Lesson 2 - Projections, lighting and transformations

Course: Information Visualization - 44156

André Fernandes (97977) 50.0%, Gonçalo Machado (98359) 50.0%

11/12/2023

Class - TP2

Developed *software* can be found **here**.

Introduction

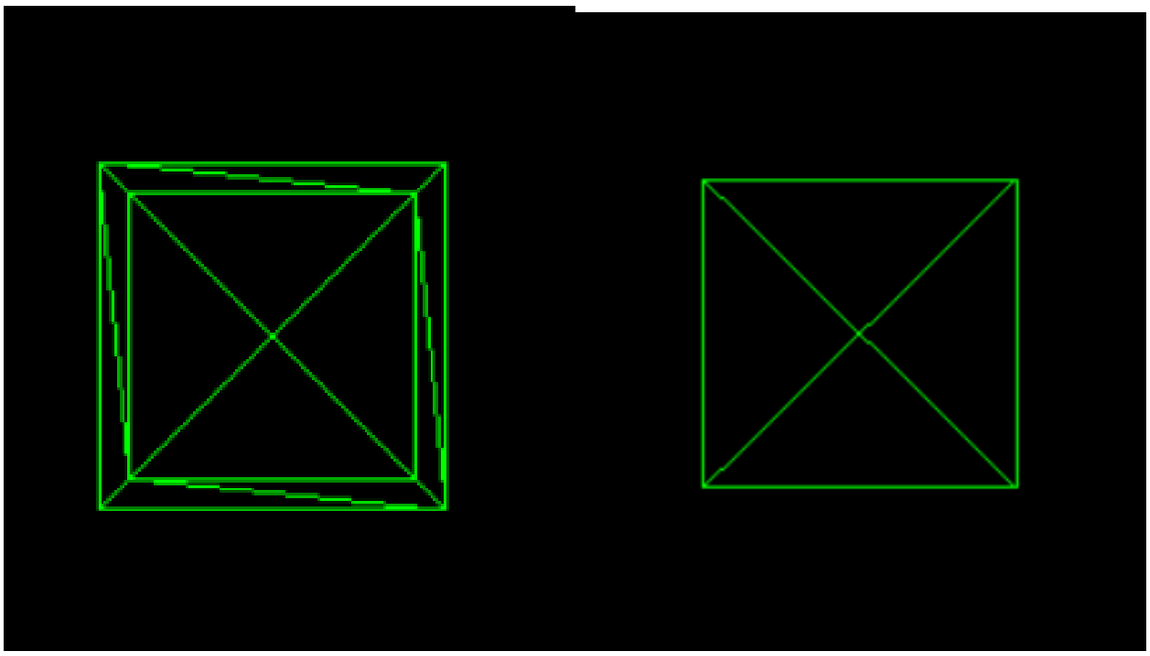
In the Lesson 2 of the Computer Graphics module, the focus will be on projections, lighting and transformations. Here, we'll unravel the intricacies of camera models, drawing distinctions between perspective and orthographic cameras. Furthermore, we dive into drawing into the design between perspective and orthographic cameras, interact also with lighting and shading and finally, transformations.

2.1 - Camera models

For the first exercise it asked to the students that they modify the first example from the last lesson to visualize the cube in wireframe and, instead of using a perspective camera, use an orthographic camera.
It is mention that the students should modify the parameters so that the world view is between -3 and 3 on the x-axis while respecting the window's aspect ratio.

The goal of this exercise is to compare the result of the two type of cameras.

In Figure 1 is represented the original cube, visualized with perspective camera, and the one visualized with the orthographic camera.



(a) Cube visualized with perspective camera. (b) Cube visualized with orthographic camera.

Figure 1: Results of 2.1 - Camera models

As is it possible to notice, in Figure 1a it is possible to see all the 6 squares that compose the cube, all six sides. On the other hand, in Figure 1b only one square is possible to see.

This results are expected because perspective viewpoints simulates the way human eyes perceive the world by creating a sense of depth and distance. Objects that are farther away appear smaller than those that are closer.
As for the orthographic viewpoints, objects are render without any sense of depth or distance meaning that all objects in the scene appear to be the same size, regardless of their distance

from the camera.

As it is mention, it is necessary to respect the aspect ration of the camera. To assure that, calculations were done to compensate the width, maintaining/assuming the height constant; Figure 2.

