**Game Programing - Simple Car Racing Game**

**Using OpenGL and C# 2020 Report:**

*Submitted by:*

**Maabul Quddus**

**RA17110200100103**

*Under the guidance of*

**Mr.S.Joseph James(Professor, Department Of Software Engineering)**

**in partial fulfilment for the award of the degree**

**Of**

**BACHELOR OF TECHNOLOGY**

**In**

**SOFTWARE ENGINEERING**

**Of**

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**S.R.M Nagar, Kattankulathur, Kancheepuram District**

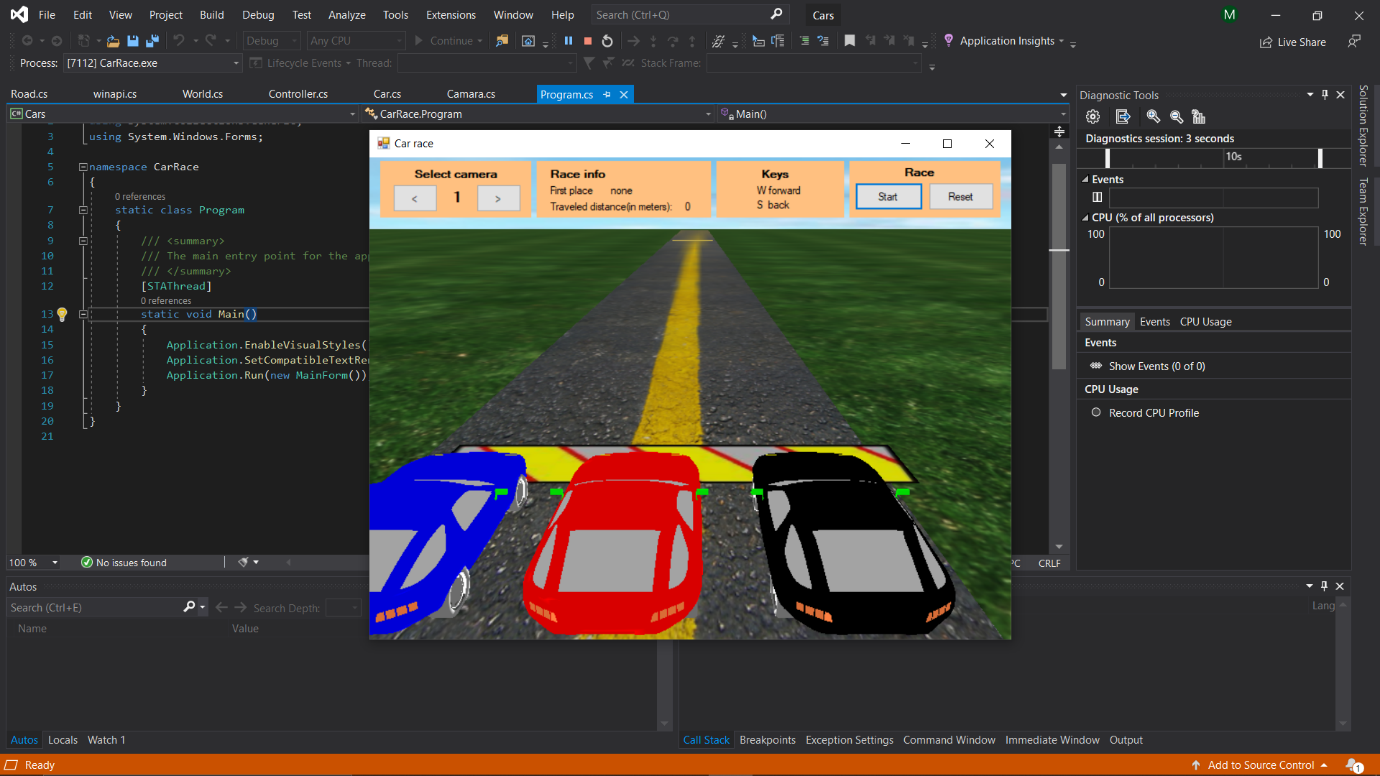
**12-13 May 2020**



Working Demo:

