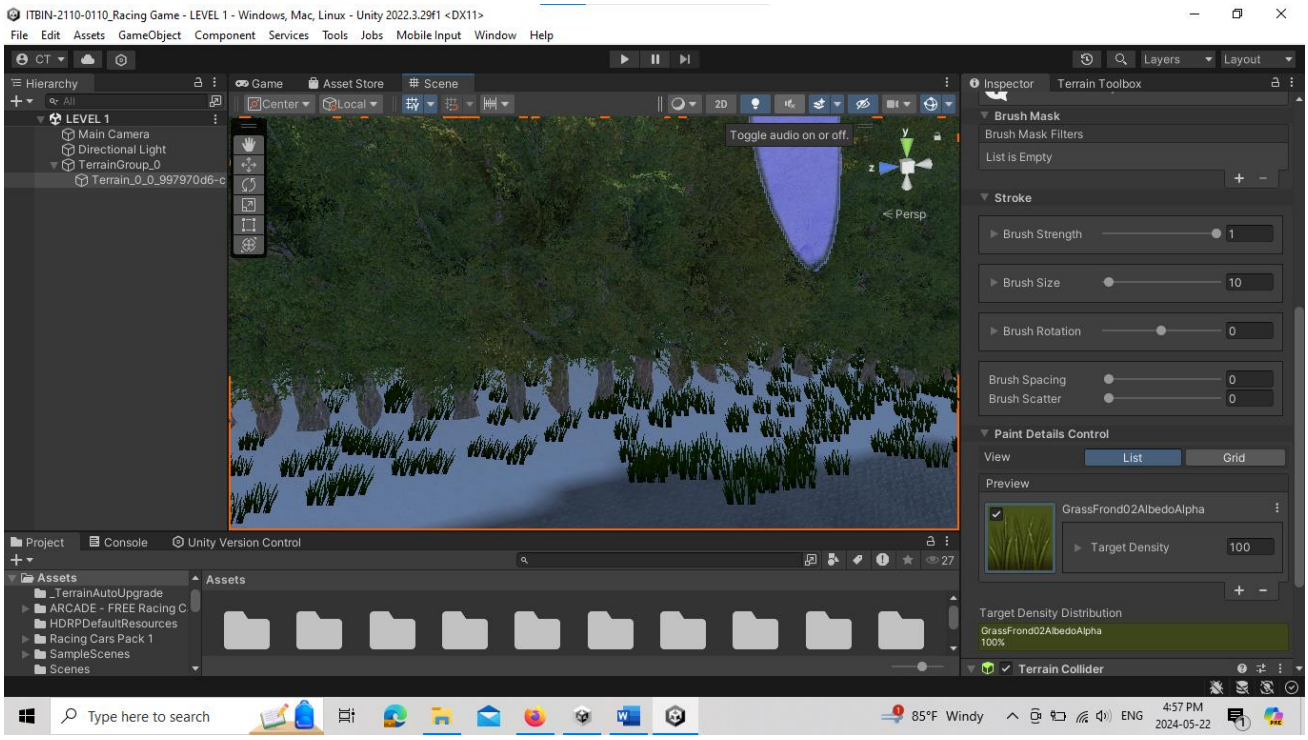




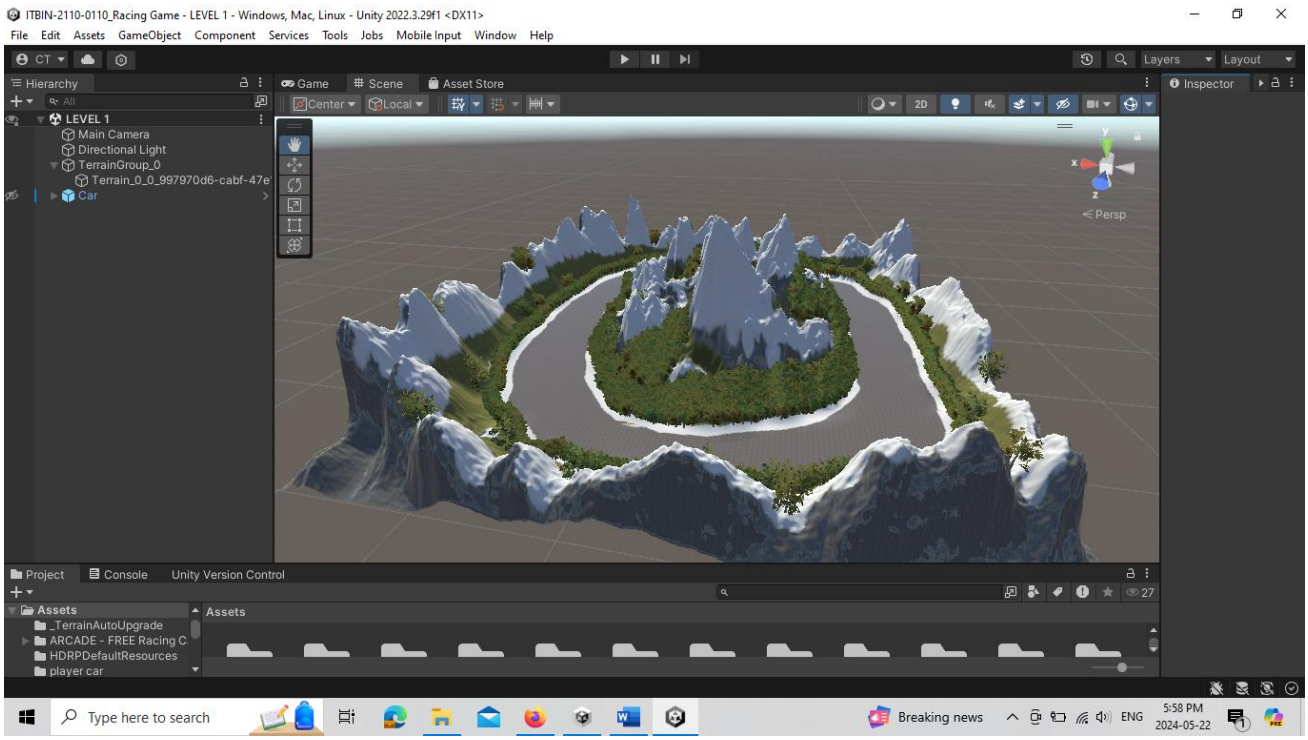
❖ After creating the mountains and road I added some textures like snow, grass, trees, etc.



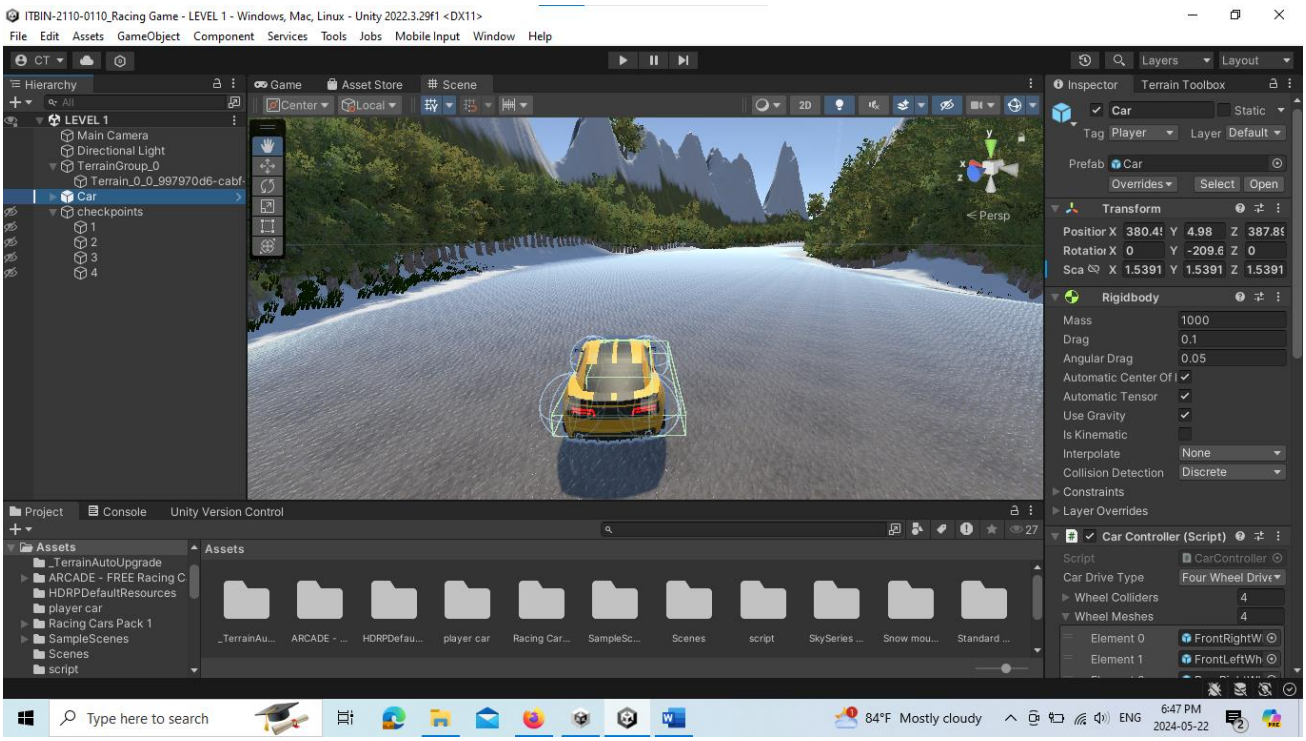




❖ you can see in below my final game track (game environment)



❖ After that I import my car into my racing track and I use some sound tracks for the car.