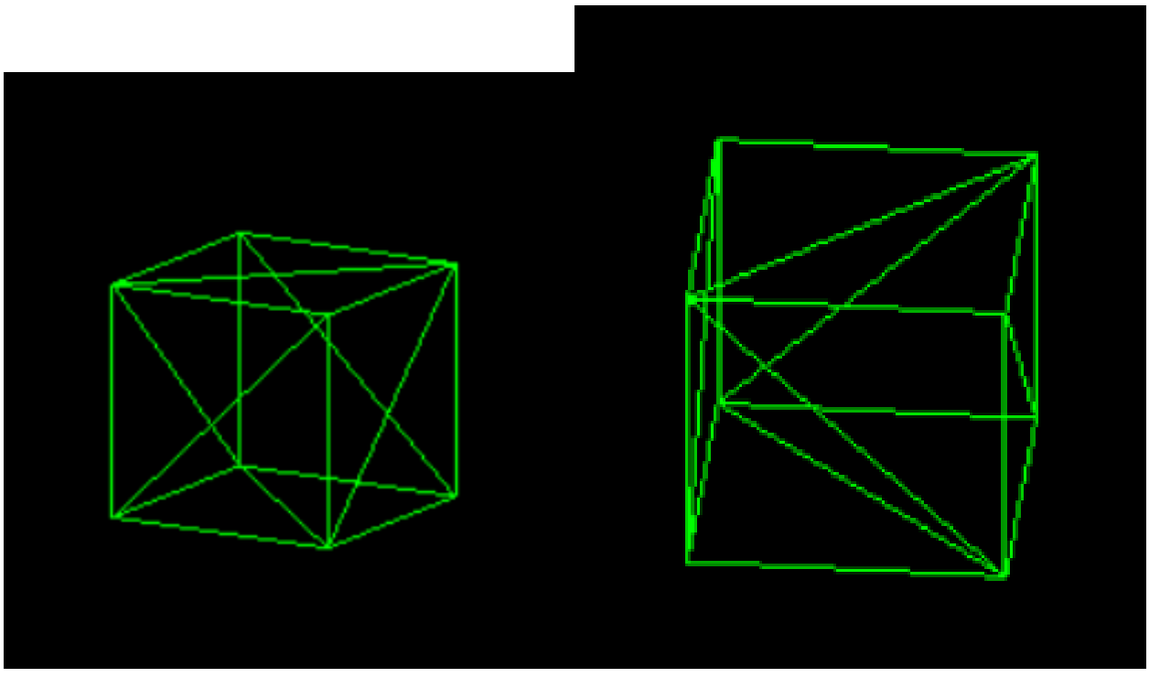
```
const correct_width_left = -(((6 * window.innerWidth) / window.innerHeight) / 2);
const correct_width_right = ((6 * window.innerWidth) / window.innerHeight) / 2;
const camera = new THREE.OrthographicCamera(correct_width_left, correct_width_right, 3, -3);
```

Figure 2: Code to apply the width correction and camera changing.

2.2 - Orbit control

In this exercise, students implement a three.js class that allow an easy control of the camera pose. To implement that, it is necessary to make an import of the class we pretended to use; we have tested the OrbitControls and TrackballControls. The controls are updated in the function that updates the view port.

Results are illustrated on Figure 3. The two images represent two different angles of view that the user desired to watch. The user clicked on the cube and dragged it chancing the angle.



(a) One angle. (b) Other angle.

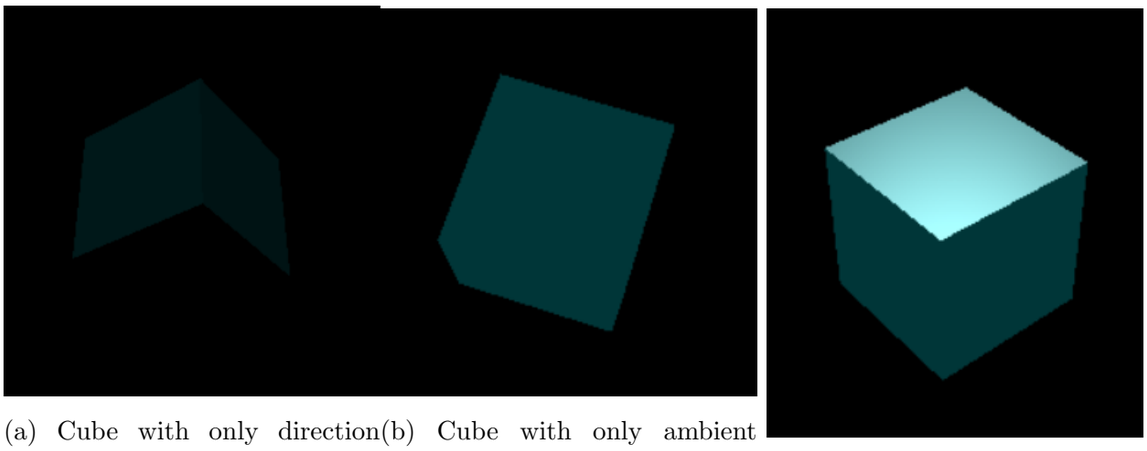
Figure 3: Results of 2.2 - Orbit control

