**Speed Life**

**BY**

**Mahmudul Hassan Fuad**
**ID: 171-15-1402**
**AND**

**Pronab Biswas**
**ID: 171-15-1396**

**AND**

**Jonayet Ahmed Shakil**

**ID: 171-15-1403**

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

Supervised By

**Md. Reduanul Haque**
**Sr. Lecturer**
Department of CSE
Daffodil International University

Co-Supervised By

**Zarin Tasnim Shejuti**
**Lecturer**
Department of CSE
Daffodil International University


**DAFFODIL INTERNATIONAL UNIVERSITY**

**DHAKA, BANGLADESH**

**DECEMBER 2020**

# APPROVAL

This Project titled "**Speed Life**", submitted by *Mahmudul Hassan Fuad* and *Pronab Biswas* and *Jonayet Ahmed Shakil* to the Department of Computer Science and Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on *05/12/2020*.

## BOARD OF EXAMINERS

**(Name)** **Chairman**
**Designation**
Department of CSE
Faculty of Science & Information Technology
Daffodil International University

**(Name)** **Internal Examiner**
**Designation**
Department of CSE
Faculty of Science & Information Technology
Daffodil International University

**(Name)** **External Examiner**
**Designation**
Department of -------
Jahangirnagar University

# DECLARATION

We hereby declare that, this project has been done by us under the supervision of **Md. Reduanul Haque, Sr. Lecturer** Department of CSE Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

**Supervised by:**

_____

**Name: Md. Reduanul Haque**
Sr. Lecturer
Department of CSE
Daffodil International University

**Co-Supervised by:**

_____

**Name: Zarin Tasnim Shejuti**
Lecturer
Designation
Department of CSE
Daffodil International University

**Submitted by:**

_____

**Mahmudul Hassan Fuad**
ID: 171-15-1402
Department of CSE
Daffodil International University

_____

**Pronab Biswas**
ID: 171-15-1396
Department of CSE
Daffodil International University

_____

**Jonayet Ahmed Shakil**
ID: 171-15-1403
Department of CSE
Daffodil International University

# ACKNOWLEDGEMENT

First we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the final year project/internship successfully.

We really grateful and wish our profound our indebtedness to **Md. Reduanul Haque**, **Sr. Lecturer**, and **Zarin Tasnim Shejuti, Lecturer** of Department of CSE Daffodil International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of "*Computer Game*" to carry out this project. His endless patience ,scholarly guidance ,continual encouragement , constant and energetic supervision, constructive criticism , valuable advice ,reading many inferior draft and correcting them at all stage have made it possible to complete this project.

We would like to express our heartiest gratitude to **Md. Reduanul Haque**, **Sr. Lecturer**, **Zarin Tasnim Shejuti, Lecturer** and Head**,** Department of CSE, for his kind help to finish our project and also to other faculty member and the staff of CSE department of Daffodil International University.

We would like to thank our entire course mate in Daffodil International University, who took part in this discuss while completing the course work.

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

# ABSTRACT

We have worked over 1 year in 3D modelling objects like cars and solid objects. Modelling 3D objects was fun but it was so lifeless to watch this models do nothing but just stay in head drive. So we have decided to make a game using our own 3d modeled cars to make use of this 3D models. So in this report, where we discussed about a case study about developing a 3D game where player can race, by the help of in Unity game engine for making the game. Unity is well known nowadays for making realistic graphics games like Racing game, FPS (1rst person shooter) games, Shooting games and so on. It also provides a chance for student package so students can use it and make their skill in this field for free. Is also has different platform export option for run the game in different platform such as PC, Android, IOS, Linux etc. For 3D modelling, we have done using "Blender" 3D modelling software. This is the most popular free platform for 3D modelling for games and all king of 3D modelling field. It also has different type of tools for making 3d model easily. So in this thesis we will try to cover the implementation of game, game engine and 3D models we used in this game. We will also cover Al physics engine, real time graphics effect also.

# TABLE OF CONTENTS

# LIST OF FIGURES

# CHAPTER 1

## Introduction

**1.1 Introduction:** Developing a game is very time consuming and hardworking process. We can play game for hours and not being hesitate for a moment but the hard work and effort to make a simple movement in games need huge thinking and hardworking ant patient. During last several years' game development have become widely popular by software developers.

**1.1.1 About graphics:** This game we are making primarily contain models of 3-dimension environment, and real time render engine. It is fully required to have 3D environment for running the game. This game is only build for 3d platform. Just in case, 2D platform could still be openable for 3D objects.

**1.1.2 Outstanding result:** The sport result must addict the player of the game. The interest of the game should be durable, and this excitement will draw the players to come back and play the game again and again.

**1.1.3 Graphical effects –** In the purpose of having a powerful result, we add modern graphical effects Using Unity Universal Render pipeline. It has some outstanding features, like real-time rendered in game and shadow effect, motion blur, and ambient occlusion. For the above expectation, we choose Unity 3D as our developing platform to develop our game. This decision we are making because this platform had many built-in tools-kit and deliver a good framework process for us to start our game development really fast in this short time possible. One of another reason that we choose Unity 3D because is used JavaScript as development language for making game.

The requirements for our game that contain 3D graphical environment was indeed an interesting but hard working process in 3D modelling field since we had not much knowledge in 3D modelling. It takes few 100 hours of working result in 3D modelling. In

ongoing research, we have to find out what 3D modelling platform or program we have to use, and finally we found some different platform studios to make 3D models that we can use to import our game project. The so we choose to make our 3d model in Blender.
The first and foremost thing for any 3D game is 3D object. After making desirable 3D object then we will be able to make game in game engine. So in this chapter we will talk in details how we make the 3D object models for the game.

# CHAPTER 2

# Game Design

**2.1 Design concept of the game:** This project what we were working, it was our choice to decide what type of game we wanted to make and develop. The idea was, we are going to make a racing game what would be perfect, because this type of game not necessarily dependent on vary amount of assets, such as human model, hard surface complex model and animated models etc. After some thinking, it had been decided by us that we are going to develop a racing game. There were two different game concept idea behind the game, which we described below. The sport concepts are evolved with our development process, as we follow to add features in the game we are making. For better explanation how the sport concept evolved in gaming, the process has been differentiating into three steps.

**2.1.1 First Conceptual Step:** In our first imagination we want to make a game almost like EA Games (Electronic Arts), called “Need for Speed Hot Pursuit 2010”, where players had a car, and there the goal was to take down the opponent’s and earn the first place to win the race. Our another concept was to make some fancy race tracks where game players could be practicing for winning the races. Though the main idea involved much greater working process, it had been decided by us that the main concept was to be the main and foremost priority, where the second concept were to be treated as our high priority.

**2.1.2 Next conceptual Step:** We were kind of lost that we thought this is a great time or step to join the errors that we are facing, that was the option to make players select cars and race with different type of experience. That is more like the steps that a lot of players

appreciate. We also decided that the racing track should be on an open road, which is decorated with walls barriers, and various objects suitable for that seen environment. Because of driving a real car at a great speed is almost impossible in busy roads, so we decided that we should always make computer controlled cars instead for practicing driving.

**2.1.3 Last Conceptual Step:** For developing the game in next level we have to take ideas from the second step about how we will implement out game for the final result. In this step we have to give up our first step idea and give this idea a low-priority for developing in our last step. So we flashback and try to make a game that will be loved by modern games as we will develop out game quality and overall display and game physics.

## 2.2 Development Process

For developing software projects, it has to fulfill many requirement and verity of qualities. Some example of quality requirements is: popularity, validity, maintenance, dependencies and usability. To fulfill this type of verity, od demands, it is very important to schedule the workflow on a well prepared strategy. For game development engineering, the term for such a technique is usually referred to software making process, which is based not one but some popular software process models.

### 2.2.1 About a discussion of Software Process Model.

Software process model may be or might be a theoretical or academicals philosophy that describes simple implementation of the way of developing any kind of software. This are based on semi or non-semi models; a process model is made for providing the leadership on how to operate. Software process models might be brief discussion of a software or program development process. The concept of the process model is analogous to an abstract java class, which can't be instantiated, but it is often implemented by another class, thus providing basic guidelines for that other class. A model may for instance demand customer involvement, but it does not state exactly how. A process implementing that

model should involve the customer within the process' activities, but is liberal to choose how.

There are few numbers of software process model, but the most common are tow. The number one is the most common, and we are about to discuss it. The second type of process model is called a paradigm process model, which is a model that is more general than a standard process model. This above discussion about a model does not store any details about the activities that cause the completion of a project

We have to find out what process model is suitable to particular software and we could have a higher chance to get benefit from such model and so. On behalf of that, we can conclude a process paradigm provides a workflow (Framework) that is adapted to make a process which is suitable for a particular project or projects. The waterfall model is widely used in present days' software engineers, which is component-based software, where the engineering and development process are evolutionary.

### 2.2.2 Discussion about waterfall Model:

Waterfall model is suitable for large and complex systems projects that has to have a long run support. There are other systems that carry this type of attribute. The main concept is to take all the activities and treat them separately. One activity is always followed by another, in the same way water travels down some falls. This description becomes even more obvious when looking at a visualization of the model

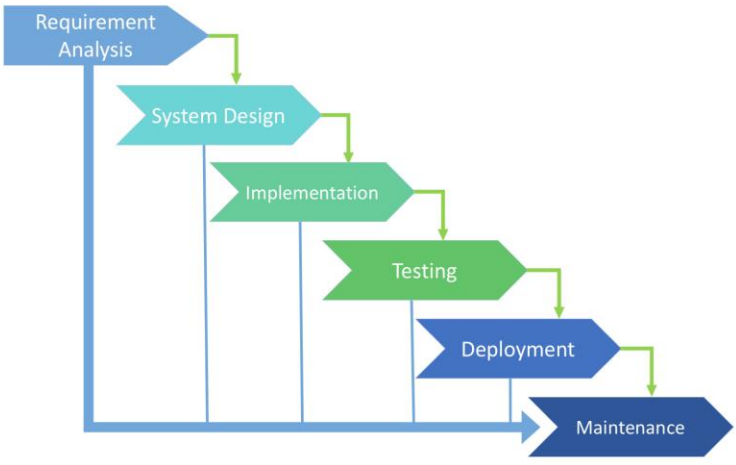



Fig 2.2.2.1: DP- Waterfall Process Model.

Some Explanation of the model:

1. Requirements definition All requirements on the system are found by talking to system users.
2. Program and software architectural design are materializing in this step where the full architecture of the system is completed in this step.
3. This part is also known as unit testing which is after implementation of software it test for bug and fix it in this step.
4. Regeneration and framework testing The units are consolidated together into a total framework. Advance testing is required.
5. Operation and support The framework is conveyed to the client and put into operation.

There is no way that one can find any software program without bug. There for it is very major necessary to have bug fixing maintenance.

# CHAPTER 3

## Game Engine

**3.1 Game Platform:** For the purpose of saving time and development process we choose Unity 3D game engine for making our game.