❖ After that I wrote code for the car. You can see in below **car controler script**.

```
using System;
using UnityEngine;

#pragma warning disable 649
namespace UnityStandardAssets.Vehicles.Car
{
    internal enum CarDriveType
    {
        FrontWheelDrive,
        RearWheelDrive,
        FourWheelDrive
    }

    internal enum SpeedType
    {
        MPH,
        KPH
    }

    public class CarController : MonoBehaviour
```

```
    {
        [SerializeField] private CarDriveType m_CarDriveType = CarDriveType.FourWheelDrive;
        [SerializeField] private WheelCollider[] m_WheelColliders = new WheelCollider[4];
        [SerializeField] private GameObject[] m_WheelMeshes = new GameObject[4];
        [SerializeField] private WheelEffects[] m_WheelEffects = new WheelEffects[4];
        [SerializeField] private Vector3 m_CentreOfMassOffset;
        [SerializeField] private float m_MaximumSteerAngle;
        [Range(0, 1)] [SerializeField] private float m_SteerHelper; // 0 is raw physics , 1 the car will grip in the direction it is facing
        [Range(0, 1)] [SerializeField] private float m_TractionControl; // 0 is no traction control, 1 is full interference
        [SerializeField] private float m_FullTorqueOverAllWheels;
        [SerializeField] private float m_ReverseTorque;
        [SerializeField] private float m_MaxHandbrakeTorque;
        [SerializeField] private float m_Downforce = 100f;
        [SerializeField] private SpeedType m_SpeedType;
        [SerializeField] private float m_Topspeed = 200;
        [SerializeField] private static int NoOfGears = 5;
        [SerializeField] private float m_RevRangeBoundary = 1f;
        [SerializeField] private float m_SlipLimit;
        [SerializeField] private float m_BrakeTorque;

        private Quaternion[] m_WheelMeshLocalRotations;
        private Vector3 m_Prevpos, m_Pos;
        private float m_SteerAngle;
        private int m_GearNum;
        private float m_GearFactor;
        private float m_OldRotation;
        private float m_CurrentTorque;
        private Rigidbody m_Rigidbody;
        private const float k_ReversingThreshold = 0.01f;

        public bool Skidding { get; private set; }
        public float BrakeInput { get; private set; }
        public float CurrentSteerAngle{ get { return m_SteerAngle; }}
        public float CurrentSpeed{ get { return m_Rigidbody.velocity.magnitude*2.23693629f; }}
        public float MaxSpeed{get { return m_Topspeed; }}
        public float Revs { get; private set; }
        public float AccelInput { get; private set; }

        // Use this for initialization
        private void Start()
        {
            m_WheelMeshLocalRotations = new Quaternion[4];
```

```
        for (int i = 0; i < 4; i++)
        {
            m_WheelMeshLocalRotations[i] = m_WheelMeshes[i].transform.localRotation;
        }
        m_WheelColliders[0].attachedRigidbody.centerOfMass = m_CentreOfMassOffset;

        m_MaxHandbrakeTorque = float.MaxValue;

        m_Rigidbody = GetComponent<Rigidbody>();
        m_CurrentTorque = m_FullTorqueOverAllWheels -
(m_TractionControl*m_FullTorqueOverAllWheels);
    }


    private void GearChanging()
    {
        float f = Mathf.Abs(CurrentSpeed/MaxSpeed);
        float upgearlimit = (1/(float) NoOfGears)*(m_GearNum + 1);
        float downgearlimit = (1/(float) NoOfGears)*m_GearNum;

        if (m_GearNum > 0 && f < downgearlimit)
        {
            m_GearNum--;
        }

        if (f > upgearlimit && (m_GearNum < (NoOfGears - 1)))
        {
            m_GearNum++;
        }
    }


    // simple function to add a curved bias towards 1 for a value in the 0-1 range
    private static float CurveFactor(float factor)
    {
        return 1 - (1 - factor)*(1 - factor);
    }


    // unclamped version of Lerp, to allow value to exceed the from-to range
    private static float ULerp(float from, float to, float value)
    {
        return (1.0f - value)*from + value*to;
    }
```

```
    private void CalculateGearFactor()
    {
        float f = (1/(float) NoOfGears);
        // gear factor is a normalised representation of the current speed within the current gear's range of speeds.
        // We smooth towards the 'target' gear factor, so that revs don't instantly snap up or down when changing gear.
        var targetGearFactor = Mathf.InverseLerp(f*m_GearNum, f*(m_GearNum + 1), Mathf.Abs(CurrentSpeed/MaxSpeed));
        m_GearFactor = Mathf.Lerp(m_GearFactor, targetGearFactor, Time.deltaTime*5f);
    }


    private void CalculateRevs()
    {
        // calculate engine revs (for display / sound)
        // (this is done in retrospect - revs are not used in force/power calculations)
        CalculateGearFactor();
        var gearNumFactor = m_GearNum/(float) NoOfGears;
        var revsRangeMin = ULerp(0f, m_RevRangeBoundary, CurveFactor(gearNumFactor));
        var revsRangeMax = ULerp(m_RevRangeBoundary, 1f, gearNumFactor);
        Revs = ULerp(revsRangeMin, revsRangeMax, m_GearFactor);
    }


    public void Move(float steering, float accel, float footbrake, float handbrake)
    {
        for (int i = 0; i < 4; i++)
        {
            Quaternion quat;
            Vector3 position;
            m_WheelColliders[i].GetWorldPose(out position, out quat);
            m_WheelMeshes[i].transform.position = position;
            m_WheelMeshes[i].transform.rotation = quat;
        }

        //clamp input values
        steering = Mathf.Clamp(steering, -1, 1);
        AccelInput = accel = Mathf.Clamp(accel, 0, 1);
        BrakeInput = footbrake = -1*Mathf.Clamp(footbrake, -1, 0);
        handbrake = Mathf.Clamp(handbrake, 0, 1);

        //Set the steer on the front wheels.
        //Assuming that wheels 0 and 1 are the front wheels.
        m_SteerAngle = steering*m_MaximumSteerAngle;
```

```
        m_WheelColliders[0].steerAngle = m_SteerAngle;
        m_WheelColliders[1].steerAngle = m_SteerAngle;

        SteerHelper();
        ApplyDrive(accel, footbrake);
        CapSpeed();

        //Set the handbrake.
        //Assuming that wheels 2 and 3 are the rear wheels.
        if (handbrake > 0f)
        {
            var hbTorque = handbrake*m_MaxHandbrakeTorque;
            m_WheelColliders[2].brakeTorque = hbTorque;
            m_WheelColliders[3].brakeTorque = hbTorque;
        }


        CalculateRevs();
        GearChanging();

        AddDownForce();
        CheckForWheelSpin();
        TractionControl();
    }


    private void CapSpeed()
    {
        float speed = m_Rigidbody.velocity.magnitude;
        switch (m_SpeedType)
        {
            case SpeedType.MPH:

                speed *= 2.23693629f;
                if (speed > m_Topspeed)
                    m_Rigidbody.velocity = (m_Topspeed/2.23693629f) *
m_Rigidbody.velocity.normalized;
                break;

            case SpeedType.KPH:
                speed *= 3.6f;
                if (speed > m_Topspeed)
                    m_Rigidbody.velocity = (m_Topspeed/3.6f) *
m_Rigidbody.velocity.normalized;
                break;
        }
```

```
    }


    private void ApplyDrive(float accel, float footbrake)
    {

        float thrustTorque;
        switch (m_CarDriveType)
        {
            case CarDriveType.FourWheelDrive:
                thrustTorque = accel * (m_CurrentTorque / 4f);
                for (int i = 0; i < 4; i++)
                {
                    m_WheelColliders[i].motorTorque = thrustTorque;
                }
                break;

            case CarDriveType.FrontWheelDrive:
                thrustTorque = accel * (m_CurrentTorque / 2f);
                m_WheelColliders[0].motorTorque = m_WheelColliders[1].motorTorque =
thrustTorque;
                break;

            case CarDriveType.RearWheelDrive:
                thrustTorque = accel * (m_CurrentTorque / 2f);
                m_WheelColliders[2].motorTorque = m_WheelColliders[3].motorTorque =
thrustTorque;
                break;

        }

        for (int i = 0; i < 4; i++)
        {
            if (CurrentSpeed > 5 && Vector3.Angle(transform.forward, m_Rigidbody.velocity)
< 50f)
            {
                m_WheelColliders[i].brakeTorque = m_BrakeTorque*footbrake;
            }
            else if (footbrake > 0)
            {
                m_WheelColliders[i].brakeTorque = 0f;
                m_WheelColliders[i].motorTorque = -m_ReverseTorque*footbrake;
            }
        }
    }
```

```
    private void SteerHelper()
    {
        for (int i = 0; i < 4; i++)
        {
            WheelHit wheelhit;
            m_WheelColliders[i].GetGroundHit(out wheelhit);
            if (wheelhit.normal == Vector3.zero)
                return; // wheels arent on the ground so dont realign the rigidbody velocity
        }

        // this if is needed to avoid gimbal lock problems that will make the car suddenly shift
direction
        if (Mathf.Abs(m_OldRotation - transform.eulerAngles.y) < 10f)
        {
            var turnadjust = (transform.eulerAngles.y - m_OldRotation) * m_SteerHelper;
            Quaternion velRotation = Quaternion.AngleAxis(turnadjust, Vector3.up);
            m_Rigidbody.velocity = velRotation * m_Rigidbody.velocity;
        }
        m_OldRotation = transform.eulerAngles.y;
    }


    // this is used to add more grip in relation to speed
    private void AddDownForce()
    {
        m_WheelColliders[0].attachedRigidbody.AddForce(-transform.up*m_Downforce*

m_WheelColliders[0].attachedRigidbody.velocity.magnitude);
    }


    // checks if the wheels are spinning and is so does three things
    // 1) emits particles
    // 2) plays tiure skidding sounds
    // 3) leaves skidmarks on the ground
    // these effects are controlled through the WheelEffects class
    private void CheckForWheelSpin()
    {
        // loop through all wheels
        for (int i = 0; i < 4; i++)
        {
            WheelHit wheelHit;
            m_WheelColliders[i].GetGroundHit(out wheelHit);

            // is the tire slipping above the given threshhold
```

```
            if (Mathf.Abs(wheelHit.forwardSlip) >= m_SlipLimit ||
Mathf.Abs(wheelHit.sidewaysSlip) >= m_SlipLimit)
            {
               m_WheelEffects[i].EmitTyreSmoke();

               // avoiding all four tires screeching at the same time
               // if they do it can lead to some strange audio artefacts
               if (!AnySkidSoundPlaying())
               {
                  m_WheelEffects[i].PlayAudio();
               }
               continue;
            }

            // if it wasnt slipping stop all the audio
            if (m_WheelEffects[i].PlayingAudio)
            {
               m_WheelEffects[i].StopAudio();
            }
            // end the trail generation
            m_WheelEffects[i].EndSkidTrail();
         }
      }

      // crude traction control that reduces the power to wheel if the car is wheel spinning too
much
      private void TractionControl()
      {
         WheelHit wheelHit;
         switch (m_CarDriveType)
         {
            case CarDriveType.FourWheelDrive:
               // loop through all wheels
               for (int i = 0; i < 4; i++)
               {
                  m_WheelColliders[i].GetGroundHit(out wheelHit);

                  AdjustTorque(wheelHit.forwardSlip);
               }
               break;

            case CarDriveType.RearWheelDrive:
               m_WheelColliders[2].GetGroundHit(out wheelHit);
               AdjustTorque(wheelHit.forwardSlip);

               m_WheelColliders[3].GetGroundHit(out wheelHit);
```

```
                AdjustTorque(wheelHit.forwardSlip);
                break;

            case CarDriveType.FrontWheelDrive:
                m_WheelColliders[0].GetGroundHit(out wheelHit);
                AdjustTorque(wheelHit.forwardSlip);

                m_WheelColliders[1].GetGroundHit(out wheelHit);
                AdjustTorque(wheelHit.forwardSlip);
                break;
        }
    }


    private void AdjustTorque(float forwardSlip)
    {
        if (forwardSlip >= m_SlipLimit && m_CurrentTorque >= 0)
        {
            m_CurrentTorque -= 10 * m_TractionControl;
        }
        else
        {
            m_CurrentTorque += 10 * m_TractionControl;
            if (m_CurrentTorque > m_FullTorqueOverAllWheels)
            {
                m_CurrentTorque = m_FullTorqueOverAllWheels;
            }
        }
    }


    private bool AnySkidSoundPlaying()
    {
        for (int i = 0; i < 4; i++)
        {
            if (m_WheelEffects[i].PlayingAudio)
            {
                return true;
            }
        }
        return false;
    }
  }
}
```

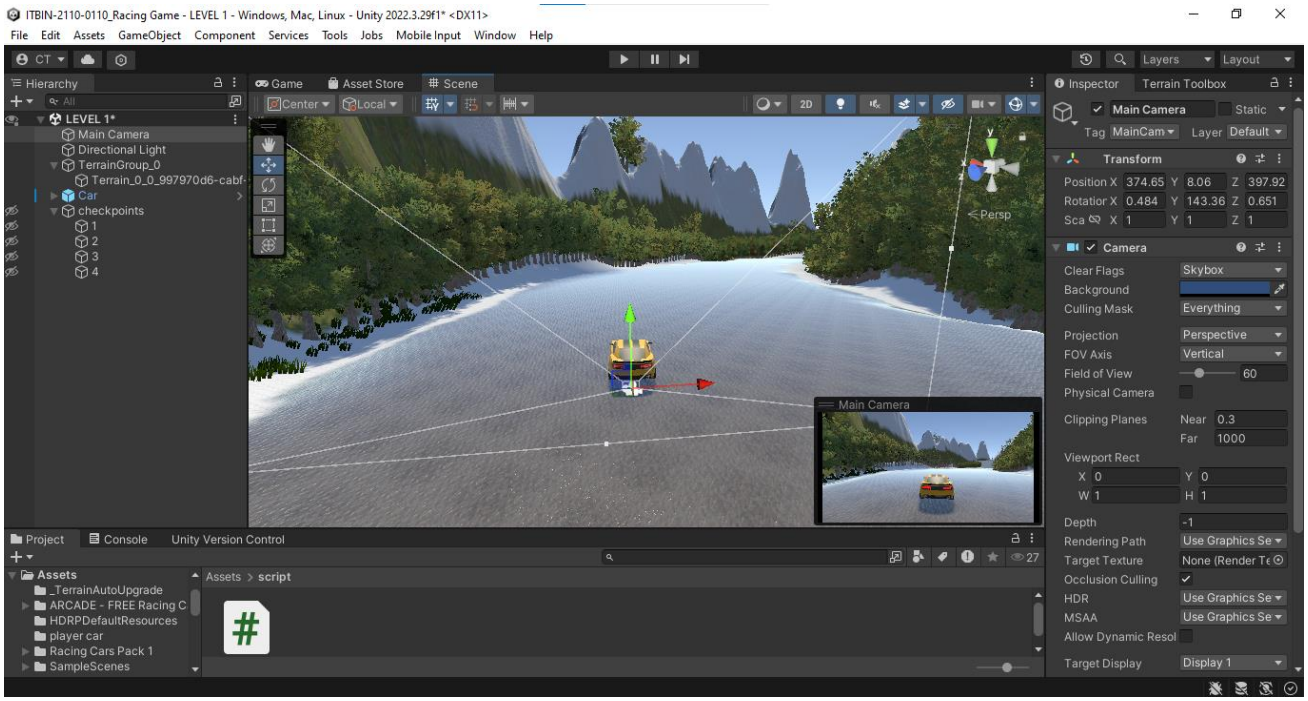
- In below you can see code of car user control

```
using System;
using UnityEngine;
using UnityStandardAssets.CrossPlatformInput;

namespace UnityStandardAssets.Vehicles.Car
{
   [RequireComponent(typeof (CarController))]
   public class CarUserControl : MonoBehaviour
   {
      private CarController m_Car; // the car controller we want to use


      private void Awake()
      {
         // get the car controller
         m_Car = GetComponent<CarController>();
      }


      private void FixedUpdate()
      {
         // pass the input to the car!
         float h = CrossPlatformInputManager.GetAxis("Horizontal");
         float v = CrossPlatformInputManager.GetAxis("Vertical");
#if !MOBILE_INPUT
         float handbrake = CrossPlatformInputManager.GetAxis("Jump");
         m_Car.Move(h, v, v, handbrake);
#else
         m_Car.Move(h, v, v, 0f);
#endif
      }
   }
}
```

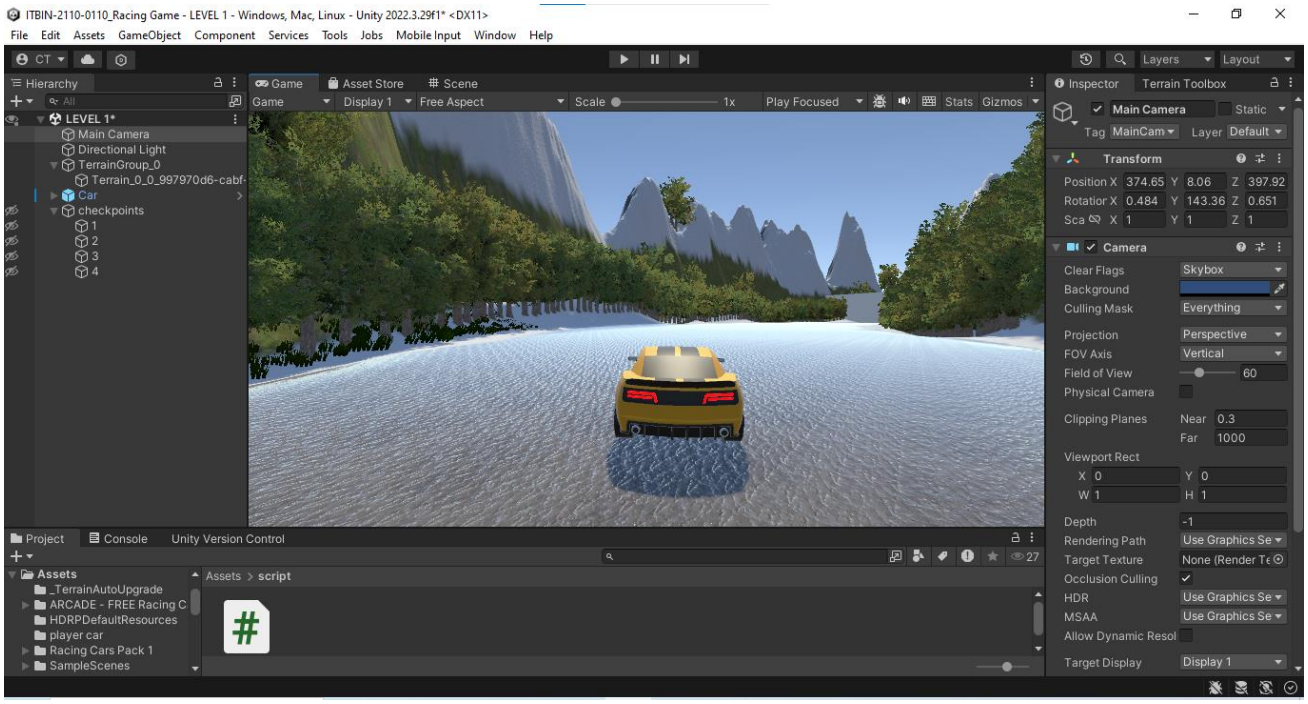
❖ After that I wrote code for my main camera. By using that code the main camera move with tha car. You can see in below

