Lesson 1 - three.js Introduction

Margarida Silva Martins, 93169 and Rui Fernandes, 92952

Information Visualization, 2021/22, MSc Software Engineering, University of Aveiro

Exercise 1.2

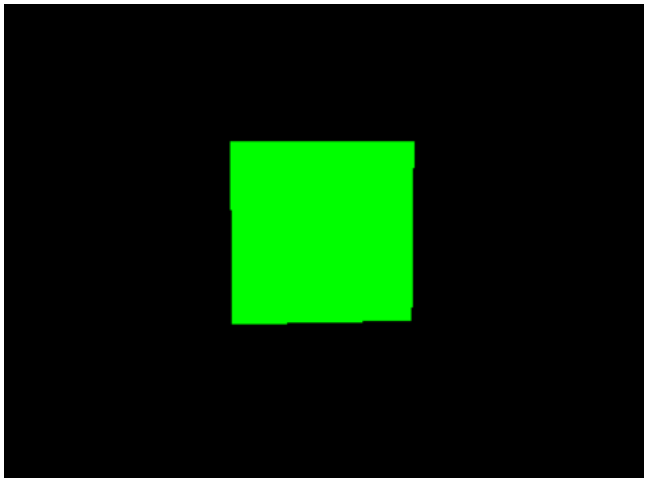

Figure 1: Exercise 1.2 final result

After following exercise 1.1 and setting up three.js, we moved on to the first real exercise.

This is the simplest exercise and functioned as a starting experiment to get used to three.js.

A simple cube was built using *BoxGeometry* giving it a simple green material and applying a very small rotation on the x and y axis.

Exercise 1.3

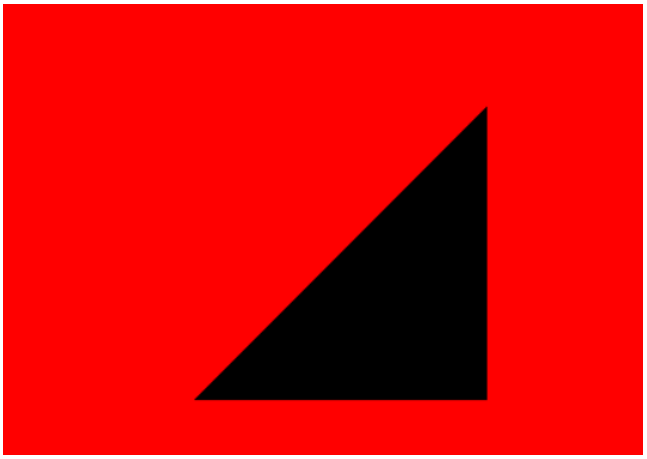

Figure 2: Exercise 1.3 final result

For exercise 1.3 some more experimentation with shape and color was made.

Firstly, instead of *BoxGeometry*, the more generic *BufferGeometry* was used. To make it a triangle, we defined the coordinates of the 3 vertices through an array, applying them by setting the position attribute to our coordinate array, and making sure we specify that the points are in 3 dimensions.

In regards to color, the triangle was changed to black by simply changing the color's hex value. As for the background, we resorted to the renderer's *setClearColor* with a red color and an opacity of 1.

Exercise 1.4

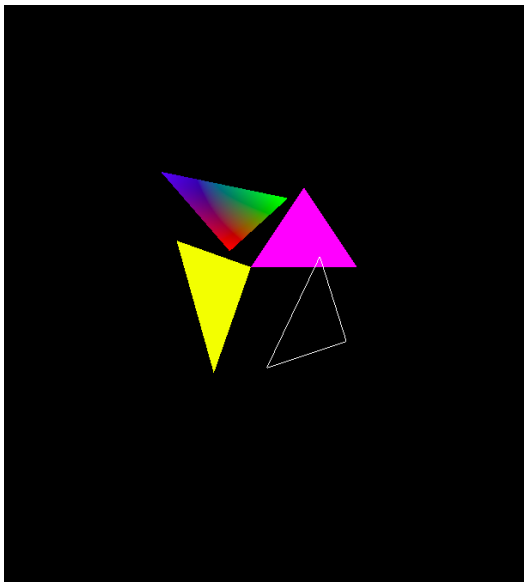

Figure 3: Exercise 1.4 final result

As Figure 3 shows in this exercise some properties of the *MeshBasicMaterial* were explored.

