University of Leeds

Coursework 2

Computer Graphics

COMP3811

Rokas Pranevicius
sc18rp

Deadline date 2021 01 11

Aims and Objectives

CyberTruck
As the aim of this graphics coursework, I decided to create a CyberTruck model fulfilling all the requirements in coursework specification. Thought that having vehicle's edgy design would make the implementation interesting and doable.
The list of features/objectives to implement:

- Plane Ground.
- Vehicle Parts.
- Animation.
- Hierarchical Modelling.
- Road Columns for the Vehicle.
- Textures.
- Normals for Lighting.
- User Interface.

Implementation was made using QT's QGLWidget and visual scenes were rendered using OpenGL.

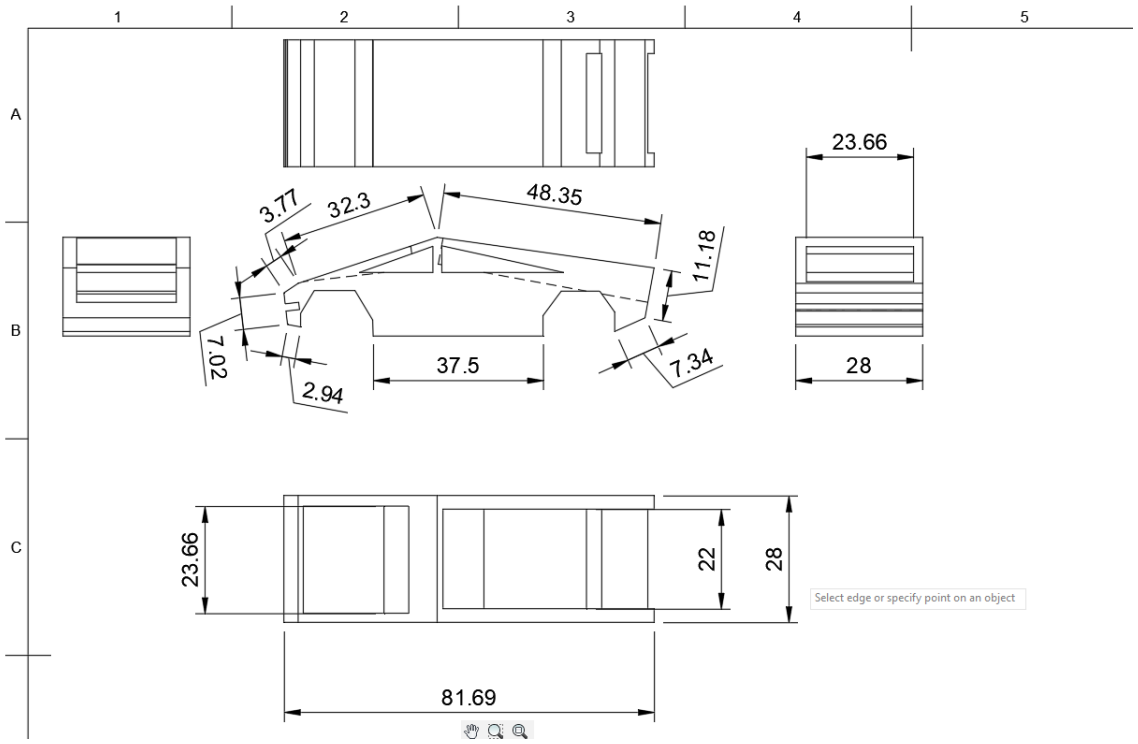
Implementation

Plane Ground

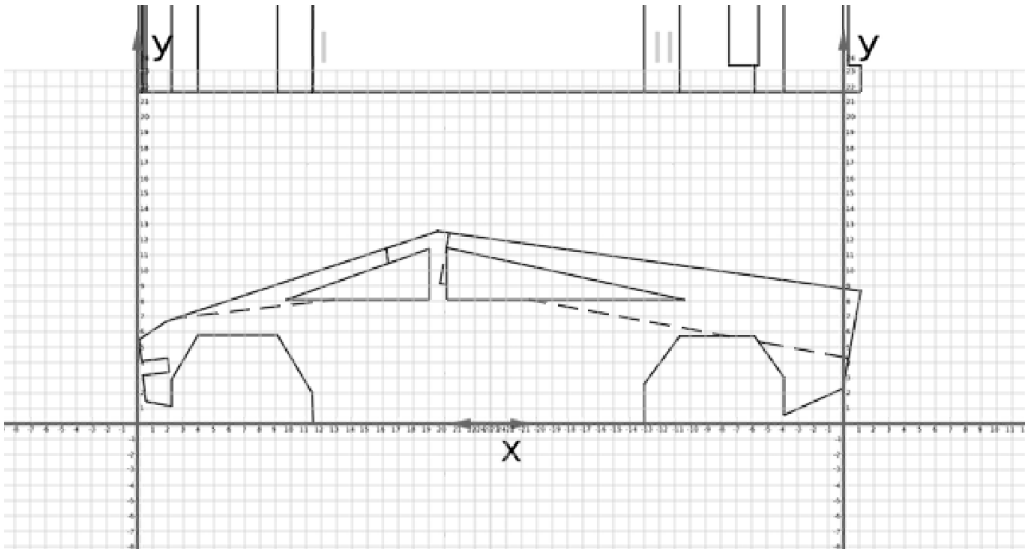
Since we don't want our vehicle to float in space, for better visualisation, the plane was implemented in the middle of a scene (y=0). It helped with testing and quicked development of vehicle parts.

Vehicle Parts

First of all, I have started my implementation by modelling the vehicle with a Fusion 360 software. Using cybertruck images, formed 3D models and hence obtained the proportional parameters of a vehicle.



Then I used a grid to calculate the coordinates of each part, which are indeed proportional to the original model of a car.



Only after the modelling, I have started drawing polygons/triangles in OpenGL.
First, I have implemented wheels with axles (with a default distance of 16.28 between two wheels). Having the base structure of a car, I kept on building up the model: Bottom Sides, Middle Sides, Top Sides, Windows, Roof(with front and back), Boot. Majority of my drawing functions used were called two times with positive z axis side and negative side, hence this saved time and space for development.
Then added some more 3D-looking elements such as bumpers over each wheel, so my model looked more alike a real cybertruck. Again, I have used same function for drawing both sides.
For material properties, I tried to replicate the original colours. Hence I used different steel materials altering between specular, diffusive, ambient properties and shiness, obtaining variations of light steel, medium steel and dark steel colours.

Animation

Now as we do have a vehicle, in order to make a scene more realistic, and to fulfil the requirements for the coursework specification, I have implemented some animations.
First of all, I have made a car to move in a circular path by specifying the radius of a circle. Then after rotating the car on y axis (not by 90, since it rotates from the centre, so to avoid transformations, I just rotated it by 70+radius/10 degrees to look more realistic).
