three.js

Gerard Antony Caramazza Vilá (gerard.caramazza.vila.15@ull.edu.es)
Edwin Plasencia Hernández (edwin.plasencia.28@ull.edu.es)

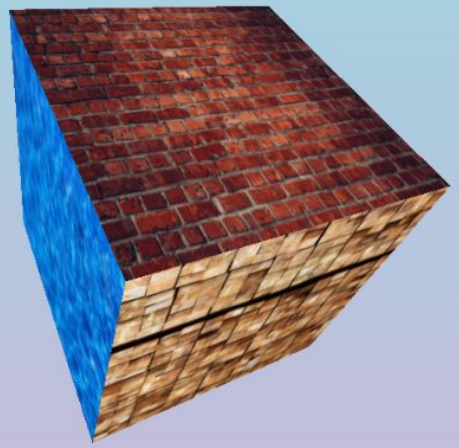
Index

Introduction

What is Three.js?

Three vs WebGL

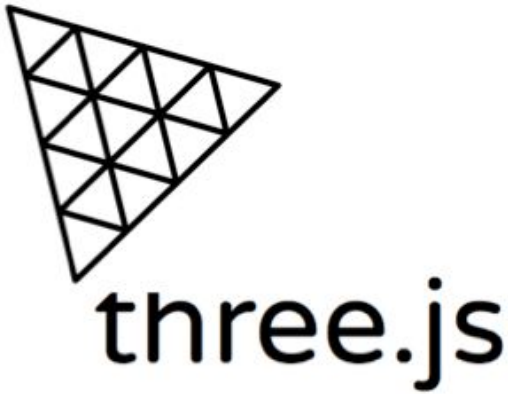
What can we do with it?

