Saurjya Mukhopadhyay

Introduction

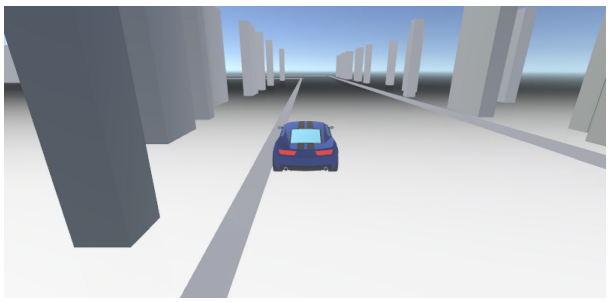
I created a driving simulator game using Unity in which the user can control a vehicle on a ground map and drive around the “streets” while avoiding certain obstacles such as buildings. Instead of creating a game with points or scores, I created more of an experience/simulator in which the user can drive around as they please. The view that the user sees when starting the scene is a hind view of the car and using the computer’s arrow keys, the user can control where the car goes on the map however they please. This project is interesting, because I feel as if a lot of modern day video and computer games are made in a similar way. I am very interested in driving cars and cars in general myself, so I decided to make my project related. The way that we can create an experience where the user can use the keyboard to control a car within a virtual reality is intriguing and inspires me to learn more about the creation of these types of games. VR makes this project’s experience more interesting because of the 3D aspect. It feels like the user is really able to drive around on the street, seeing objects pass by and new objects pop up the further the vehicle travels. I could develop this project even further in the future to model particular cities, such as LA or New York (which is my goal for winter break). Overall, the project is still in the stages of full development. The car’s driving functionality and the streets / track in the scenes are completely finished, but there are many other things that could be added to make this a game people would enjoy daily.

Related Works

There are many related works from which I got inspiration from for my project. Originally, I wanted to create a visual simulator of driving from the point of view of a driver in the driver's seat. After the professor mentioned augmented reality, I was inspired by the augmented reality feature that is found in a lot of modern day cars in 2020. Essentially, this feature allows the dashboard of the car to become a full view of the road in front of them in real time. Surrounding vehicles will move in real time as you view your dashboard, so looking at the road basically is not essential. I wanted to make something like that for this project, so I would create a view from the driver's seat in which you can view the dashboard and the road in front of you. However, I felt that this project would be very complex, so I decided to make a simple driving car controller. There are many video/computer games that are similar to the simulator that I have created, such as Forza Horizon, Need for Speed, and Grand Theft Auto. These incredible games act as the main inspiration for how I modeled my project, especially the view that the camera gives us. However, my game is strictly for driving in free-roam mode around the city I have created. More components could be added in the future, such as a competitive racing aspect like Need for Speed or the ability to have a character and exit the vehicle and walk around the city, like Grand Theft Auto. However, my game is very simple compared to all these games, but the controls are even more simple which is an advantage.

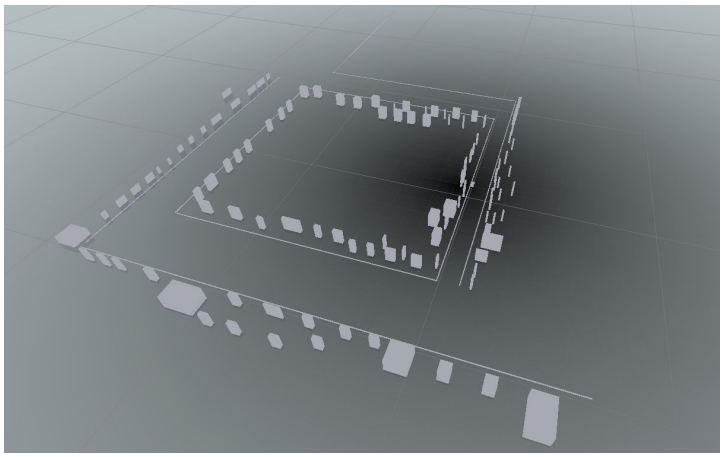
Design

The design of my project is very simple. There is a ground level plane on which the car drives, so this is basically the road. The ground is what I had to create first. For the roads in my project, I used very long horizontal cubs to act as the curbs to the "sidewalks". There are basically 5 main roads on which you can drive on, but it goes in a square format as shown in the picture of the map from a high viewpoint below. The simulator is designed so that you can make loops on these streets and return to the same position if you prefer. In between the "roads", there are many buildings placed, which you have to drive around or you will crash (all of them are also considered "game objects".) Then, most importantly, there is the car game object. I imported the design of the car from a package from iMena games. Around this game object, I placed a box collider, which will help us create this driving simulator. Obviously, there is a main camera that follows the car along its route, which captures the footage that the user sees when he or she uses this simulator. The view of what the user will see is shown in the picture below; the camera is placed almost right behind the vehicle.