2.3 - Lighting and materials

In this exercise, lights are going to be added. Before that, we got back to the perspective camera, disabled the wireframe and turned the rotation of the cube back on.

We were asked to create create a DirectionalLight at position (0, 5, 0) with color 0xffffff, that is, white, and intensity equal to 1.0.
In order for the object interact with the light, that is, light reflection is possible to see, it is needed a diferent material type from MeshBasicMaterial, in this case, a MeshPhongMaterial material. Finally, we added an ambient light.

In Figure 4a it is represented the cube with only direction light; in Figure 4b the cube with only ambient light; and finally, in Figure 4c, the cube with both.



(a) Cube with only direction light. (b) Cube with only ambient light. (c) Cube with both lines

Figure 4: Results of 2.3 - Lighting and materials

In Figure 4a, the direction light, we can see an ambient with its majority dark/off and a glimpse of the cube. In some angles it is possible to see a light reflection, which makes sense, because a direction light is pointing only in one direction, as the name says.

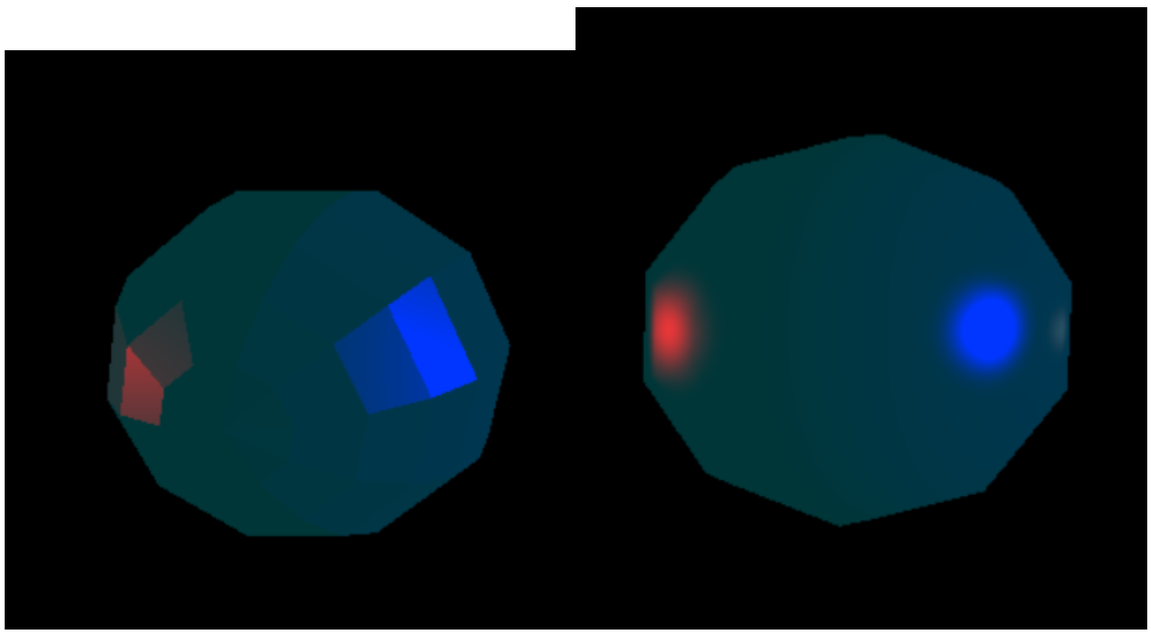
In Figure 4b its only the ambient light and as it is possible to notice, the cube is seen 100% and in a constant way which also makes sense.