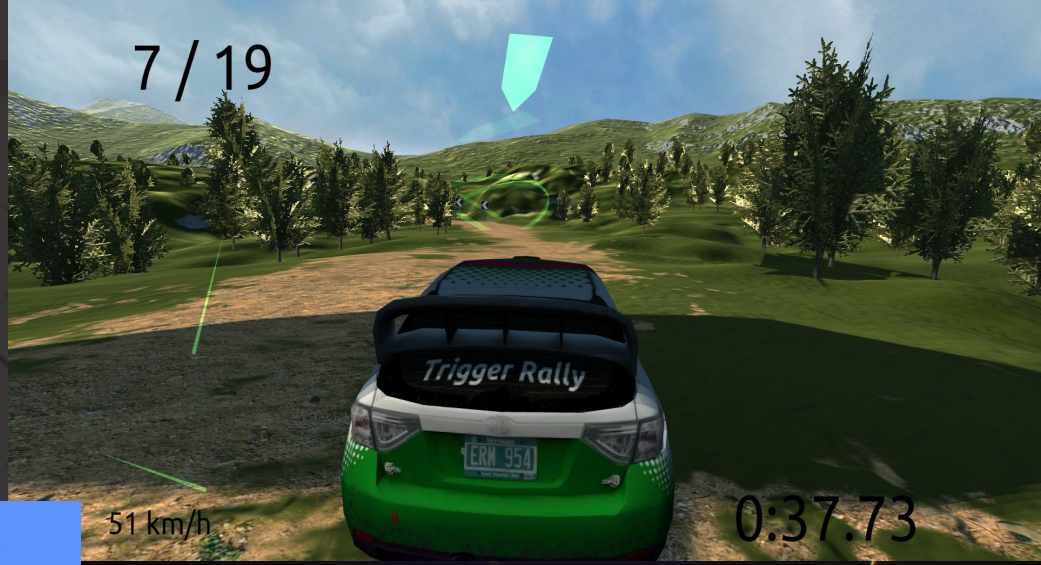


Abstraction

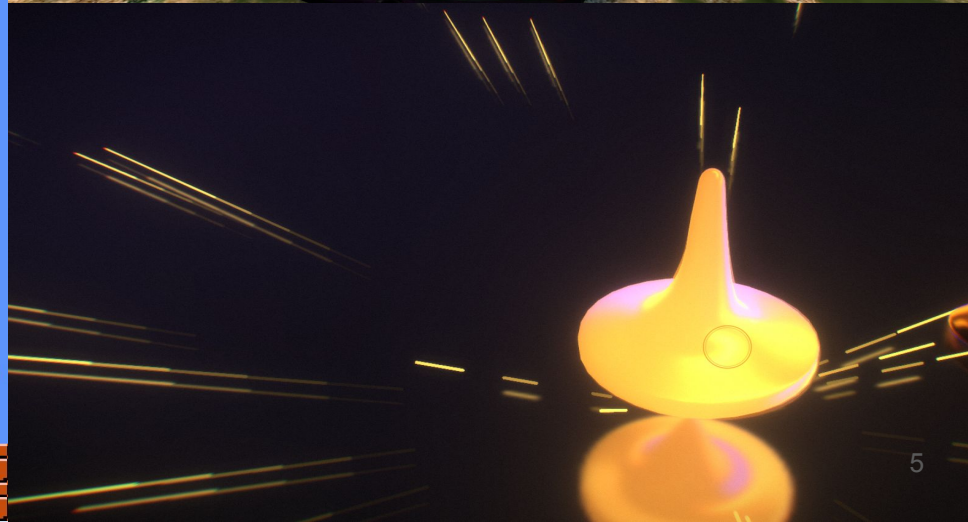
Low level

High level

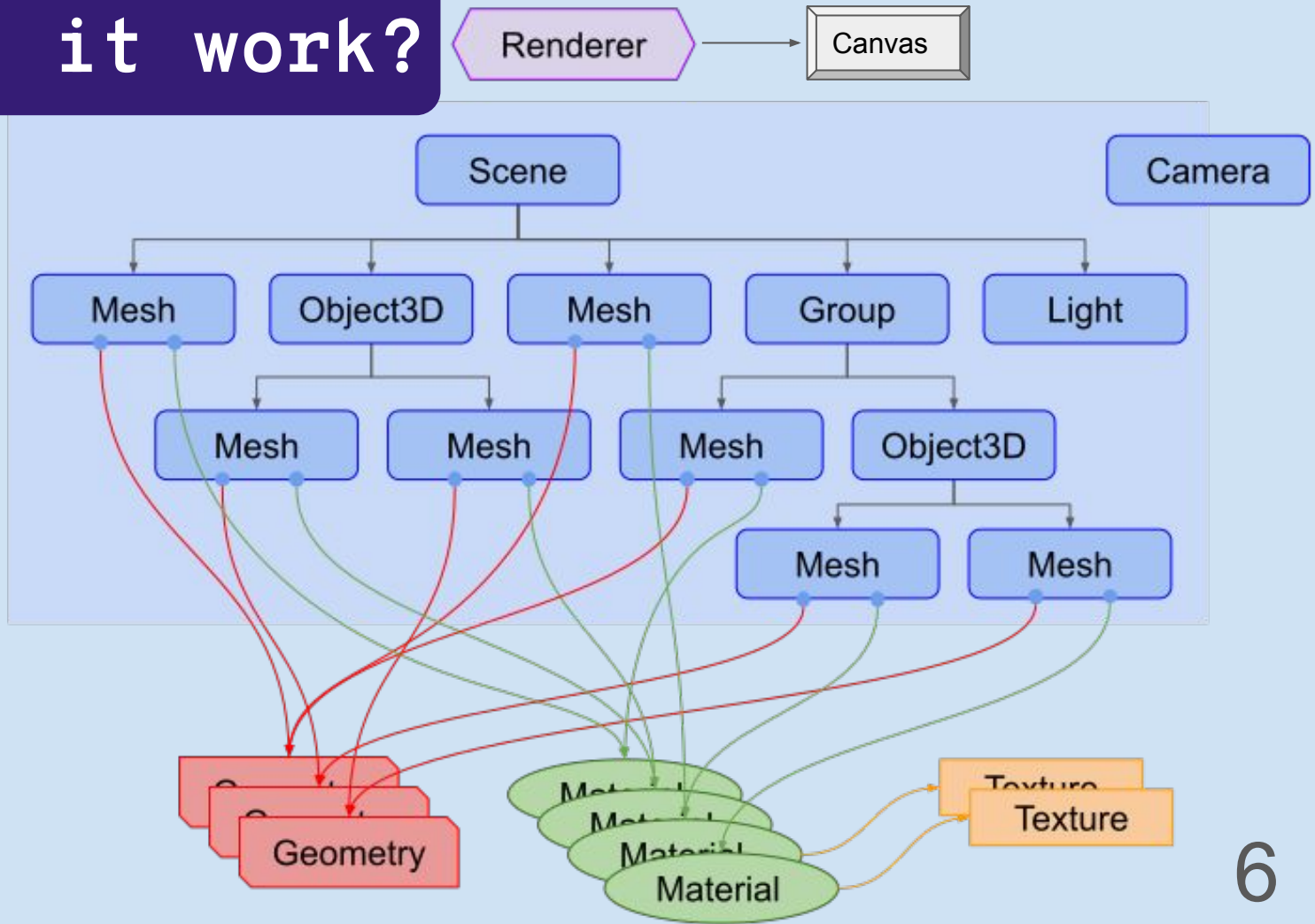



7 / 19
Trigger Rally
ERN 954
51 km/h
0:37.73


MARIO
000000
x00
WORLD
1-1
TIME
JELLY
MARIO BROS.
USE ARROW KEYS!
EAT TO START!