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

public class MainCameraMovement : MonoBehaviour
{
    public Transform  target;
    public float distance = 5.0f;
    public float height = 2.0f;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (target == null) return;
        transform.position = target.position;
        transform.position = transform.position - target.rotation * Vector3.forward * distance;
        transform.position = new Vector3(transform.position.x, transform.position.y + height, transform.position.z);
        transform.LookAt(target);


    }
}
```

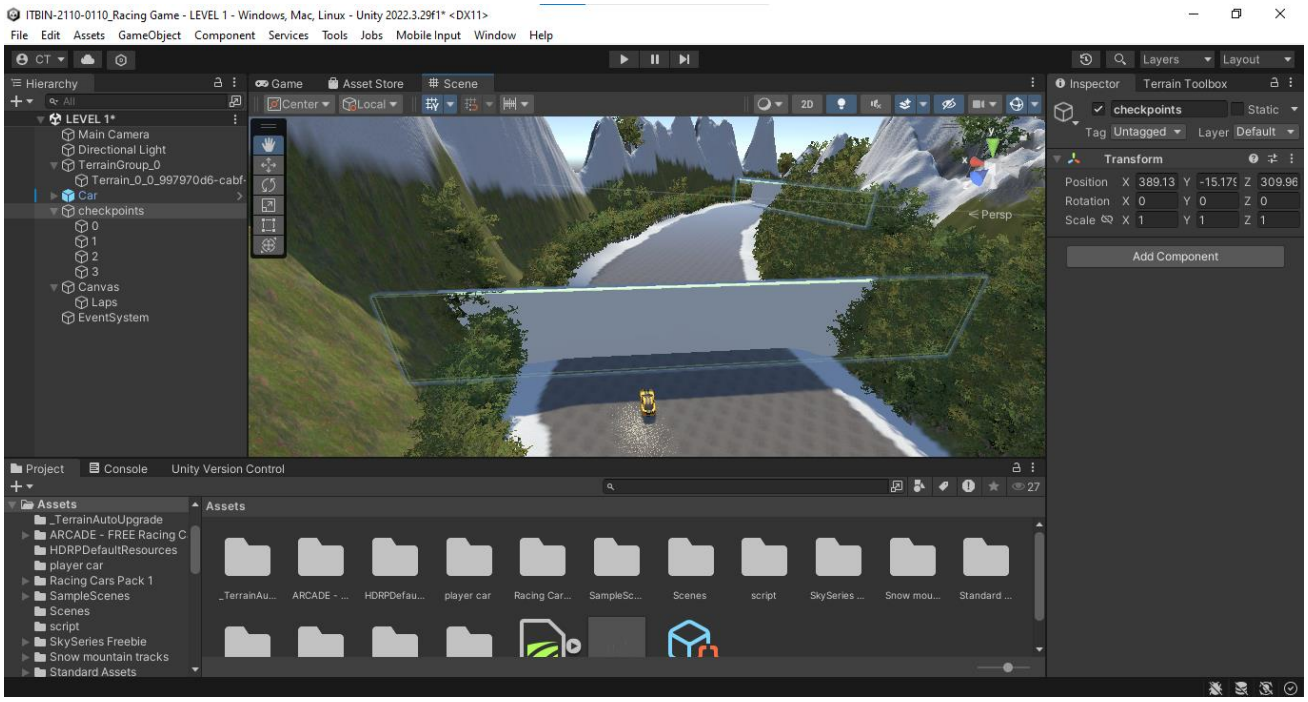
❖ You can see in below, my camera angale.

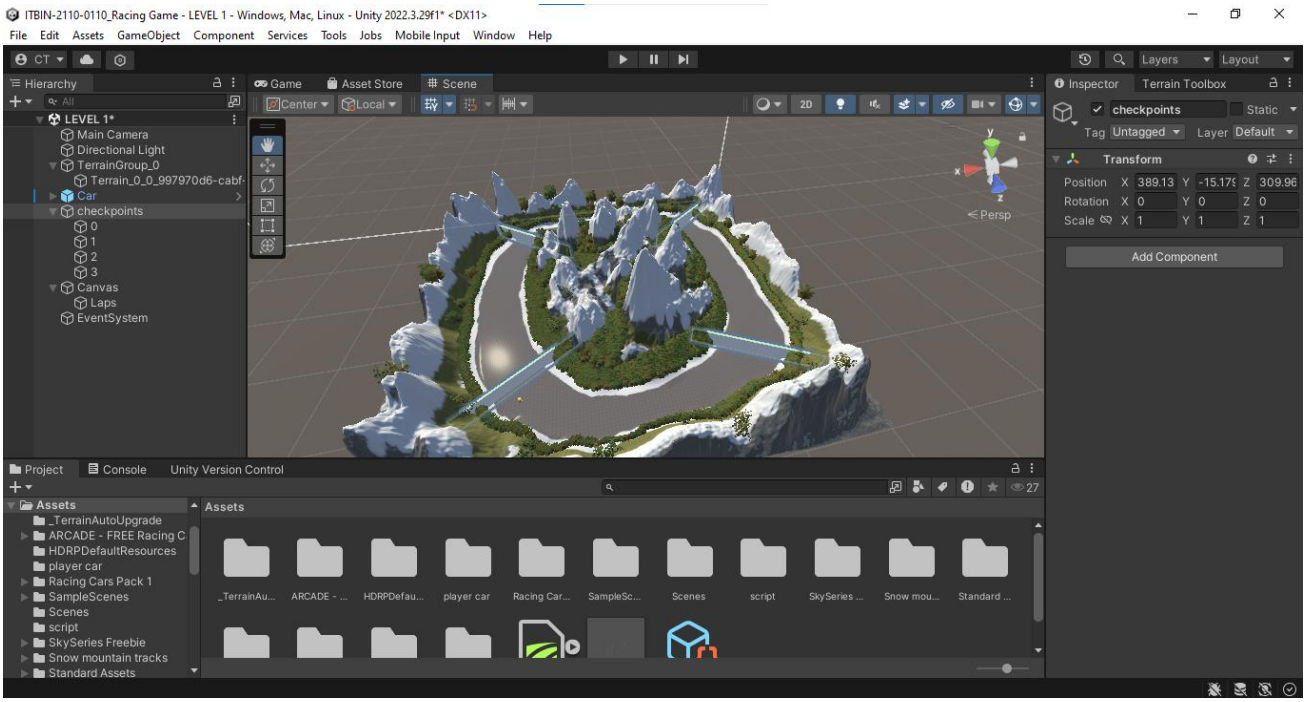


❖ Below image shows a game view to you.

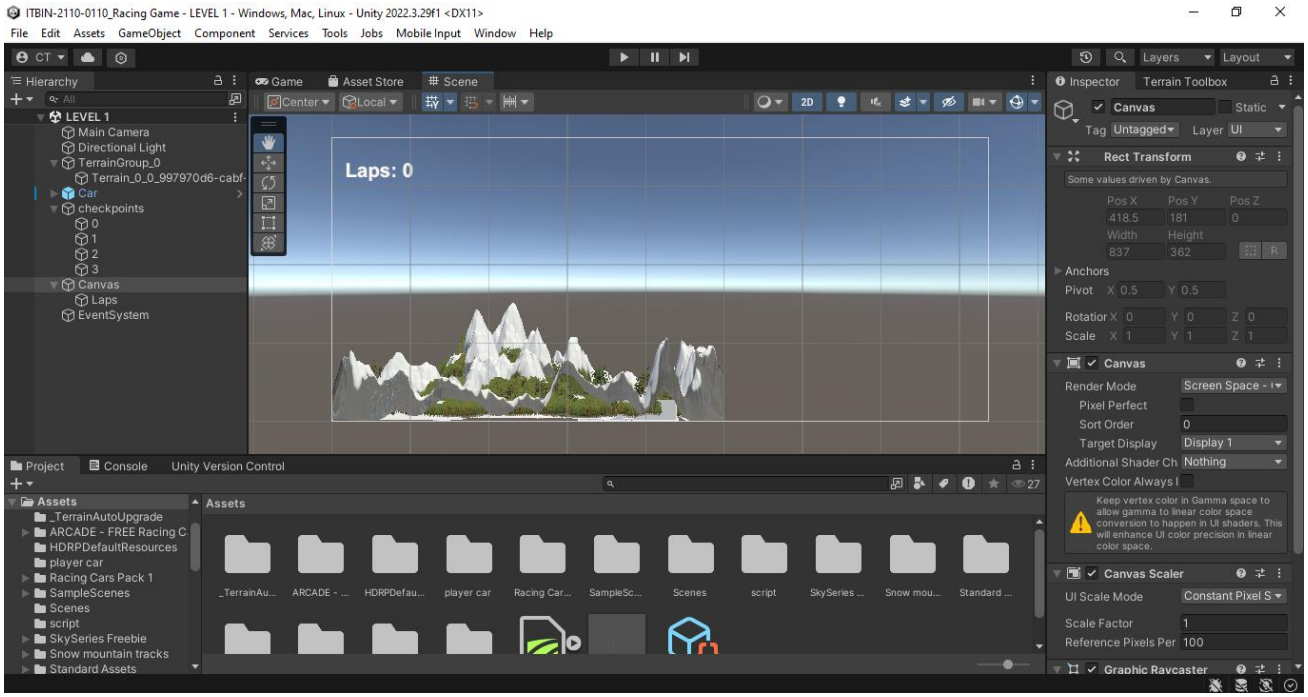


❖ After that I added four checkpoints. For checkpoints I used four cubes. I renamed those four cubes 0 to 3.





❖ After that I add UI text called **Laps.**



❖ After that I wrote code for count laps, by using four checkpoints. That code you can see in below.

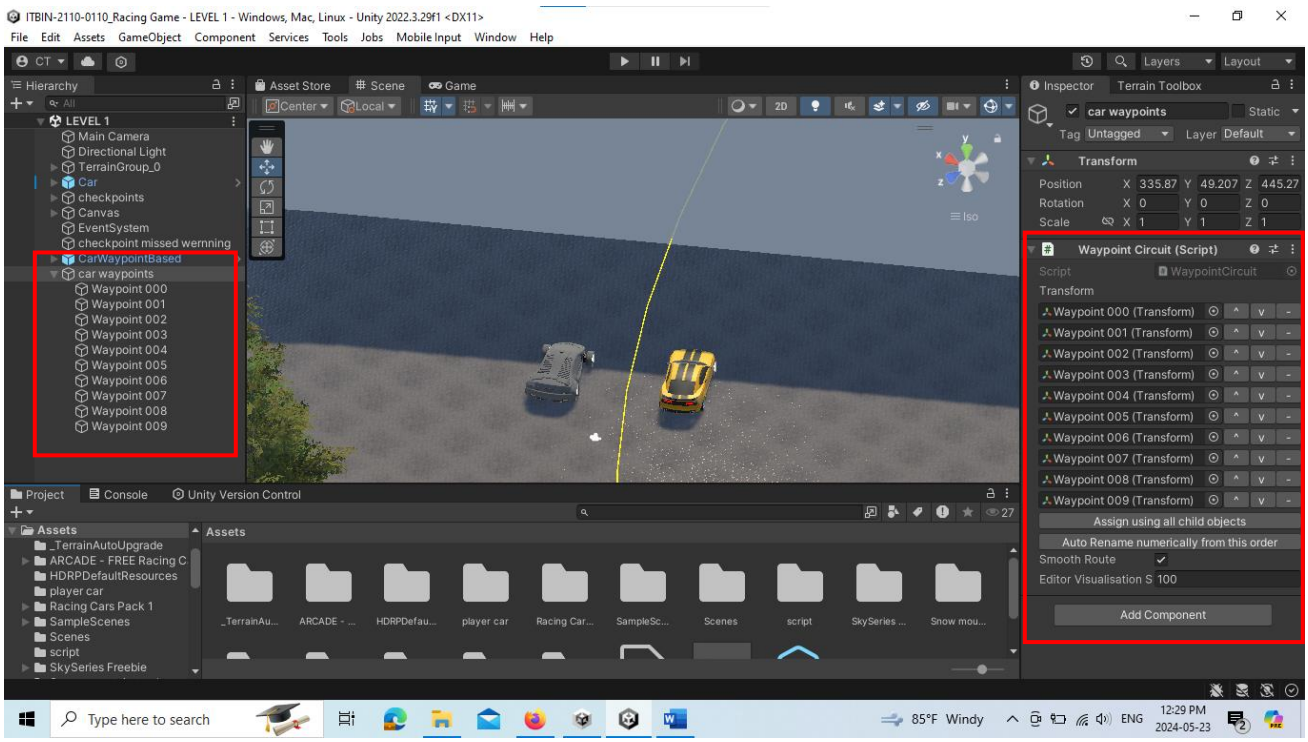
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;

public class Checkpoints : MonoBehaviour
{
    public int lap = 0;
    public int checkpoint = -1;
    int checkpointCount;
    int nextCheckpoint = 0;
    Dictionary<int, bool> visited = new Dictionary<int, bool>();
    public Text lapText;
    // Start is called before the first frame update
    void Start()
    {
        GameObject[] checkpoints = GameObject.FindGameObjectsWithTag("checkpoint");
        checkpointCount = checkpoints.Length;

        foreach(GameObject cp in checkpoints)
        {
```

```
            visited.Add(Int32.Parse(cp.name), false);
        }

    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.tag == "checkpoint")
        {
            int checkpointCurrent = int.Parse(other.gameObject.name);
            if (checkpointCurrent == nextCheckpoint)
            {
                visited[checkpointCurrent] = true;
                checkpoint = checkpointCurrent;
                if (checkpoint == 0)
                {
                    lap++;
                    lapText.text = "Lap:" + lap;
                }
                nextCheckpoint++;
                if(nextCheckpoint >= checkpointCount)
                {
                    var keys = new List<int>(visited.Keys);
                    foreach(int key in keys)
                    {
                        visited[key] = false;
                    }
                    nextCheckpoint = 0;
                }
            }
        }
    }
}
```