Finally, combining the two lights, Figure 4c, it is possible to always see the cube but, is some angles it is possible to see the reflection of the direction light.

2.4 - Shading

In this exercise, students were asked to forget the cube and make two spheres.
To the question "What do the widthSegments and heightSegments parameters correspond to?", this two parameters change the "form factor" of the sphere, that is, these parameters make the sphere smoother, with a more sphere-like shape, without seeing the segments that make them up.
Then, we added ambient light and directional light from the previous exercise located between the two spheres with y=5.
We were encouraged to modify the flatShading option of one of the materials by toggling between true and false and observe the result. The results are represented in Figure 5.

As it is possible to see, the difference is noticed when the light hits the sphere. When toggled true, the reflection is seen in the same shape of the segments that compose the object; when toggle false, it is sign as a "normal"/expected reflection.



(a) flatShading = true. (b) flatShading = false.

Figure 5: Results of 2.4 - Shading; flatShading

As Optional/HomeWork, we applied MeshLambertMaterial type material to the spheres. This materials, the Lambertian, scatter light evenly in all directions so the specular coefficient and brightness are ignored. We also modify the properties of the spheres selecting values from a table to see the effects of different materials; we used emerald and gold.
We also added a red directional light in position (-5,0,0) blue directional light in position (5,0,0) and a green spotlight light in position (0,0,-5) with angle Math.PI/20 and target object in (-2.5,0,0). The result obtained is represented in Figure 6.

Figure 6: Results of 2.4 - Shading; Optional

2.5 - Transparency

Here, we added to the previews exercise two spheres with a slightly larger size around original spheres.
We have used a glassed type material and played with the opacity and transparency parameters. The results can be found in Figure 7.

As it is possible to notice, now the light reflection is hitting the transparent object but, a

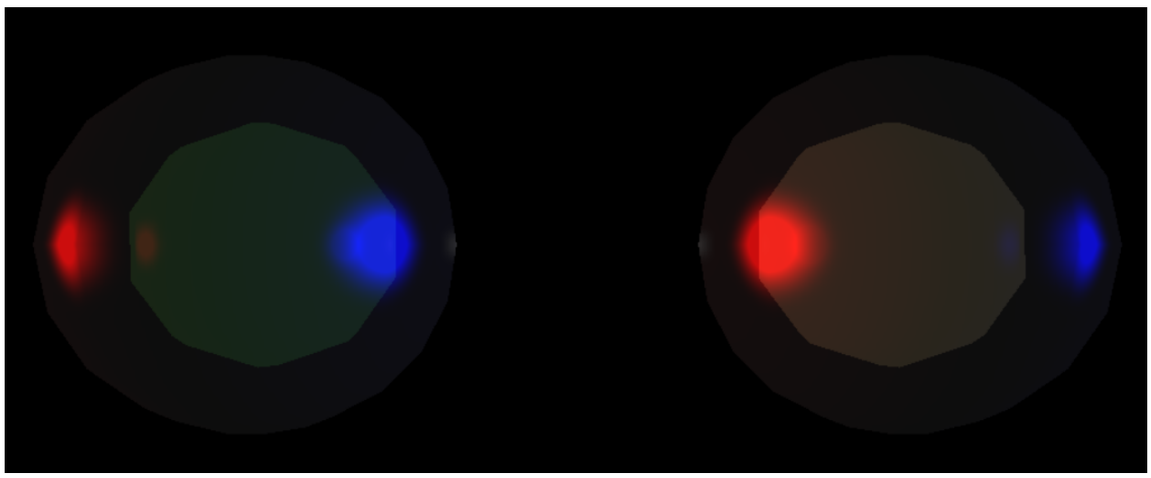


Figure 7: Results of 2.5 - Transparency

slight beam of light passes though the glass type material and hits the inner sphere. This made us concluded that this kind of material has realistic interactions with light because it reflects it, absorbs it and lets it through.

2.6 - Transformations (scale and rotation)

In exercise 6, we have created a new scene that consisted of a box and four spheres as it is possible to see in Figure 8.