Unity game Engine: Unity is a well-known game engine for today's generation. It can provide game for android, Windows, Linux, Console Like PS4, Mac OS etc. First announced just for Mac OS, at Apple's Worldwide Developers Conference in 2005, it's since been extended to focus on quite fifteen platforms. Unity game engine have many package for verity of worker, it is available for anyone to use to individuals or commercial companies. This software makes around US$100,000 of annual gross sales in verity of packages. It was March 3, 2015 when it is release of Unity 5.0. This game engine is popular for making games which has capability to focus on games like multiple games story based games strategy games etc.

**3.2 User Interface:** It is not important that we need to import anything at this point at this point at all, but one thing is to specify that the file or project save location is on the dialog window and there it is located. It is a place to store all type of files that our project will contain. It helps to find any broken links and missing links while we are developing our game development process.

**3.2.1 Viewport:** It is the point where we place any 3D object in your Unity Game Engine environment. This engine has real time update feature while previewing and changes in the game engine instantly. In the top right we can see the manipulator which allows the user to change variety of view in the platform. In default we are in perspective view we can change the view by clicking in the middle box in the top right corner in viewport. We can move the 3d platform using by clicking Middle mouse button and hover the mouse. We can also rotate the seen using by clicking Right mouse button and hover the mouse in such direction.

**3.2.2 Mode Change (Game and Editing):** If we want to see how the game will look like or let say we want to see a certain point of the game that how it will look in the game we can change the view port mood from edit to game mood, where we will be able to see how it will look. Another thing is that, if we don't like any certain point and we want to change the point and remake a view seen after watching the game view, this game mood will help a lot in this type of situation.

**3.2.3 List menu (Hierarchy view):** This is the list view that we are creating in the game project package. It will show all of the objects, seen the we are making or importing from the computer. We can also inspect objects that are in an object like, a car is an object and it contains many objects like bumper, front fender, side kit, roof, head light, back light, seats, dashboard, body-kit so and so on. So with the help of the hierarchy option it is easy to find.

**3.2.4 Project/GameAssets view:** this is often an inventory of all custom assets for our game, which contains- graphical assets, sound asset, scripts, prefabs (pre- assembled game objects) and much more.

**3.2.5 Inspect tools/ examiner:** It will be completely blank if nothing is selected by us in the first place. This inspect-tools allows us to observe and change any individual settings of variety in selecting or deleting any game objects which is known as asset, farther more we can change some global settings in the game that will help a lot of time saving in long run. This contains sensitive option that can very helpful if we just edit it from here rather than go and change in the code.

**3.2.6 Graphical Icon for Movement:** This graphical icon is very helpful for moving object, scale object, rotating object etc. The icon on its right, which seems like four arrows, allows you to maneuver a specific object around. We call this transforming the object. The next icon allows for rotation of the thing, and therefore the final one allows for uniform scaling of the thing.

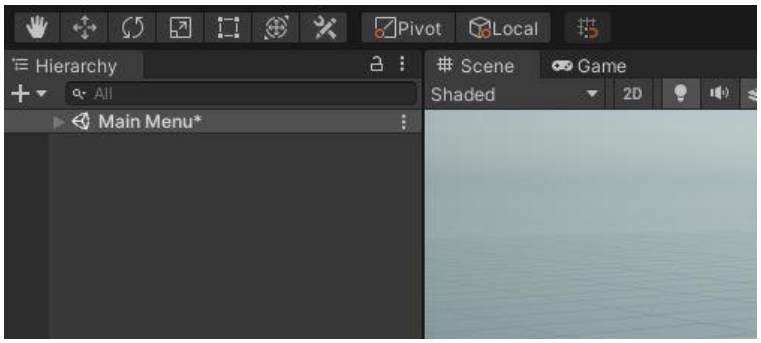



Fig 3.2.6.1: Graphical Icon.

**Quick Movement Tool:** The shortcut of this option is 'Q' which allow us to rotate and move around in the scene-view to move user camera view.

**Translate Tool:** The shortcut for this is 'X' which active selection tool.

**Rotate Tool:** The shortcut of this option is 'E'. It helps to rotate any set of object in X, Y, Z location.

**3.2.7 Playback Bar:** Following option allow us to play pause any seen we are running and we can also stop the seen in the middle of the process. The easiest way to run and stop the game in development process.

Fig 3.2.7.1: Playback Bar.

# CHAPTER 4

## Game implementation in Game Engine.

**4.1 Car Physics:** In this game we use Realistic car controller V3 for controlling and handling of the cars. This section will describe all of the things about how we implement our game and first thing we would like to discuss about the "Torque" angular velocity utilization. It has many benefit for simulating the car in realistic motion, and one of another important thing is that we don't need to apply any multiplication of movement with respect to time. In every frame it applies a certain amount of force to speed-up specific object. For the anisotropic function of the controls script it will never go sideways. It is a very necessary script tools for running any sorts of racing or wheel based game. We can also apply different settings by using this script like car drifting effect etc.

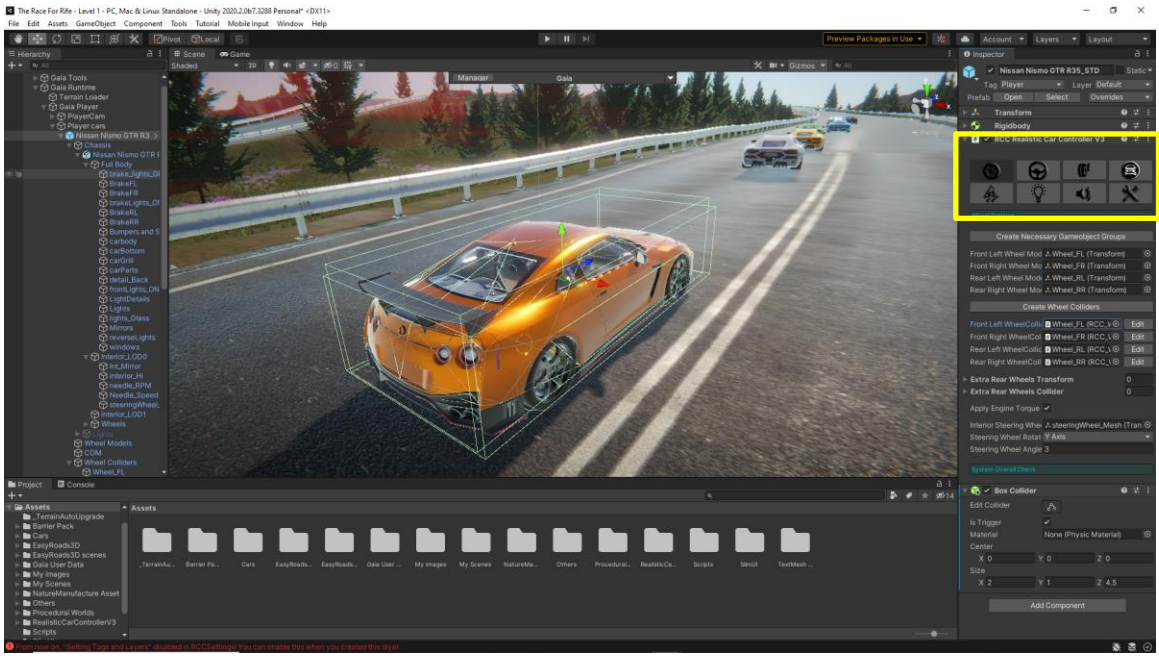
For lower the complexity we will use three basic simple steps to create the primary ray casting of a car for the player. In the beginning of the process would we will add some colliders for the car, it could be other object covers the car or the car mesh collider along. Then we will add collider to every four wheel so that we can have the traction effect on the tracks.

All of the procedure are added by the menu-bar from the Realistic car controller v3 menu which we discussed in details in below:

**4.2 Advance Car Physics:** Here we will discuss detailed about the car physics that we used. We used realistic car controller v3 for both player car and the AI cars. The AI cars run by script "RCC AI controller Script.

- Player Car:

Fig 4.2.1: Realistic car controller Unity 3D



The car is mainly controlled by this controller script of Unity game engine. It makes our workflow easier for implement the realistic car driving effect.

Her first we see a wheel logo this section is for controlling the rotation of the wheel obj of the car. We can see front and rare wheel of the car are being assigned in this necessary geometry group, and wheel collider are assigned under the create wheel collider menu. There are also starring wheel rotation option so, when the wheel rotates in left and right direction the starring wheel will also rotate. It gives the whole seen a realistic look. The wheel collider script helps the car object to stay on top of the track. If there were no collider of wheel object, then the wheel object will be drowning from the track. As if the tire object has no friction of the ground.

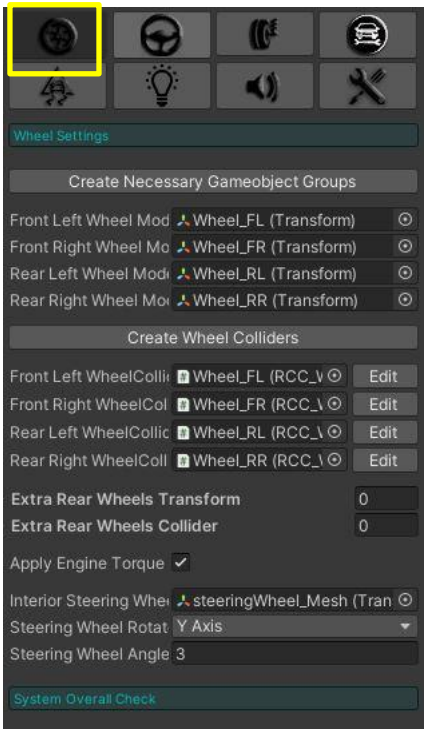



Fig 4.2.2: RCC Controller

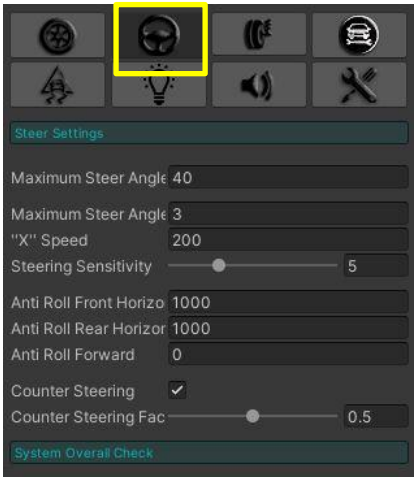



Fig 4.2.3: RCC Controller

Here we can see the steering wheel physics option. Here we can apply the car handling option. This will create different driving experience of different cars. We can make a car handling easy or hard from this section. We assigned the steer angle 40 for better rotation of the car that this assigned to.

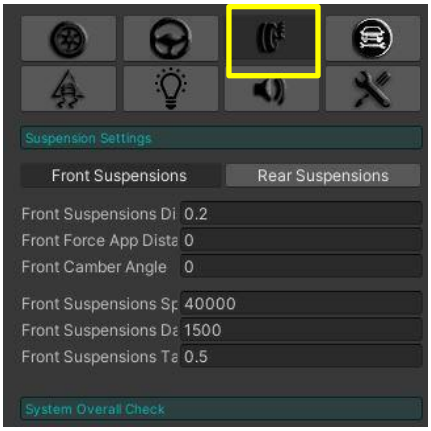



Fig 4.2.4: RCC Controller

This picture indicates the suspension effect of the car. By applying suitable values for front and back suspension, the car will have suspension effect against any bump on the road collation and against gravity like real life car do. We assigned it as suitable value as we tested what value works the best o such cars.