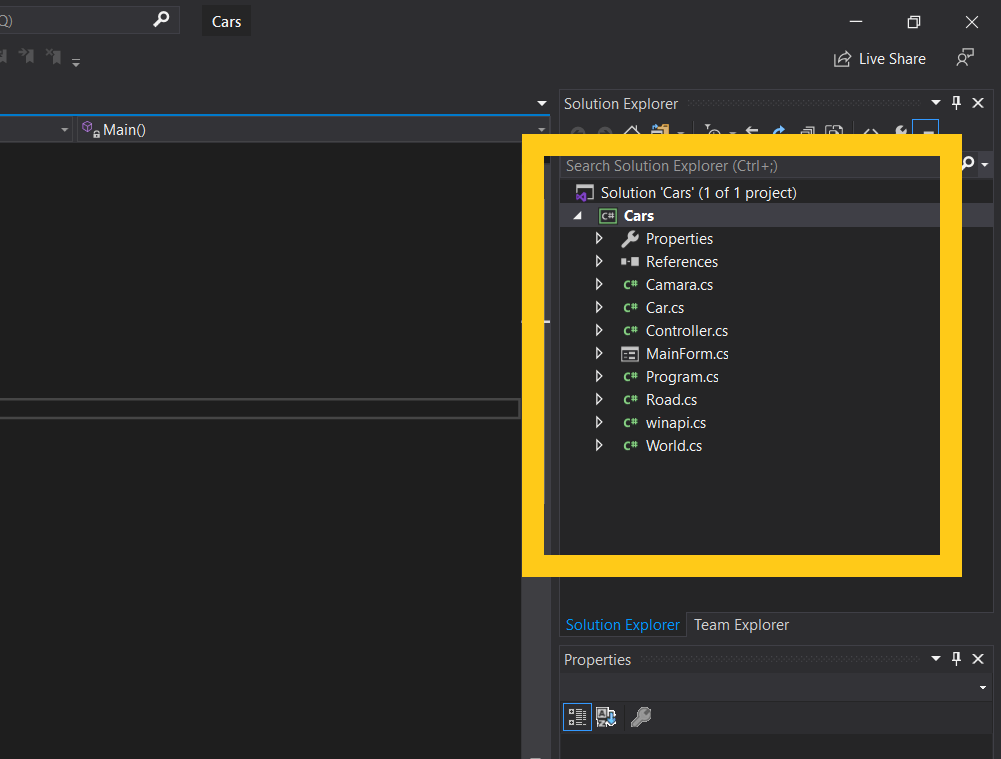
This is a simple straightforward car race game with minimum lines of code. There are 3 cars that begin in a start line, they randomly change their speeds on the race and the race ends on an end line, quite simple.

The Video of the Demo is enclosed for reference.



## **Introduction**

This is a simple straightforward car race game with minimum lines of code (LOC). There are 3 cars that begin in a start line, they randomly change their speeds on the race and the race ends on an end line, quite simple. The user can follow the race with the keys W (forward), S (backward) and can change the camera takes. At the end of the race, a message is shown telling which car was the winner and you are able to reset the race and start over again. The car race is a simulation and the user can’t take control over any car.



Above is the project solution explorer. As you can see, there are only a few classes. I will give a brief explanation of what each class does specifically.

**Objective**

The Objective of this game is to determine the winner among the three cars. The user can follow the race cars with the help of camera.cs which can be manipulated with the keys W (front) and S (Back).

Furthermore, the mainform.cs class has a design for showing the cameras the race info and keys that can be used you can reset the race and start the race using the buttons in mainform.cs.

**Explanation**

### **Camara.cs**

This class has two methods: one for selecting a camera position and another for initializing the camera. The code is very simple.

**Highlights**

public class Camara

{

public void SelectCamara(int camara)

{

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

Gl.glLoadIdentity();

switch (camara)

{

case 0:

{

Glu.gluLookAt(2, 3, 14, 2, 0, 5, 0, 1, 0);

break;

}

case 1:

{

Glu.gluLookAt(-2, 7, 10, 1, 3, 1, 0, 1, 0);

break;

}

case 2:

{

Glu.gluLookAt(0, 35, -15, 1, 0, -16, 0, 1, 0);

break;

}

case 3:

{

Glu.gluLookAt(3, 3, -47, 1, 0, 1, 0, 1, 0);

break;

}

}

}

public void SetPerspective()

{

*//select the projection matrix*

Gl.glMatrixMode(Gl.GL\_PROJECTION);

*//reset it*

Gl.glLoadIdentity();

*//55 = vision angle*

*//1 = aspect ratio*

*//0.1f = minimum draw distance*

*//1000 = maximum draw distance*

Glu.gluPerspective(55, 1, 0.1f, 1000);

SelectCamara(0);

}

}

### **Car.cs**

A 3D model is a set of meshes and this car is not an exception. In this class, I make a little tweak to draw the car. I draw the tires and the rims separate from the car because they have a rolling movement besides a displacement movement. Then, I make a list of the meshes that roll and calculate their pivot point which is its centre and insert it to a separate list.