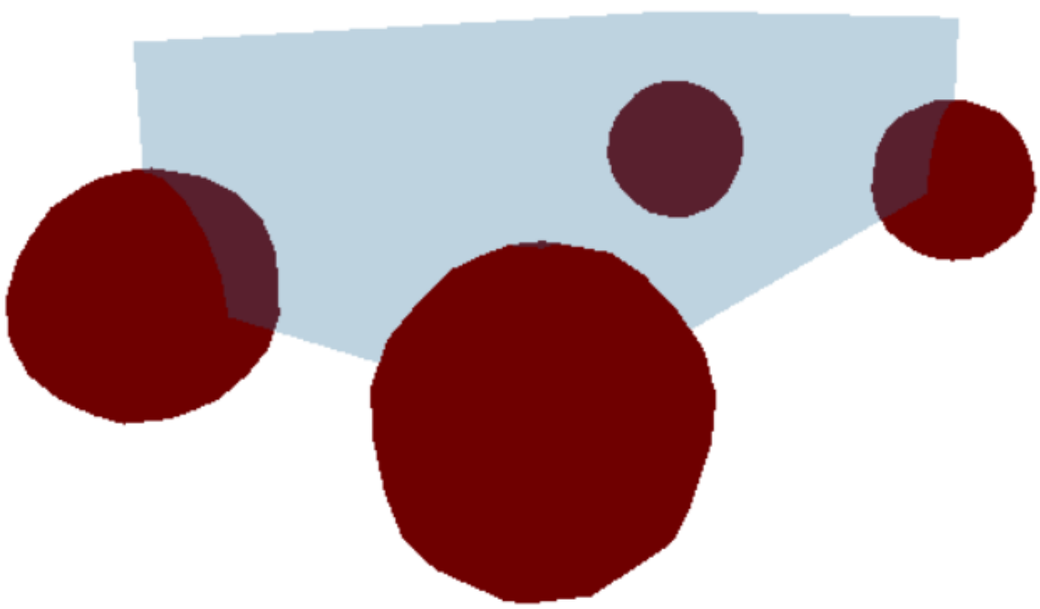


Figure 8: Results of 2.6 - Transformations (scale and rotation)

We were informed that instead of adding multiple separate meshes, we could add multiple meshes into a single THREE.Object3D() via the add command - possible to see in the code developed **here**.

We were also asked to view the transformation matrices of the parallelipiped and of one of the spheres on the console. The transformation matrix is a 4 by 4 matrix used to store

transformations done to an object and to perform further transformations.

$$\begin{bmatrix} a & b & c & \text{tx} \\ d & e & f & \text{ty} \\ g & h & i & \text{tz} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In this matrix:

- a, b, c represent the scaling and rotation components along the x, y, and z axes.
- d, e, f represent the scaling and rotation components along the x, y, and z axes.
- g, h, i represent the scaling and rotation components along the x, y, and z axes.
- tx represents the translation along the x-axis.
- ty represents the translation along the y-axis.
- tz represents the translation along the z-axis.

The top-left 3x3 matrix ($\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$) handles scaling and rotation, and the rightmost column ($\begin{bmatrix} \text{tx} \\ \text{ty} \\ \text{tz} \end{bmatrix}$) handles translation. The fourth row (0, 0, 0, 1) is a standard part of 4x4 matrices used in computer graphics.

This matrix is represented as an array of 16 elements with a column-major, meaning that the consecutive elements of a column are next to each other. For example, the element with index 0 is the a variable in the example matrix, and the element with index 1 is the d variable in the example matrix.

As we can see from Fig 9, the parallelipiped has a very simple transformation matrix, since we place it with coordinates $0, 0, 0$ and perform no transformation to it. The *Sphere_1* matrix is a little more complex, with some scaling and rotation components performed on the y and z axis due to the animation of the spheres. There is also a translation, which is represented in the elements with index 12,13 and 14, which correspond to the coordinates of the sphere.

