WebGL Assignment

Alexandre Santos 80106
Pedro Salgado 80051

***Resumo* – Aplicação 3D que consiste num jogo em que o objetivo é controlar um carro de forma a desviar dos obstáculos que vão surgindo.**

***Abstract* – Application 3D consisting of a game in which the aim is to avoid the various obstacles that are constantly appearing in the map.**

I. Introduction

With this assignment we wanted to develop a game in which we could apply what was taught in the classes and expand our knowledge even more in web and computer graphics programming. We also wanted to have fun making this assignment and make the best use of what was taught in classes, so an idea emerged, controlling a car in a 3D environment in a way that it could switch between lanes, using the keyboard, in order to avoid obstacles was a good goal. These obstacles, cubes, for example, will appear in the game to come close to the car as this is moving, and the aim is to switch roads with using keyboard keys in order to avoid them.

II. Application Exhibition

We present to the user the possibility to choose one of the 3 levels as shown in Fig. 1, what differs between them is the minimum number of lanes occupied with obstacles, the speed of them and the time in which objects are generated. Level Easy has the minimum of one object thrown to the user at a speed x, the level Medium has the minimum of 2 obstacles at a speed of x + y and the last one more hardcore, level Expert has the minimum of 3 obstacles thrown at the user at a speed x + y + z, where x,y,z > 0. The object generator works in an inverse way, the more challenging the level the smaller is the time between the generations. Other features not exposed in the image are the obstacle and time counter, which are used to calculate the final score, and game-over which ends the game when the car collides with any of the obstacles making the score twinkle and blocking the game.

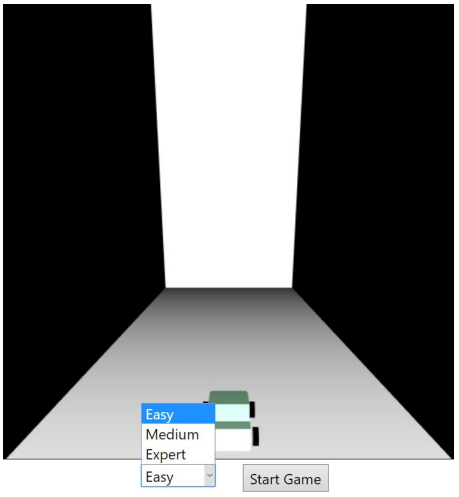

Fig 1 - Levels presented to the user

III. Development

A. *Polygon*

Using the best of Object Oriented Programming we created the Polygon class to be the most generic and used by other classes as it's parent. *Scenario*, *Road*, *Objects(*Sphere, Cube and Pyramid *)* and *Car* all descend from Polygon. This class will have the variables that need to be manipulated by its child classes, like the values for translations, scaling, rotating, use of the light, etc.

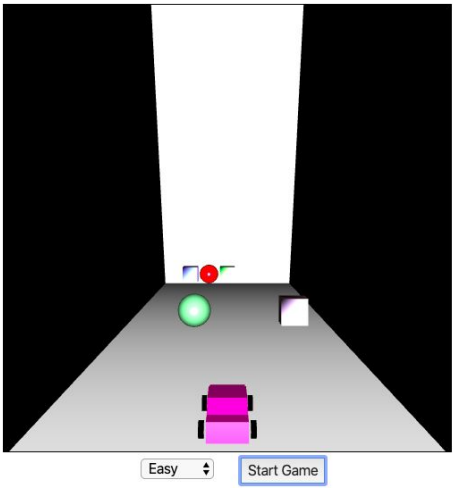

Fig. 2 - Application frame.

B. Scenario

Referring to the environment in which the game is played, basically consists of two triangles which will design the two black walls of the game that together with the road gives an even better sense of depth. The black color are obtained with all light coefficients of the triangles at 0.

C. Road

Class that illustrates the layer in which the car and obstacles are on, the gray bottom area. It is also formed with two triangles, that in combination with perspective and the two black walls as we said before gives a bigger sensation of depth.

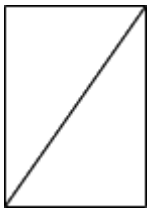

Fig 3 - Schema of road

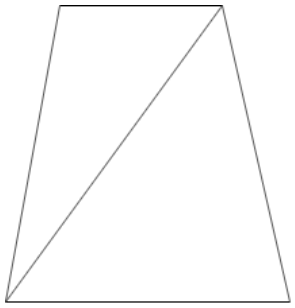

Fig 3.1 - Schema of road with perspective

D. Objects

Objects present in the game are Cubes, Spheres and Pyramids. Each has its own class and has the ability to rotate in the YY axis.. Each object will randomly generate its light coefficients in the moment of its creation so that in different stages of the game, which means the position of the object throughout the scene, and in different games it will have different colors and will be affected by light in unique ways.

E. Track

There are 5 tracks in the game, each Object and Car will be in one track at a time and only the car changes track using the A and D keys, for moving left and right respectively.

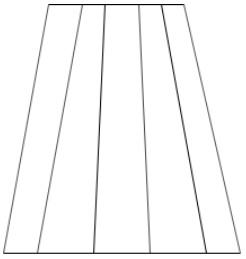

Fig 4 - Track distribution

F. Car

As you can see in Fig 5, the car is drawn using the assembling, like legos, of some polygons. 1 rectangular prism for the base, 1 cube for the user's cabin and 2 triangular prisms for the windows.

Like the objects each time the game begins the light coefficients are randomly generated, but in this case all polygons in the car have the same since the car must be cohesive piece.