In this picture what we can see is the car engine configuration panel. This is where we configure the RPM, Speed, Horse power of the cars, gear shifting delay etc. We can also assign what type of car it would be. We can choose FWD which is a forward engine car. We can also select back engine cars like AWD which indicates the all-wheel drive cars. We can also assign how many gear the will have. Gear shifting delay and threshold is used for the delay between the next gear of the engine. The engine torque is used for assigned for the torque of the engine. The brake torque ration controls how fast the car will slow down from it velocity. We can also have assigned the top speed of the car in maximum speed section.

The Engine RPM section is the most important part of the car engine section this will effect on the driving of the car that how fast the car gains top speed fast. Higher RPM giver the car higher horse power to gain speed fast and lower rpm giver the car lower horse power where the car will speed up slowly.

There are other options where we can assign Nitro and Turbo effect of that will give the car more realism compare to real life.

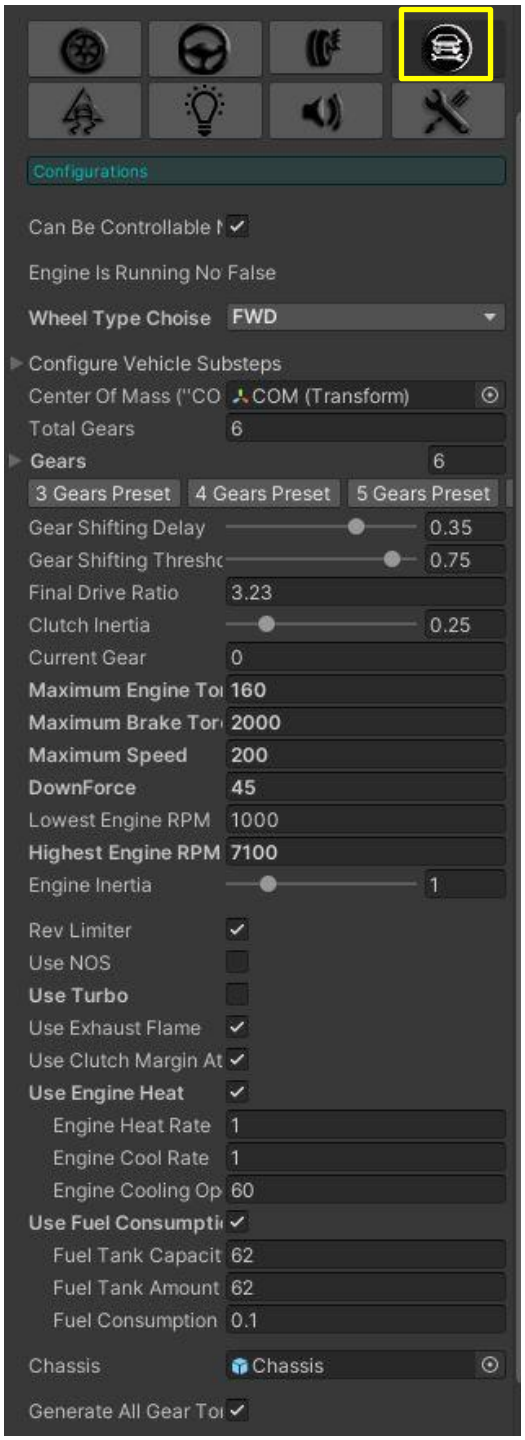



Fig 4.2.5: RCC Controller

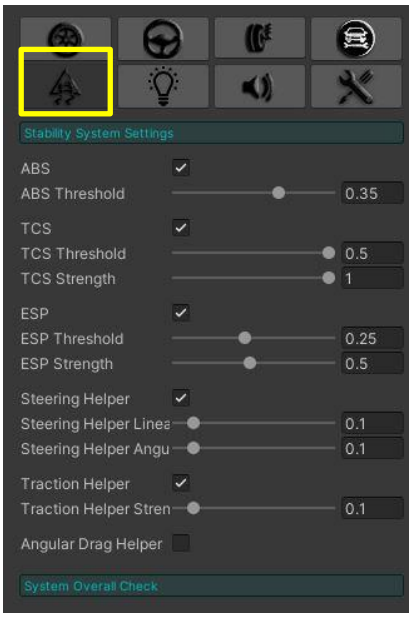



Fig 4.2.6: RCC Controller

In this section what we can see is the Abs Braking system of the controller UI. It makes the car driving experience more real. We can achieve the drift and break effect like real life by using this option.

- AI Cars: From the beginning till now we were busy making and optimizing on getting a car under the control human player. But we really need another car on the track, because it is a racing game and other car will try to take us down or make it hard for winning the race. It turns to be the foremost difficult task so far. But with the help of RCC AI car controller script it become very easy for us to implement it in our game. To begin the section, we'll start by creating a car that drives itself round the race track. The AI car object will be duplicated in the scene to make multiple opponent cars. The opponent "AI" cars of this game also run through this script and since its AI so it needs one more script to rum itself, which is RCC AI Controller Script.

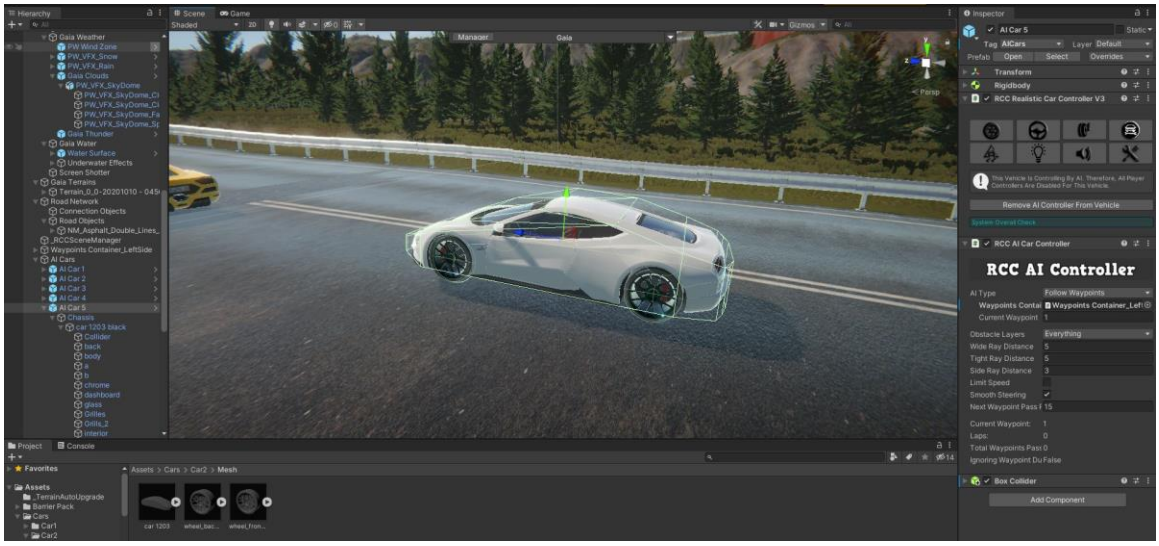


Fig 4.2.7: AI Cars.

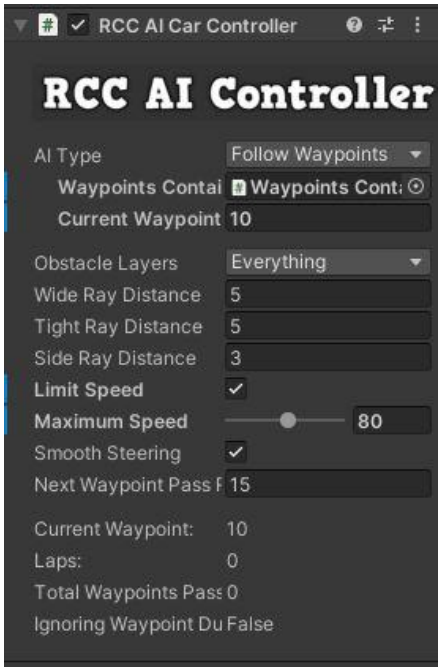



Fig 4.2.8: RCC AI Cars Controller Script Menu Unity.

The AI cars runs by using several waypoints set on the road. This RCC AI controller makes it easy and time saving for us to make the waypoint faster. There is a picture below where we can see how the waypoints for the AI cars are look like.


Fig 4.2.9: RCC AI Cars Controller Script Menu Unity.

The light blue circle gismo type mesh are the waypoints for the AI Cars.

**4.3 The Game Camera:** The game camera is simply the camera where the device can capture the world or seen ahead to the players. The polish of this shorter flat solid branch is to make and manage a tally photographic to camera to follow the participant 's cable elevator railroad car because it washes rung the car t runway. In a racing plot of ground, one among the foremost park camera position is from a third-mortal perspective; that's, behind and slightly above the player. This chore will display you path to make during a lone amongst one in every of one among these photographic cameras then to maneuver it along in a fluid mode with the player railroad line car. In more advanced plot you would possibly wish to line up numerous camera to capture the natural process from different angles. This more composite plant special outcome is completed by enabling and disabling various camera supported drug substance abuser remark.

We start again and open the Camera Control Scripts for Smooth Camera view. Now we have to attach the final Camera script to the camera that we want to smooth. Then we connect the target variable to the car object. We also did it earlier for the car camera movement. Now we are seeing the result of the camera script that smothers the rotation view along y-axis which is the height while still maintaining a static horizontal distance. This script gives you tons of control over how the camera behavior will be needed in the

game because we are allowed to change many the variables. For every of these smoothed values we calculate a wanted value and therefore the current value.

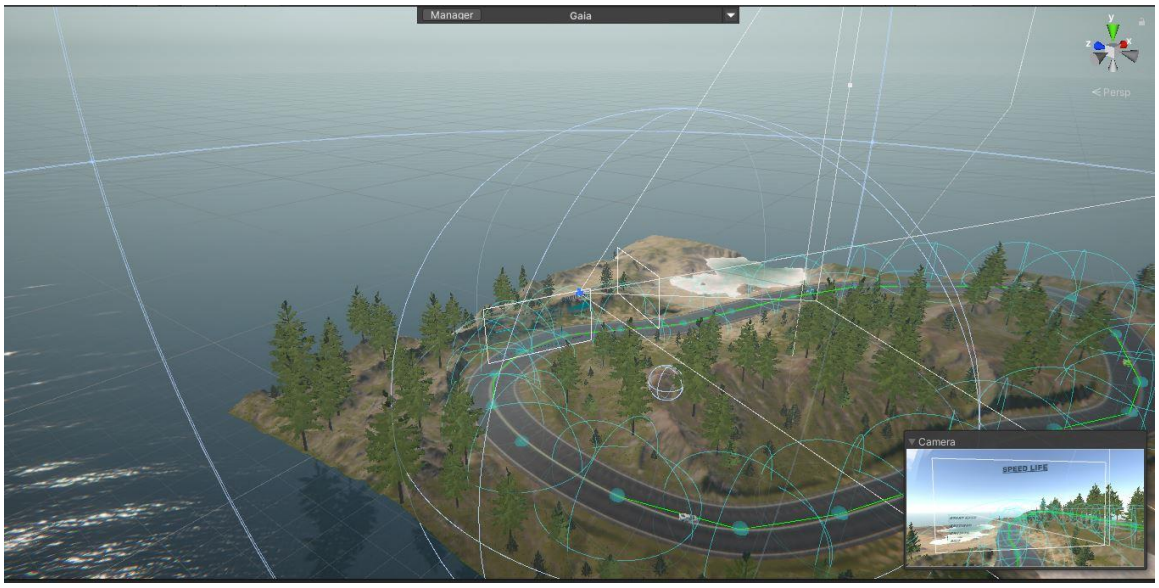



Fig 4.3.1: Game Camera Unity.