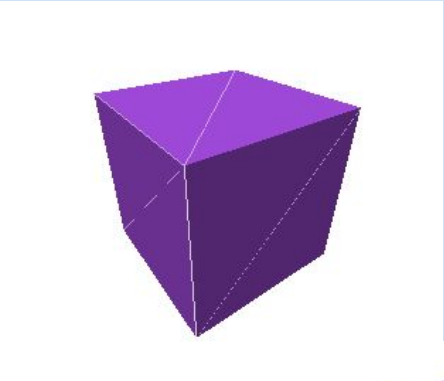


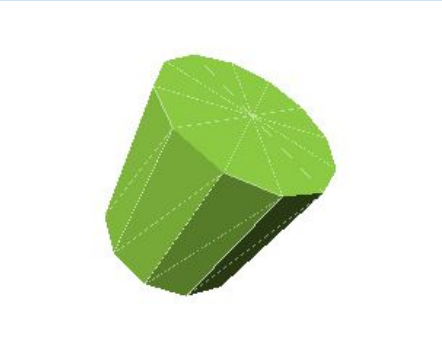
How does it work?

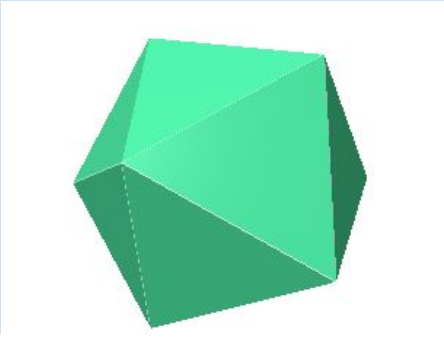


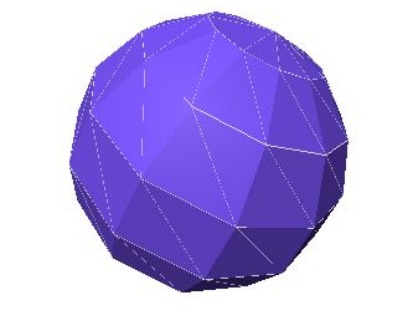
Geometry

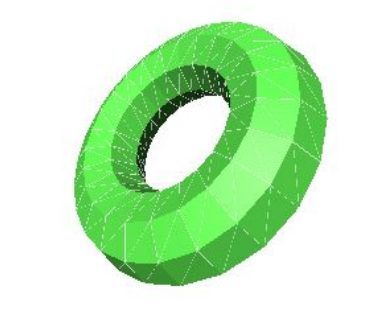
Shape of the object





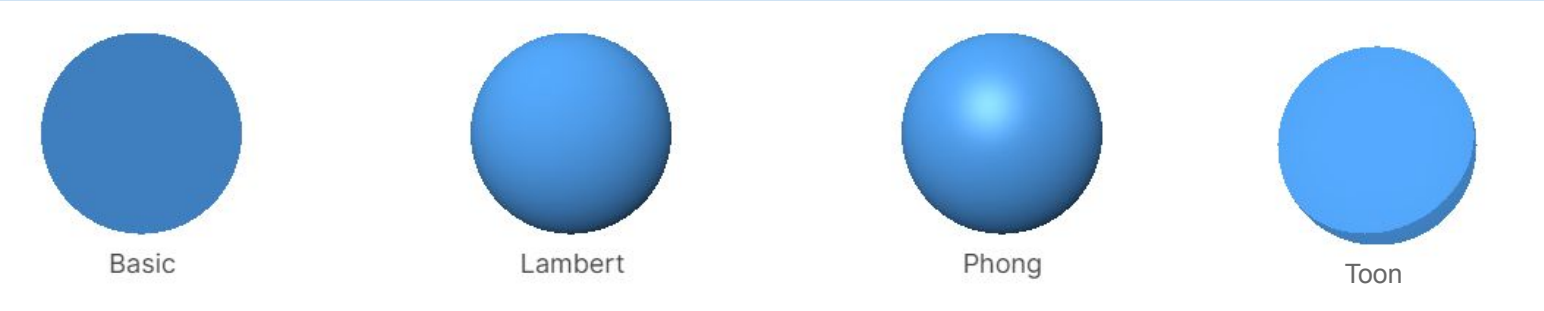




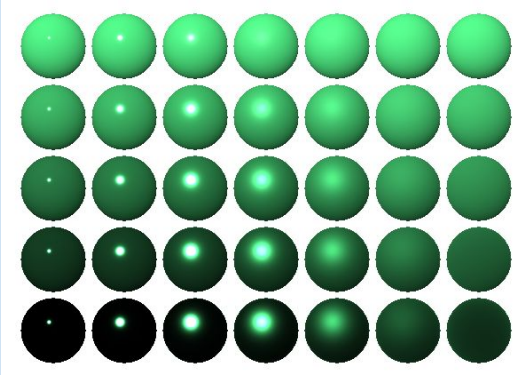


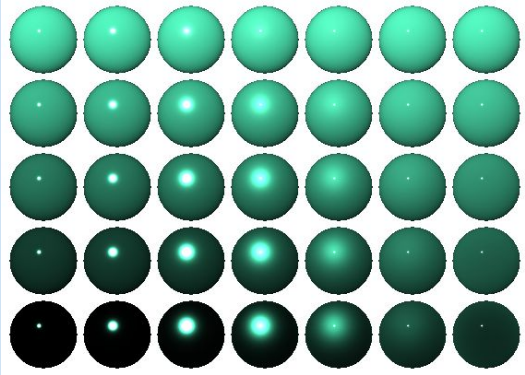
Material

Surface properties of the object



Standard

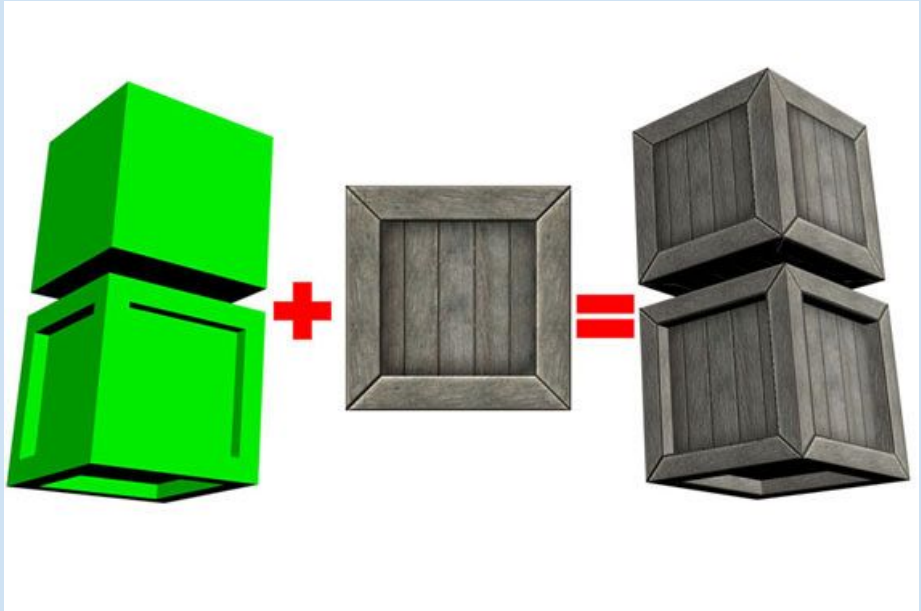




Clear Coat

Textures

Surface appearance of the object



Getting started

Installation

```
npm install three
```

Basic webpage

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Test</title>
    <script type="module" defer src="../src/index.js"></script>
  </head>
  <body>
    <canvas id="canvasBase" width="1920" height="965"></canvas>
  </body
</html>
```

Our script

Import

```
1 import * as THREE from '../node_modules/three/build/three.module.js'
```

Getting our canvas

```
1 const CANVAS = document.getElementById('canvasBase');
```

Creating our renderer

```
1 const RENDERER = new THREE.WebGLRenderer({
2   canvas: CANVAS,
3   alpha: true
4 });
```