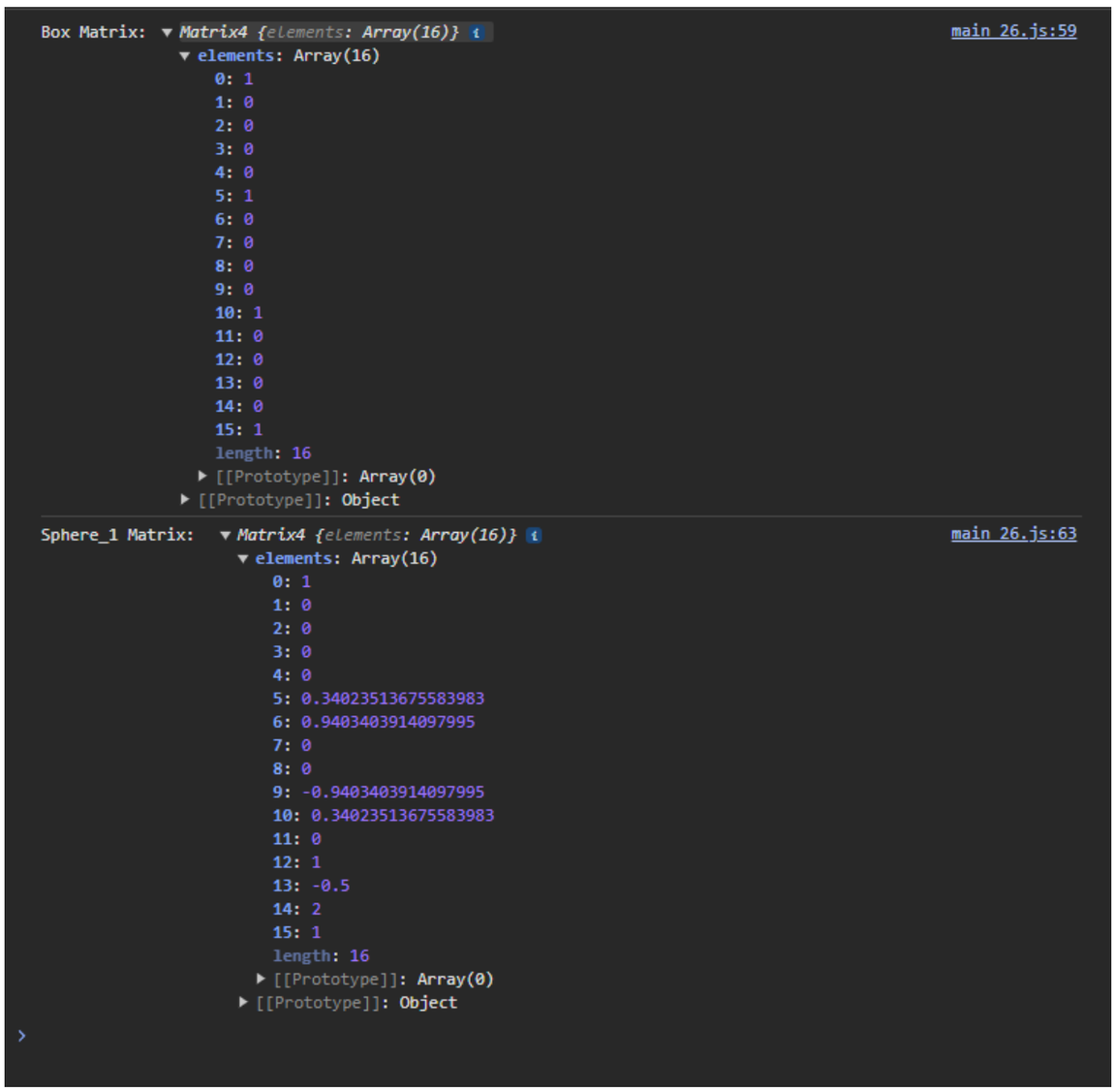


Figure 9: Transformation matrices of the parallelipiped ('Box') and of one of the spheres('Sphere_1')

2.7 - Transformations (rotations)

For the finally exercise, we were asked to add to the previews exercise a new object that represents a coordinate system using three red, green, and blue cylinders (CylinderGeometry) for each axis.
Another modification made was changing the spheres to cylinders.
The result is represented in Figure 10.

Note that an animation was requested. This consists in a moving car along a predefined circuit, in this case, a rotation of radius 1 around the point (0,0,-1)). Unfortunately, the animation is not possible to show but, once again, it is possible to see it **here**.

The axis cylinders were added to the box, like the spheres in Exercise 2.6.
It was necessary to rotate the cylinders that represent the wheels in $\pi/2$.
For the three cylinders to represent a coordinate system, each of the cylinders needs the correct orientation. To the x axis, the cylinder was rotated $\pi/2$ in relation to the z; to the y axis, the cylinder was not rotated; and finally, the z axis, the cylinder was rotated $\pi/2$ in relation to the x.