For a vehicle to actually move, I had to connect vehicle widget to timer object for communication.

The car is moving, however, the wheels are not. Hence I implemented a function to calculate the angle of updated wheels that depended on the speed of a vehicle and wheel radius. I have made the updates within the same slot, hance the same two objects communicated through the same two slots.

The vehicle is moving in a circular path. Hence the wheels should be rotating not only on the z-axis(depending on speed) but also on the y axis to indicate the rotation of a whole vehicle around the circle, because in real world, vehicle drives in a circle only if wheels are turned left or right (in my case around y-axis).
The formula for calculating the angle by which the wheel should be rotated on the y axis is[1]:

$\sin(alpha)=\frac{(l-2*r)}{R}$ where alpha - is the angle in radians, l - is the length of a vehicle, r - is the radius of a wheel, and R - is the radius of a circle.

Now the vehicle is moving in a circle and the wheels are rotating responsively to speed and radius on z and y axes.

Hierarchical Modelling

For this step, I had a variety of options, since my cybertruck had many polygons and vehicles, in general, are hierarchical.
For the first feature of hierarchical modelling, I decided to make windows open/closing using glTranslatef(). To make it hierarchycal, they do translate within the driving vehicle.
Then I made doors to open or close (max 60 degrees – realistic). They are also hierarchically connected with the rotating vehicle and with the windows which are responsive to the door position. Hence while the vehicle is rotating, doors can open and windows might translate and it keeps the whole object connected.

Then I implemented a boot to open or close. The boot is also hierarchically connected to the whole vehicle so it can translate(open or close) while a vehicle is moving.

So there are windows, doors and boot hierarchically connected to the vehicle.

Road Columns for the Vehicle

For better circular path visualisation road columns were implemented. They are separated into inner and outer circle road columns, whose purpose is to outline the path that a vehicle is moving(as in the race field). Each obstacle consists of a Cylinder and two Torus objects from the glut library. I set the material properties to be shiny red and brass colours, so that they are easy to distinguish from the vehicle.

