Lesson 3 - Texture and Interaction

Daniel Gomes
DETI, UA
Universidade de Aveiro
Aveiro, Portugal
Número Mecanográfico: 93015

Mário Silva
DETI, UA
Universidade de Aveiro
Aveiro, Portugal
Número Mecanográfico: 93430

***Abstract*—This report aims to describe the work developed in the context of the Information Visualization course of the University of Aveiro.**

***Index Terms*—Three.js, Texture, Interaction, Illumination**

I. Using a texture in a plane

For the first exercise, we used a simple scene with a cube, but instead of a box geometry, it was used the plane geometry

The *TextureLoader* from *Three.js* uses the *ImageLoader* internally for loading files. We used the *TextureLoader* for applying an image to the plane texture, on the *map* attribute of the material.

```
const geometry = new THREE.PlaneGeometry( 3, 3 );
var textloader = new THREE.TextureLoader();
var tex = textloader.load("lena.jpg")
const material = new THREE.MeshBasicMaterial({
    map: tex
});
var plane = new THREE.Mesh(geometry, material);
```

By changing the plane geometry size, it can be seen on the figure below 1, that the loaded image does not maintain its aspect ratio, and instead, just covers the plane geometry.

Fig. 1. Two Scenes with a Plane size 3×3 (left) and 6×3 (right)

II. Texture on a cube

For this exercise, it was used the box geometry to make a cube, and then, used several textures, one for each face of the cube. By using a list of materials, all with mapped with different image loaded as texture, and with different colors, we obtain the following result 2.

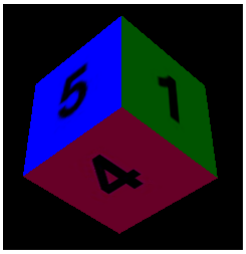

Fig. 2. Spinning cube with different textures on each face

III. Texture and Lighting

On the third exercise, the objective was to recreate the earth with an illumination effect coming from the sun. For this, it was used a sphere geometry, loaded with a texture with an image of the earth's map.

To recreate the sun's illumination on earth, it was used two different lights, a dark ambient light, to be able to see a little of the other side of the earth that doesn't get the direct sunlight, and a bright directional light coming from the side representing the sunlight.

```
var alight = new THREE.AmbientLight(0x333333);
scene.add(alight)
var dLight = new THREE.DirectionalLight(0xffffff);
dLight.position.set(1, 0, 0)
scene.add(dLight)
```

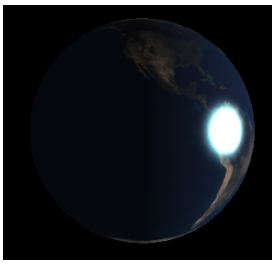

Fig. 3. Scene recreating the earth with sunlight effects

IV. Interaction and Lighting activation

The next exercises are about creating a way of interaction with the scene, in this case, keyboard interaction. This can be done by adding an *Event Listener* in the document, each time a key is pressed.

```
function onDocumentKeyDown (event) {
  var keyCode = event.which;
  console.log(keyCode)

  if (keyCode == "76") {
    if (scene.children.includes(dLight)) {
      scene.remove(dLight);
    } else {
      scene.add(dLight)
    }
 // if (sphere_mesh.material == mesh_phong_material){
 //   sphere_mesh.material = mesh_basic_material
 // } else {
 //   sphere_mesh.material = mesh_phong_material
 // }
  } else if (keyCode == "187") {
    dLight.intensity += 0.2;
  } else if (keyCode == "189") {
    dLight.intensity -= 0.2;
  }
}
document.addEventListener("keydown",
    onDocumentKeyDown, false)
```

We modified the previous exercise's code, to allow turning on/off the directional light via the L key, by removing the light from the scene, which can be seen on the figure 4, and also tested it by changing the material of the sphere to a *MeshBasicMaterial*, showed on the figure **??**.

It's possible to observe that by removing the light, the rest of the earth gets darker because of the ambient light, but by changing the material to *MeshBasicMaterial*, the earth becomes bright, this is because this material is not affected by lights.