When I draw a car, I draw the chassis, accessories, and the tires separately.

The chassis takes the colour of the car, the accessories are the same in the three cars, and the tires and rims go in a list because I rotate them.

This portion of code goes on the drawing function of the car and contains the race logic:

**Highlights**

if (Controller.StartedRace == true)

{

*//move the object the traveled distance*

Gl.glTranslatef(0, 0, traveledDistance);

if (traveledDistance > -59)

{

tireAngle -= 24;

traveledDistance -= speed;

}

else

if (Controller.FinishedRace == false)

{

Controller.FinishedRace = true;

}

counter++;

*// if counter == 30 i change the speed*

if (counter == 30)

{

counter = 0;

speed = 0.2f + (float)randomizer.NextDouble() / 20f;

}

}

If the race has started, then in each frame, I translate the car on the Z axis through the travelled distance which increases by adding a number called speed, and every 30 frames I randomize that speed. This will keep the race interesting giving the same shots to every car to win the race.

### **World.cs**

This class is for drawing the skybox and the terrain (not the road). As you will see later on the project, it is very simple..

### **Road.cs**

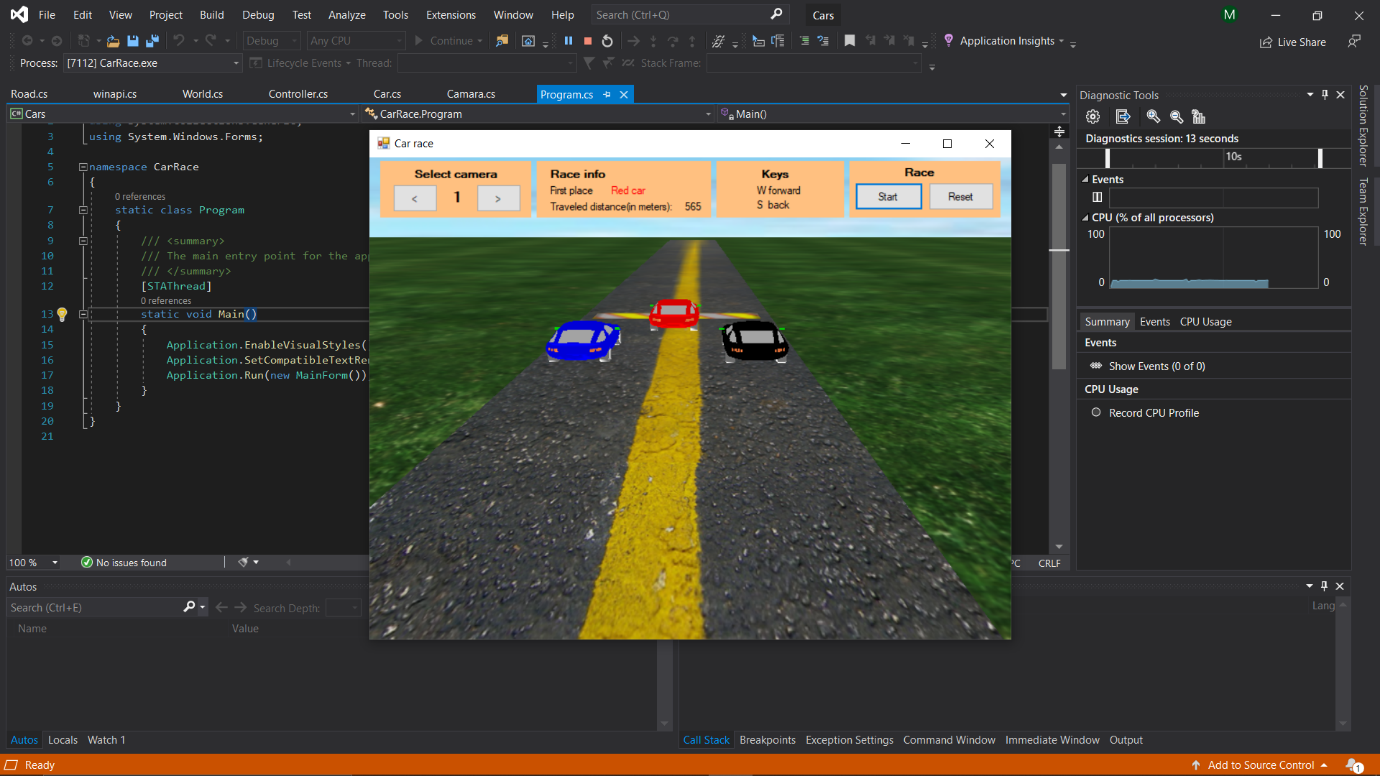
This class draws the road and the begin and end line. For drawing the road, I made a for loop drawing quads containing the texture of the road.