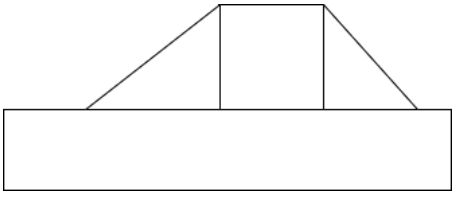

Fig. 5 - Car assemble

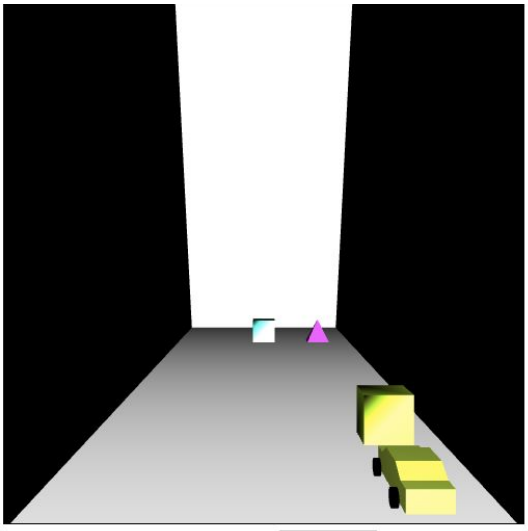

Fig. 6 The car on the rightmost lane.

G. *Wheels*

The wheels points were obtained with a python script that uses the trigonometric circle to generate all points and consequent triangles.

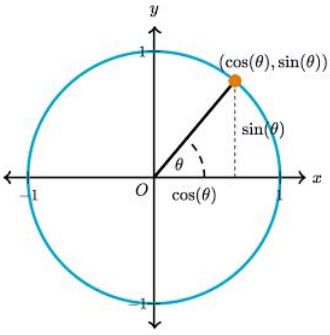

Fig. 7 - Base image for Cylinder script

Although the trigonometric circle gives 2D points, the script since the wheels will be fixed at XX axis uses the cos Θ for the z points and sin Θ for the y points.

Each base of the cylinders used to draw the wheels itself have 360 points, one point for each integer angle, plus the center one. Since the wheels are seen in different sides of the car, the left side had to be drawn in one way, while the right had to be drawn in the opposite direction. The right wheels were generated from 360º to 0º and the left from 0º to 360º to simplify the process of triangle drawing in a counter clockwise direction.

Fig 7 - Single wheel

According to the position of the car only some wheels will be shown, on the center all wheels are shown, on the two right lanes only the left wheels are shown, as you can see in Fig. 6, and on the left lanes only the right wheels are shown.

H. *Features*

- The game will only start when the uses want it to, by selecting level and pushing the start button.
- The user has access to time passed, obstacles avoided and immediate score.
- When a collision occurs the game is over.
- The light sources combined with light coefficients make the objects different.
- If game-over occurs a pop-up will appear to ask the user if he wants to play again.

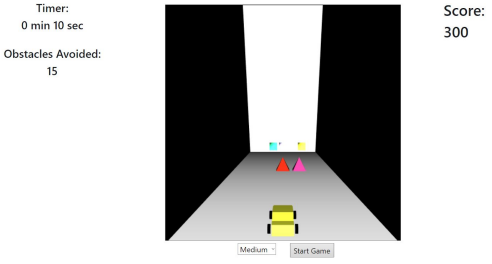

Fig.8 - Mid game scene

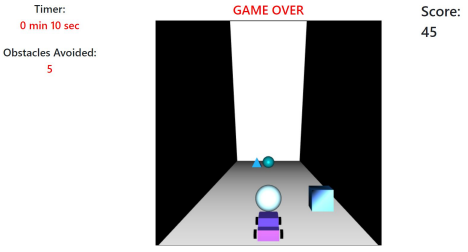

Fig. 9 - Game over - layout.

IV. Light Sources

To make use of all RGB spectrum, in viewers positions exists 5 light sources, red, green, blue, white and black. Also at other positions exist 2 light sources, one white and one black to make the road with that grey color and also to make objects appear to be made of reflective materials.

As we already said although the light sources keep the same, the objects will have different color due to it's randomly generated light coefficients.

V. Textures

We tried to use textures, although we did not want to use it in all polygons because we wanted to make the best out of the light. So we gave up on that idea because alternating between light and textures it's a costly and highly complex to implement, we would have to rebuild most of the shaders functions and the ones that they use, and reload both shaders in runtime to do it.

VI. Difficulties and Challenges

During the game execution we face some challenges, where to put the car, if close or far to the user position, in eye line or below, the scaling of the objects and the car to make them fit to one track only, and more. The most challenging problem that we had was, no matter the position of the car in the YY axis the objects, even if their YY position was lower than the car's, they would pass above the car. We assumed that was because the car was not a single piece and using perspective that happened, of course that when we introduced collisions that no longer occurred, although we couldn't make the collisions in the most correct way that was vertices collisions which means, if the objects and car vertices were in the same spot at the same time a collision occurred, in our solution, we take into consideration the car's length to get the front of it and if an object as it's translations coefficients in the same spot than a collision occurred and the game stops.

VII. Conclusion

With this work we are now capable of developing a computer graphics program using the basic principles of an app of this type. We learned the concepts of lights and to make the best use of them, for example, how light coefficients can give the illusion that an object is made of certain material, textures, and how to apply them and make the objects look more realistic although in irregular objects that needs to be combined with normals to fit the texture and make the surface rugged. And most importantly, we learned how to work with perspective so that the user can get the best of our application getting a 3D experience without feeling that something doesn't look right. For example, if an object is far away then it's intuitive to be smaller than the ones which are closer.

VIII. Work Distribution

Alexandre Santos - 60 %
Pedro Salgado - 40 %