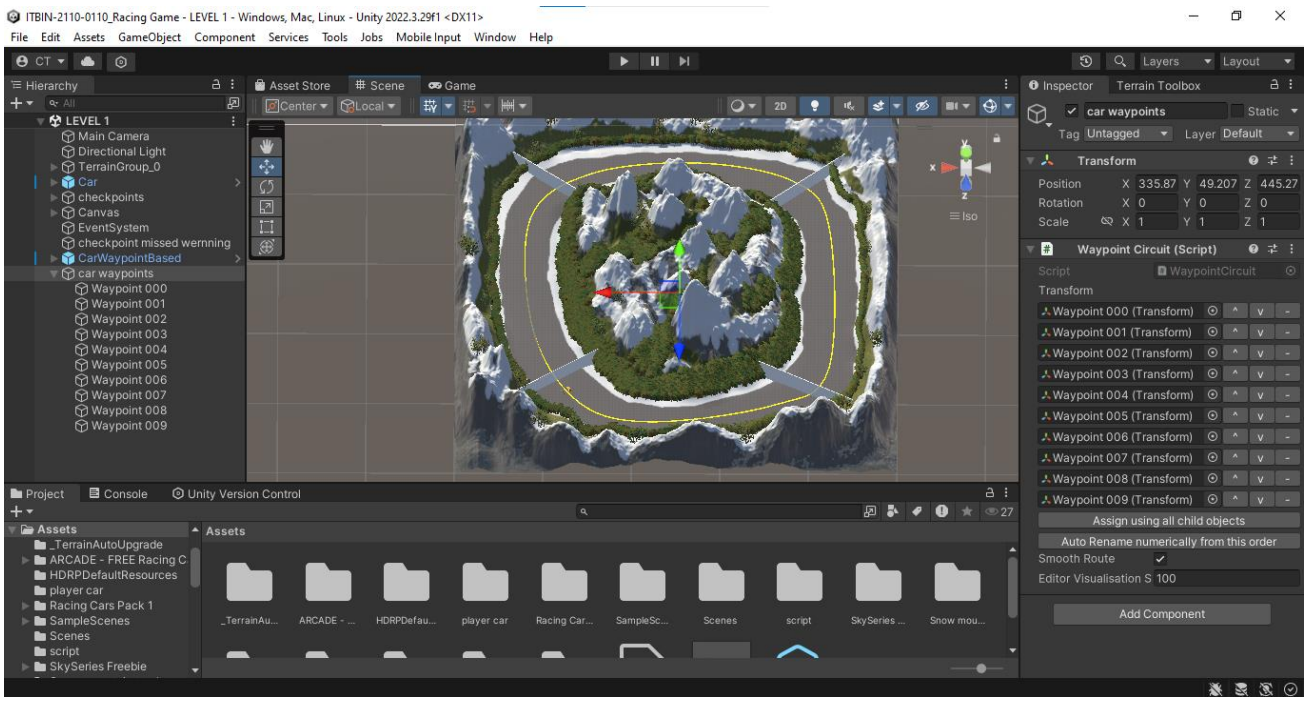
❖ In below you can see game view.

❖ After that I add another AI car for my opponent.

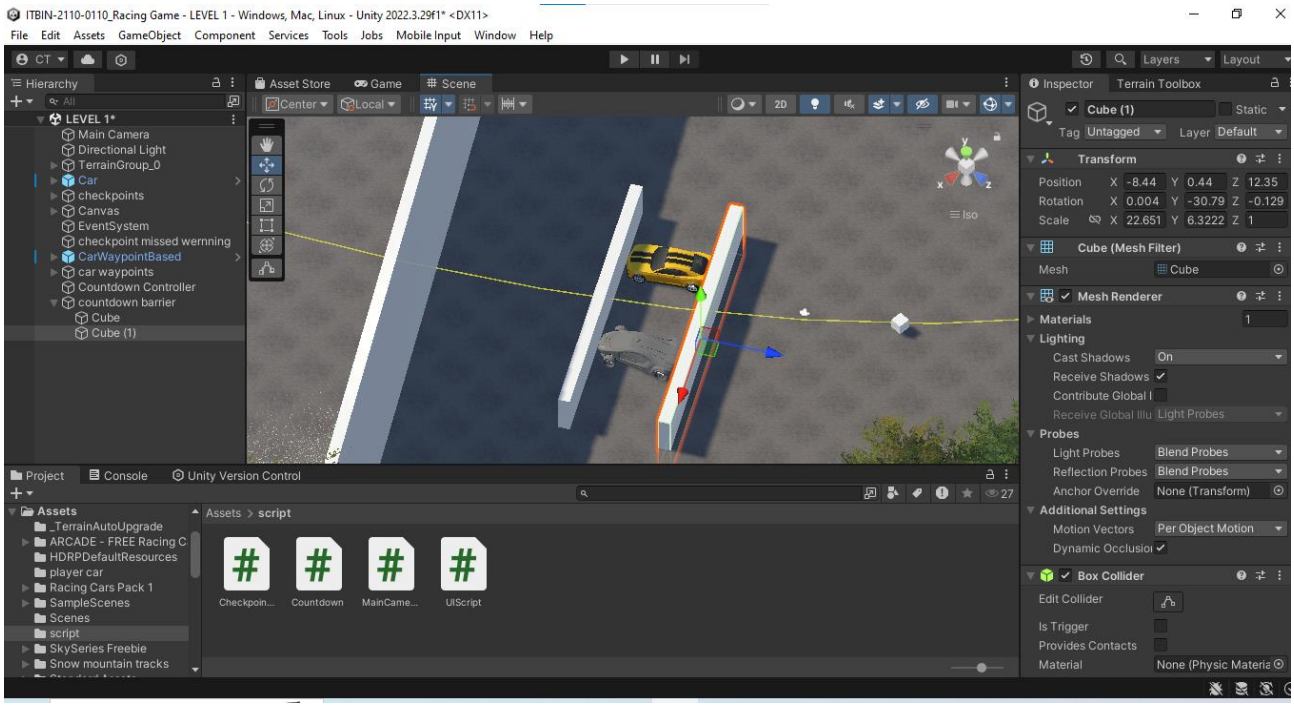
❖ Then I add waypoints for the AI car. The AI car goes across those waypoints.

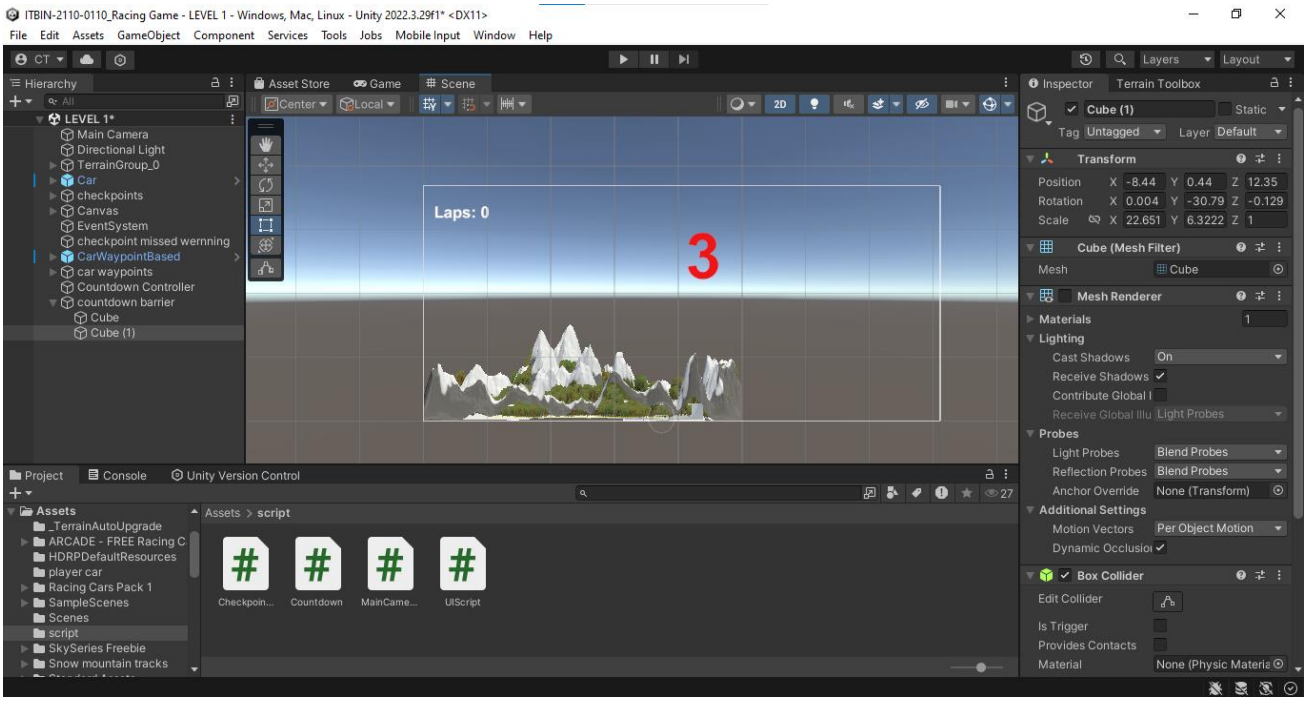


- The AI cars go across the yellow line. You can see that yellow line in below image.

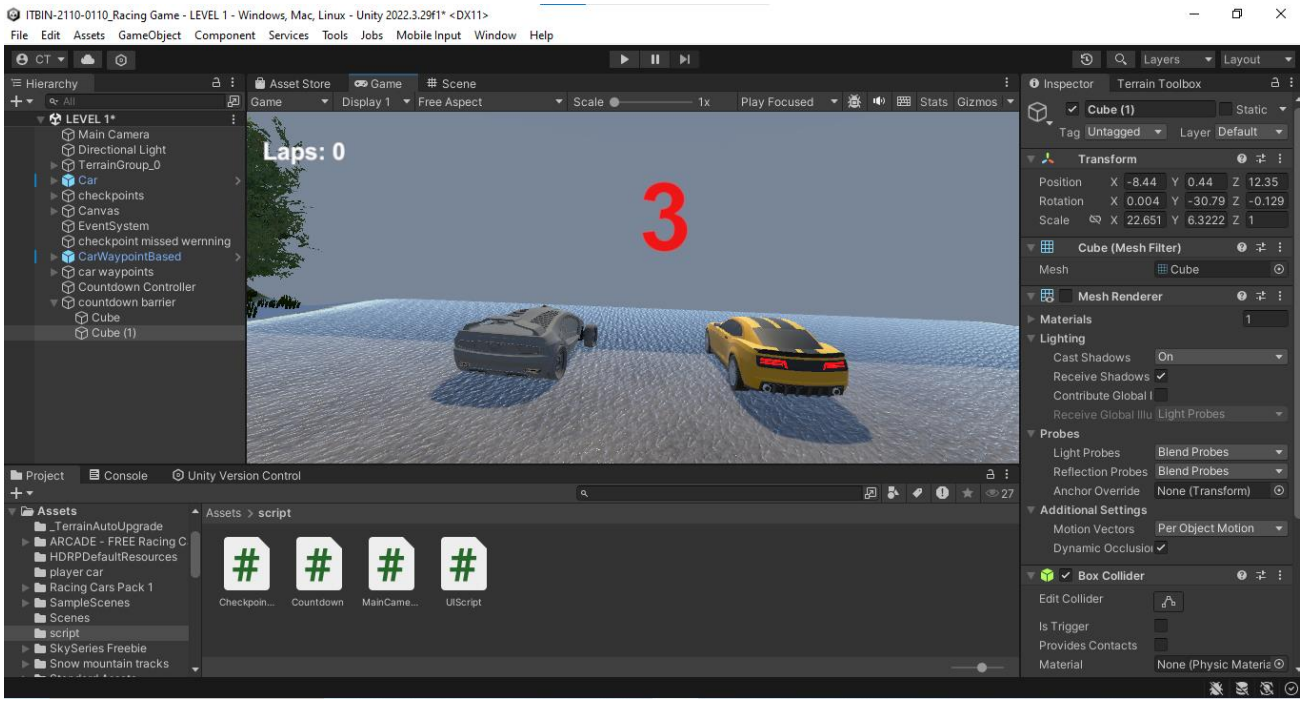


- After that I add a countdown for my game. Only After counting, we can play the game. For that, I added two barriers, one for the front of the cars and another one for the back of the cars. After the countdown, those barriers are automatically destroyed. I wrote code for that. That code and UI text you can see in below screen shots.





❖ Game view you can see in below,



❖ C# code you can see in below,

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

public class Countdown : MonoBehaviour
{
    public GameObject carcontrol;
    public GameObject CountdownUI;
    // Start is called before the first frame update
    void Start()
    {
        StartCoroutine(CountStart());
    }

    IEnumerator CountStart()
    {
        yield return new WaitForSeconds(0.5f);
        CountdownUI.GetComponent<Text>().text = "3";
        yield return new WaitForSeconds(1.0f);
        CountdownUI.GetComponent<Text>().text = "2";
        yield return new WaitForSeconds(1.0f);
        CountdownUI.GetComponent<Text>().text = "1";
        yield return new WaitForSeconds(1.0f);
        CountdownUI.GetComponent<Text>().text = "Go";
        yield return new WaitForSeconds(1.0f);
        CountdownUI.SetActive(false);
        carcontrol.SetActive(false);
    }
}
```

❖ When the car is accident, I can press “R” and reenter to the game from the final checked checkpoint. That code you can see in below.

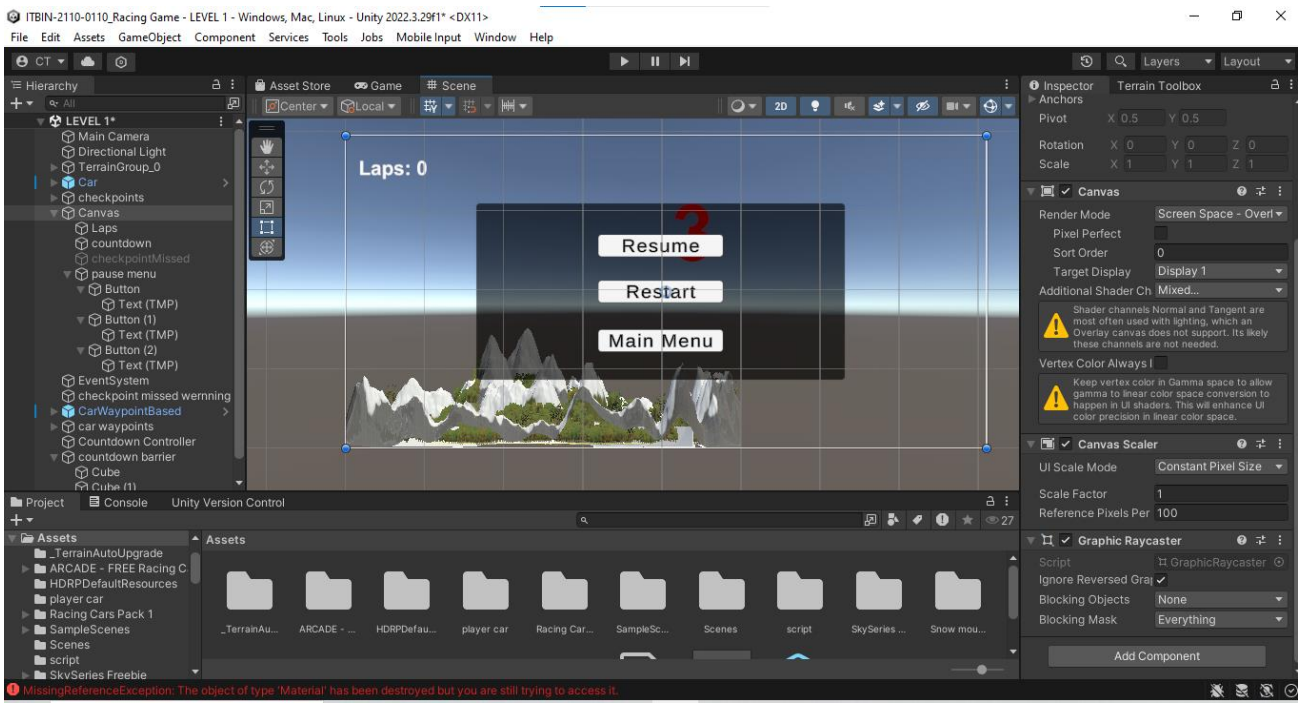
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class resetplayerCar : MonoBehaviour
{
    Checkpoints checkpoints;
    Rigidbody rb;
    // Start is called before the first frame update
    void Start()
    {
        rb = GetComponent<Rigidbody>();
        checkpoints = GetComponent<Checkpoints>();
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.R))
        {
            gameObject.transform.position = checkpoints.PrevCheckpoint.transform.position;
        }
    }
}
```

❖ After that I first add Pause C# script to the game. By using that code The player can pause the game by pressing the “Escape” key on the keyboard. That code you can see in below.