```
1 RENDERER.render(SCENE, CAMERA);
```

Parameters[1]:

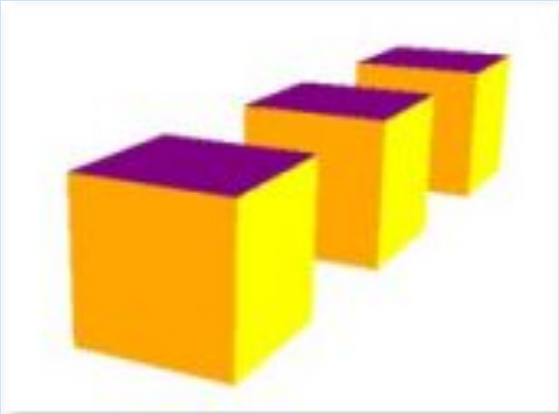
- canvas
- alpha
- antialias
- precision
- and more…

Creating our camera

Orthographic camera

```
OrthographicCamera()
```

Perspective camera

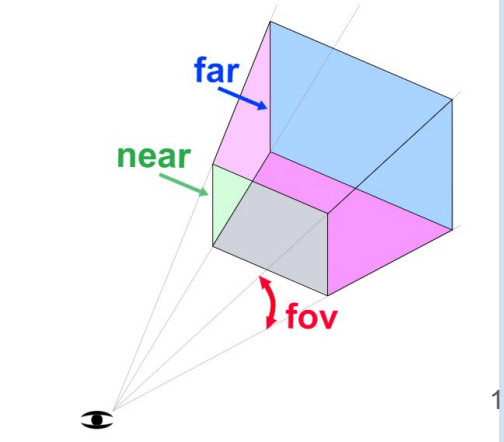


```
PerspectiveCamera()
```

Creating our camera

```
1 const FOV = 90;
2 const ASPECT_RATIO = (CANVAS.width / CANVAS.height);
3 const NEAR = 0.1;
4 const FAR = 50;
5 const CAMERA = new THREE.OrthographicCamera(FOV, ASPECT_RATIO, NEAR, FAR);
```

FIELD OF VIEW
ASPECT RATIO
NEAR DISTANCE
FAR DISTANCE



Creating a scene

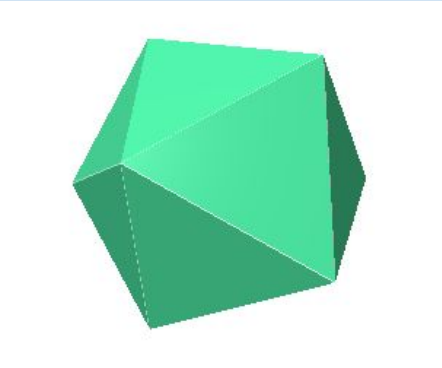
What will be rendered by the renderer

```
1 const SCENE = new THREE.Scene();
2 SCENE.add(/*SOMETHING*/);
```

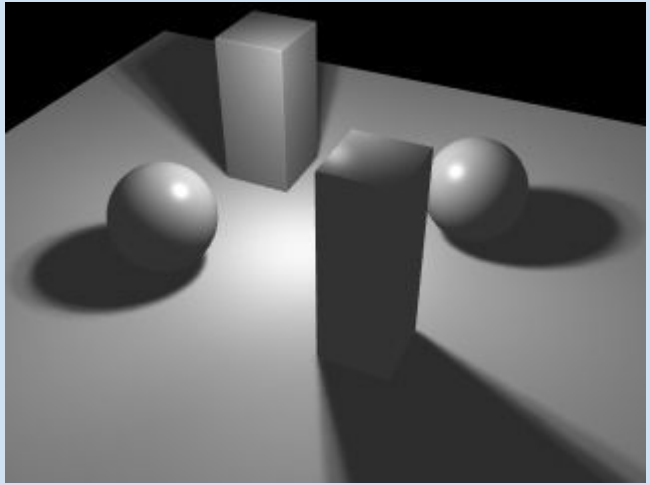
SOMETHING = Objects and/or lights

Adding things to our scene

Objects



Lights



Objects – Geometry

SphereGeometry(radius, widthSegments, heightSegments)

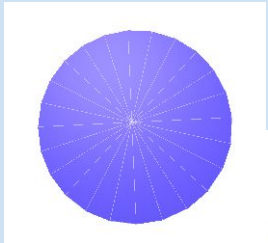
PlaneGeometry(width, height)

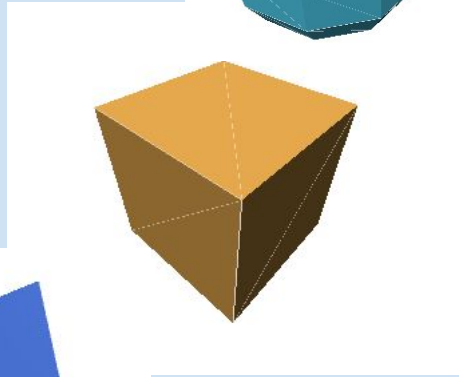
CircleGeometry(radius, segments)

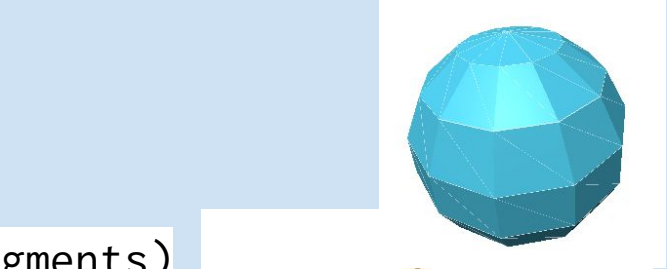
BoxGeometry(width, height, depth)

TextGeometry(text, {font, size, etc})

and more…







