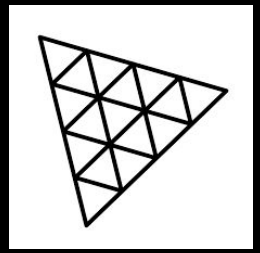
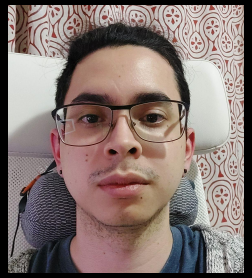
Three.js



Who we are



Julio Iván Carrasco Armas
alu0101110639@ull.edu.es

Francisco Marqués Armas
alu0101438412@ull.edu.es

Table of contents

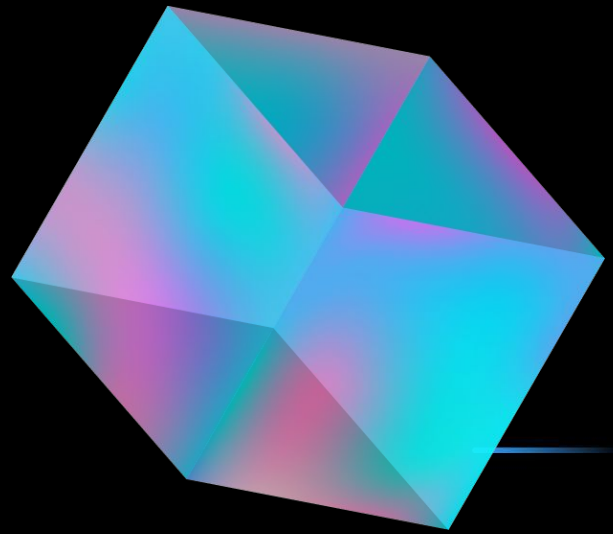
01

Introduction

What is three.js

Three.js is a JavaScript library and an API used to to create and display animated 3D computer graphics in a web browser using WebGL

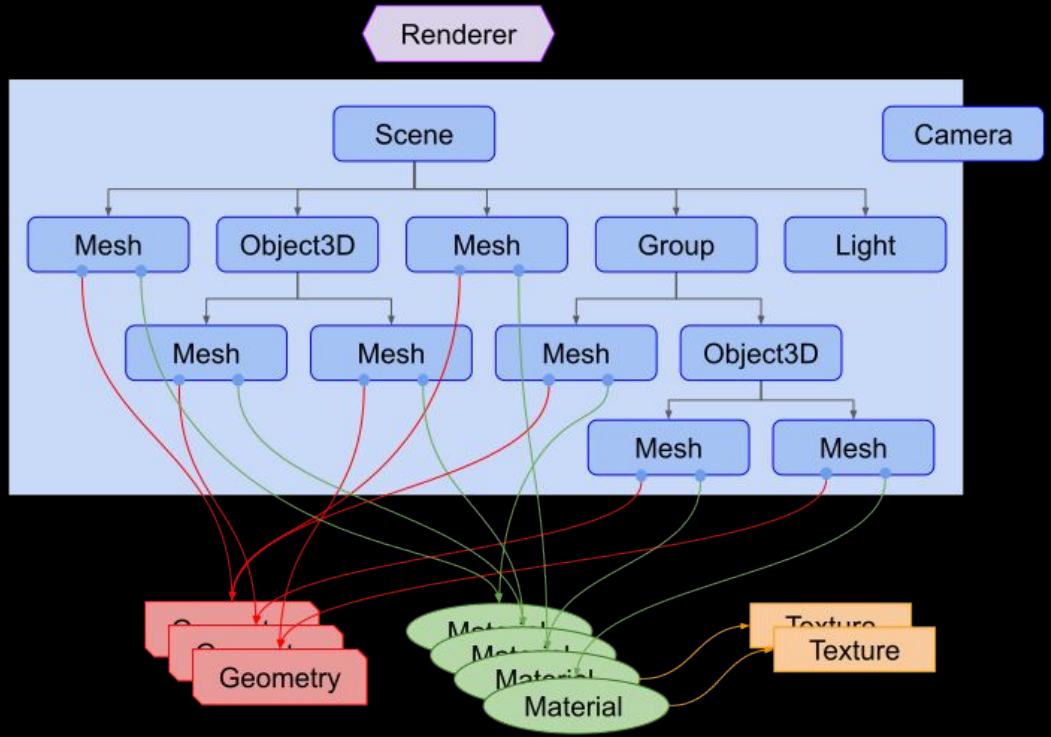
https://threejs.org/examples/#webgl_animation_keyframes



Three.js and WebGL

- Both can be used to draw graphics, but WebGL is considered low-level.
- Lights, shadows, materials, etc. all are integrated in Three.js
- WebGL is the renderer
- We can output the result to a <canvas>

Structure of a program



02

Preparation

Installation

Simple installation with npm

```
npm install three
```

Import

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

Import from CDN

We add this script to our html body

```
<script  type="importmap">{
            "imports": {
                "three": "https://threejs.org/build/three.module.js",
                "three/addons/": "https://threejs.org/examples/jsm/"
            }
        }</script>
        <script async
src="https://unpkg.com/es-module-shims@1.6.3/dist/es-module-shims.js"></script>
```

