Lesson 2 – Projections, lighting and transformations

Pedro Tavares – 93103 Ricardo Cruz – 93118

MSc Robotics and Intelligent Systems MSc Communication and Web Technologies

Information Visualization, University of Aveiro, 2022

Camera models

The first exercise from the practical class focuses on the camera models available, changing the perspective of the user. Until now, we have been using the perspective camera, that like the name says, shows the world in our perspective. By doing so, objects can be deformed, and we can’t take measures. In this exercise we changed the camera model to Orthographic, that differently from the perspective camera, maintains the measures of the objects.

To achieve this, we passed on to the Options of the camera, a position of -3 and 3 on left and right, -3 \* (window.innerHeight / window.innerWidth) and 3 \* (window.innerHeight / window.innerWidth) as top and bottom. Differently from the previous used camera, in this one we do not pass a field of view and instead, we pass the position the camera is.

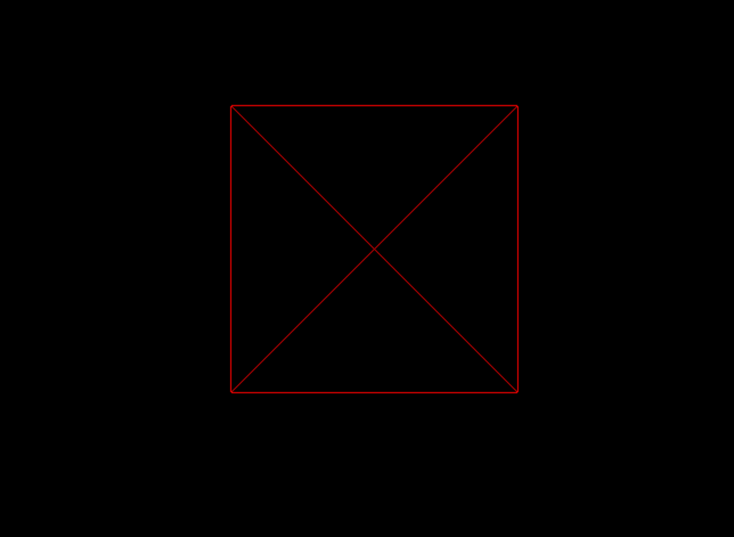


Figure 1: Output from ex. 1.

Comparing with the results obtained in the last lesson, we can see what was stated above. In the orthographic camera, the measures are exact, and they are not affected by the perspective. On the other hand, on the perspective camera, things get more real, and we see the cube as if we were there.

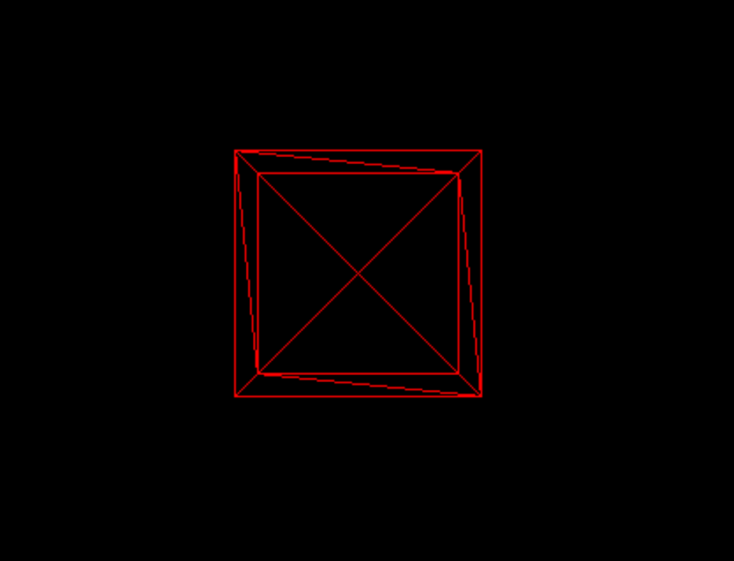


Figure 2: Output from ex.1. Lesson 1

Finally, as an extra, we added a function that ensures the cube aspect ratio does not change when the window is resized. For that, we added an eventListener on resize.

# Uma imagem com texto Descrição gerada automaticamente

Figure 3. Event listener for exercise 1

# Orbit Control

For the second exercise, the objective is to is to add controls so we can easily control the camera pose. For this, we used the OrbitControls class.

We first imported the script into our HTML and created the variable controls and finally we updated it.

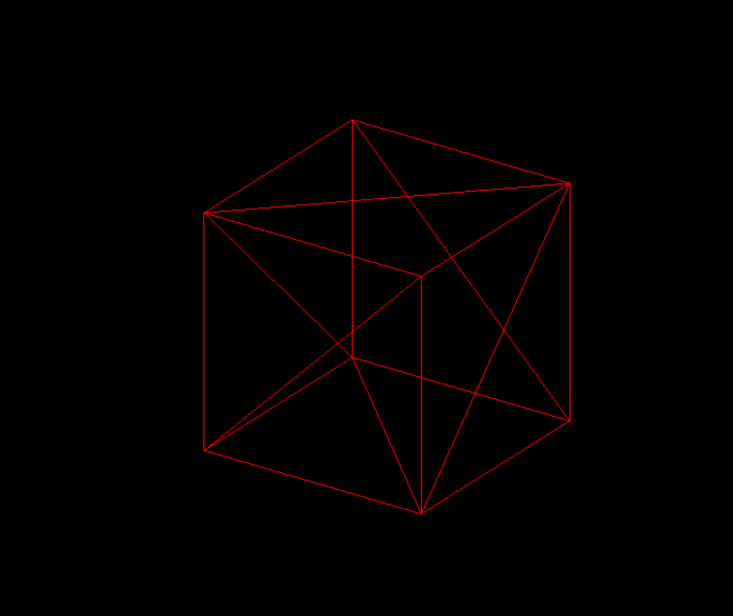


Figure 4: Output from ex. 2 using Orbit Controls.