```
1 const CUBE = new THREE.BoxGeometry(1, 1, 1);
```

Objects – Materials

Shininess

MeshBasicMaterial()

No light effects

MeshLambertMaterial()

Light effects

only on vertices

MeshPhongMaterial()

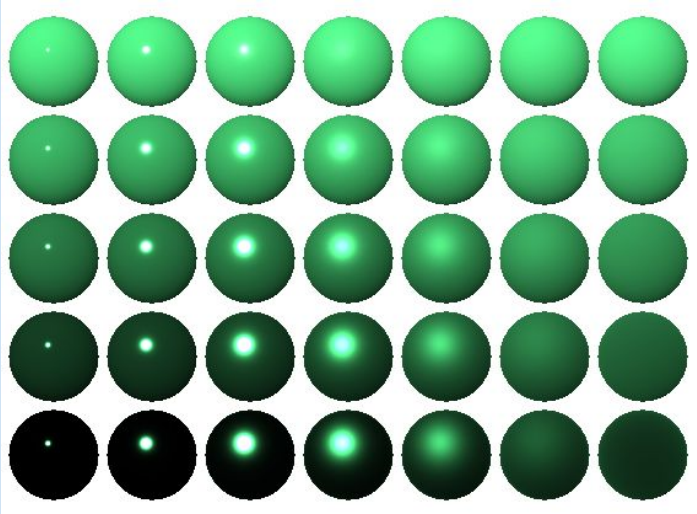
Light effects

everywhere

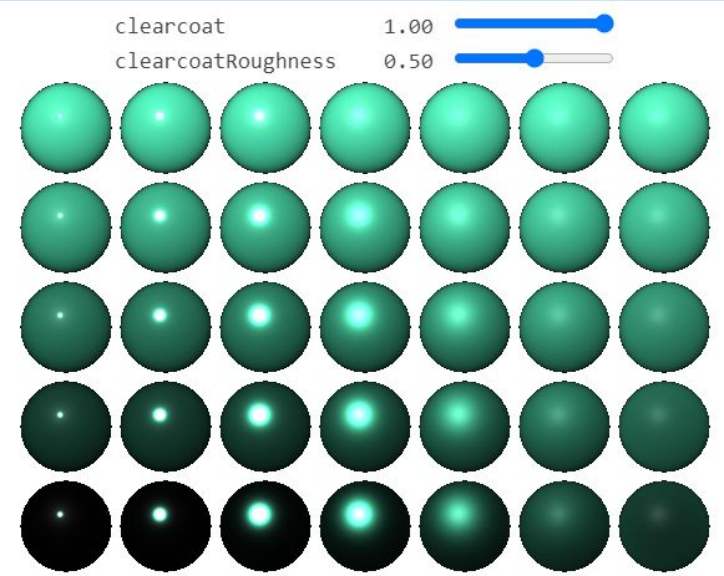
Objects – PBR Materials

Roughness

Metalness



MeshStandardMaterial()



MeshClearCoatMaterial()

Objects – Materials

Our example:

```
1 const MATERIAL = new THREE.MeshBasicMaterial({
2   color: 'red',
3   transparent: true,
4   opacity: 0.9,
5 });
```

Rendering the scene

```
CAMERA.position.set(1, 1, 1);
CAMERA.lookAt(0, 0, 0);
const OBJECT = new THREE.Mesh(CUBE, MATERIAL);
SCENE.add(OBJECT);
RENDERER.render(SCENE, CAMERA);
```

Let’s spice it up – Lights

First let’s change the material