### **Controller.cs**

This class manages the drawing of the entire scene as well as methods to restart the race, querying which cars go first, among others.

### **Mainform.cs**

This form is where everything is drawn; it has a timer which draws the scene at a regular interval and the buttons to select the cameras and to restart the race. Inside the code, you will find the graphics initialization function, the lighting setup, the resize screen algorithm, etc.



As you can see in the given screenshot the red car reaches first which enables the program to prompt a dialog box saying the red car wins the details on Race Info gets updates before the dialog box.

**About The Software**

OpenGL is a cross-language, cross-platform application programming interface for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit, to achieve hardware-accelerated rendering.

C# is a general-purpose, multi-paradigm programming language encompassing strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented, and component-oriented programming disciplines

Microsoft Visual Studio is an integrated development environment from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services, games and mobile apps

Complete Code for Reference

### **Car.cs**

using System;

using System.Collections.Generic;

using System.Text;

using System.Drawing;

using ShadowEngine;

using Tao.OpenGl;

using System.Windows.Forms;

using ShadowEngine.ContentLoading;

namespace CarRace

{

class Car

{

float tireAngle;

static int carColor = 1;

Color color;

Position pos;

float speed;

float traveledDistance = 0;

static Random randomizer = new Random(); // to randomize the speed value

int counter;

ModelContainer m;

List<Mesh> tires = new List<Mesh>();

Mesh body; // this is the chassis of the car

int texture;

public float TraveledDistance

{

get { return traveledDistance; }

}

public Car()

{

if (carColor == 1)

{

color = Color.Blue;

pos = new Position(0, 10);

}

if (carColor == 2)

{

color = Color.Red;

pos = new Position(1.5f, 10);

}

if (carColor == 3)

{

color = Color.Black;

pos = new Position(3, 10);

}

carColor++;

speed = 0.2f + (float)randomizer.NextDouble() / 18f;

}

public void ResetRace()

{

traveledDistance = 0;

}

public void Create()

{

m = ContentManager.GetModelByName("carro.3DS");

m.CreateDisplayList(); //optimice the model and load it in opengl display lists

m.ScaleX = 0.1f;

m.ScaleY = 0.1f;

m.ScaleZ = 0.1f;

foreach (var item in m.GetMeshes)

{

item.CalcCenterPoint();// calculate th epivot point

switch (item.Name)

{

case "tireA":

case "tireB":

case "tireC":

case "tire":

case "rimA":

case "rimB":

case "rimC":

case "rim":

tires.Add(item);

break;

case "body":

body = item;

break;

}

}

if (color == Color.Blue)

{

texture = ContentManager.GetTextureByName("bodyBlue.jpg");

}

if (color == Color.Red)

{

texture = ContentManager.GetTextureByName("bodyRed.jpg");

}

if (color == Color.Black)

{

texture = ContentManager.GetTextureByName("bodyBlack.jpg");

m.RemoveMeshesWithName("body");

m.RemoveMeshesWithName("tire");

m.RemoveMeshesWithName("rim");

}

}

public void Draw()

{

Gl.glPushMatrix();

Gl.glTranslatef(pos.x, 0, pos.y);

#region race logic

if (Controller.StartedRace == true)

{

//move the object the travelled distance

Gl.glTranslatef(0, 0, traveledDistance);

if (traveledDistance > -59)

{

tireAngle -= 24;

traveledDistance -= speed;

}

else

if (Controller.FinishedRace == false)

{

Controller.FinishedRace = true;

}

counter++;

// if counter == 30 i change the speed

if (counter == 30)

{

counter = 0;

speed = 0.2f + (float)randomizer.NextDouble() / 20f;

}

}

#endregion

m.DrawWithTextures(); // draw the car accessories

#region draw chasis

Gl.glPushMatrix();

Gl.glScalef(0.1f, 0.1f, 0.1f);

