Lesson 1 - Introduction to three.js

Artur Romão (nMec 98470), Eva Bartolomeu (nMec 98513)
Information Visualization, 2022 (MSc Informatics Engineering, University of Aveiro)

First example

In the first exercise we created a 3D scene, a camera and a renderer. To this scene we added a cube. Then we define a "*render()*" function to continuously render the scene, updating the rotation of the cube with *cube.rotation.x += 0,01* and *cube.rotation.y += 0,01* and also the camera's projection matrix.

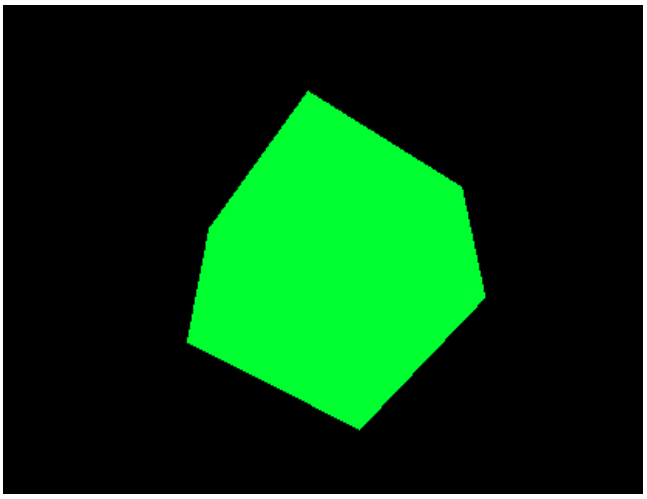

Figure 1 - Result of the first example.

2D Primitives

In the second exercise we created a black triangle. The triangle is created with the THREE.BufferGeometry geometry and the MeshBasicMaterial material.
In order to make this figure a triangle, we define the coordinates of 3 vertices through the Float32Array object. Then we change the position of the geometry to these coordinates, indicating that the points are in three dimensions.
In the material of this figure, we changed the color to black, and finally we set the renderer color to red, through the setClearColor function.

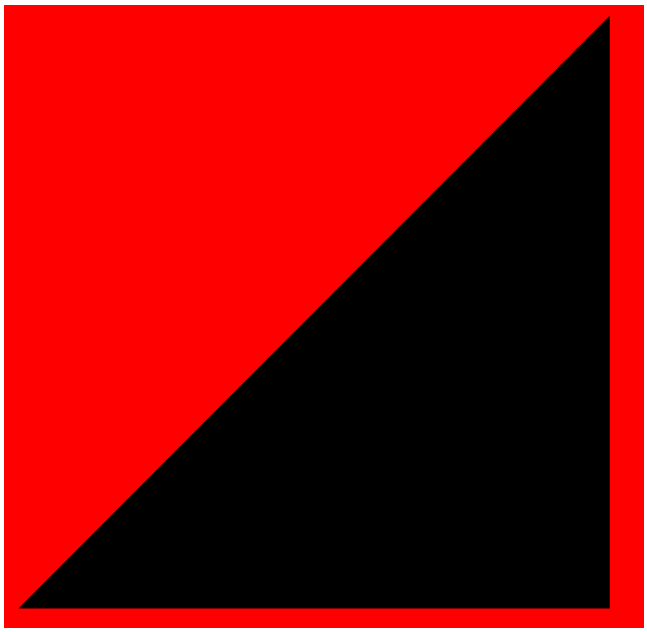

Figure 2 - Result of the 2D primitives.

2D Primitives (Addition of color)

In the next exercise we build a 2D scene with four triangles. The triangles are defined using THREE.BufferGeometry objects and THREE.BufferAttribute objects, and are rendered using THREE.MeshBasicMaterial objects. In order to define the vertices coordinates of each triangle, we used Float32Array and to define the colors of each triangle we used Uint8Array. Then, we need to set the positions and the colors of the triangles as geometry attributes and create the triangles as THREE.Mesh objects. The script continuously renders the scene of the four triangles.