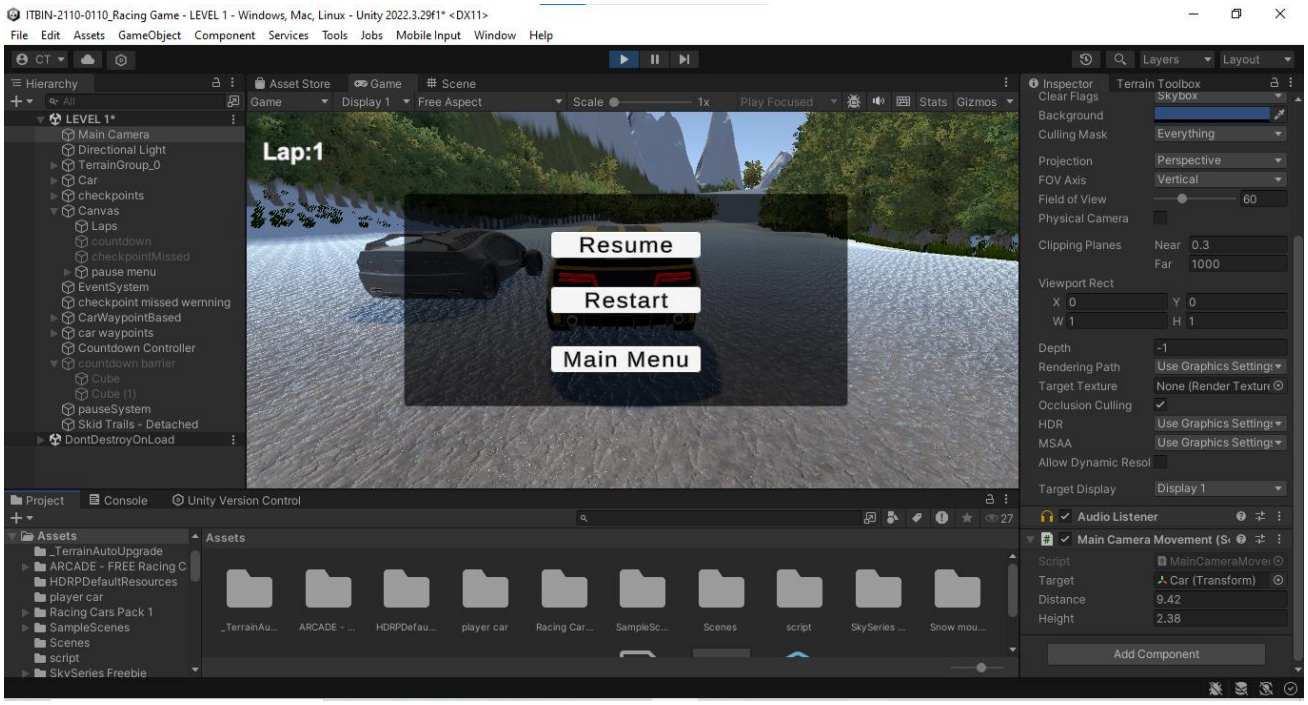
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Pause : MonoBehaviour
{
   [HideInInspector]
   public GameObject car;
   [HideInInspector]
   public GameObject AIcar1;

   int x = 0;
   // Start is called before the first frame update
   void Start()
   {
      car = GameObject.FindGameObjectWithTag("Player");
      AIcar1 = GameObject.FindGameObjectWithTag("AIcar1");
   }

   // Update is called once per frame
   void Update()
   {
      if (Input.GetKeyDown(KeyCode.Escape))
      {
         if (x == 0) {

            car.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMin
= 0;
            car.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMax
= 0;


AIcar1.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMin = 0;

AIcar1.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMax = 0;

            Time.timeScale = 0;
            x = 1;
         }
         else if (x == 1)
         {
            car.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMin
= 1;
```

```
            car.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMax
= 5;


AIcar1.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMin = 1;

AIcar1.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMax = 5;



                Time.timeScale = 1;
                x = 0;
            }
        }
    }
}
```

- Secondly, I create UI pannel for My pause menu And add three Buttons called Resume, Restart and Main Menu



❖ When the player clicks the “Escape” key on the keyboard then the pause menu panel display.



❖ Thereafter I updated the code of the Resume button and I wrote new code for the Restart button, when the user clicks on that button by using a mouse those buttons work properly. The updated code and new code you can see in below.

Resume Button code

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

public class Pause : MonoBehaviour
{
    [HideInInspector]
    public GameObject car;
    [HideInInspector]
    public GameObject AIcar1;

    public GameObject pauseMenu;

    int x = 0;
    // Start is called before the first frame update
    void Start()
    {
        car = GameObject.FindGameObjectWithTag("Player");
        AIcar1 = GameObject.FindGameObjectWithTag("AIcar1");
```

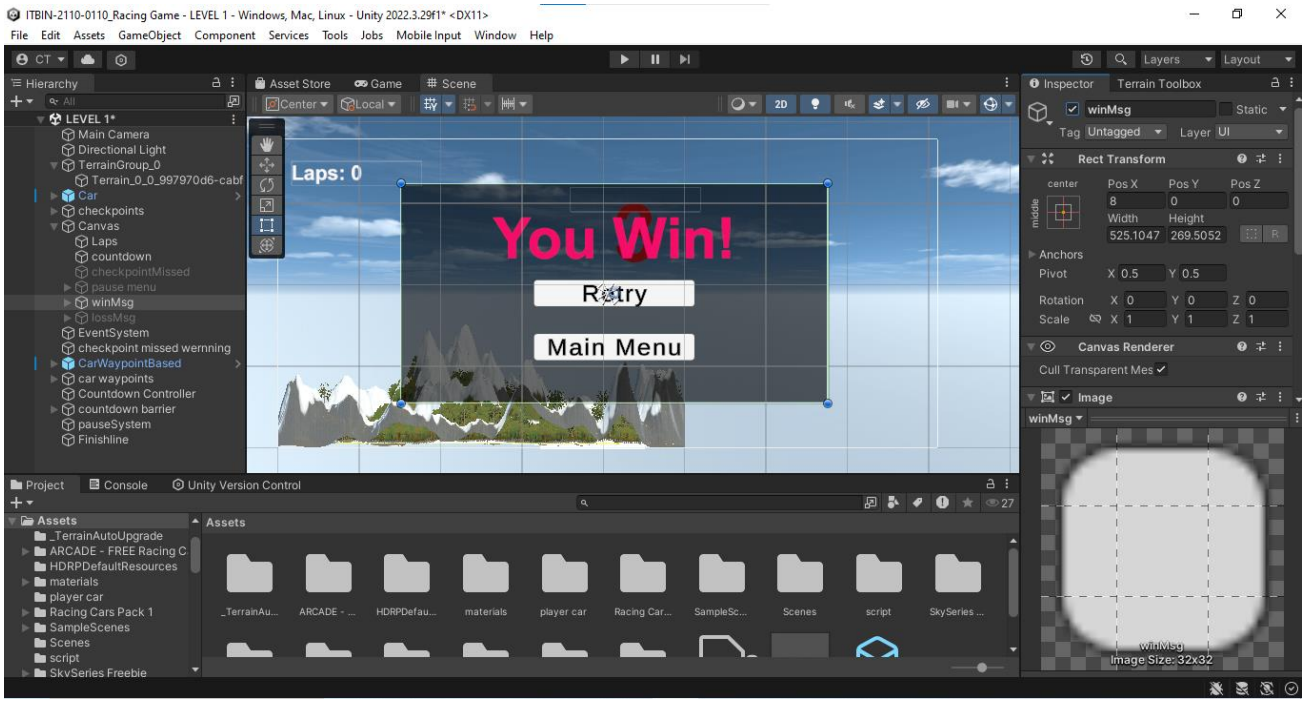
```
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (x == 0) {

                car.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMin
= 0;
                car.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMax
= 0;


AIcar1.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMin = 0;

AIcar1.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMax = 0;

                Time.timeScale = 0;
                x = 1;

                pauseMenu.SetActive(true);
            }
            else if (x == 1)
            {
                resume();
            }
        }
    }

    public void resume()
    {
        car.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMin = 1;
        car.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMax = 5;

        AIcar1.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMin  =
1;
        AIcar1.GetComponent<UnityStandardAssets.Vehicles.Car.CarAudio>().lowPitchMax  =
5;



        Time.timeScale = 1;
        x = 0;
```

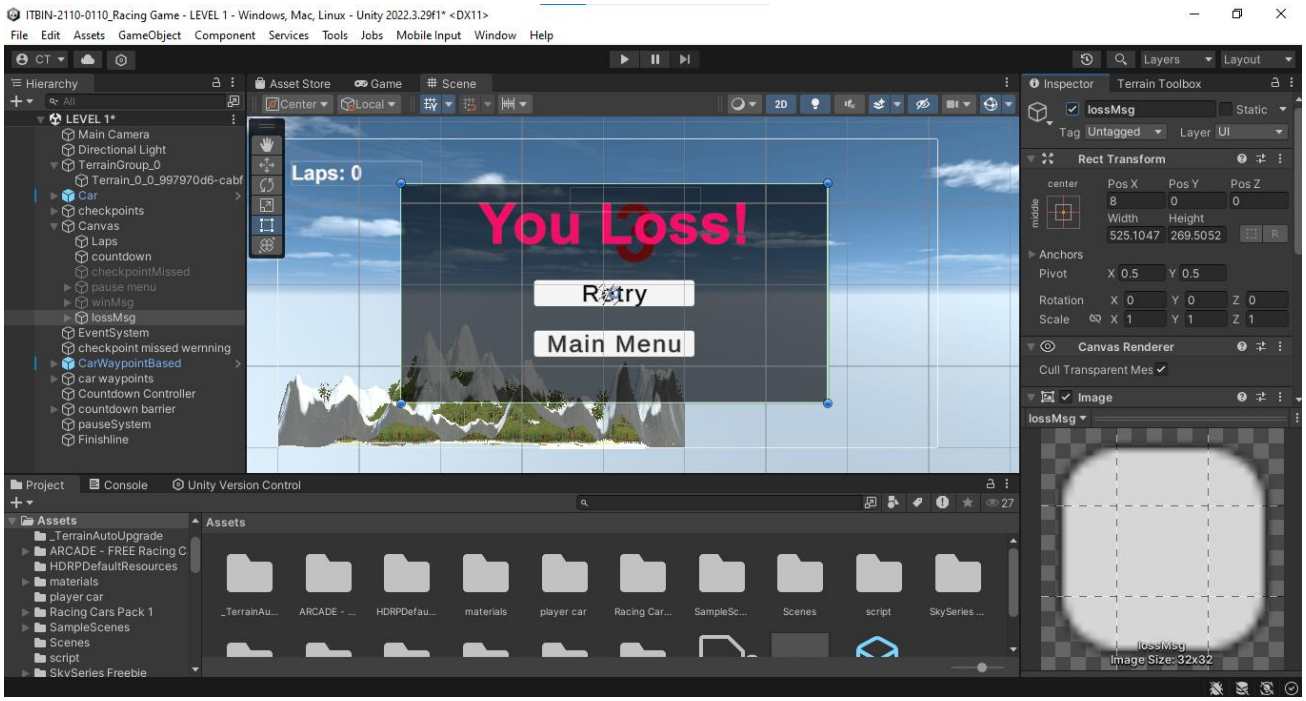
```
        pauseMenu.SetActive(false);
    }

}
```