Gl.glEnable(Gl.GL\_TEXTURE\_2D);// enable textures

Gl.glBindTexture(Gl.GL\_TEXTURE\_2D, texture);

body.Draw();

Gl.glDisable(Gl.GL\_TEXTURE\_2D);

Gl.glPopMatrix();

#endregion

#region draw rolling tires

foreach (var item in tires)

{

Gl.glColor3f(0.5f, 0.5f, 0.5f);

Gl.glPushMatrix();

Gl.glScalef(0.1f, 0.1f, 0.1f);

Gl.glTranslatef(item.CenterPoint.X, item.CenterPoint.Y, item.CenterPoint.Z);

Gl.glRotatef(tireAngle, 1, 0, 0);

Gl.glTranslatef(-item.CenterPoint.X, -item.CenterPoint.Y, -item.CenterPoint.Z);

item.Draw();

Gl.glPopMatrix();

Gl.glColor3f(1, 1, 1);

}

#endregion

Gl.glPopMatrix();

}

}

}

**Camera.cs**

using System;

using System.Collections.Generic;

using System.Text;

using Tao.OpenGl;

namespace CarRace

{

public class Camara

{

public void SelectCamara(int camara)

{

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

Gl.glLoadIdentity();

switch (camara)

{

case 0:

{

Glu.gluLookAt(2, 3, 14, 2, 0, 5, 0, 1, 0);

break;

}

case 1:

{

Glu.gluLookAt(-2, 7, 10, 1, 3, 1, 0, 1, 0);

break;

}

case 2:

{

Glu.gluLookAt(0, 35, -15, 1, 0, -16, 0, 1, 0);

break;

}

case 3:

{

Glu.gluLookAt(3, 3, -47, 1, 0, 1, 0, 1, 0);

break;

}

}

}

public void SetPerspective()

{

//select the projection matrix

Gl.glMatrixMode(Gl.GL\_PROJECTION);

//reset it

Gl.glLoadIdentity();

//55 = vision angle

//1 = aspect ratio

//0.1f = minimum draw distance

//1000 = maximum draw distance

Glu.gluPerspective(55, 1, 0.1f, 1000);

SelectCamara(0);

}

}

}

**Controller.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace CarRace

{

public struct Position

{

public float x;

public float y;

public Position(float x, float y)

{

this.x = x;

this.y = y;

}

}

public class Controller

{

Car[] cars = new Car[3];

World world = new World();

Camara camara = new Camara();

Road road = new Road();

static bool startedRace;

static int winner = -1;

public static int Winner

{

get { return Controller.winner; }

set { Controller.winner = value; }

}

static bool finishedRace;

public static bool FinishedRace

{

get { return Controller.finishedRace; }

set { Controller.finishedRace = value; }

}

public static bool StartedRace

{

get { return Controller.startedRace; }

set { Controller.startedRace = value; }

}

public Controller()

{

for (int i = 0; i < cars.Length; i++)

{

cars[i] = new Car();

}

}

public Camara Camara

{

get { return camara; }

}

public int GetFirstPlace()

{

float menor = cars[0].TraveledDistance;

int lugar = 0;

for (int i = 1; i < cars.Length; i++)

{

if (menor > cars[i].TraveledDistance)

{

lugar = i;

menor = cars[i].TraveledDistance;

}

}

return lugar;

}

public int GetSecondPlace()

{

int primero = GetFirstPlace();

switch (primero)

{

case 0:

{

if (cars[1].TraveledDistance > cars[2].TraveledDistance)

{

return 1;

}

else

return 2;

}

case 1:

{

if (cars[0].TraveledDistance > cars[2].TraveledDistance)

{

return 0;

}

else

return 2;

}

case 2:

{

if (cars[0].TraveledDistance > cars[1].TraveledDistance)

{

return 0;

}

else

return 1;

}

default:

break;

}

return -1;

}

public int GetThirdPlace()

{

for (int i = 0; i < cars.Length; i++)

{

if (i != GetFirstPlace() || i != GetSecondPlace())

{

return i;

}

}

return -1;

}

public float GetDistanceInMeters(int carro)

{

return cars[carro].TraveledDistance \* -10;

}

public void ResetRace()

{

startedRace = false;

finishedRace = false;

for (int i = 0; i < cars.Length; i++)

{

cars[i].ResetRace();

}

}

public void CreateObjects()

{

for (int i = 0; i < cars.Length; i++)

{

cars[i].Create();

}

road.Create();

}

public void DrawScene()

{

//draw the world (sky & terrain)

world.Draw();