Finally, we tried other camera controls such as TrackballControls, FirstPersonControls and FlyControls.

# Lighting and materials

In this exercise, the objective is to add lighting to the scene.

We started by getting back into the perspective camera, disabling the wireframe and enabling the cube rotation.

We then created a directional white light and added it to the scene. However, there were not any visible changes. For that to happen, we had to change the material of the object, and replaced the MeshBasicMaterial by MeshPhongMaterial.

Finally, we added an ambient light to the scene.

Uma imagem com texto, céu noturno

Descrição gerada automaticamente

Figure 5: Output from ex. 3.

# Shading

The first step of this exercise is to change from a cube to a sphere. We started by creating a sphere with radius 1. We then activated the wireframe and set the widthSegments and heightSegments to 10. With the wireframe active, we could see what these values referred to.

Secondly, we disabled the wireframe again, and created another sphere with the same characteristics and set the position of on in (-2.5,0,0) and the other in (2.5,0,0).

Thirdly, we added an ambient light and a directional white light between the two spheres in y = 5. We applied the MeshPhongMaterial to see the lighting. Then in one of them we toggled the flatShading attribute to true.



Figure 5: Two spheres with MeshPhongMaterial. The left has flatShading active.

We then created a new material, MeshLambertMaterial with the same characteristics as the others. We removed the specular and shininess components since Lambertian materials scatter light evenly in all directions.

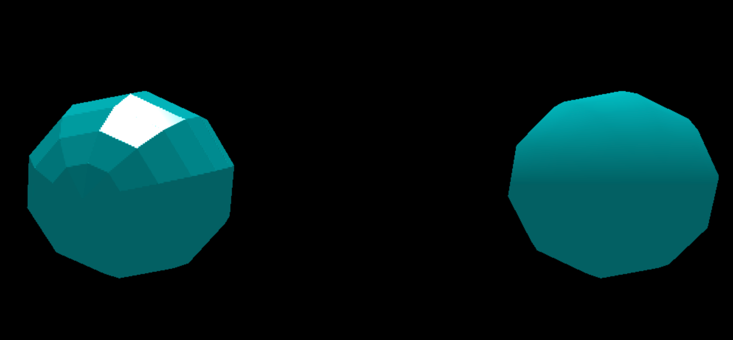


Figure 6: Two spheres. The left with MeshPhongMaterial and flatShading. The right with MeshLambertMaterial

Fourthly, we then created an emerald material and applied it to one of the spheres.



Figure 6: Two spheres. The left with MeshPhongMaterial and flatShading. The right with MeshPhong and an emerald Effect.

To finish this step, we then created a javascript object that contains the properties to represent different effects, i.e, emerald, gold, silver, etc, and tested it in one of the spheres.

Uma imagem com texto

Descrição gerada automaticamente

Figure 7: Javascript Object defined to contain different types of materials. To apply, change the in the const optional\_material

Finally, to finish this exercise, we opted to apply the white plastic material to one of the spheres. Then we created two directional lights, one blue and one red, and one spotlight light.



Figure 8: Two spheres. The left with MeshPhongMaterial and flatShading. The right with MeshPhong and an White plastic Effect. Added blue and red directional lights and a green spotlight.

# Transparency

In this exercise the objective is to apply the transparency effect. To achieve this, we used the example from the previous exercise.

We started by creating a new material, a MeshPhongMaterial with the properties to create a glass effect.

Uma imagem com texto

Descrição gerada automaticamente

Figure 9: Code to replicate glass effect.

We then created two new spheres with twice the size of the originals. We then placed them in the same x position.

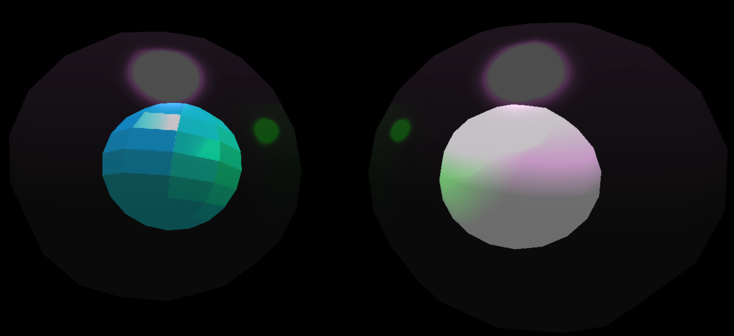


Figure 8: Two spheres with same properties as Figure 8 and two more spheres with twice the size that are transparent

As we can see, the transparent spheres let the light go through and reflect it as well.

References

1. Three.js. (2022). *Orthographic Camera* Obtido de <https://threejs.org/docs/#api/en/cameras/OrthographicCamera>
2. Three.js. (2022). *Directional Light*. Obtido de <https://threejs.org/docs/#api/en/lights/DirectionalLight>