The top right triangle has the property vertexColors set as true which allows to color the triangle considering the color associated with each vertex. In

this case the triangle has one blue vertex, a red one and the other green. This results in a rainbow effect.

The top left triangle as well as the bottom right one would not be visible if the property *side* was not set as *DoubleSide*. This happens because the default *side* value is *FrontSide* which means that only the front side of faces will be rendered.

The bottom left triangle only has its structure (wireframe) represented. This is done by changing the default value of the wireframe property to true.

Exercise 1.5

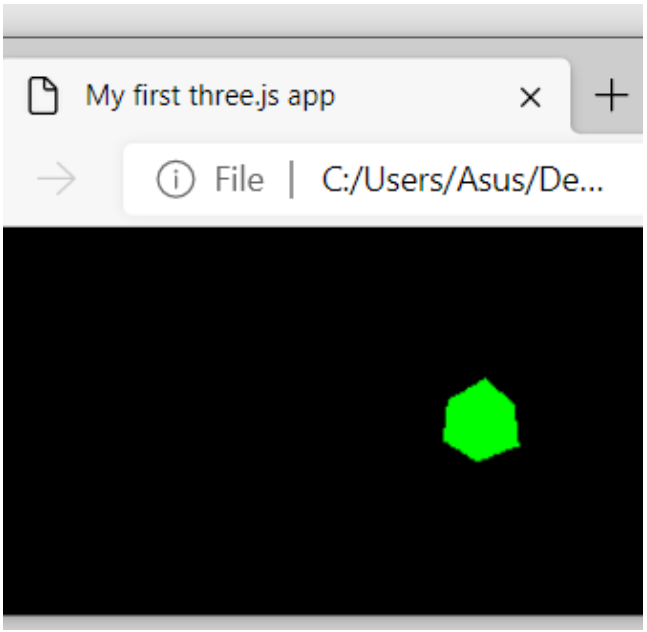

Figure 4: Exercise 1.5 final result

Exercise's 1.5 objective pertained to the viewport, since in exercise 1.2 if the browser window was redimensioned, the cube stayed in the same position per say, so it would disappear if the window was made smaller.

To avoid this, an event listener for resizing was created, that set the renderer's size and camera's aspect according to the change in window.

This made it so, as we can see in Figure 4, resizing the window would now keep the cube anyways.

Exercise 1.6

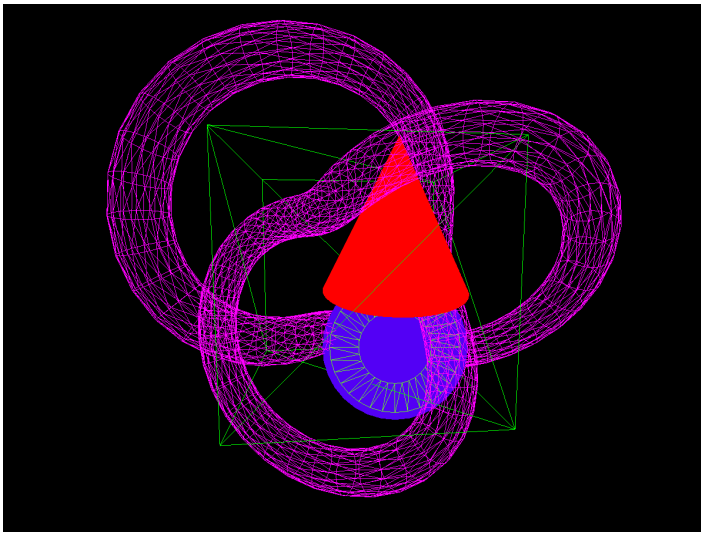

Figure 5: Exercise 1.6 final result

Exercise's 1.6 goal was to explore different geometries.

The first geometry is the wireframe of a cube slightly rotated in the x and y axes. This geometry was created using the *BoxGeometry* method.

The second one is a circle created with the *CircleGeometry* method, the circle has radius 1 and is built using 32 segments. The circle was translated.

The third geometry, created using the RingGeometry method, is the wireframe of a ring with 0.5 of inner radius and 0.9 of outer radius and 32 segments.

A red cone is the fourth geometry, it was rotated in the y axis in order to be easier to perceive its 3 dimensions. The cone was built using the ConeGeometry method, it has radius 1 and height 2, 32 radius segments and 1 height segment.

The final geometry is a more artistic one, the wireframe of a Torus knot, built using the TorusKnotGeometry. The Torus knot has a global radius of 2 and a tubular one of 0.3. It was rotated on the x axis.