**4.4 Creating Terrain:** Terrain is the base where all of the seen and objet took places. We can make different type of terrain in Unity. Now we will talk about how we make the terrain of the game.

Simply we right click on hierarchy menu and then > 3D object and > Terrain and our terrain is being created. This is how the default terrain will look like:

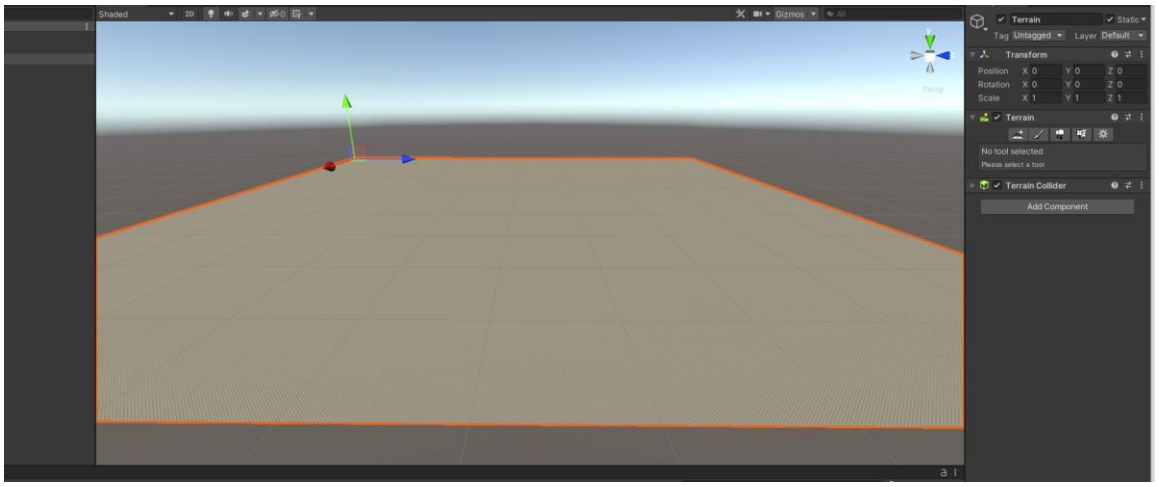



Fig 4.4.1: default terrain Unity.

Now for the shape of the terrain that we want to give it we will use Terrain brushes. There are different types of brusher for editing terrain in Unity we choose what is suitable for us for making the terrain.

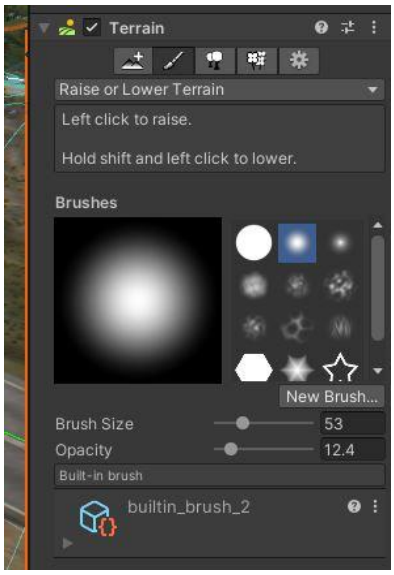



Fig 4.4.2: Brush editor for terrain Unity.

In the way of making out terrain the terrain will start to look like this.

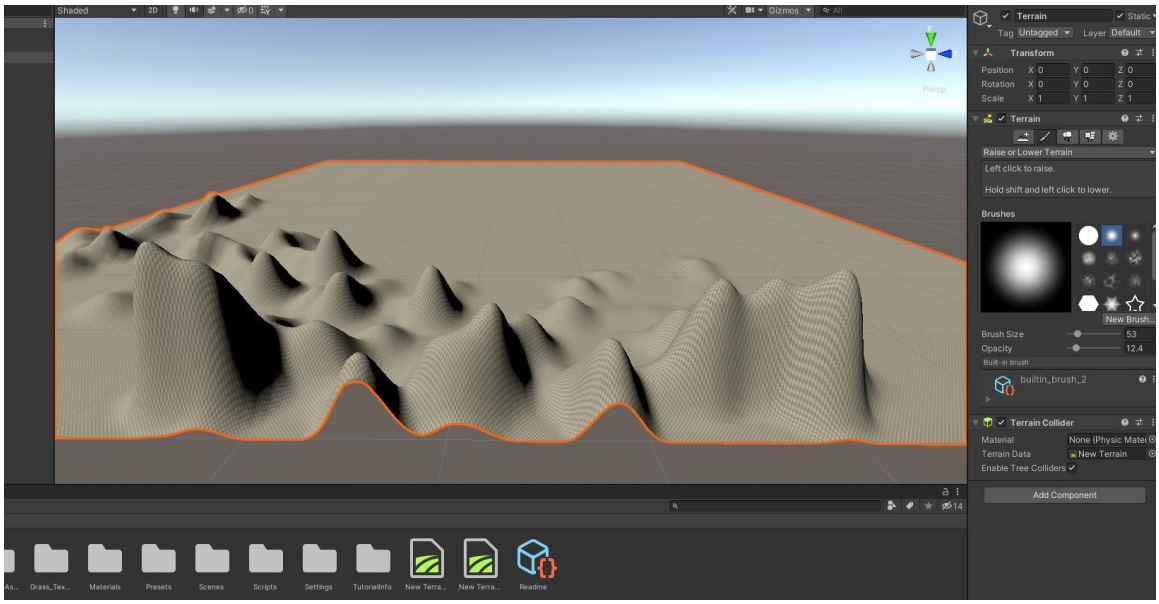


Fig 4.4.3: Ongoing terrain editing.

After a long hard work on shaping the terrain and assigning the material, trees, and grasses the final result of the terrain we achieved was very good. This is our final result of editing of the terrain.

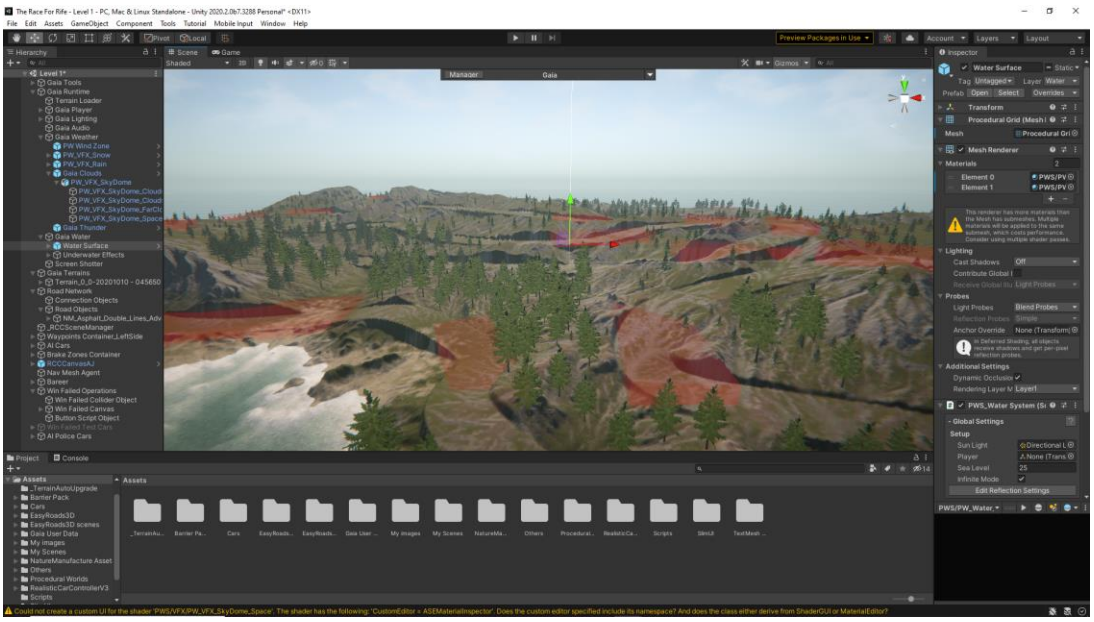


Fig 4.4.4: Final Level 01 Terrain.

**4.5 Track Design:** After finishing the terrain editing we add road mesh where the actual racing will happen. The final result was very good. This picture shows how it looks in the game.

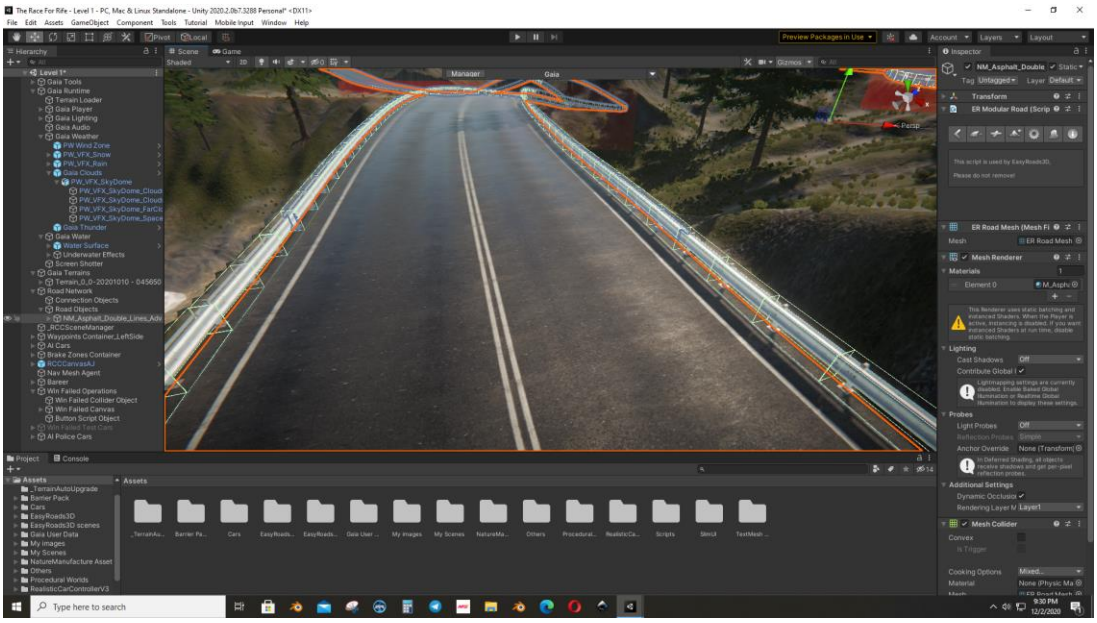


Fig 4.5.1: Road mesh on the Terrain.

