Three.js - Lesson 3

.

Visualização de Informação

Bruno Bastos 93302
DETI
Universidade de Aveiro

Leandro Silva 93446
DETI
Universidade de Aveiro

***Abstract*—Report that shows the resolution of the second Lesson of Three.js of the Information Visualization course. It provides the explanation for the proposed exercises as well as some of the obstacles that appear during the development.**

***Index Terms*—Three.js**

I. Introduction

In computer graphics, texture mapping is a method for defining the surface texture or color information in a 3D model. It allows the user to better understand what is the object. In this report we will shows examples on how to add textures to objects and how to interact with them using Three.js.

II. Texture Plane

In the first example we added a texture to a plane.

```
var texloader = new THREE.TextureLoader();
var tex = texloader.load("./lena.jpg");

const geometry = new THREE.PlaneGeometry(10, 10);
const material = new THREE.MeshBasicMaterial({ map:
    tex });
const plane = new THREE.Mesh(geometry, material);
scene.add(plane);
```

Using the Three.js texture loader, we loaded the ”lena.jpg” image. Then, the texture was added to the material using ”map” attribute. Figure 1 shows the final result.

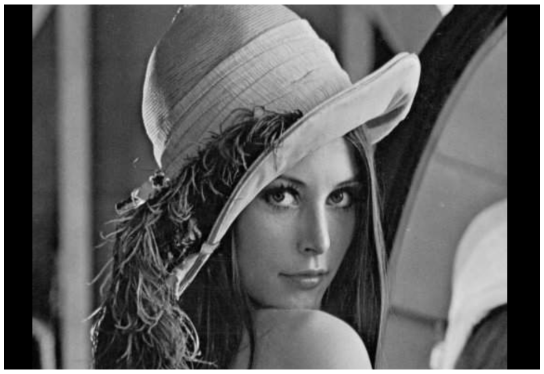

Fig. 1. Plane with a texture.

III. Cube Texture

The next exercise uses a cube mesh and the objective is to add a texture to each of its faces.

```
const materials = [];
for(let i=1; i <= 6; i++) {
    materials.push(new THREE.MeshBasicMaterial({
        map: new
        THREE.TextureLoader().load('Im'+i+'.jpg')
        }));
}
```

Instead of loading one image, we needed to load every face of the cube using the texture loader, create the material and save it to an array. When creating the object just need to provide the array of materials instead of just one.

Figure 2 shows the final result for this exercise.

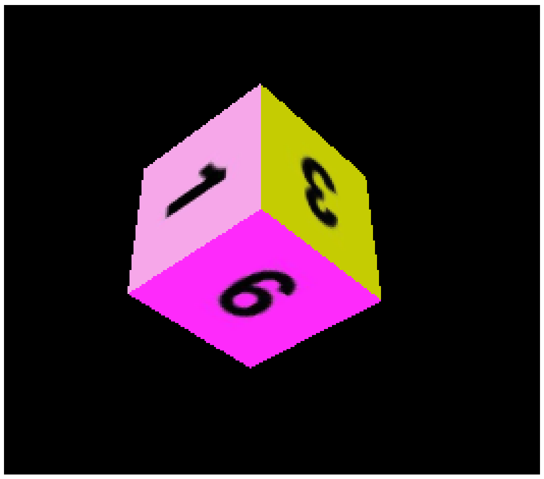

Fig. 2. Cube with different texture on each face.

IV. Earth Texture

In this exercise, we need to create a sphere with a texture of the earth. The earth needs a fixed z inclination and a y rotation speed.

```
let y_speed = 0.0025;
let z_incl = 0.41;
```

```
// sphere
const sphereGeometry = new THREE.SphereGeometry(1,
    32, 32);
let material = new THREE.MeshPhongMaterial({
        map: new THREE.TextureLoader().load(
            'earth_surface_2048.jpg'),
        specular: '#a9fcff',
        shininess: 100,
        flatShading: true
});
let earth = new THREE.Mesh(sphereGeometry,
    material);
earth.rotation.z = z_incl;
scene.add(earth)
```

The next step is to add directional lighting and ambient lighting.

```
// directional light
const directionalLight = new
    THREE.DirectionalLight(0xffffff, 1.0);
directionalLight.target.position.set(1,0,0)
scene.add(directionalLight);
scene.add(directionalLight.target);

// ambient light
const alight = new THREE.AmbientLight(0x333333);
scene.add(alight);
```