//draw the road

road.Draw();

//draw cars

for (int i = 0; i < cars.Length; i++)

{

cars[i].Draw();

}

}

}

}

**World.cs**

using System;

using System.Collections.Generic;

using System.Text;

using ShadowEngine;

using Tao.OpenGl;

namespace CarRace

{

class World

{

public void Draw()

{

int width = 240;

int height = 200;

int length = 240;

//start in this coordinates

int x = 10;

int y = -3;

int z = 7;

//center the square

x = x - width / 2;

y = y - height / 2;

z = z - length / 2;

Gl.glEnable(Gl.GL\_TEXTURE\_2D);

Gl.glBindTexture(Gl.GL\_TEXTURE\_2D, ContentManager.GetTextureByName("back.jpg"));

//start drawing quads

Gl.glBegin(Gl.GL\_QUADS);

Gl.glNormal3d(-1, 1, 1);

Gl.glTexCoord2f(1.0f, 0.0f); Gl.glVertex3d(x + width, y, z);

Gl.glNormal3d(-1, -1, 1);

Gl.glTexCoord2f(1.0f, 1.0f); Gl.glVertex3d(x + width, y + height, z);

Gl.glNormal3d(1, -1, 1);

Gl.glTexCoord2f(0.0f, 1.0f); Gl.glVertex3d(x, y + height, z);

Gl.glNormal3d(1, 1, 1);

Gl.glTexCoord2f(0.0f, 0.0f); Gl.glVertex3d(x, y, z);

Gl.glEnd();

Gl.glBindTexture(Gl.GL\_TEXTURE\_2D, ContentManager.GetTextureByName("front.jpg"));

Gl.glBegin(Gl.GL\_QUADS);

Gl.glNormal3d(1, 1, -1);

Gl.glTexCoord2f(1.0f, 0.0f); Gl.glVertex3d(x, y, z + length);

Gl.glNormal3d(1, -1, -1);

Gl.glTexCoord2f(1.0f, 1.0f); Gl.glVertex3d(x, y + height, z + length);

Gl.glNormal3d(-1, -1, -1);

Gl.glTexCoord2f(0.0f, 1.0f); Gl.glVertex3d(x + width, y + height, z + length);

Gl.glNormal3d(-1, 1, -1);

Gl.glTexCoord2f(0.0f, 0.0f); Gl.glVertex3d(x + width, y, z + length);

Gl.glEnd();

Gl.glBindTexture(Gl.GL\_TEXTURE\_2D, ContentManager.GetTextureByName("top.jpg"));

Gl.glBegin(Gl.GL\_QUADS);

Gl.glNormal3d(-1, -1, 1);

Gl.glTexCoord2f(1.0f, 0.0f); Gl.glVertex3d(x + width, y + height, z);

Gl.glNormal3d(-1, -1, -1);

Gl.glTexCoord2f(1.0f, 1.0f); Gl.glVertex3d(x + width, y + height, z + length);

Gl.glNormal3d(1, -1, -1);

Gl.glTexCoord2f(0.0f, 1.0f); Gl.glVertex3d(x, y + height, z + length);

Gl.glNormal3d(1, -1, 1);

Gl.glTexCoord2f(0.0f, 0.0f); Gl.glVertex3d(x, y + height, z);

Gl.glEnd();

Gl.glBindTexture(Gl.GL\_TEXTURE\_2D, ContentManager.GetTextureByName("left.jpg"));

Gl.glBegin(Gl.GL\_QUADS);

Gl.glNormal3d(1, -1, 1);

Gl.glTexCoord2f(0.0f, 1.0f); Gl.glVertex3d(x, y + height, z);

Gl.glNormal3d(1, -1, -1);

Gl.glTexCoord2f(1.0f, 1.0f); Gl.glVertex3d(x, y + height, z + length);

Gl.glNormal3d(1, 1, -1);

Gl.glTexCoord2f(1.0f, 0.0f); Gl.glVertex3d(x, y, z + length);

Gl.glNormal3d(1, 1, 1);

Gl.glTexCoord2f(0.0f, 0.0f); Gl.glVertex3d(x, y, z);

Gl.glEnd();

Gl.glBindTexture(Gl.GL\_TEXTURE\_2D, ContentManager.GetTextureByName("right.jpg"));

Gl.glBegin(Gl.GL\_QUADS);

Gl.glNormal3d(-1, 1, 1);

Gl.glTexCoord2f(0.0f, 0.0f); Gl.glVertex3d(x + width, y, z);