**4.6 The procedure of making a track in unity is given below:**

Select New EasyRoads3D Object from the EasyRoads3D Menu. A dialog window will appear where you'll name the new road object. The new road objects are going to be

added to the hierarchy panel after clicking the Create Object button. After creating we will see in the inspector menu this tool bar.

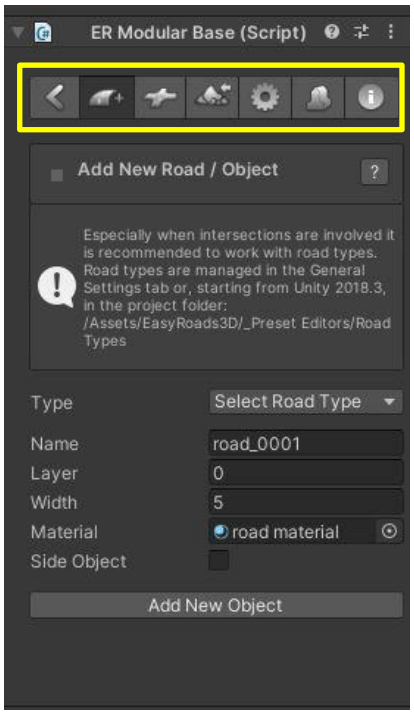



Fig 4.6.1: Road inspector option.

Move the mouse to the position where we would like the road to start out and click on the left push button while holding the [shift] key. Continue adding markers according the form of the specified road. We will see that the road system will create surfaces representing the road and the affected surrounding.

Insert road markers works similar as adding road markers but rather than adding the marker at the top of the road, it'll insert the marker between the two closest markers to the mouse position. We make sure in Insert Markers mode by activating the Insert Markers toolbar tab from the right inspector menu.
After we give our waypoint and finally create the road waypoint the final simple result will look like this given picture:


Fig4.6.2: Creating road mesh in unity.

In terrain the after assigning the material to the road and barrier on the road the road looks like this:

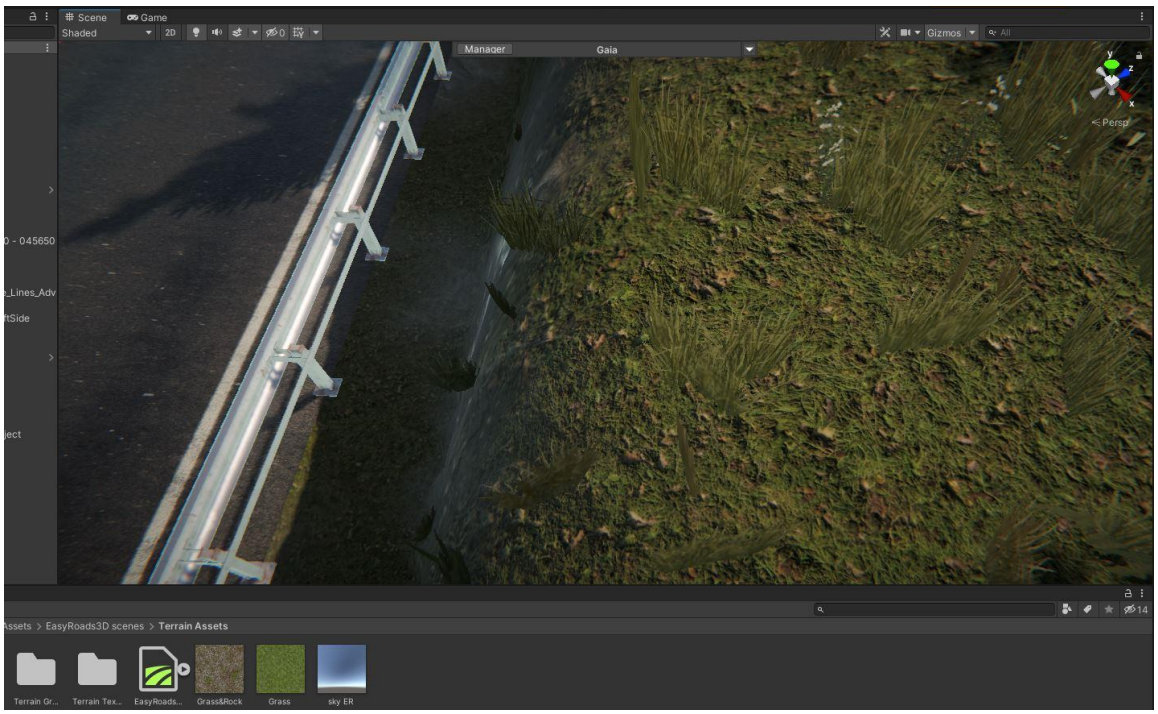



Fig 4.6.3: Final Road with material and barrier.

**4.7 Side Object Geometry Controls:**

1. Index selection: Whenever we select some value/s that is/are going to be displayed in this index selection menu.

2. Object Tracer: This tool could be used to analyze the geometry and it will also by itself define the form of the geometry accurately. We just have to make sure that the vertices wont to connect the start and end of the assets with one another are at the same position or null position, what will be shown in the base of our modelling platform or software.

3. Trace On: counting on the rotation of the asset we may need to switch between x-axis, y-axis and z-axis until we clearly see the form that the procedural geometry should have. By default, objects are traced on the z-axis when opening the editor window and no points are stored yet for this side object.

4. Auto Connect: Below things is described where we will clearly see that the dots are positioned correctly but they are doing not connect within the right order. Auto Connect are going to be enabled once we select the primary point, from this it will connect all points according the closest neighbor. Depending on the geometry structure it's going to simplify reordering of point indexes.
5. Mirror Horizontally: This mirrors the form horizontally and is beneficial when correcting the form of a duplicated side object or when it seems that tracing the geometry of the beginning or end asset leads to a flipped shape.

6. Mirror Vertically: This mirrors the shape vertically.

7. Flip Faces: it's going to happen that the traditional point within the wrong direction, we will notice this when rendering the side object. This button will flip the faces so it will appear correctly. This feature is also available in the Side Object Inspector.

# CHAPTER 5

# Modelling Cars.

## 5.1 Blender:

Blender is open source 3d modelling platform for anyone to start 3d modelling and get use to 3D modelling profession. With Blender, you'll create 3D visualizations like still images, 3D animations, VFX shots, and video editing. It is compatible to individuals and little studios who enjoy its unified pipeline and responsive development process.

Being a cross-platform application, Blender runs on Linux, Mac OS, also in Windows systems. It also has relatively small memory and drive requirements compared to other 3D modelling software like “Autodesk MAYA”, “Autodesk 3DS MAX” “Fusion 360” etc. Its interface uses OpenGL to supply a uniform experience across all supported hardware and platforms.

Blender is user friendly and easy to use for making any kind of 3D model. For first time there will be some difficulties for user but with the flow of time everything will be easy as user gain knowledge about how it works. There are tons of hot keys for speed up the modelling process in blender. So one can easily master this and speedup their modelling speed greatly.

## 5.2 Platform:

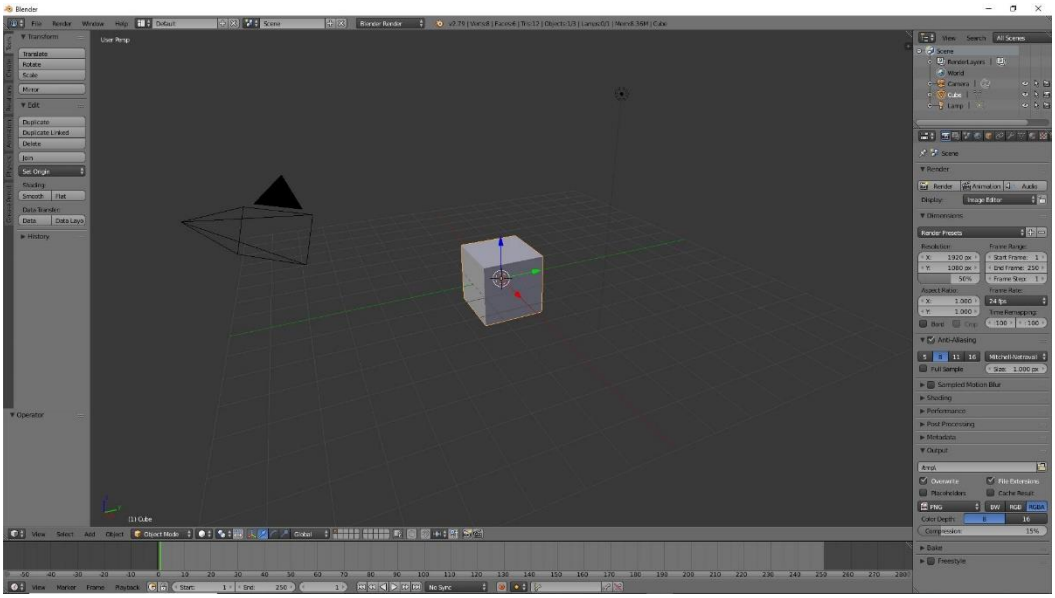


Fig 5.2.1: Blender Platform

What we are looking at is the first seen when we open the Blender 3D modelling software. In the beginning we have our default 3D object Cube artificial lamp and a camera in the middle center which is called 3D platform. In top we have our preference settings and save options. In left we have our all desirable mesh tools and so on. In the right side we have our 'Render' 'Modifier' 'World' and 'Material' options and details of our individual meshes on the top. In the bottom we have our layer LODS (level of details) some mesh options to select vertex, edge, and faces, and key frame for purpose of animations.

We can move our 3D platform using mouse. To move the platform, we have to click and press middle mouse button and hover the mouse to move the platform in various direction. To zoom in and out we use scroll wheel. For changing the view of platform from User perspective view to User Orthographic view and vice versa we use numpad key 5 and we can change view angle along 'x', 'y', 'z' such as 'side', 'top', 'front' using numpad key 3, 7, 1 respectively. We select any object using 'Right mouse button'. To edit an object, we have to go to edit mode by selecting the object first and hit the tab key in the keyboard and start editing the object shapes. We can start our modelling from different shapes. To add any available shapes, we simply click and 'Add' in left bottom side or we use hot key 'Ctrl+A' to add any basic available basic shapes.

So that was some basic instruction to observe and model 3D objects in blender.

## 5.3 Modelling Cars in Blender:

Modelling car in any 3d modelling software is very time consuming. In case of blender there is no exception for it. However, modelling is fun. The concept cars we have made from our own imagination and skill by working 1,000 of hours in blender day by day. Although the procedure I same but the making procedure of real world super cars that we have used in the game is given below. So this is the first thing we need to do.

**5.3.1. Setup Cars blueprint:** To setup blueprint we have to find the blueprint of the car that we will use in the game. We can simply find blueprint from various website like - www.the-blueprints.com , www.pinterest.com , www.drawingdatabase.com etc.

Once we have found our blueprint we need to setup in our 3d modeling software for modelling the cat object.