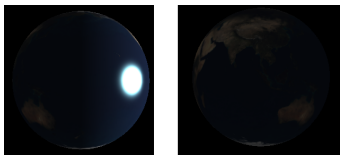

Fig. 4. Scene switching the direction light

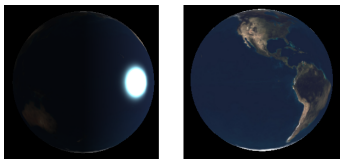

Fig. 5. Scene switching the sphere's material

We also added an option to increase and decrease the light intensity, in our case we chose the directional light, using the '+' and '-' keys. It can be observed on the figure below 6, a very bright earth by pressing the '+' key a couple of times, and a dark earth by pressing '-'.

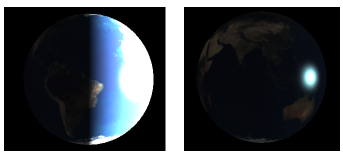

Fig. 6. Scene switching the directional light intensity

V. Modify Position and Rotation

The following exercise aimed to add the features of increasing and decreasing the speed of the rotation around both y and z axis. Thus, taking into consideration the previous exercise, there have been added more 4 keydown events responses: when the left and right arrow are pressed, the rotation of the y axis will increase and decrease, respectively, and on the up and down arrow keys the rotation of the z axis will increase and decrease too. To do so, firstly, two auxiliar variables for the and y and z rotation values have been created, which were necessary to keep the rotation values updates on every rendering of the scene. On the snippet bellow it is shown the update of the rotations values on the *animate* method, through this variables.

```
function animate() {
    requestAnimationFrame(animate);
    renderer.render(scene, camera);
    sphere_mesh.rotation.y += y_rotation;
    sphere_mesh.rotation.z += z_rotation;
}
```

Now, in order to reach the final goal of this exercise it is only necessary to increase/decreasing the auxiliar variables values according to the correct key down event, which can bee verified on the snippet below.

```
else if (keyCode == "37") {
    y_rotation+=0.01;
} else if (keyCode == "39") {
    y_rotation-=0.01;
} else if(keyCode == "38") {
    z_rotation+=0.01;
} else if (keyCode == "40") {
    z_rotation-=0.01;
}
```

VI. Concatenation of transformations / addition of the moon

On the final exercise of this assignment, it was intended to add new model representing the Moon, which should rotate around the Earth. In order to this, the first step taken was changing the mesh added to scene, to an *Object3D* that would contain both earth's and moon's meshes.

Next up, it was necessary to create the moon model. The difference of this model to the earth's one, consisted on the calculation of the rotation and position values that are considerably different since the goal was to make the moon rotate around the earth.

```
//Moon settings
var DISTANCE_FROM_EARTH = 356400;
var PERIOD = 28;
var INCLINATION = 0.089;
var SIZE_IN_EARTHS = 1 / 3.7;
var EARTH_RADIUS = 6371;

var sphere2 = new THREE.SphereGeometry(
    SIZE_IN_EARTHS, 32, 32
);
tex2 = textloader.load("moon_1024.jpg")
const mesh_phong_material2=new THREE.MeshPhongMaterial({
    specular: '#a9fcff',
    shininess: 100,
    map: tex2
});
const moon = new THREE.Mesh(
```

```
    sphere2,
    mesh_phong_material2
);

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;
moon.rotation.y += (earth.rotation.y / PERIOD);
planets.add(moon);
scene.add(planets);
```

Finally, once again, it is necessary to change the animate method so that the transformations are applied to both earth and moon. Thus, the *Object3D* previously created, will have its rotation values updated on this method, as it can be seen on the snippet below.

```
function animate() {
    requestAnimationFrame(animate);
    renderer.render(scene, camera);
    planets.rotation.y += y_rotation;
    planets.rotation.z += z_rotation;
}
```

The final result of this exercise can be visualized on the figure bellow 7.

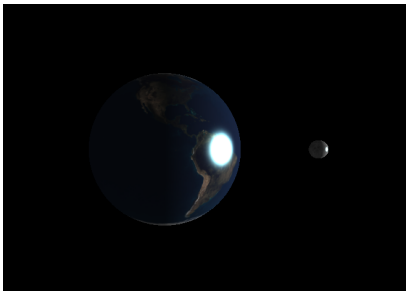

Fig. 7. Scene with Earth and Moon

VII. Conclusion

With this assignment we managed to gather more knowledge on the *Three.js* Technologies, and more how use textures, manage illumination effects on textures, and create interactions, in this case, keyboard interactions with the objects, such as adding and removing effects.