Textures

My scene looked plain and dull without any textures. Since the coursework specification asked to use all provided textures(Marc photo and Map), I have added 3 wall planes made of Map ppm image and added Marc ppm image on the front windshield.
I have used Image.cpp and Image.h files provided in a tutorial and later used the class to create image objects which I used for textures.
I also implemented a race track for the vehicle as an additional texture. The racetrack gives a more realistic scene.
However, for a large radius, the texture should be larger as well. I spend a considerable amount of time thinking of a function that would scale the texture proportionally. I have noticed that my texture is not perfectly symetrical, hence additional scalings and tranformings were needed depending on radius. In the end, I decided to calculate a few points(radius and scale size) manually and have used software tools[2] to make a practically working exponential curve fitting function. It appeared to be:

$Scale = 0.0357506 * R^{0.827275} - 0.11963$ where R is the radius of a circle.

Now, as we do have textures on the walls, Marc's portrait on the front windshield and a racetrack which scales, the scene looks more natural.

Normals for Lighting

We want the scene and object to be responsive to light. In order to do that, we need to calculate a normal vector for each polygon or triangle. Since glut objects do have normals calculated correctly by default, I only had to calculate normals for my drawings.
The function takes an array of 9 parameters(3 vertices coordinates - x,y,z) and first, checks which z-axis side contains more vertices. The front side should be counter-clockwise, and the side in the negative z-axis – clockwise.

Then I create vectors for given array parameters, calculate cross product and returned normalised result. Apply this function before drawing any polygon.

As a result, all polygons (given three vertice coordinates) light up gradually and correctly.

User Interface

I have used two types of user interface QSliders and Dialog Window (on double mouse click event). Starting with sliders, horizontal view slider helped to test the scene and object by rotating on the y axis. It is also a great way to scan the scene for the user. Constraints for this widget are set as a minimum (-90 degrees) and maximum(90 degrees).
Then the zoom slider scales the whole scene and object, to look like it is zooming in to the centre of a scene (0,0,0) or out(to see the big picture of a scene). The constraints for the widget are minimum – (-7_ and maximum 40. so that's within glOrtho parameters.
Open doors widget opens two front door polygons by rotating them on the y axis. I set the maximum angle to 60 degrees, and default as a minimum – 0.
The Open windows slider transforms the windows among the y axis – vertically, so it seems that the windows are opening. The widget minimum is set as default – 0, and maximum as 15(windows are wide open, so there is no point on doing maximum larger).
Open boot widget opens the vehicle's boot. Default is closed boot – value 0 and maximum is opened boot - value 50.
The last widget is to control the view vertically rotating the scene on the x-axis. For the widget, I created the separate layout and later added it to the main window layout. The minimum value is set as 0 (looking at the scene from the 0th y position, making a side view) and maximum as 90 degrees, making a view from a top.

For a better UI, that allow to actually change the behaviour of a vehicle, I have used a dialog.ui and ui_dialog.h templates provided by Graphics Module Block3.
The dialog opens on double mouse click and allows to change numeric values as well as boolean values. After prefered changes are made, thr user is forced to confirm or cancel the changes.

Firstly, the speed(default 0.4) is allowed to be changed in a range between 0(vehicle does not move) to 40(because setting speed over 40 does not change the visuals). The wheels rotation speed is set accordingly to the requested change.
The radius of the vehicle is allowed to be set as well. While the default is 50, the user can set it in the range of 30 ti 150 inclusively. Having a radius less than 30, the vehicle does not fit to the road anymore and having more than 150 causes inconvenience to visualise the scene. The change in radius also makes different rotation angle of the wheels on the y-axes, changes the number of obstacles of the inner and outer circle, and also, scales the texture using the mentioned exponential formula.

The width of a vehicle originally is proportional to the vehicle model – 8.14 on each side(starting from the z=0), hence making 2*8.14=16.28 full car length. It actually does not make sense to change the width of a vehicle, however, I found it fun. For that reason, I do not have any constraints on the width.

References

Wheel Turning Angle, https://www.assignmentexpert.com/homework-answers/physics/classical-mechanics/question-153417
Exponential Function, https://www.dcode.fr/function-equation-finder

For my coursework, I have used Github - https://github.com/Rocyzas/OpenGl_f.

Screenshot of a Model