

[**Three-Dimensional Visualization and Animation**](https://fenix.tecnico.ulisboa.pt/disciplinas/AVT351795/2017-2018/1-semestre)

1st Semester 2017/2018



Project Report

Group 13:

78223 João Carvalho

78414 Ricardo Silva

78516 João Loureiro

# Introduction

The first part of the project was to implement the classic micro machines game where we control a car and drive around the track while avoiding obstacles and try to complete the laps as fast as possible. During this stage we implemented the movement of the objects, collisions, lighting and cameras. To specify the location of the multiple elements in the table we wrote a script that displays a grid to the user and the user can paint colours by clicking on the grid squares. Depending on the number of clicks the grid element shows a different colour that corresponds to a different game element (light grey – empty spot; dark grey – cheerio; yellow – butter; green – starting position for the car; orange - candles). When the user is done he can save and a file is created containing a grid of numbers where each number corresponds to a different game element. Then we parse the file in our game to generate the game elements on their final positions.

|  |  |
| --- | --- |
|  |  |
| Fig. 1 – Track generator | Fig. 2 – Result from loading the track file generated by the track generator |

We also implemented the multi texturing of the table using two textures that are mixed according to a black and red mask. The mask has with fuzzy transitions between black and red to obtain smooth transitions between the two textures.

|  |  |
| --- | --- |
|  |  |
| Fig. 3 – Mask texture applied to the table | Fig. 4 – Multi texturing of the table using the mask to decide where each texture goes |

The objective of the second part of the project was to implement six different techniques to improve the overall graphics of the application:

1. Stencil Test Based Effects
2. Transparent Objects
3. Fog
4. Particle Systems
5. 2D Lens Flare effect
6. Billboards

We also implemented some extras to improve the game experience.

# Stencil Test

We used stencil testing to implement the reflection of the rear-view mirror inside the car.

## Method

To accomplish this goal, we used an extra camera located inside the car and oriented towards the road behind the car. First we render the scene as usual, then we render the mirror with the stencil testing active and finally we render the scene using the backwards camera. By using this technique, we are able to create the illusion of a reflection in the rear-view mirror.

## Results

|  |  |
| --- | --- |
|  |  |
| Fig. 5 – View from inside the car | Fig. 6 – View from inside the car |

# Translucid Objects

The translucid objects we implemented were the car front, back and side windows.

## Method

To accomplish this goal, we set the alpha of the material used for the windows to 0.34 and enabled the blending mechanism in OpenGL.   
Since we import the objects from .obj files, we can just modify the files in blender or simply edit the .mtl files directly.

|  |
| --- |
|  |
| Fig. 7 – Window material in the car.mtl file |

Special attention is needed when drawing transparent objects. We must make sure than the transparent objects are drawn after the solid objects. Otherwise the solid objects would not be drawn due to the depth testing.

## Results

|  |  |
| --- | --- |
|  |  |
| Fig. 8 – View from inside the car | Fig. 9 – View from outside the car |

# 2D Lens Flare

We implemented a 2D lens flare consisting of 5 textures consisting of circles, a hexagon and sunbursts.

## Method

To do this we start by converting the sun’s world coordinates into screen coordinates since we will be using screen coordinates to indicate where the flare’s textures will be located relatively to the sun.   
We then set the flare elements’ size, alpha and position according to the sun position.  
Finally, we apply the textures according to the attributes set before.

## Results

|  |  |
| --- | --- |
|  |  |
| Fig. 10 – Lens Flare Textures | Fig. 11 – Lens Flare Textures |

# Billboards

We used billboards to draw the Christmas tree in the centre of the track and to draw the sun in the sky. We decided to implement true cylindrical billboards to achieve the best quality possible.

## Method

To draw the billboards we apply a texture to a plane that rotates according to the position of the camera.

## Results

|  |  |
| --- | --- |
|  |  |
| Fig. 12 – Sun Billboard Texture | Fig. 13 – Christmas Tree Billboard Texture |

# Particles

We used a particle system to implement a snowing effect. Each particle is a cross configuration billboard with a texture applied.

## Method

To implement the snowing effect, we used a particle system where each snowflake is a particle. The particles get spawned at a random position in the sky with a random velocity towards the ground and affected by gravity. When a particle reaches the ground it disappears and a new particle gets spawned in the sky.

## Results

|  |  |
| --- | --- |
|  |  |
| Fig. 14 – Snow Particles during the day | Fig. 15 – Snow Particles during the night |

# Fog

We implemented the fog effect with the ability to turn it on and off and using an exponential function for the fog density.

## Method

The first thing we did to create the fog was to set the clear colour the same colour as the fog. We choose a different colour depending on if it’s day or night and if it’s raining or not.  
Then to calculate the fog density we used an exponential function, as said before, and to get the distance using the length.

## Results

|  |  |
| --- | --- |
|  |  |
| Fig. 16 – Fog during the night | Fig. 17 – Fog during the day with snow |

# Extras

In additional to all the requirements in the exercises, we also added two extras:

## Music

To contribute to the atmosphere of the game we added music using a SFML.

## Weather

To add to the atmosphere of the game we change the colour of the sky depending on the current time of the day and weather using the clearing colour of OpenGL.

## Scoring System and finishing Line

Our scoring system consists of tracking the time a player takes to complete a lap around the track. The best time and the current time is displayed in the tittle of the window. To complete the system, we added a finishing line, using a texture, to the place where the game starts and finishes counting the time.

|  |
| --- |
|  |
| Fig. 18 – Finishing Line |

## Custom .obj and .mtl loader

To load .obj models we wrote a custom loader than parses .obj files and the corresponding .mtl files and creates the meshes in our application. The loader retrieves all the necessary elements (vertices, normals, texture coordinates and material) and then creates the VBOs and VAOs necessary to draw the meshes.

# Conclusion

We completed the project successfully by implementing all the required objects/effects using the six required techniques, plus some additional effects to complement the game. During the development of this project we acquired new skills and had the opportunity to learn more about modern OpenGL.