In platform press ‘a’ to select all and press ‘x’ and enter to delete all unwanted objects. Then press ‘5’ in numpad in the keyboard to go to orthogonal view of the platform and pressing ‘3’ in numpad in the keyboard we will make our ‘Right orthogonal’ view.

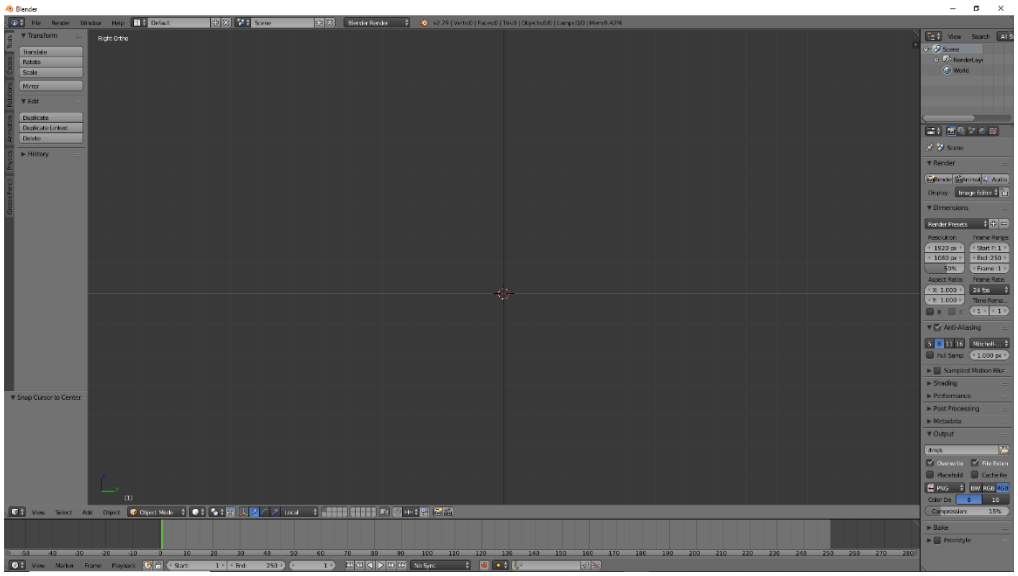


Fig 5.3.1.1: Blender Platform

So we open our Properties menu by pressing ‘N’ in the keyboard and Check the ‘Background Images’ then expand the background image option and add four image by clicking add image four times.

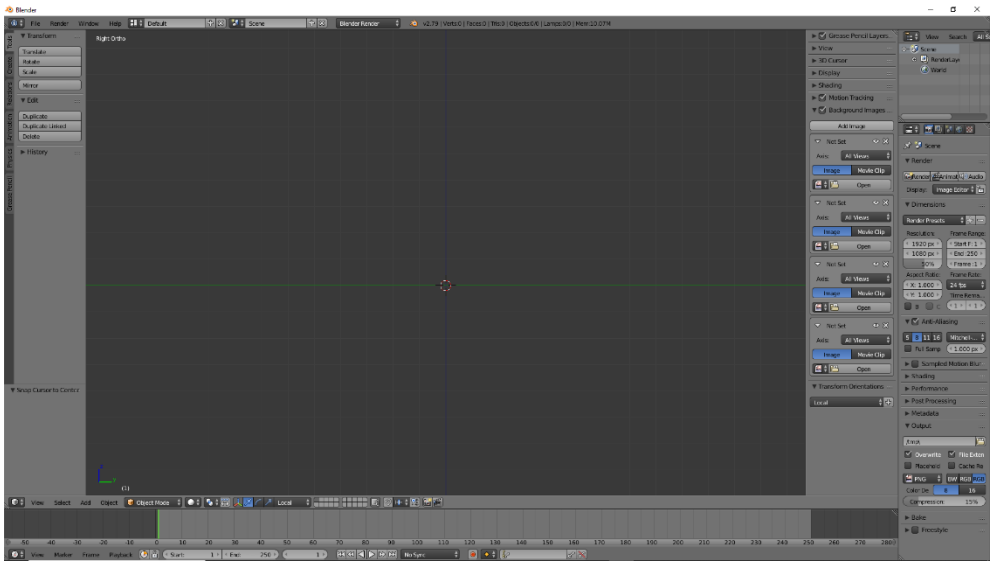


Fig 5.3.1.2: Blender Platform

After adding the image, it will look like this.

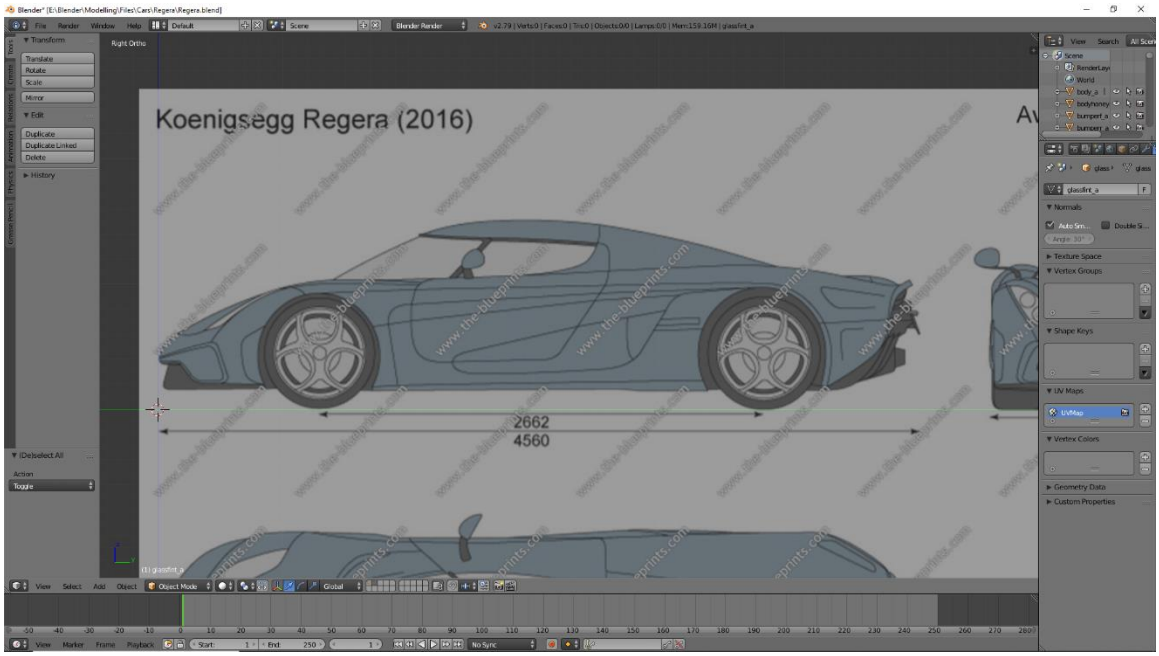



Fig 5.3.1.3: Side View of the Blueprint-Blender Platform

We have to reposition all four view angle to the horizontally and vertically along the x, y, z axis. The image 'Fig A4: Blender Platform' shows the perfect positioned Right side view image of the blueprint along Z & Y axis. So for the 'Top View' 'Front View' and 'Back View' of the blueprint image after repositioned it will be look like the image below.