Gl.glNormal3d(-1, 1, -1);

Gl.glTexCoord2f(1.0f, 0.0f); Gl.glVertex3d(x + width, y, z + length);

Gl.glNormal3d(-1, -1, -1);

Gl.glTexCoord2f(1.0f, 1.0f); Gl.glVertex3d(x + width, y + height, z + length);

Gl.glNormal3d(-1, -1, 1);

Gl.glTexCoord2f(0.0f, 1.0f); Gl.glVertex3d(x + width, y + height, z);

Gl.glEnd();

Gl.glBindTexture(Gl.GL\_TEXTURE\_2D, ContentManager.GetTextureByName("cesped.jpg"));

Gl.glBegin(Gl.GL\_QUADS);

Gl.glNormal3d(1, 1, 1);

Gl.glTexCoord2f(16.0f, 0.0f); Gl.glVertex3d(x, -0.2f, z);

Gl.glNormal3d(1, 1, -1);

Gl.glTexCoord2f(16.0f, 16.0f); Gl.glVertex3d(x, -0.2f, z + length);

Gl.glNormal3d(-1, 1, -1);

Gl.glTexCoord2f(0.0f, 16.0f); Gl.glVertex3d(x + width, -0.2f, z + length);

Gl.glNormal3d(-1, 1, 1);

Gl.glTexCoord2f(0.0f, 0.0f); Gl.glVertex3d(x + width, -0.2f, z);

Gl.glEnd();

}

}

}

**Road.cs**

using System.Collections.Generic;

using System.Text;

using ShadowEngine;

using Tao.OpenGl;

namespace CarRace

{

class Road

{

int initList;

public void Create()

{

initList = Gl.glGenLists(1);

Gl.glNewList(initList, Gl.GL\_COMPILE);

int texturaDelimitador = ContentManager.GetTextureByName("delimitador.jpg");

//start line

Gl.glEnable(Gl.GL\_TEXTURE\_2D);

Gl.glBindTexture(Gl.GL\_TEXTURE\_2D, texturaDelimitador);

Gl.glPushMatrix();

Gl.glTranslatef(0, 0, 8);

Gl.glBegin(Gl.GL\_QUADS);

Gl.glTexCoord2f(0.0f, 0.0f); Gl.glVertex3f(-0.5f, 0.1f, 0);

Gl.glTexCoord2f(0.0f, 1.0f); Gl.glVertex3f(-0.5f, 0.1f, 1);

Gl.glTexCoord2f(1.0f, 1.0f); Gl.glVertex3f(4f, 0.1f, 1);

Gl.glTexCoord2f(1.0f, 0.0f); Gl.glVertex3f(4f, 0.1f, 0);

Gl.glEnd();

Gl.glPopMatrix();

//end line

Gl.glPushMatrix();

Gl.glTranslatef(0, 0, -50);

Gl.glBegin(Gl.GL\_QUADS);

Gl.glTexCoord2f(0.0f, 0.0f); Gl.glVertex3f(-0.5f, 0.05f, 0);

Gl.glTexCoord2f(0.0f, 1.0f); Gl.glVertex3f(-0.5f, 0.05f, 1);

Gl.glTexCoord2f(1.0f, 1.0f); Gl.glVertex3f(3.5f, 0.05f, 1);

Gl.glTexCoord2f(1.0f, 0.0f); Gl.glVertex3f(3.5f, 0.05f, 0);

Gl.glEnd();

Gl.glPopMatrix();

int texturaAsfalto = ContentManager.GetTextureByName("asfalto.jpg");

Gl.glEnable(Gl.GL\_TEXTURE\_2D);

Gl.glBindTexture(Gl.GL\_TEXTURE\_2D, texturaAsfalto);

Gl.glPushMatrix();

Gl.glTranslatef(0, 0, -100);

int count = 0;

for (int y = 0; y < 40; y++)// this for loop draws the road

{

Gl.glBegin(Gl.GL\_QUADS);

Gl.glTexCoord2f(0.0f, 0.0f); Gl.glVertex3f(-0.8f, 0, count);

Gl.glTexCoord2f(0.0f, 1.0f); Gl.glVertex3f(-0.8f, 0, count + 10);

Gl.glTexCoord2f(1.0f, 1.0f); Gl.glVertex3f(3.8f, 0, count + 10);

Gl.glTexCoord2f(1.0f, 0.0f); Gl.glVertex3f(3.8f, 0, count);