Restart Code

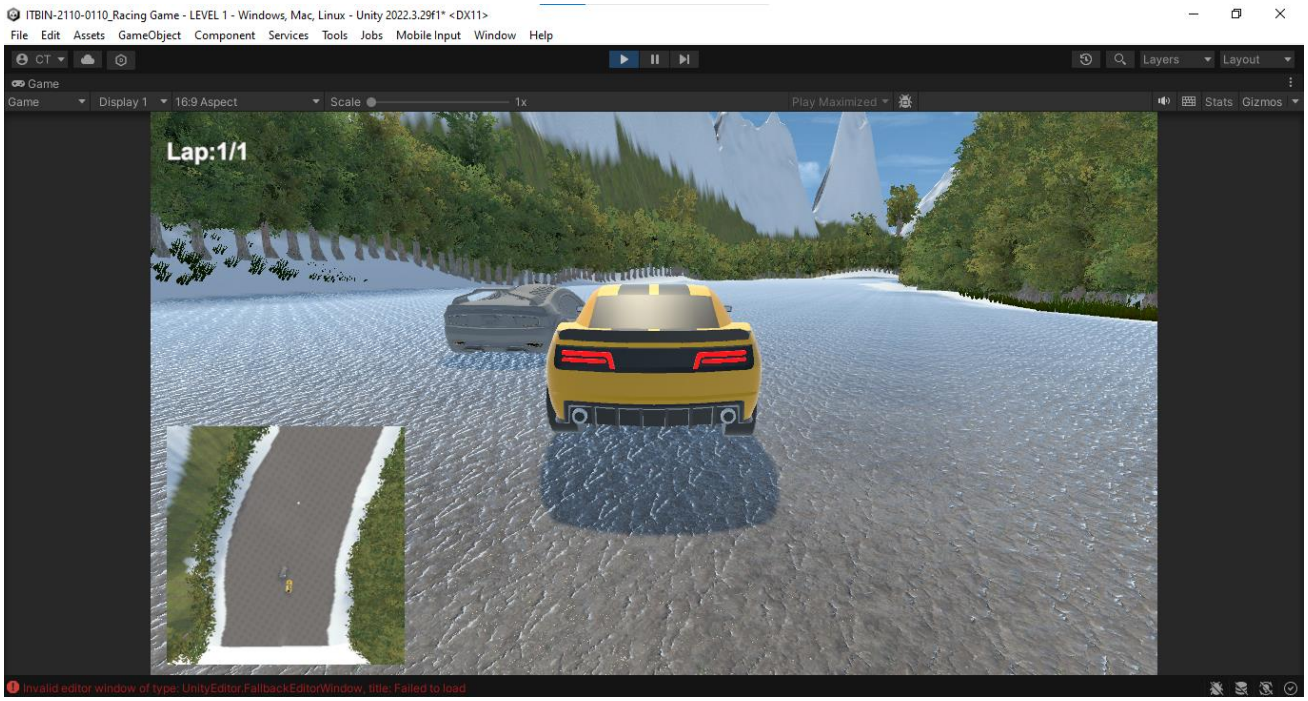
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;
using UnityEngine;

public class Restart : MonoBehaviour
{
 // Start is called before the first frame update
 void Start()
{