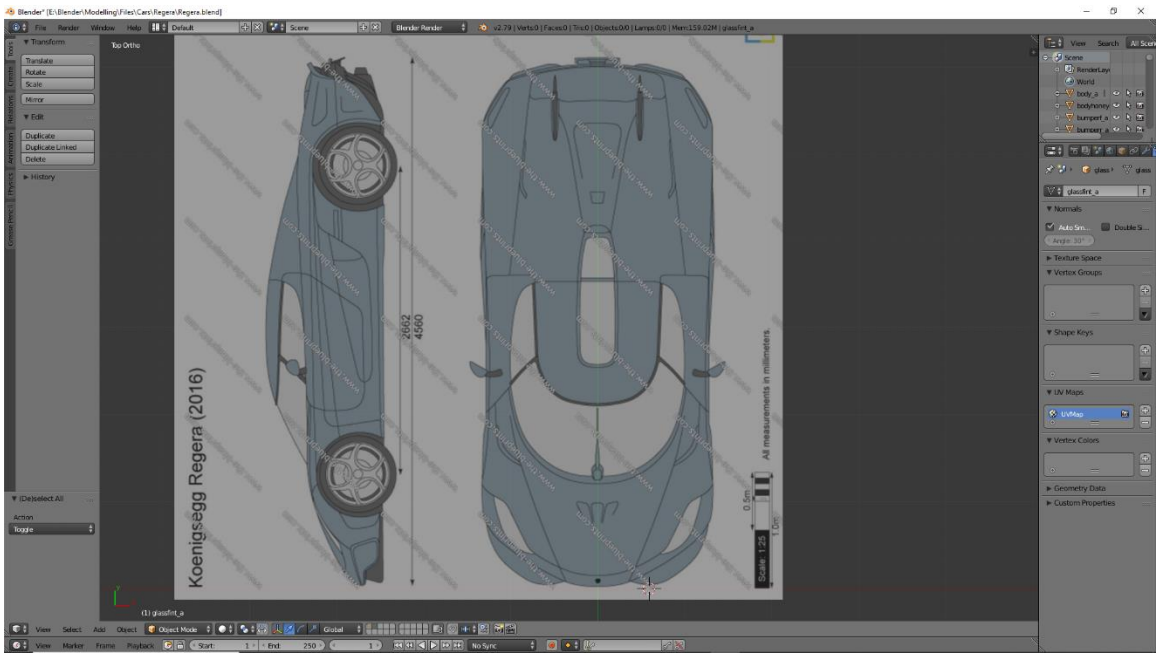



Fig 5.3.1.4: Top View of the Blueprint- Blender Platform

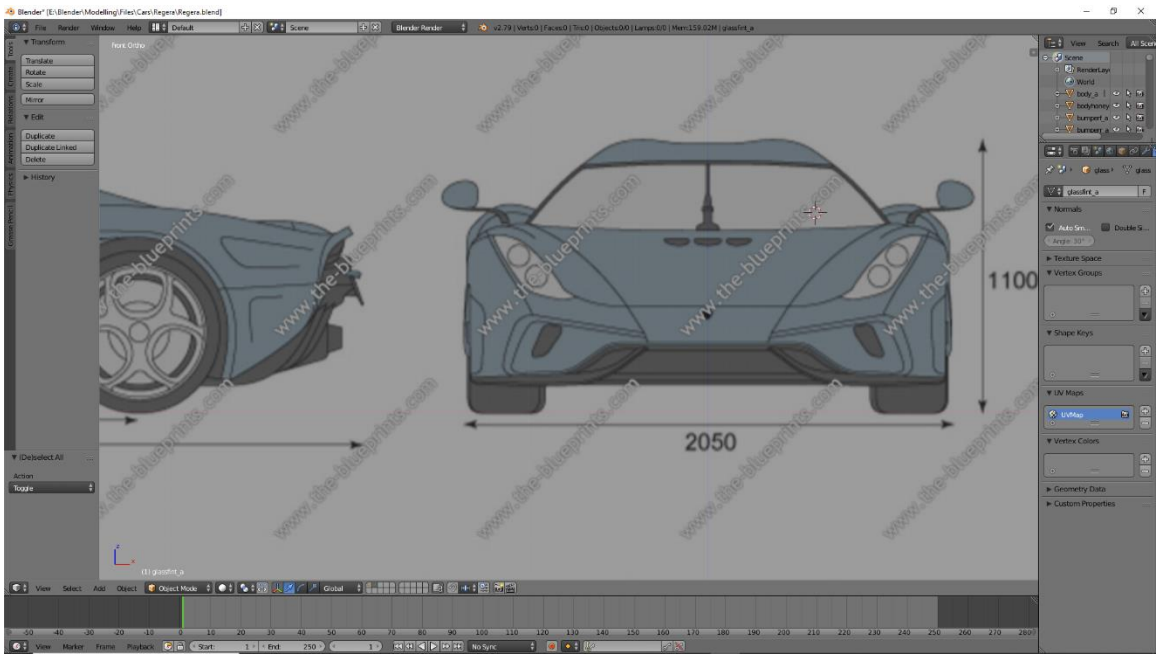



Fig 5.3.1.5: Front View of the Blueprint-Blender Platform

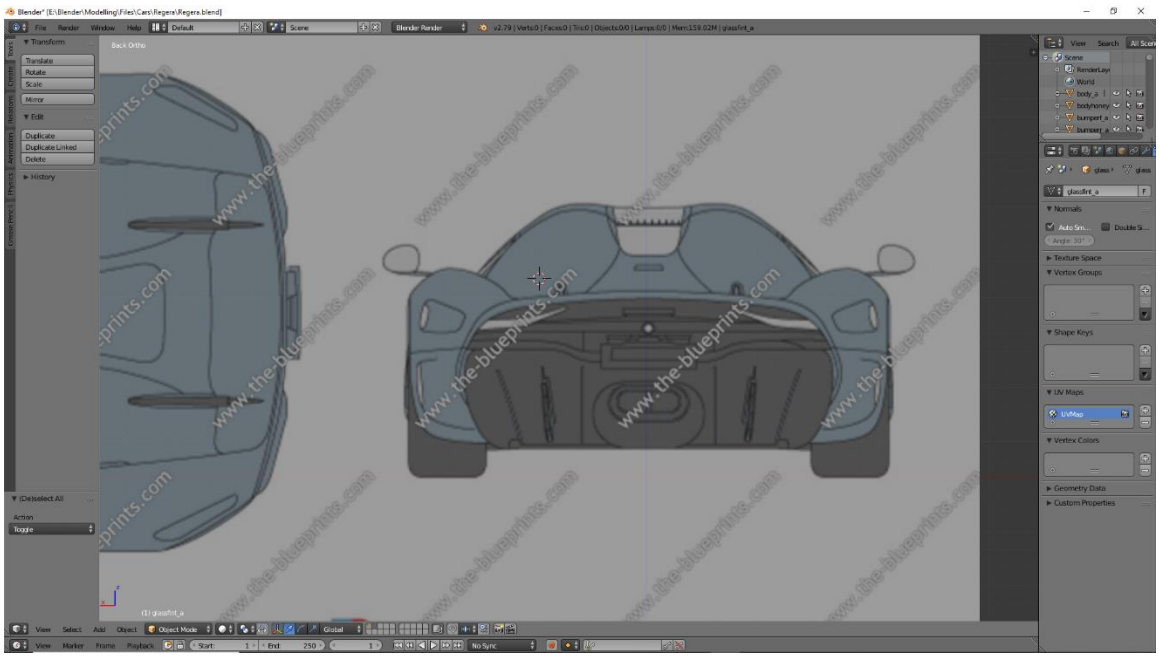


Fig 5.3.1.6: Back View of the Blueprint-Blender Platform

This setup of the Blueprint is the first and far most step for modelling car model. It is very mediatory to make a perfect 3D model of the car we desire.

After setting up the blueprint we start modelling our car.

### 5.3.2. Start creating Object:

We start modeling our object from the basic shape. We add basic shapes in the way of modelling as we needed.

So we add a mesh and start editing the polygon and give the desirable shape of the car using the blueprint. We used various modifiers to make our modelling along the way which helps to reduce modelling time and effort. Although the complex modeling of car is very time consuming and there are many flow of working process, thus for the simplicity of the major understanding we will show some workflow images that we took in the way of our modelling for this car 'Koenigsegg Regera 2016'.

Here are some images that describes our workflow about how we create the car objects and make the whole set of object and give it a realistic look.

**5.3.3. UV mapping:** UV Mapping is that the process of transferring a 3D mesh from a 3D model to a 2D space to further texture the model. UV Maps represent the essential principle of making textures, which is employed by all applications.

We have to UV map all of the objects to assign color and texture to the mesh that we have created.

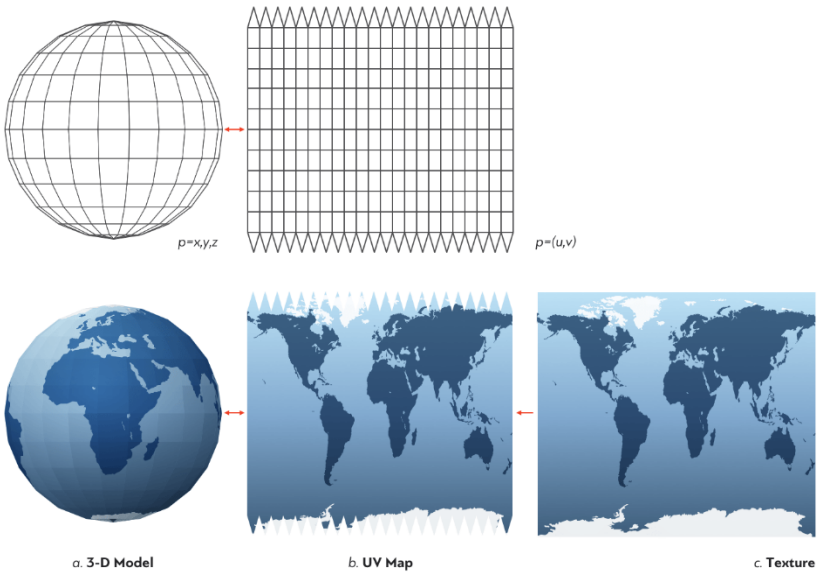



Fig 5.3.3.1: UV Mapping and texture making.

For UV mapping in blender, we select a single mesh object then press 'Tab' key to go to edit mood and press 'U' (shortcut key for UV mapping) in and select 'Unwrap'.

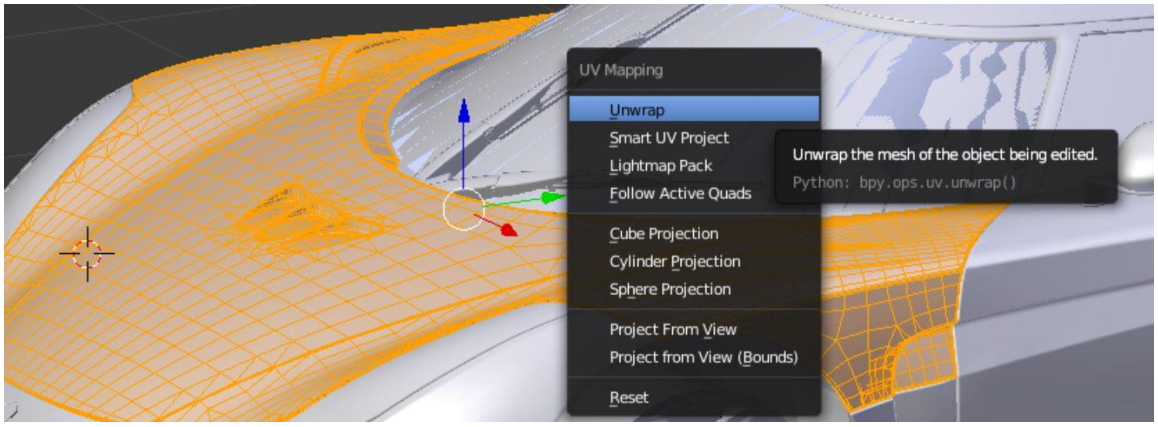



Fig 5.3.3.2: UV Mapping and texture making.

In the bottom side we expand and select the option 'UV/Image editing'. It will open a platform for making our UV edit

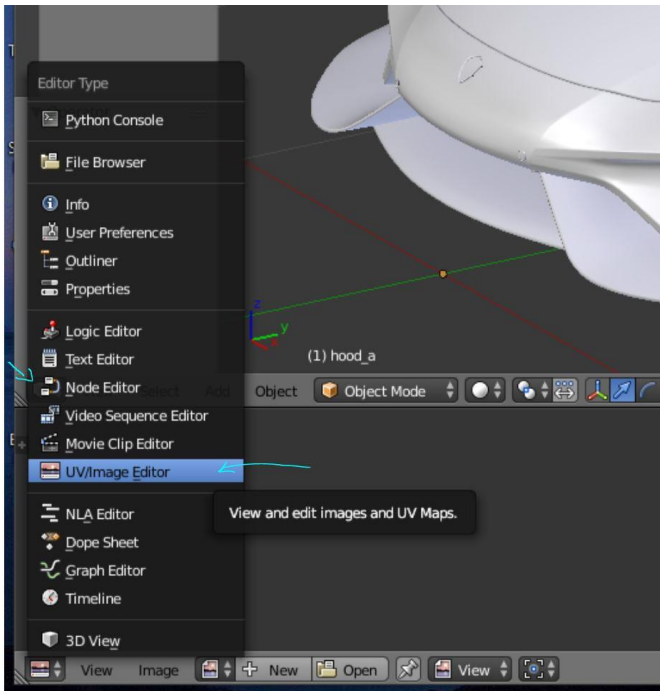



Fig 5.3.3.3: UV Mapping and texture making.

Here it will visible the 2D form of the object that we selected.

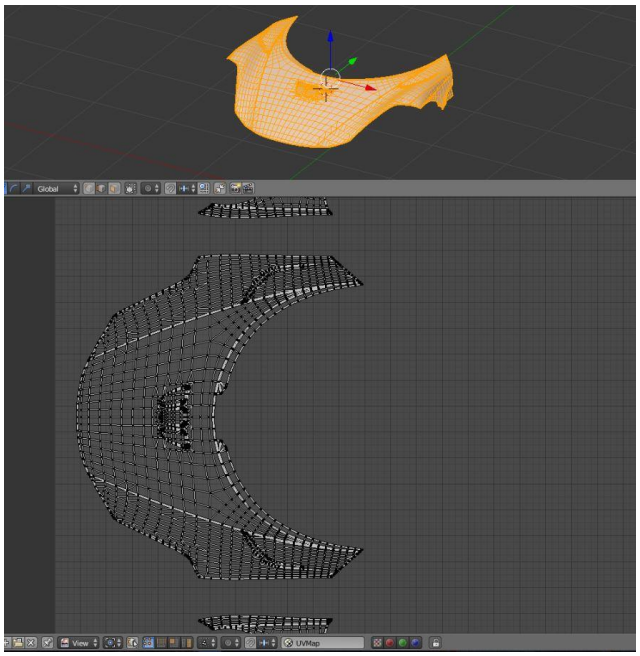


Fig 5.3.3.4: UV Mapping and texture making.