Gl.glEnd();

count += 10;

}

Gl.glPopMatrix();

Gl.glEndList();

}

public void Draw()

{

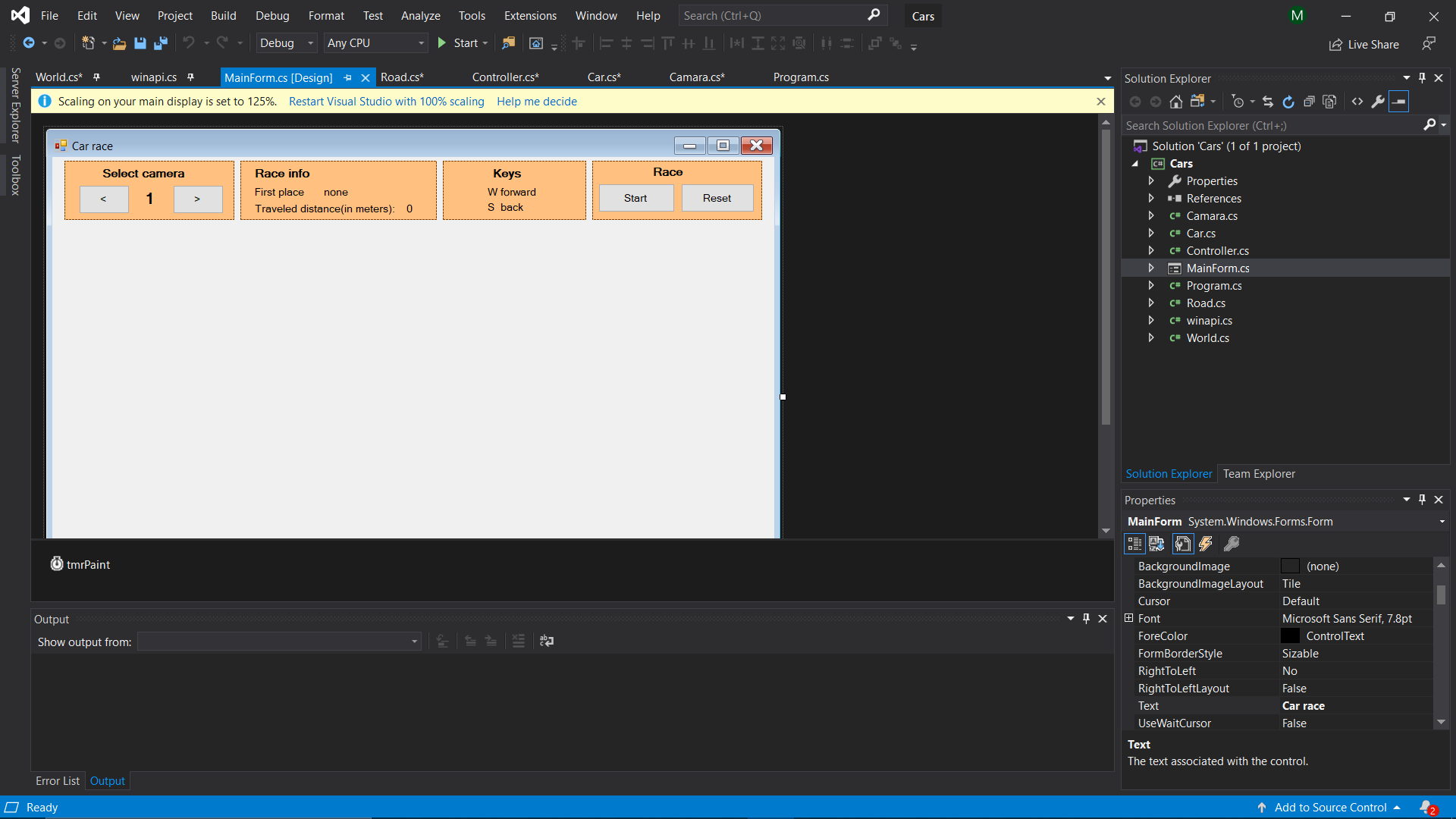
Gl.glCallList(initList);

}

}

}

**MainForm.cs**

****

**Program.cs**

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace CarRace

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

}

**Conclusion**

This project was successfully developed and executed with the help of OpenGL and Visual Studio.

This project highlights the important topics for development of games with extensive study in the same would enable me to create better games. With the help of multiple files and graphic tools this game can be improved further with addition of control to cars and addition in textures of the cars and world.