In the next part we need to add interaction to the sphere.

```
 function onDocumentKeyDown(event){
    // Get the key code of the pressed key
    var keyCode = event.which;
    if(keyCode == 76) { // L
        directionalLight.visible =
            !directionalLight.visible;
    } else if (keyCode == 171) { // +
        directionalLight.intensity += 0.2;
    } else if (keyCode == 173) { // -
        directionalLight.intensity -= 0.2;
    } else if (keyCode == 37) { // ->
        y_speed += 0.0005;
    } else if (keyCode == 39) { // <-
        y_speed -= 0.0005;
    } else if (keyCode == 38) { // ^
        z_incl += 0.05;
    } else if (keyCode == 40) { // v
        z_incl -= 0.05;
    }
}
```

When pressing a specific key we want to do an action. If we press ”L” we want to toggle the directional light, making it on and off. The ”+” and ”-”, increase and decrease, respectively, the light intensity of the directional light. The left and right arrows, decrease and increase the y rotation speed of the sphere. The up and down arrows, increase and decrease the z inclination of the sphere.

Figure 3 shows the final result for the earth sphere.

V. Moon

Now that we have the earth, we need to add the moon to rotate around the earth. The first important thing it that

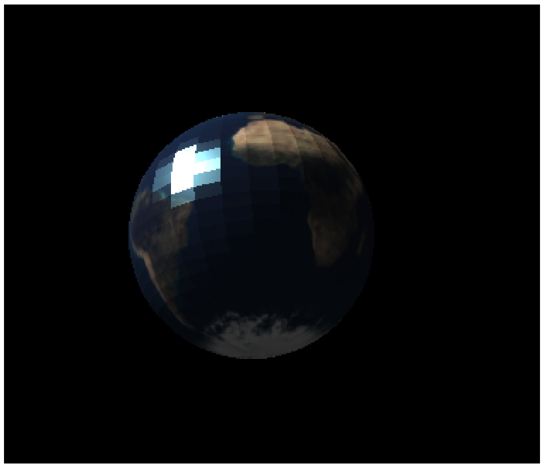

Fig. 3. Sphere with earth texture affected by light.

the moon needs to be attach to the earth so that every transformation applied to the earth is also applied to the moon.

```
const DISTANCE_FROM_EARTH = 356400;
const PERIOD = 28;
const INCLINATION = 0.089;
const SIZE_IN_EARTHS = 1 / 3.7;
const EARTH_RADIUS = 6371;
```

First we define some constants to help position the moon in the right place.

```
const moon_geometry = new
    THREE.SphereGeometry(SIZE_IN_EARTHS, 32, 32);
let moon_material = new THREE.MeshPhongMaterial({
        map: new THREE.TextureLoader().load(
            'moon_1024.jpg'),
        specular: '#a9fcff',
        shininess: 100,
        flatShading: true
});
let moon = new THREE.Mesh(moon_geometry,
    moon_material);
```

The moon is created as a sphere and its size is choose according to the earth’s size. Now we need to apply some transformations for the moon in order to make it rotate around the earth.

```
let distance = DISTANCE_FROM_EARTH / EARTH_RADIUS;
moon.position.set(Math.sqrt(distance / 2),
    0,-Math.sqrt(distance / 2));

// Rotate the moon so it shows its moon-face toward
    earth
moon.rotation.y = Math.PI;
moon.rotation.x = INCLINATION;

// For animation
moon.rotation.y += (earth.rotation.y / PERIOD);

scene.add(moon)
earth.attach(moon);
```

The moon is also attached to the earth using earth.attach(moon).

Figure 4 the moon rotation around the earth.

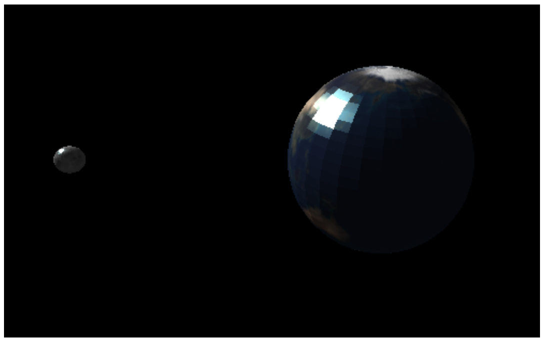

Fig. 4. Moon rotating around earth.

VI. Conclusion

The exercises were fully implemented without many difficulties, all due to the well documented Three.js library. Overall, we think that the objective of understanding and getting familiar with the Three.js library was fulfilled.