Here we will see our 2D mesh form of the 3D mesh we have created. Now it is ready to assign textures.

### 5.3.4. Assigning materials:

After Unwrapping all object meshes we have to assign materials to the objects. To add material in blender. This is the look without any material of the car:

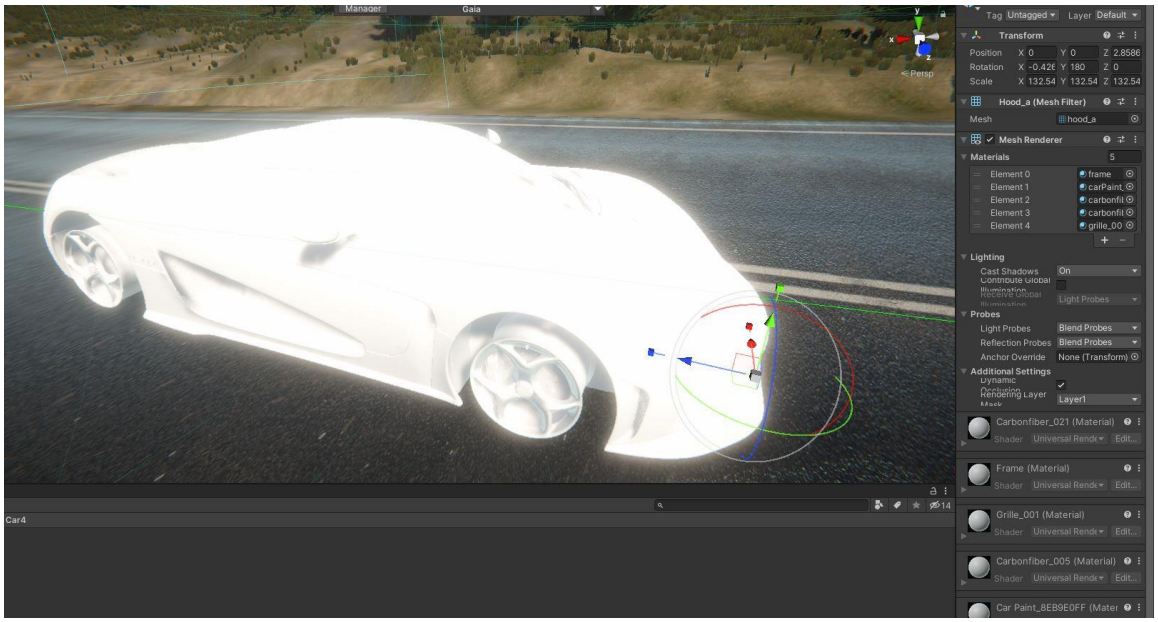



Fig 5.3.4.1: Raw 3d model without any material.

For assigning a material first we select an object mesh to put material on and assign material from the left side menu bar.

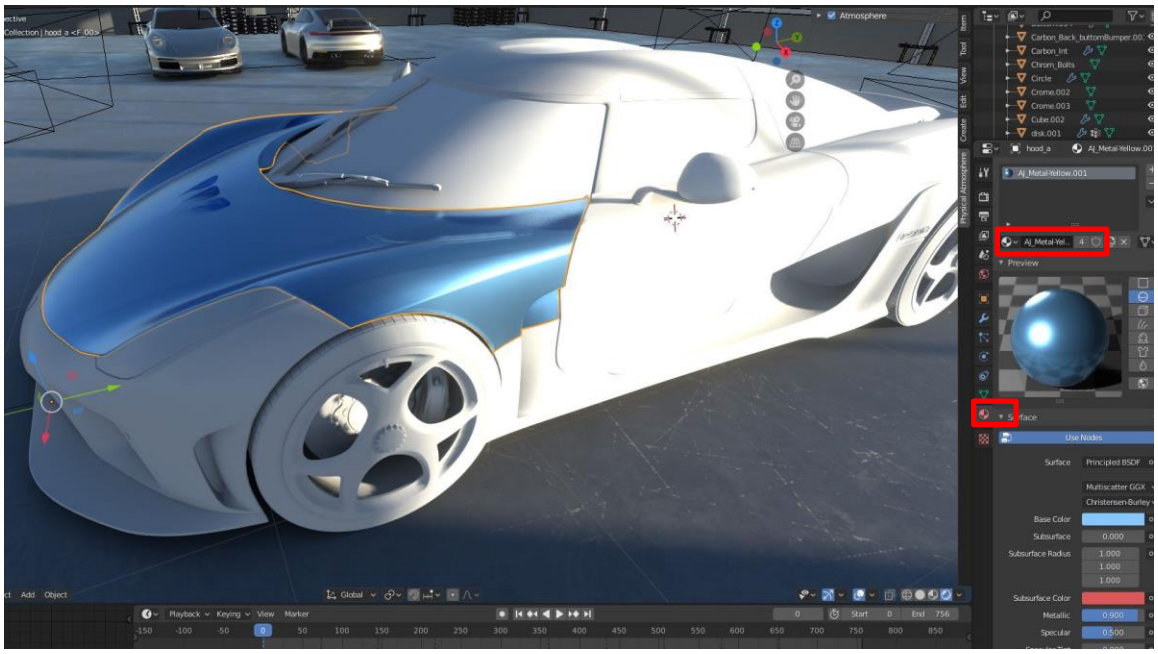


Fig 5.3.4.2: Assigning material in body parts.

Then we will work our procedure on making desirable material by changing base color, and change the value of roughness of the surface, making clear cote and things like that in the node editor.

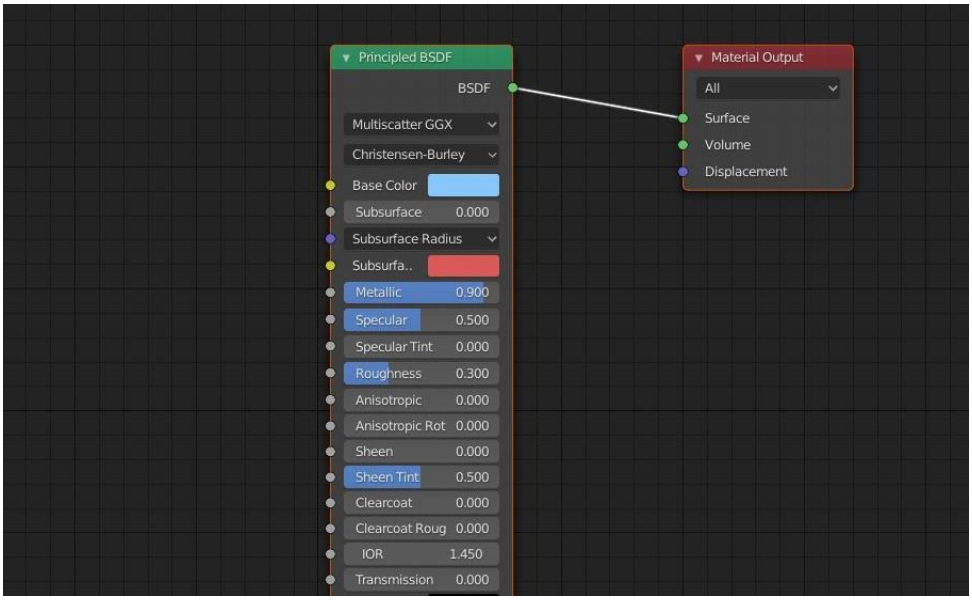



Fig 5.3.4.3: Material node editor.

After finishing assigning all of the material in every object of the car here are some final result of the look of our car.

Fig 5.3.4.4: Final Look View 1.


Fig 5.3.4.5: Final Look View 2.

Fig 5.3.4.6: Final Look View 3.

### 5.3.5. Exporting for Game Engine:

After finishing texturing and assigning material to the objects meshes it is finally ready to export for game engine.


Fig 5.3.5.1: Export for game (FBX format).

# CHAPTER 6

## Coding

**6.1 Code**: The code is not much as this is a development based project but there are some code the we have to write on our own.

Win Fail Code:

```
using System.Collections;
using System.Collections.Generic;
using System.Runtime.CompilerServices;
using UnityEngine;
using UnityEngine.UI;
public class WinFailedCollider : MonoBehaviour
{
    public GameObject WinPanel;
    public GameObject FailedPanel;
    public GameObject WinFailedObject;
    // Start is called before the first frame update
    void Start()
    {
        WinPanel.SetActive(false);
        FailedPanel.SetActive(false);
        WinFailedObject.SetActive(true);
    }
    // Update is called once per frame
    void Update()
    {
    }
    void OnTriggerEnter(Collider other)
```

```
    {
        if (other.tag == "Player")
        {
            WinPanel.SetActive(true);
            FailedPanel.SetActive(false);
            WinFailedObject.SetActive(false);
        }
        if (other.tag == "AICars")
        {
            FailedPanel.SetActive(true);
            WinPanel.SetActive(false);
            WinFailedObject.SetActive(false);
        }
    }
}
```

Main Menu Button Script:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class MyButton : MonoBehaviour
{
    public string GoToMainMenu = "";
    public string CurrentLevel = "";
    void Start()
  {
    }
```

```
    // Update is called once per frame
    void Update()
    {
    }public void GoTo_MainMenu()
    {
        if (GoToMainMenu == "Main Menu")
        {
            //StartCoroutine(LoadAsynchronously(GoToMainMenu));
            SceneManager.LoadScene("Main Menu");
        }
    }public void Play_Again()
    {
        if (CurrentLevel == "Level 1")
        {
            //StartCoroutine(LoadAsynchronously(GoToMainMenu));
            SceneManager.LoadScene("Level 1");
        }
    }
}
```

There are some major car controller and AI car controller script that too large so we Uploaded it in our GitHub .
**So the link is given below:**

**GitHub Link:** MahmudulHassanFuad/Game-project-code.: The Code script we used in our game project (github.com)

# CHAPTER 7

# Conclusion

## 7.1 Conclusion:

It was a good experience for all of our group members. Our goal was to make a different type of game and we believe that we are going in right direction. We would like to make this project big and try to make a full publishable game one day.

For larger projects, we expect that planning and having an honest structure, become even more important. For example, if we wanted to make some major changes to our game, we would probably have to discard a lot of previous work, because our lack of good structure and generally written code. So we will gradually to make our skill in coding.

We understood that there are many more to explore and our knowledge was so little in amount. But we achieve a success in making out first step. So we hope we can overcome any obstacle in our way.

# Reference

This are tutorials were very helpful in the way of our journey. We found this specific channels video very helpful for making our 3D models and implement game in our unity game engine.

1. Making racing game in unity: https://www.youtube.com/watch?v=ehDRTdRGd1w
   **Author:** Jimmy Vegas
2. Material Making in blender: https://www.youtube.com/watch?v=moKFSMJwpmE
   **Author:** Grant Abbitt
3. Gaia - Unity Asset Store: https://www.youtube.com/watch?v=lEJFBil9idcb
4. Master Car Creation in Blender: https://www.youtube.com/watch?v=dwoq53Bjb3Y
   **Author:** CG Masters