 }

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

}

 public void RestartGame()
{
SceneManager.LoadScene(SceneManager.GetActiveScene().name);
Time.timeScale = 1;
}
}
```

- After that I add two UI panels to display the win message and the Loss message. I added Two Buttons such as Retry and Main Menu. When the player wins the game that win message panel automatically displays otherwise the Lost message displays. Those two UI panels you can see in below screen shots.

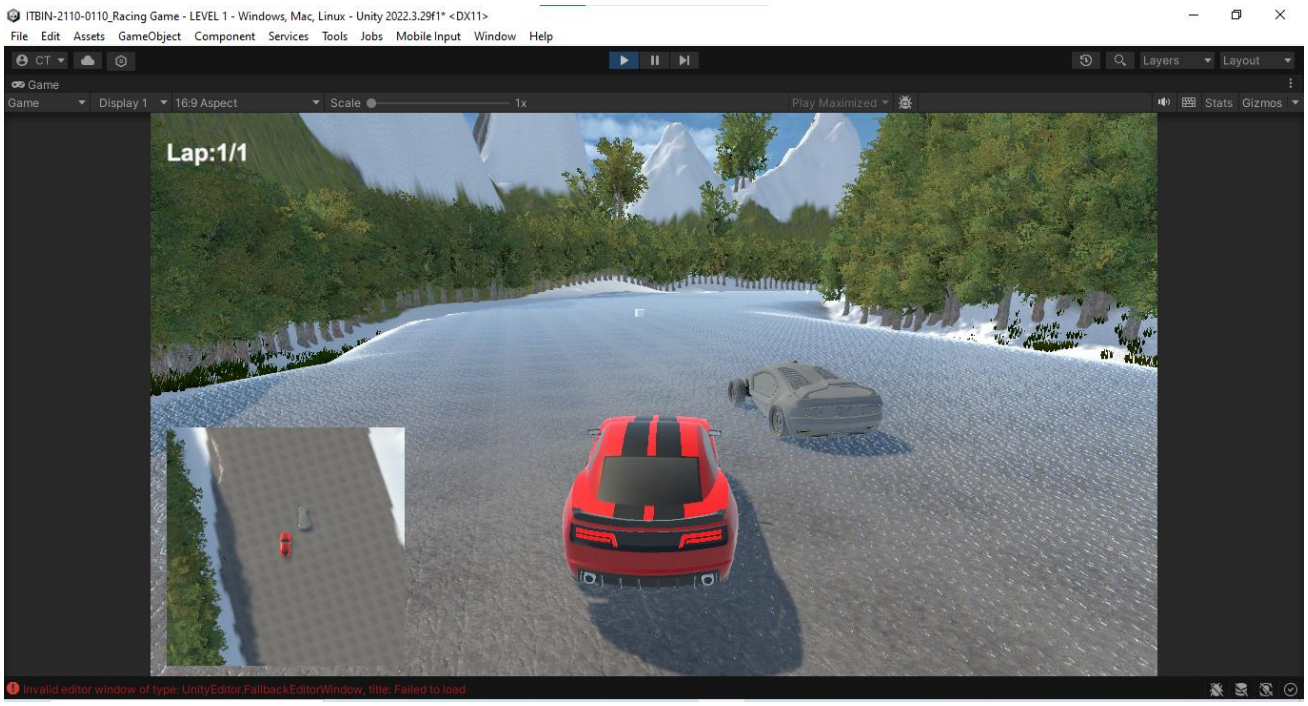




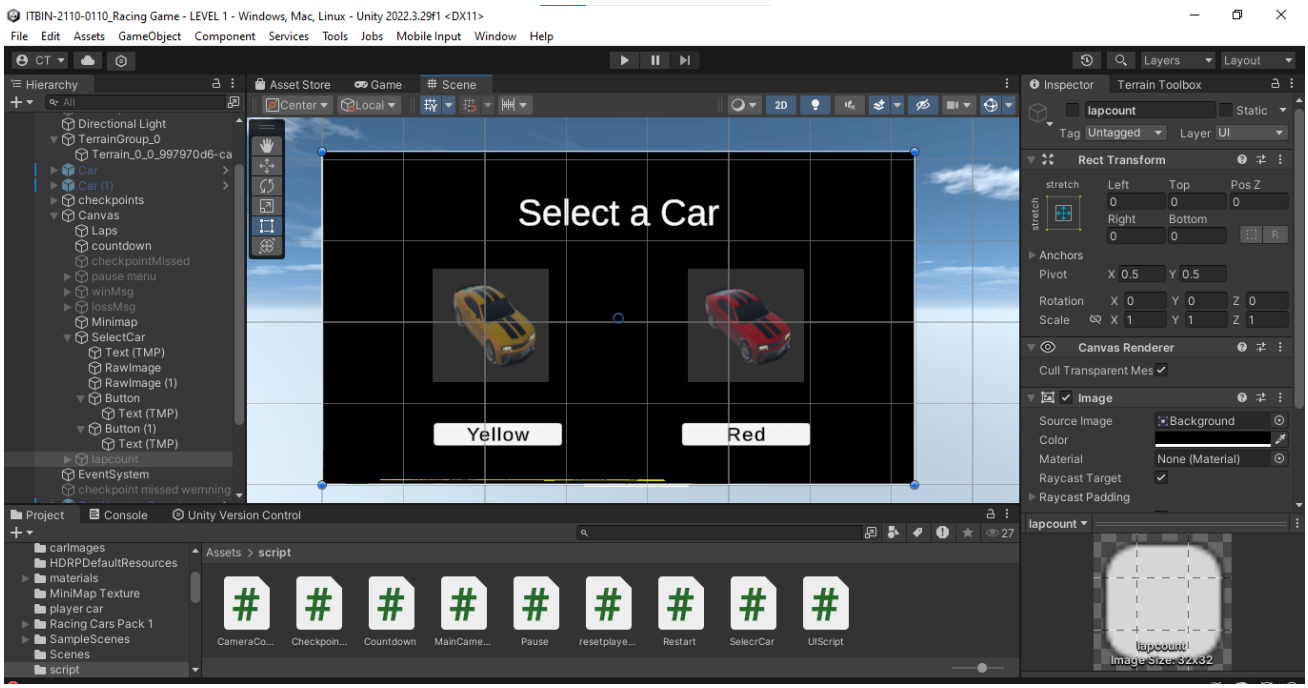
❖ After that I added Mini Map to my game. You can see that Minimap in the left-hand side corner of the game window.

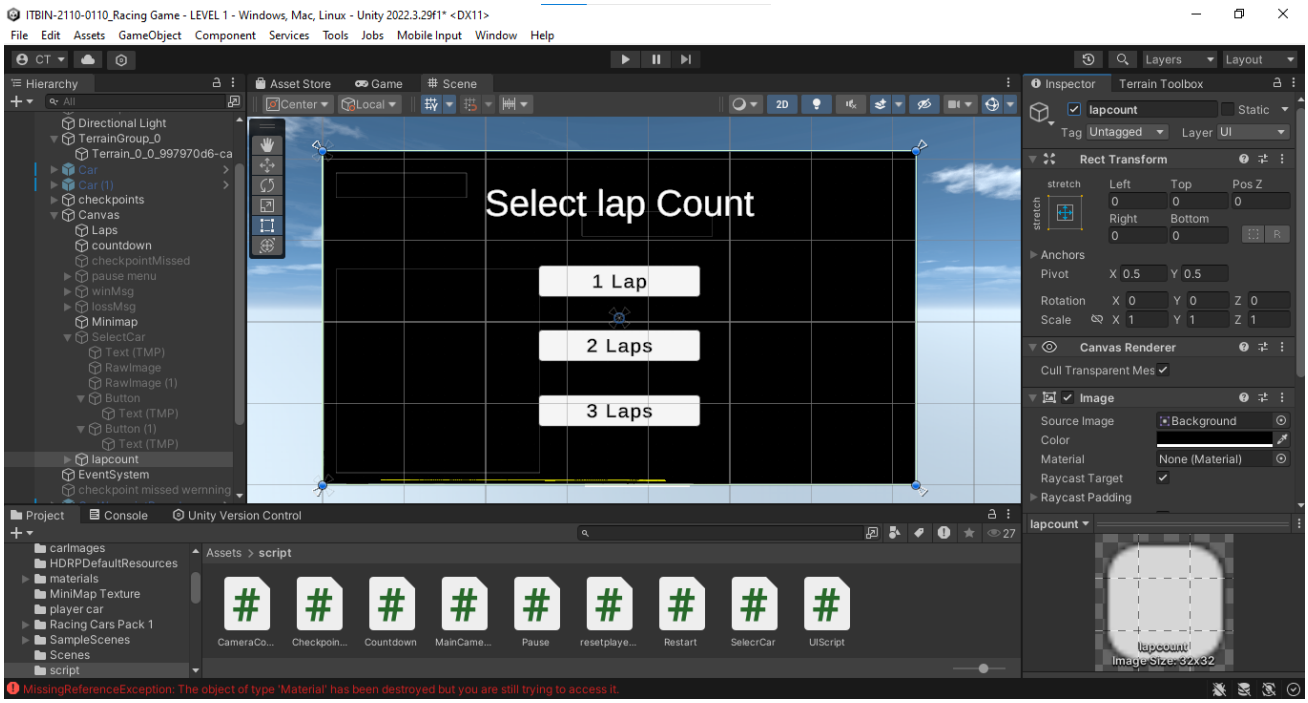


❖ After that I added another Red car to my game. The purpose of adding another car is for players to choose their desired color.

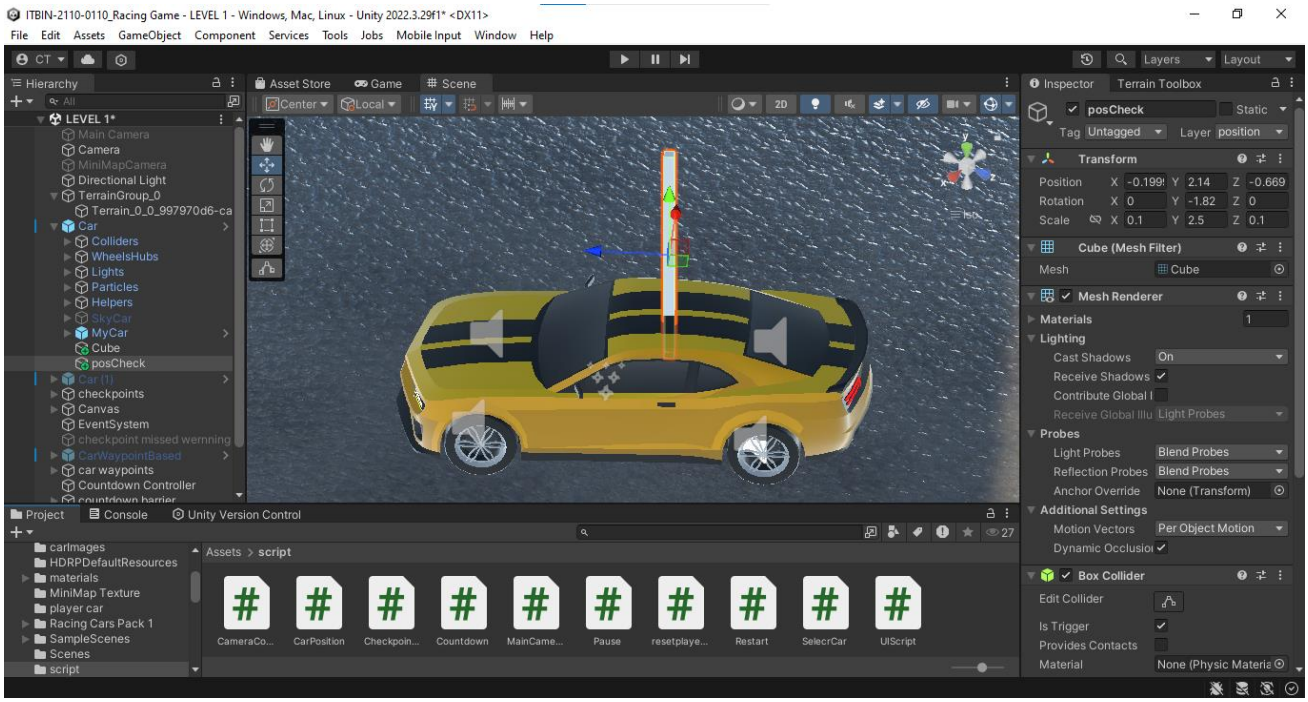


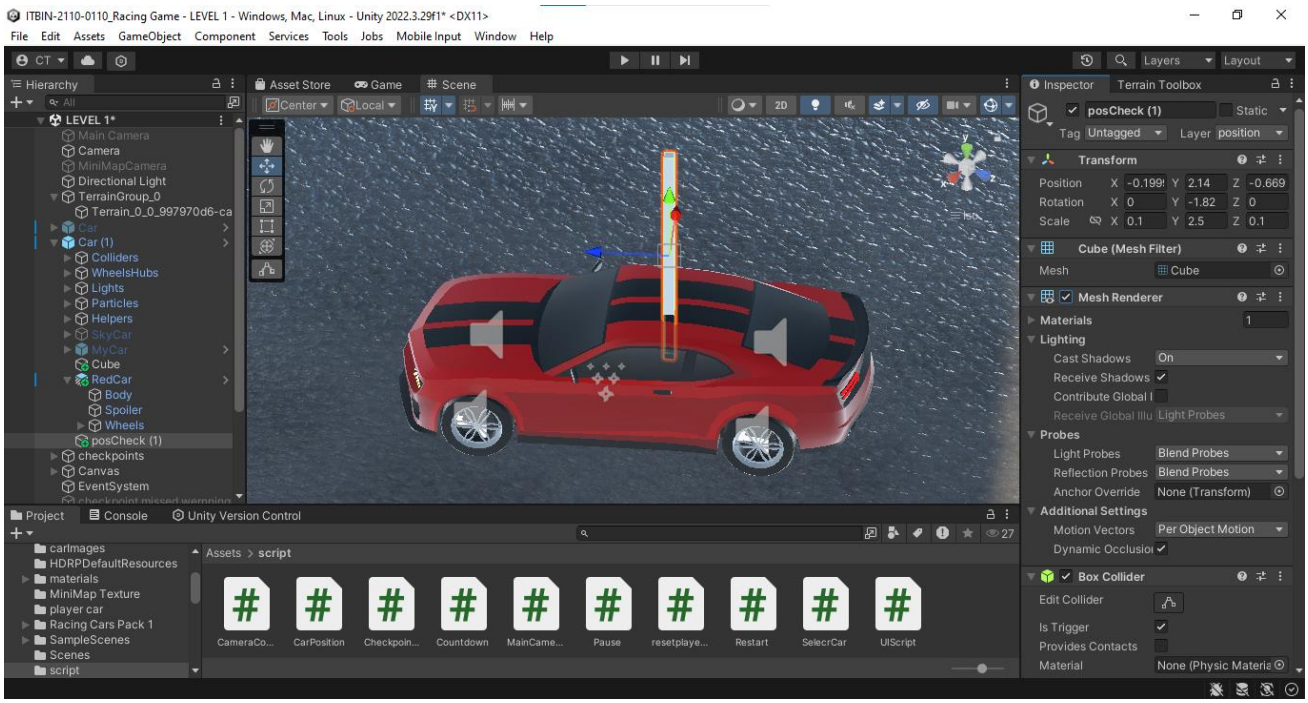
❖ Now I created two UI Pannels Called, **“SelectCar”** and **“lapcount”**. By using select car pannel the players can choose cars and By using the lapcount panel the users can select laps. Those two panela you can see in below.



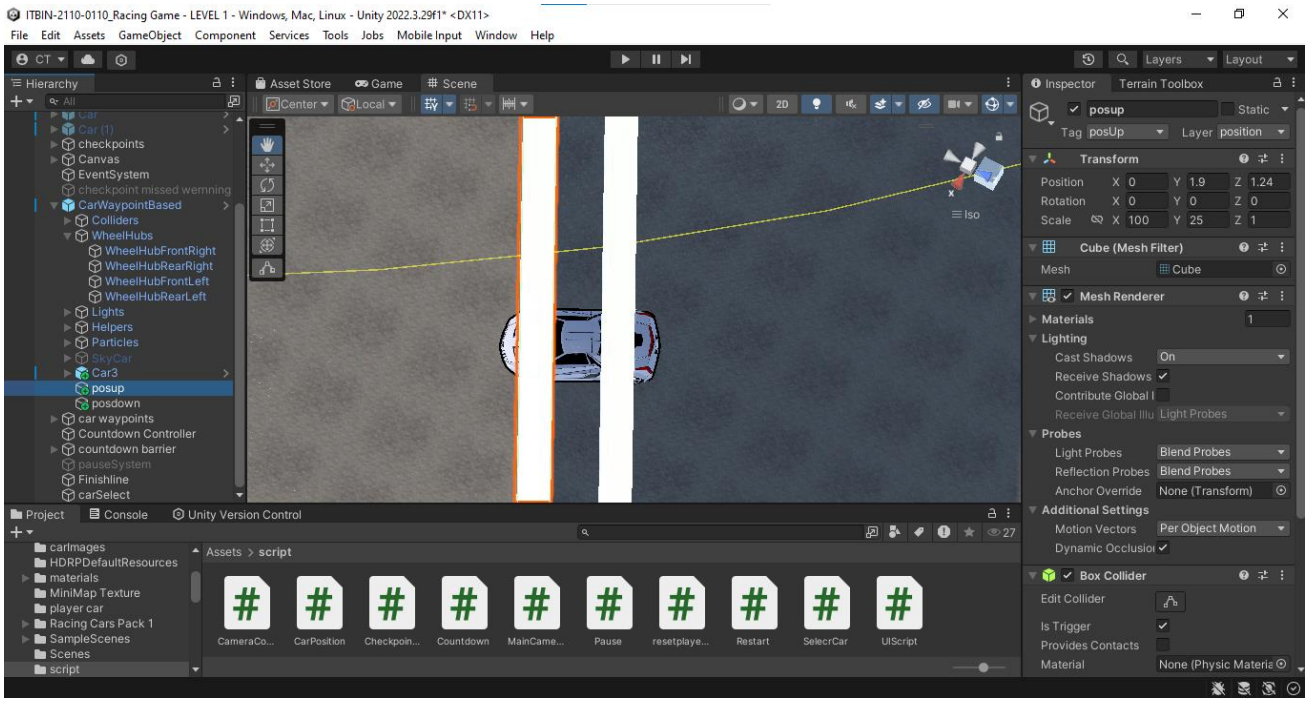


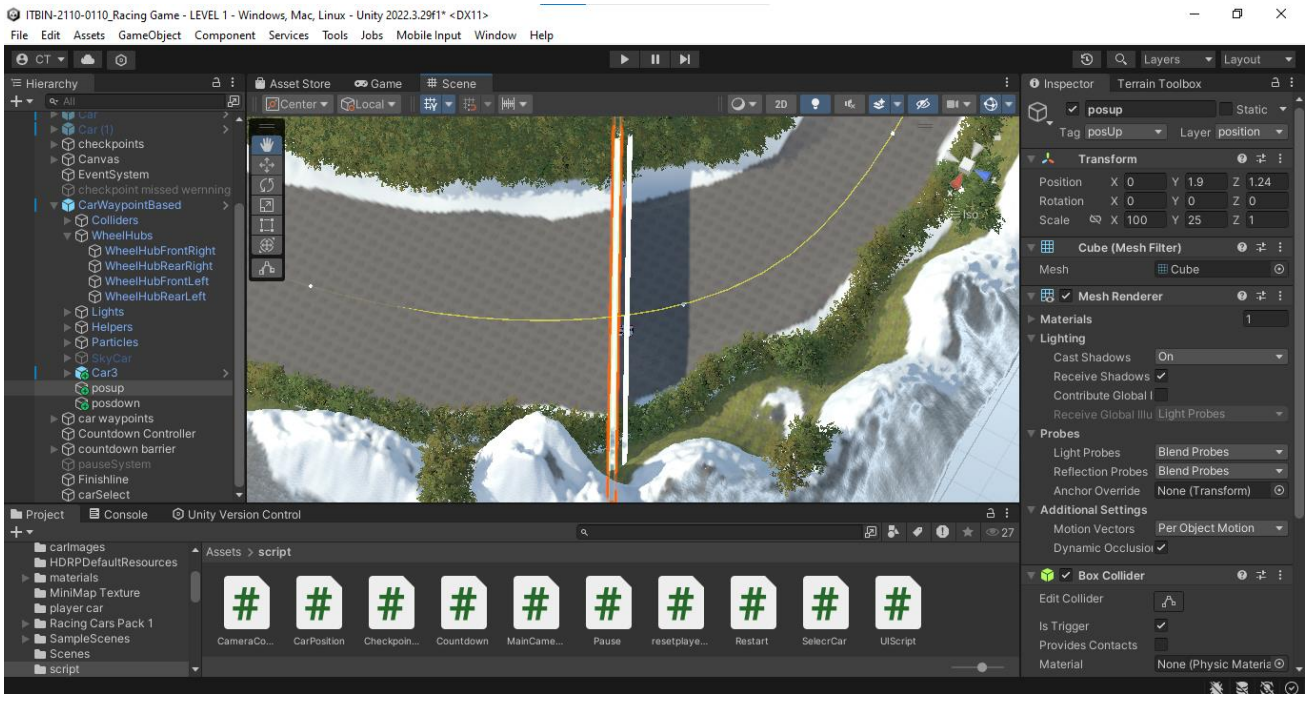
❖ After that I add the Position system to my game. To add that position system, firstly I add 3D game object to the two Player Cars. Like below,





❖ After adding those game objects, I add another Two game objects for the AI car. Those Objects you can see in below.



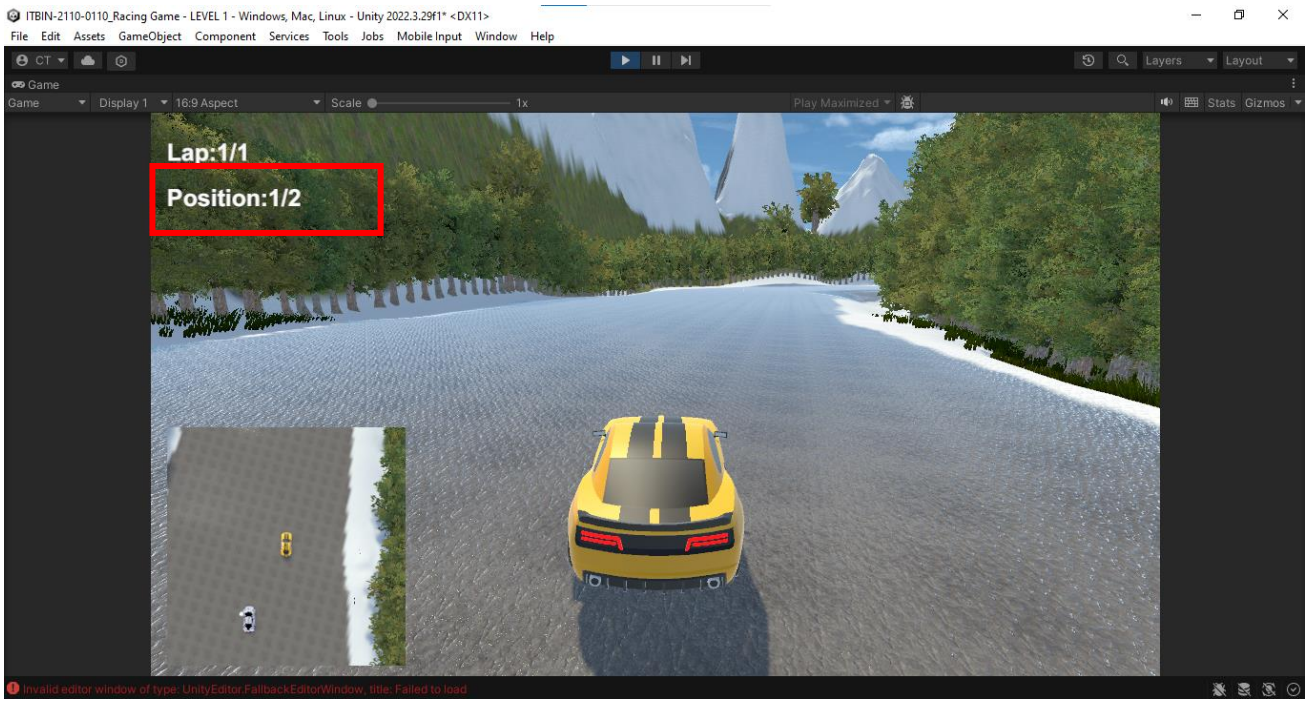


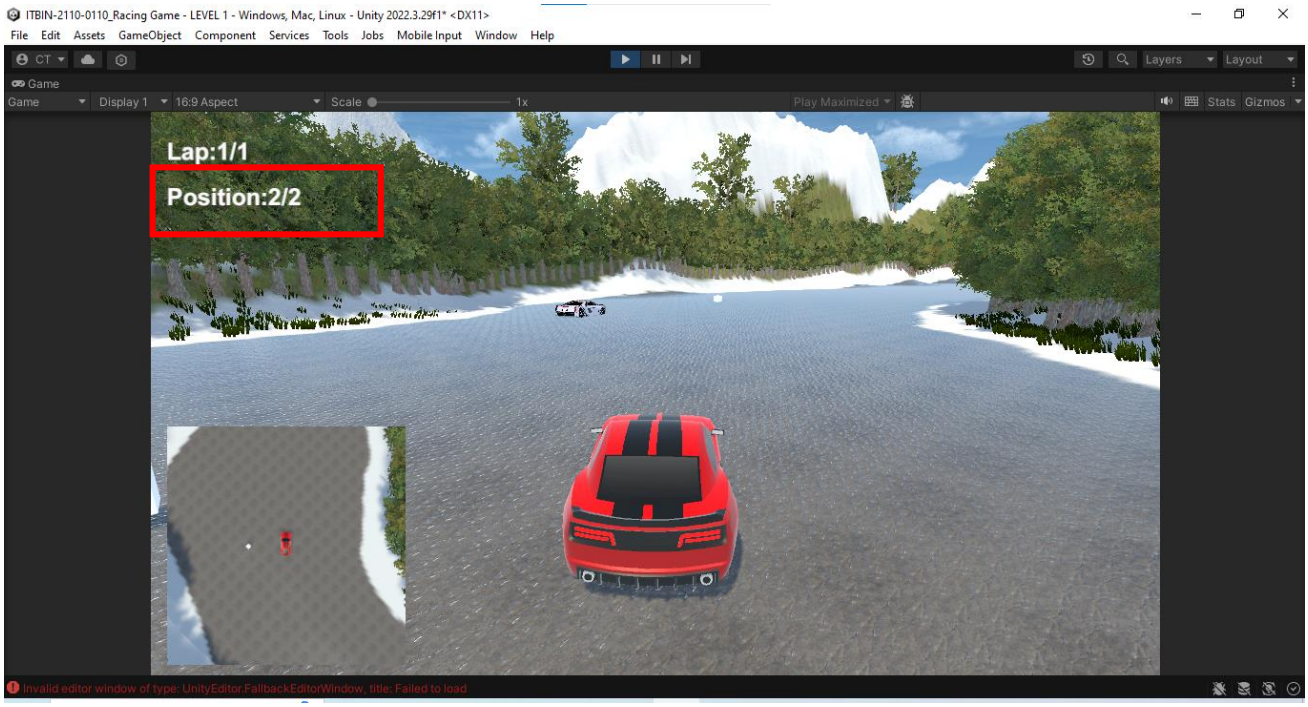
❖ The purpose of adding Those two game objects, By using those two game objects, The Player car can identify the correct position. When The player car altakes the AI car, The player car's Game Object identifies the AI car's Game objects. That is the mechanism. The C# script you can see in below.

❖ **C# script**

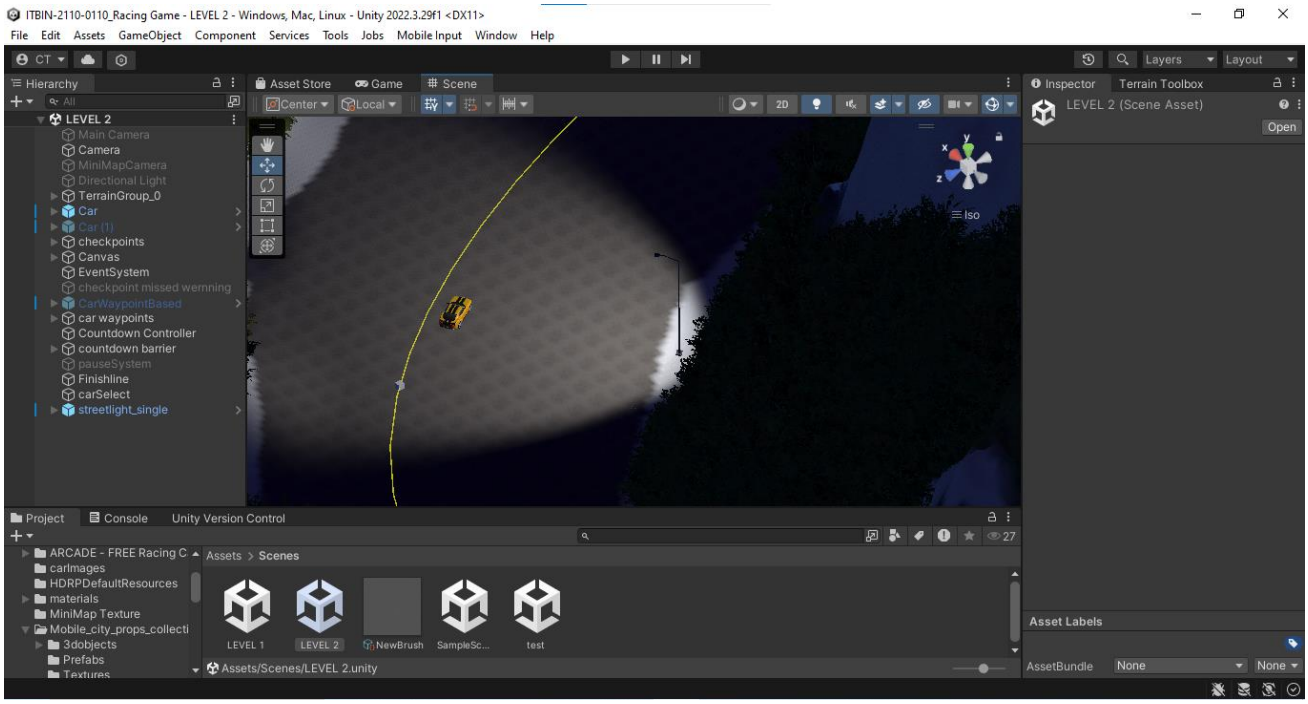
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class CarPosition : MonoBehaviour
{
    public GameObject PositionText;
    public int pos = 2;

    bool x = true;


    bool p = false;


    private void OnTriggerExit(Collider other)
    {
        if(other.tag == "posUp" && pos>1 && x == true )
        {
            pos = pos - 1;
            PositionText.GetComponent<Text>().text="Position:" + pos + "/2";
            x = false;
            p = true;
        }

        else if (other.tag == "posDown" && p == true)
        {
            pos = pos + 1;
            PositionText.GetComponent<Text>().text = "Position:" + pos + "/2";
            p = false;
            x = true;
        }

    }

}
```