```
1 const MATERIAL = new THREE.MeshPhongMaterial({
2   color: 'gray',
3   transparent: true,
4   opacity: 0.9,
5 });
```

Lights – Ambient light

```
const COLOR = 'white';
const INTENSITY = 1.5;
const LIGHT = new THREE.AmbientLight(COLOR, INTENSITY);
SCENE.add(LIGHT);
```

Lights – Hemisphere light

```
1 const COLOR_SKY = 'white';
2 const COLOR_GROUND = 'red';
3 const INTENSITY = 1;
4 const LIGHT = new THREE.HemisphereLight(COLOR_SKY, COLOR_GROUND, INTENSITY);
5 SCENE.add(LIGHT);
```

Lights – Directional light

```
1 const COLOR = 'white';
2 const INTENSITY = 1.5;
3 const LIGHT = new THREE.AmbientLight(COLOR, INTENSITY);
4 SCENE.add(LIGHT);
```

Lights – Point light

```
1 const COLOR = 'white';
2 const INTENSITY = 1;
3 const LIGHT = new THREE.PointLight(COLOR, INTENSITY);
4 LIGHT.position.set(5, 10, 4);
5 SCENE.add(LIGHT);
```

Lights – Spot light

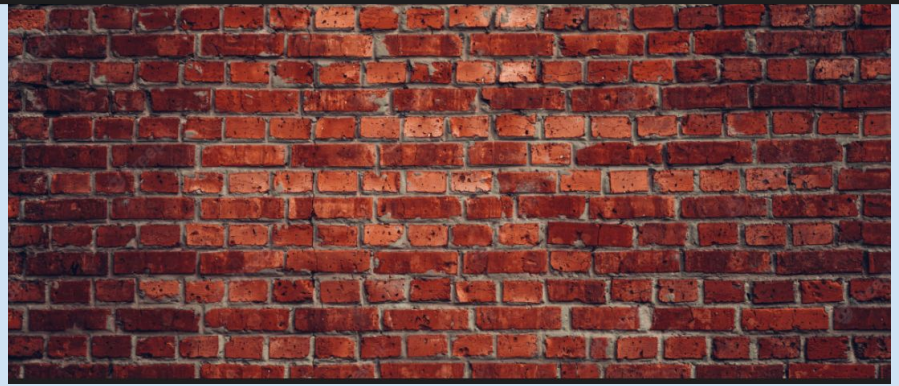
```
1 const COLOR = 'white';
2 const INTENSITY = 1;
3 const LIGHT = new THREE.SpotLight(COLOR, INTENSITY);
4 LIGHT.position.set(5, 10, 4);
5 LIGHT.target.position.set(0, 0, 0);
6 SCENE.add(LIGHT);
```

Textures

Texture loader

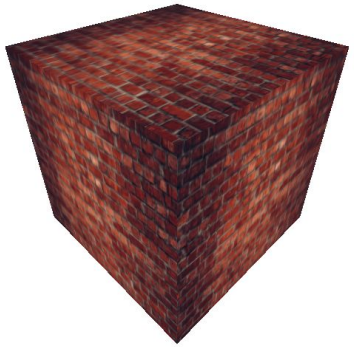
```
1 const LOADER = new THREE.TextureLoader();
```

Our texture:



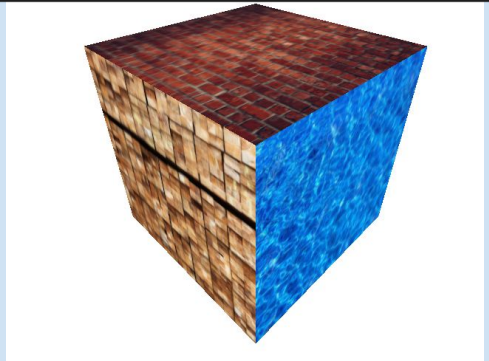
Applying a texture to a material

```
1 const MATERIAL = new THREE.MeshBasicMaterial({
2   color: 'white',
3   map: LOADER.load('./src/textures/bricks.jpg')
4 });
```



Applying a texture to a material