The way in which you control this car and drive it along the streets is using the keys on your computer's keyboard. The up arrow key is used to accelerate forward and the down arrow key is used to accelerate backwards (but can also be used as a brake pad if you are in motion going forward). The right arrow key is to turn the car right and the left arrow key is to turn the car left. Lastly, you can break completely using the space key (but if you are in full swing forward or backward motion, it will take time to break as the friction in this VR experience functions properly). I chose to use the keyboard as the interface to control the car, because I believe that it would be the simplest option. I am a gamer myself and using these keys on a keyboard is easier

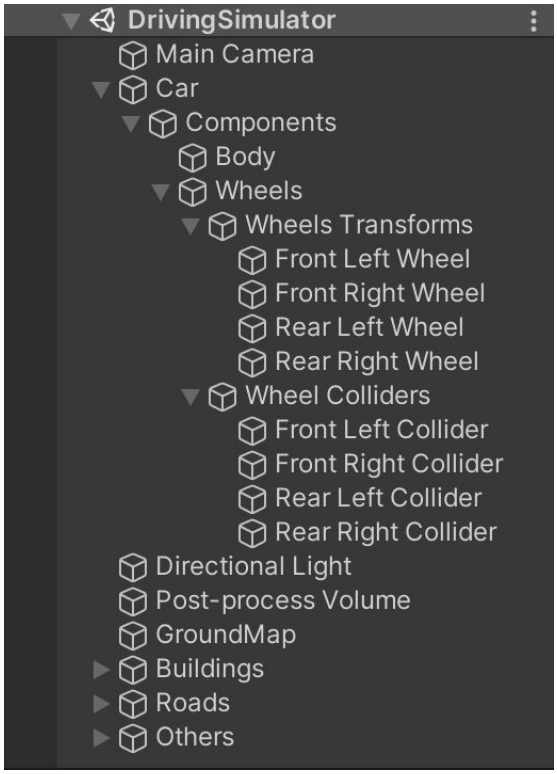
than any other way I could have done it. My design works best when the user believes that he is driving an actual car and considers speed while turning, etc. If you treat the car in VR as if you are driving an actual car, it will be a good experience. For example, if you spam the front arrow key as much as possible and try to make a turn instantly, the car will not be able to turn within time and probably crash into a curb or even worse, a building.



Ground Map

Implementation

In order to implement my driving car, there are two main scripts that I wrote. One script is the main driving script with all of the important functions inside of it, such as a motor function, a break function as well as a steering function. For this script to function properly, the setup of the wheels of the car is very important. The car object has 4 wheels: the front left, front right, rear left, and rear right wheels. Each wheel has a wheel transform and a wheel collider, so there are 4 wheel transforms and 4 wheel colliders in total. Essentially, after every input that the user gives (up, down, left, right arrow keys), the wheel positions of all four wheels are updated within an UpdateWheelPosition function that I have in the main driving script. This is what gets the car to actually move on the map. I watched some very good tutorials online to figure out how to get the wheels to work. Another very important function in this script is the GetInput function, which correctly assigns each input from the user to the variables. There is also a Motor function, which handles the acceleration of the car. There is a Steering function, which handles the turning of the car's wheels and the angle in which it turns. Lastly, there is a Breaks function, which handles the car's breaks when the user inputs a space key. This script is the most important script for the car and I left a few variables that I can set within the unity inspector section. I can set the max steering angle, which is the amount that the car will turn once the user gives those particular inputs. I have set this angle to 45, because if it is too high, the car will end up turning very easily and doing circles (if it is too less, the car will barely turn when the user presses the turn keys). I can also set the gas and break amounts, which basically controls how hard the car will accelerate and how hard the car will break. The second script I wrote for this driving car is the camera script. This script is simple and is implemented so that the camera follows the car perfectly around its route. Within this script, there is a Translation function and a Rotation function. Both functions handle the translation and rotation of the camera as it moves along with the car. I learned about and used the Lerp function for vectors in unity within these functions to make them work along with Time.deltatime. This script allows me to set how far away from the car that the camera will be positioned. I chose to position it right behind the car from a higher view, so that the car and surrounding objects are seen very easily (example of what it looks like was shown in the Design section).



The above picture shows all the game objects that I have set for this project. The car game object is the main one, which contains the body of the car, the wheels, and all the colliders involved. I used the ARCADE - Free Racing Cars asset from Imena Games to import the model of the car that I wanted to use. The GroundMap game object is just the ground plane I used for the roads. The main camera and directional light are also both important game objects for the visuals of this project. I used Skybox as an existing asset also. The Buildings game object contains all the "buildings", which I used simple cubes and squares to create. The Roads game object contains all the sidewalk curbs as I mentioned before. Lastly, the Others game object contains all the other cars on the roads which the user has to avoid.

Lessons Learned

I learned many lessons from this project, but probably the most important one is how to create a simulator of a moving car. Ever since I started this class, I've been wanting to learn how to do this, so I am very happy that I was able to make something like this for the final project. I learned a lot about how the wheel colliders and transforms work, which was a crucial aspect of this project. I also needed to read up more about the rigid body components in order to get the physics all worked out properly. First, I was having trouble making sure that the wheel colliders worked, but I was able to make them work eventually. Another obstacle I faced was that at first, my box collider was not big enough so it did not fit in the wheels of the car. It is important for the wheels to be inside of the box collider, because if they are not, the whole driving script will not work properly. After a long while, I figured this out and adjusted the box collider and all of a sudden, I had a functioning driving car. Something I learned in a broader aspect of VR projects and game design in general is that these projects take a long time to develop. It is not easy work and not a one day's job, which goes to show how much effort professional game designers put in. Creating my simple simulator took a lot of time for me, so I can only imagine how long it would take to create something like the Need for Speed or Forza Horizon video games. I gained a lot of respect for game design because of this class and this project; it is something that I never researched on or did any prior work for, but I am very interested now.