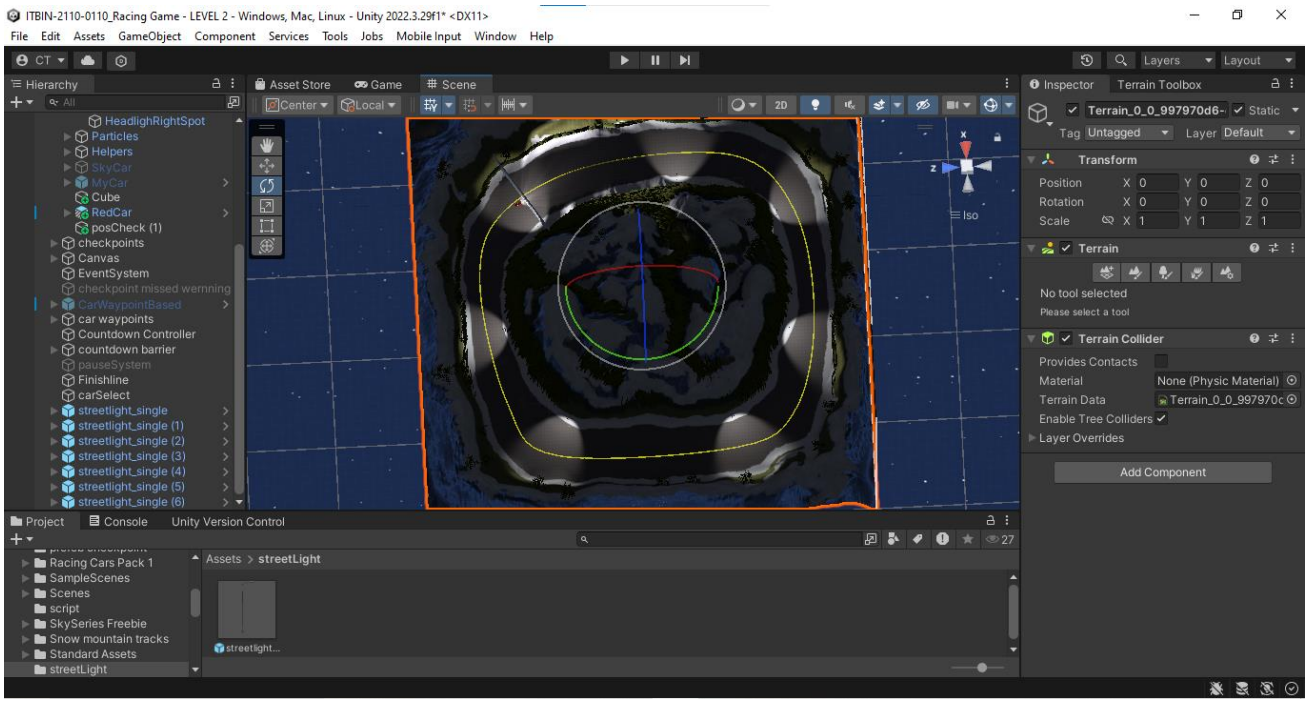
❖ After that I add UI text to display the car's position. You can see that Position system in images below. That is the Game View.



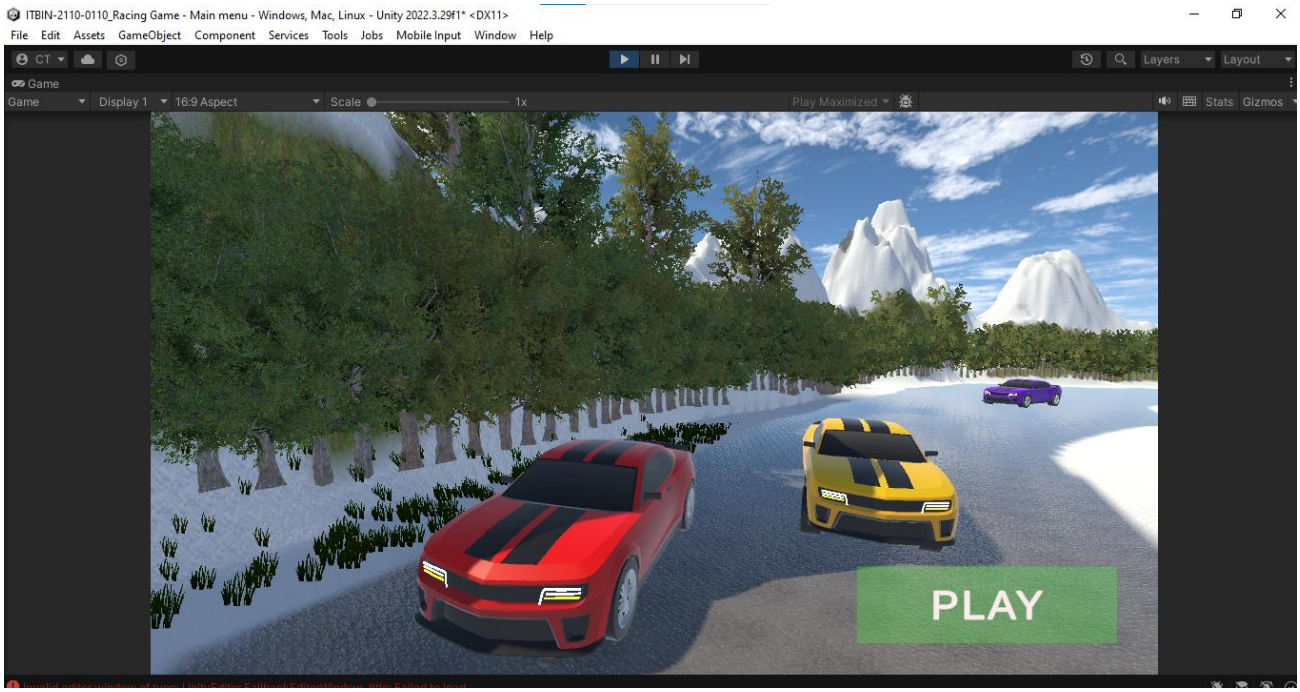


- After that I add another Level for the game. That Level is Night Race. To create another Level I duplicate My Previous level(Day Race) , Turn of the directional light, and add a night Skybox to the sky. I add Spot Lights to the Race track. You can see in below image.

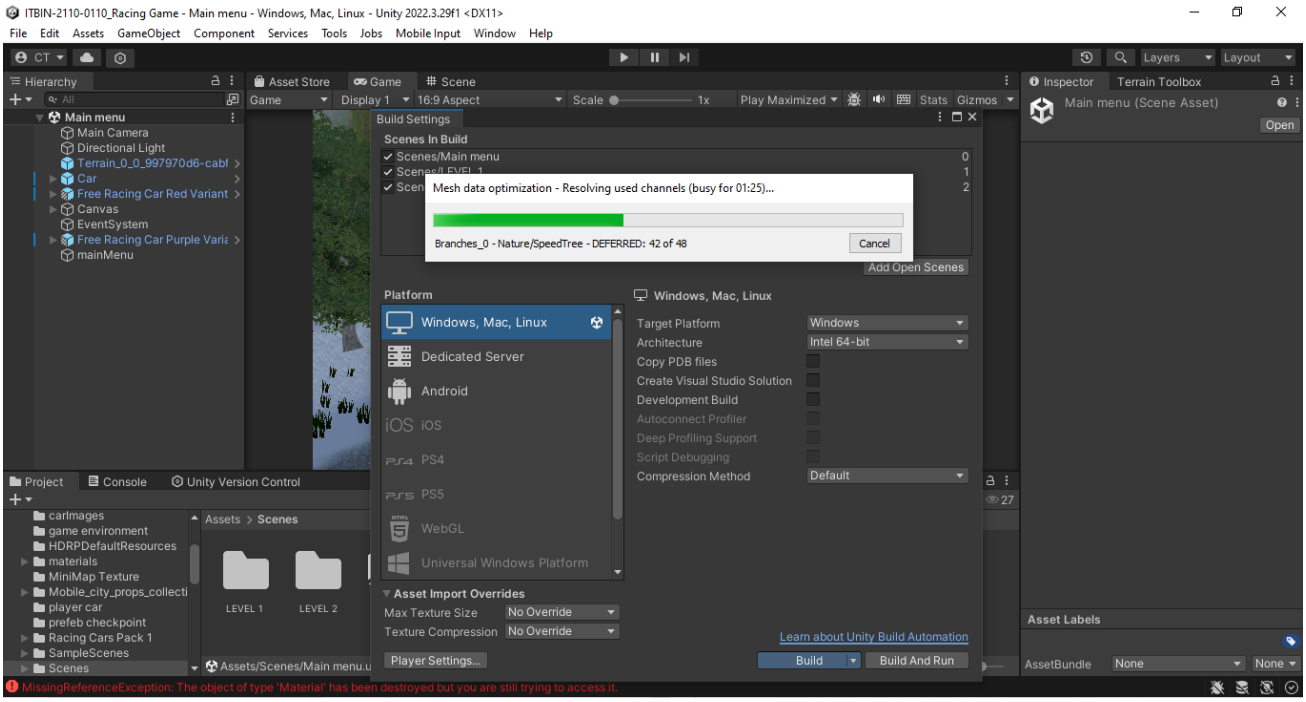




❖ After that I create a new scene as my car game's main menu. When the player opens the game the main menu is displayed first.



❖ After creating the main menu I export the My car race game to my computer.



- You can see Unity file, .exe/.apk and Demonstration screen record by using below link.

Google drive link - https://drive.google.com/drive/folders/1wwOvsrYC70SuIEcTvtFo8MqWOUQuRlLy?usp=sharing