```
1 const MATERIALS = [
2     new THREE.MeshBasicMaterial({ map: LOADER.load('./src/textures/water.webp') }),
3     new THREE.MeshBasicMaterial({ map: LOADER.load('./src/textures/wood.jpg') }),
4     new THREE.MeshBasicMaterial({ map: LOADER.load('./src/textures/bricks.jpg') }),
5     new THREE.MeshBasicMaterial({ map: LOADER.load('./src/textures/water.webp') }),
6     new THREE.MeshBasicMaterial({ map: LOADER.load('./src/textures/wood.jpg') }),
7     new THREE.MeshBasicMaterial({ map: LOADER.load('./src/textures/bricks.jpg') }),
8 ];
```



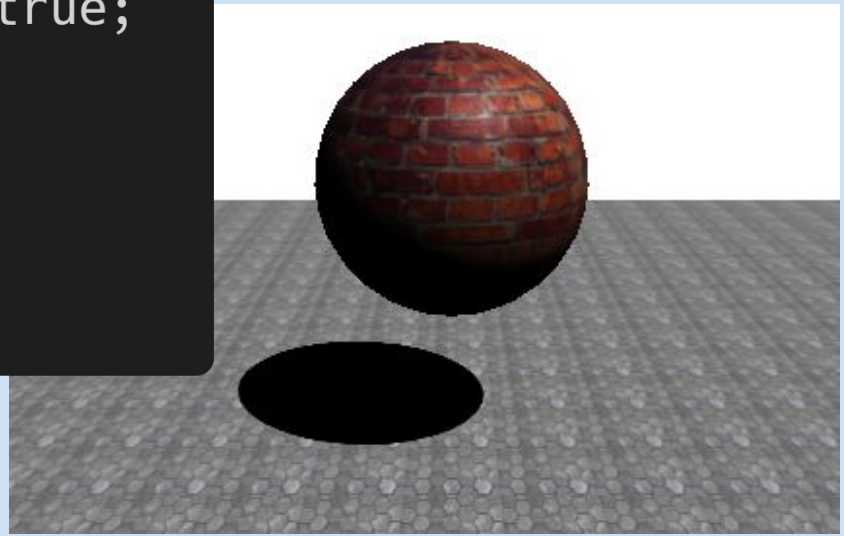
More things – Fog

```
1 const SCENE = new THREE.Scene();
2 {
3   const NEAR = 1;
4   const FAR = 2;
5   const COLOR = 'purple';
6   SCENE.fog = new THREE.Fog(COLOR, NEAR, FAR);
7   SCENE.background = new THREE.Color(COLOR);
8 };
```

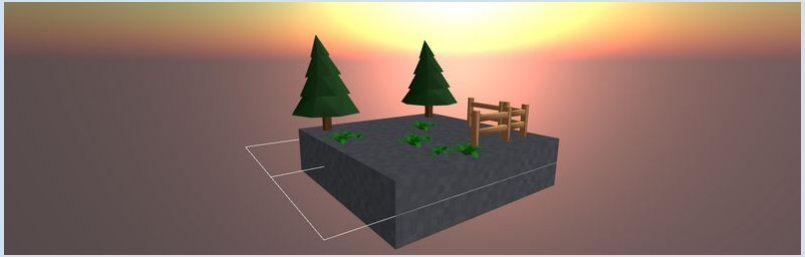
Fog with color and vision parameters

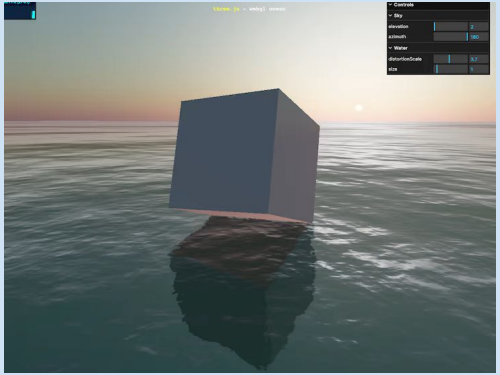
Shadows

```
RENDERER.shadowMap.enabled = true;
FLOOR.receiveShadow = true;
SPHERE.castShadow = true;
SPHERE.receiveShadow = true;
LIGHT.castShadow = true;
```



Final thoughts





Strong tool

Easy to use

Many possibilities

Programmer-friendly

Great amounts of documentation online

Bibliography

1 https://threejs.org/docs/index.html#api/en/renderers/WebGLRenderer
2 https://threejs.org/manual/
3 https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene
4 https://github.com/josdirksen/threejs-cookbook
5 https://davidlyons.dev/threejs-intro/#slide-0
6 https://riptutorial.com/three-js
7 https://threejs.org/examples/#webgl_animation_keyframes

About Us

Should you have any more specific questions please let us know!

Gerard Antony Caramazza Vilá

gerard.caramazza.vila.15@ull.edu.es

EDWIN PLASENCIA HERNANDEZ

edwin.plasencia.28@ull.edu.es

THANK YOU FOR YOUR ATTENTION!