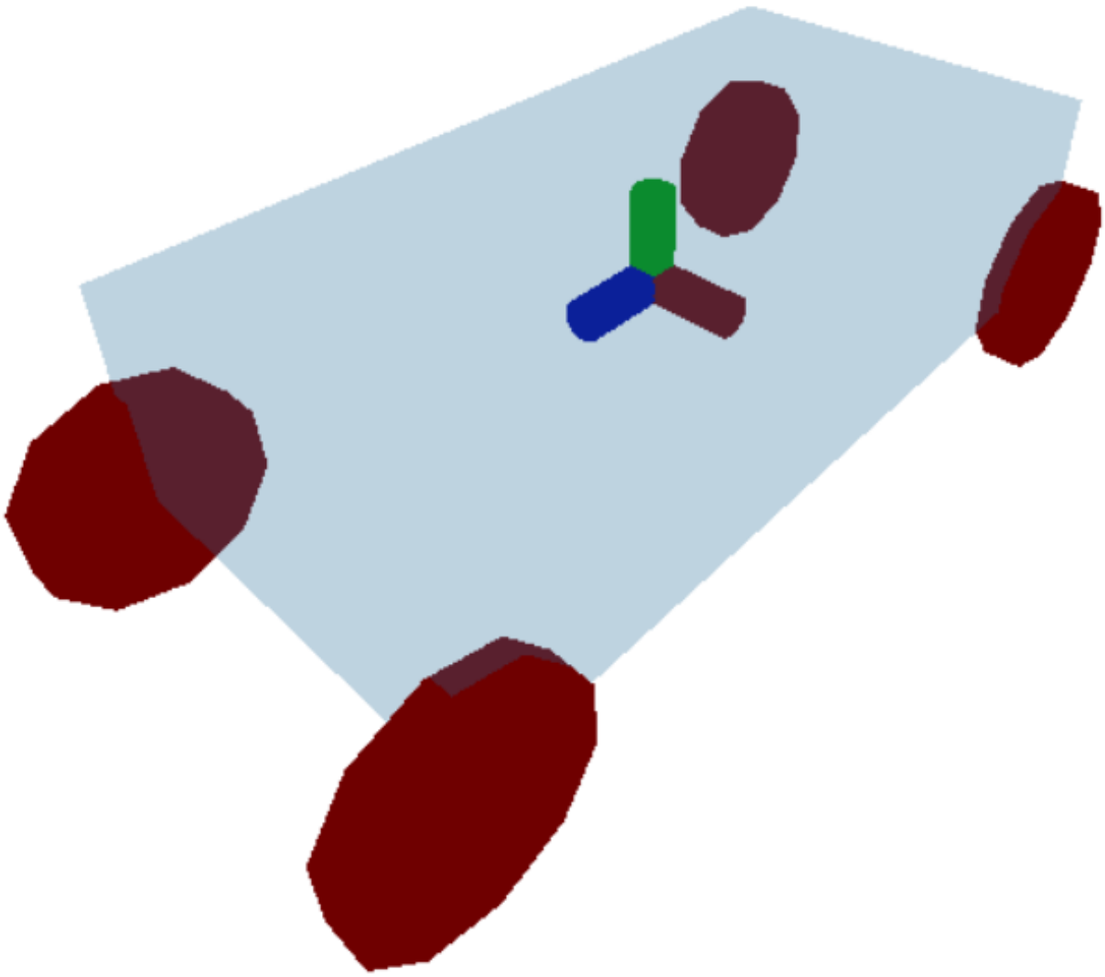


Figure 10: Results of 2.7 - Transformations (rotations)

To make the animation, we had to translate/update the box position (that include all the other objects, the axis and "wheels), which was achieved by using the following calculations:

```
const radius = 1;
const angle = Date.now() * 0.001;

const translation_X = 0 + radius * Math.sin(angle);
const translation_Y = 0;
const translation_Z = -1 + radius * Math.cos(angle);

box.position.set(translation_X, translation_Y, translation_Z);
```

The translation variables consist in the some of the point coordinates (0, 0, -1), then the sum of the radius of the rotation and, finally, the multiplication of the rotation angle. The Y axis remains constant throughout the animation.

Conclusion

To sum up the Lesson 2 - Projections, lighting and transformations, focusing on camera models, lighting, shading, transparency, and transformations, was concluded with success as we explored differences between perspective and orthographic cameras, the effects of lighting

on materials, and spatial manipulations completing successfully the exercises and presenting the expected results.