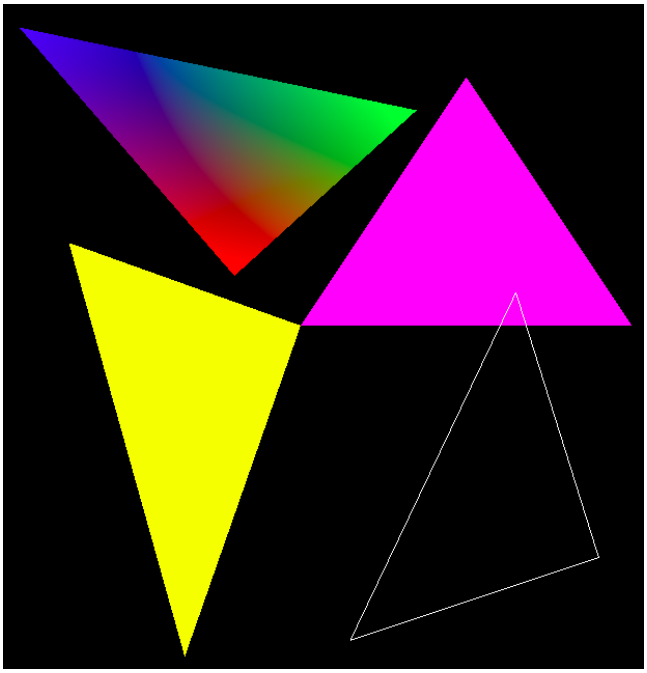

Figure 3 - Result of the 2D primitives (Addition of color) .

Viewport Update

In this exercise we were supposed to redefine the first exercise and update the visualization window so that the cube could continue to be visualized although the viewport (visualization window) changed. In order to do that, we added an event listener to the window object to ensure that the 3D scene is correctly displayed when the browser window is resized, by doing:

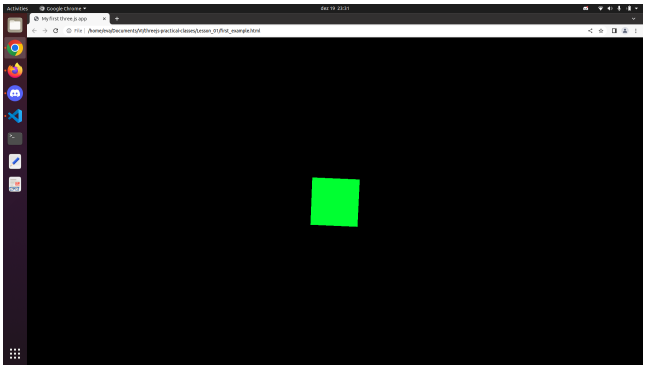

Figure 4 - Result with the fullscreen window.

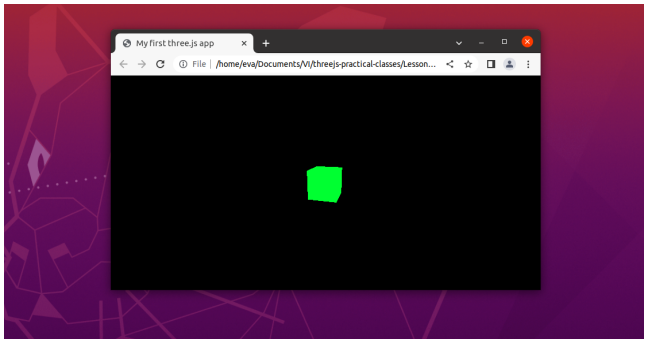

Figure 5 - Result with minimized window.

Other primitives

In this exercise we developed different types of geometries.

We implement a wireframe torus, with the THREE.TorusGeometry geometry. With a radius of 1, tube of 0.5, radialSegements 16 and tubularSegments 100. We place this figure with the pink color and with the positions (4,2,-1).

Next, we implemented a sphere, with the geometry THREE.SphereGeometry. With a radius of 1, widthSegments 32 and heightSegements 32. We place this sphere with the positions (0,1,-3) and with the red color.

We create a wireframe cylinder, with the geometry THREE.CylinderGeometry. With a radius of 1, height of 2 and with a radialSegements of 32. We place this figure in the positions (7,-3,-3) and with the color green.

Finally, we also create a wireframe cone, with the THREE.ConeGeometry geometry. With a radius of 1, height of 2 and radialSegments of 32. We position the cone at positions (-7,-3,-3) and change the color to blue.

In addition, we applied some animations, we made the torus rotate around its x axis, the cone rotate around its y axis, the cylinder perform a horizontal movement back and forth, and the sphere perform a horizontal and vertical movement at the same time, giving the impression that you are jumping.

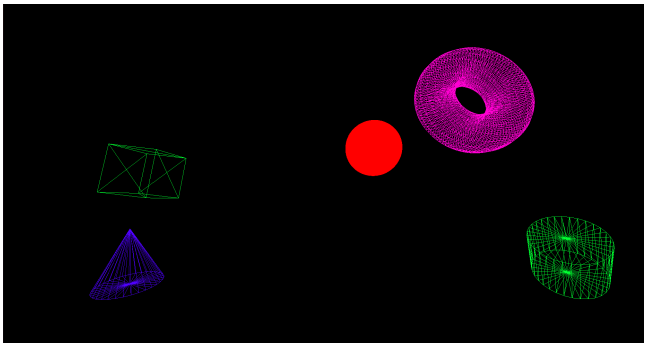

Figure 6 - Result of the other primitives.