Lesson 2 - three.js Projections, lighting and transformations

Margarida Silva Martins, 93169 and Rui Fernandes, 92952
Information Visualization, 2021/22, MSc Software Engineering, University of Aveiro

Exercise 2.1 - Camera models

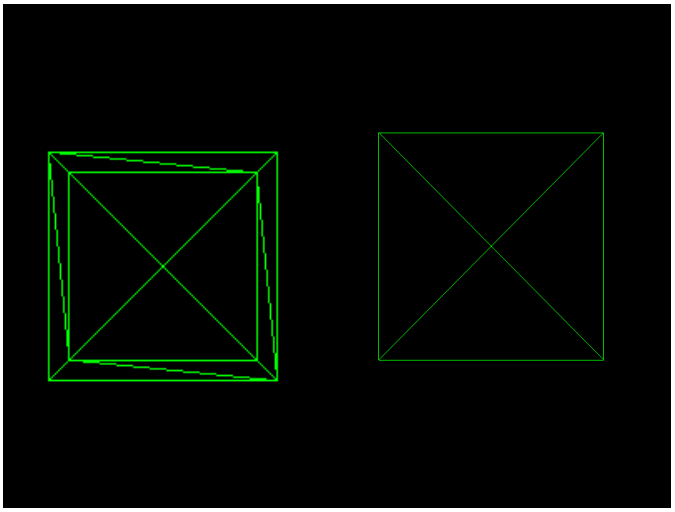

Figure 1: Perspective (left) and Orthographic (right) Cameras

Three.js supports two different camera types: Perspective and Orthographic.

In exercise 2.1 an Orthographic camera was used instead of the Perspective Camera used in the previous lesson.

The Perspective camera uses perspective projection. It is used in rendering 3D scenes because the viewers see no difference between a real scene and its representation using this model.

The Orthographic camera uses orthographic projection. it is used in rendering 2D scenes because the object size is constant regardless of its distance to the camera.

As we can observe from Figure 1 the perspective camera gives us a sense of depth inexistent in the orthographic one.

Exercise 2.2 - Orbit Control

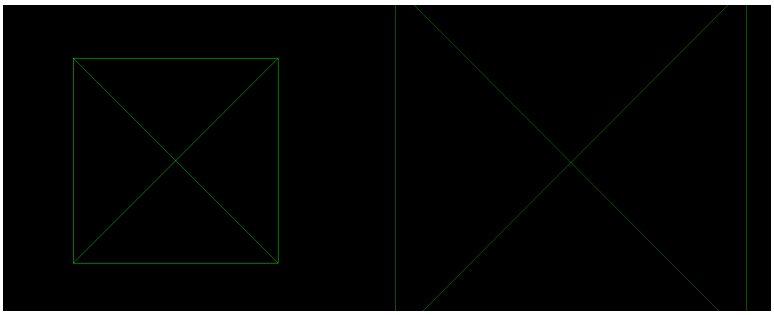

Figure 2: OrbitControls mouse scroll effect

Exercise 2.2 consisted in experimenting with the controls provided by Three.js.
Three.js has 8 different types of controls one of each is the OrbitControls.
This type of controls allows the camera to orbit around a target. By dragging the mouse cursor it is possible to rotate the cube. Using the mouse scroll it is possible to bring the cube closer as seen in Figure 2.

Exercise 2.3 - Lighting and Materials

Figure 3: Exercise 2.3 final result

Exercise 2.3's goal was to see the effect of light in a cube geometry.

First it was created a white directional light, a light that gets emitted in a specific direction, using the *DirectionalLight* method . This type of light can cast shadows.

However this was not enough in order to see shadows in the cube. The *MeshBasicMaterial* is not affected by light so another material type was used, the *MeshPhongMaterial,* where the shading is calculated using a *Phong* shading model.

AmbientLight was also added to the scene in order for the user to be able to see the color of the material.

Exercise 2.4 - Shading

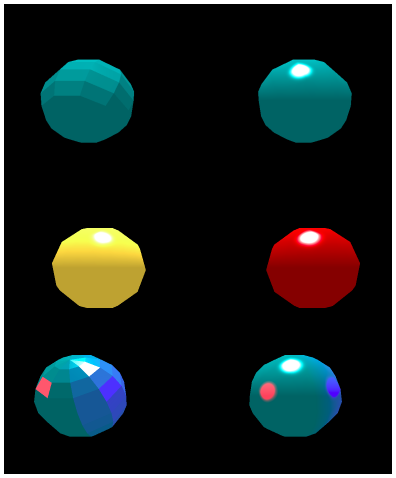

Figure 4: Exercise 2.4 final results

In this exercise instead of using a *BoxGeometry* it was used a *SphereGeometry* in order to create two similar spheres side by side.

Keeping with the same directional and ambient light from the previous exercise, one of the spheres (left) has the *flatShading* as true while the other has it as false (right).

In flat shading the lightning value is calculated only once for each face while the phong shading is more realistic. Flat shading requires much less computational power.

Other combinations of the sphere's colors were tested.

In the final exercice three different lights were added to the scene two directional lights and one spotlight. In the spotlight the light expands along a cone that increases in size the further from the light it gets.

Exercise 2.5 - Transparency

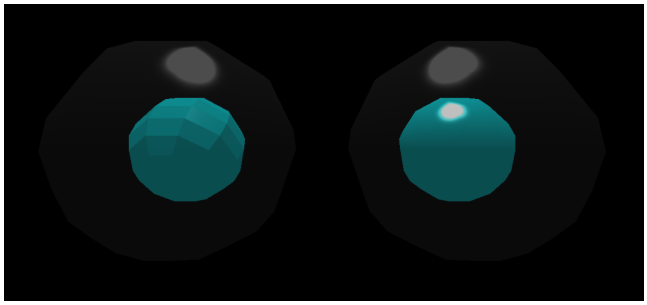

Figure 5: Exercise 2.5 final result

Exercise 2.5 was quite simple, following the instructions two bigger spheres were made centered on the same points of the prior spheres. The difference is these ones were made with low opacity and transparency on. This gives off a glass effect, letting us see the prior spheres inside, while the lighting still affects the glass ones.

Exercise 2.6 - Transformations (scale and rotation)

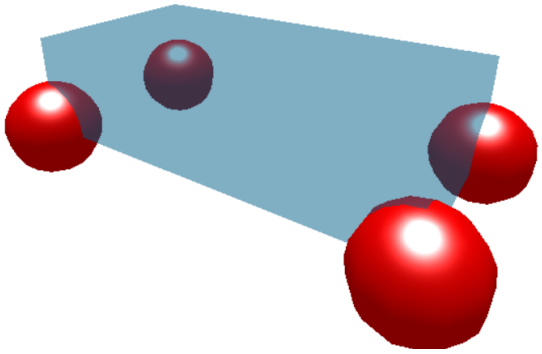

Figure 6: Exercise 2.6 final result

Exercise's 2.6 goal was to make the presented car like shape.

Firstly we made a cubic shape with *boxGeometry*, then using the shape property, we expanded it to the desired proportions.

Following that we made 4 spheres, positioned on the lower vertices of the parallelepiped.

Following the guide, we added all the meshes into a single *Object3D.*

Also included were some of the previous techniques, like orbit controls and some ambient and directional lights.

Exercise 2.7 - Transformations (rotations)

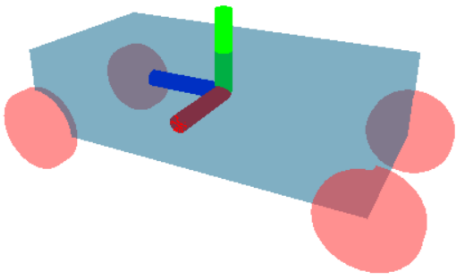

Figure 7: Exercise 2.7 final result

In regards to the last exercise, it was a simple repurposing of the previous.

This time the wheels were instead cylinders, created with the *CylinderGeometry* and having to apply a rotation to it on the z axis.

Then three more cylinders were made to represent the directional axis, this was made through different rotations and positioning.

Finally, a simple rotation was applied on the whole object.