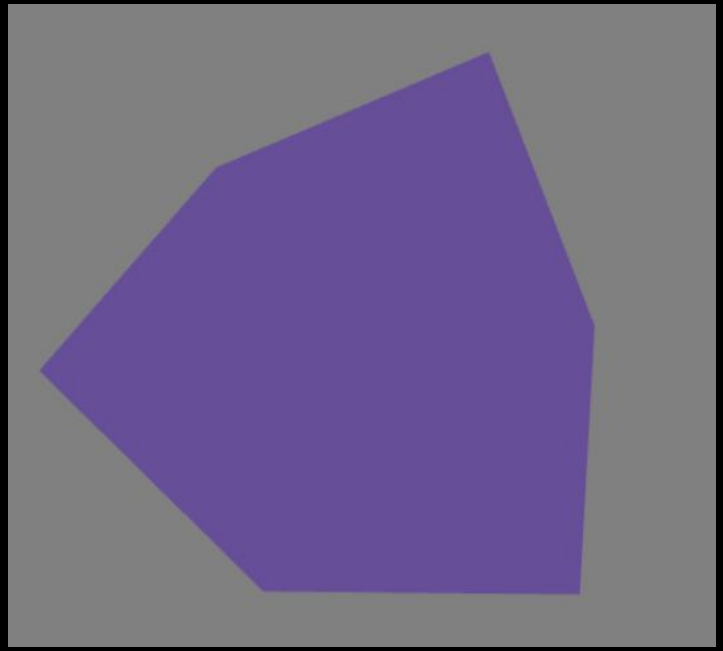
Creating the renderer

```
let CANVAS = document.getElementById('canvasBase'); // Canvas
const RENDERER = new THREE.WebGLRenderer({   // Renderer
canvas: CANVAS,
  alpha: true           // For transparency
});
```

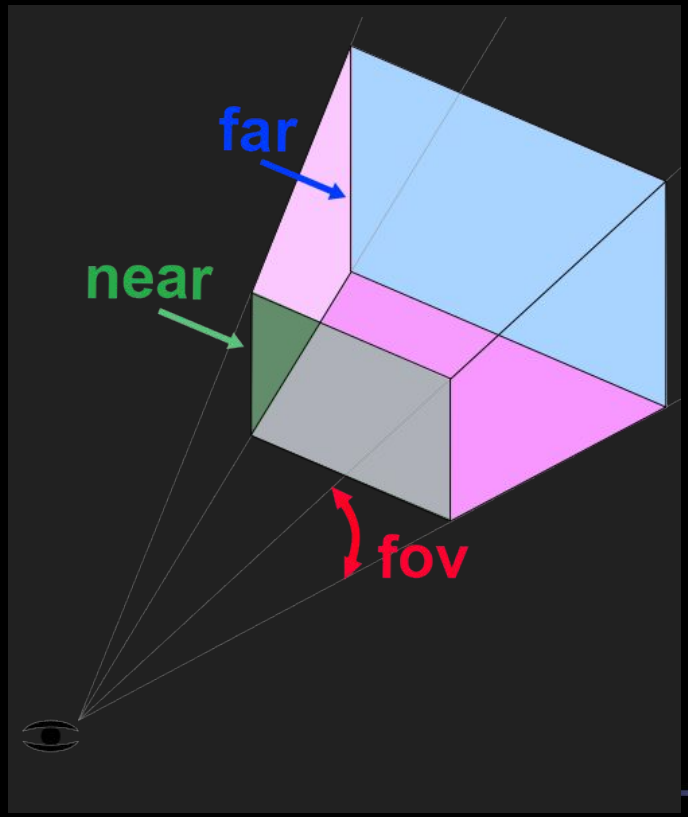
```
RENDERER.render(SCENE, CAMERA);
```

Basic example

ex-basic.js



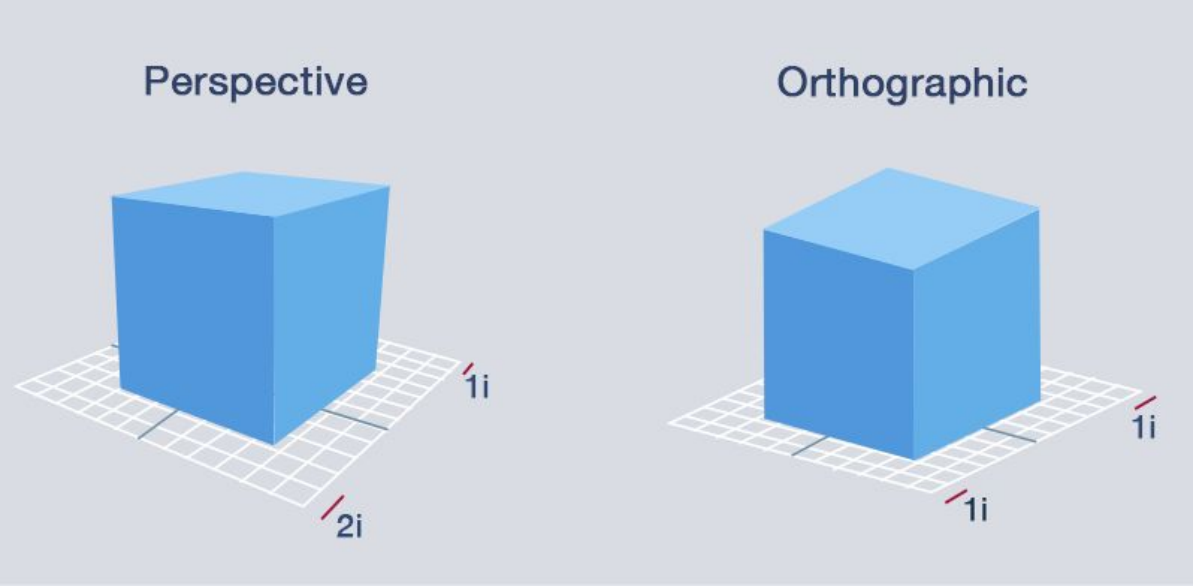
Camera



- Near
- Far
- Fov

Camera

[ex-1-view-example.js]

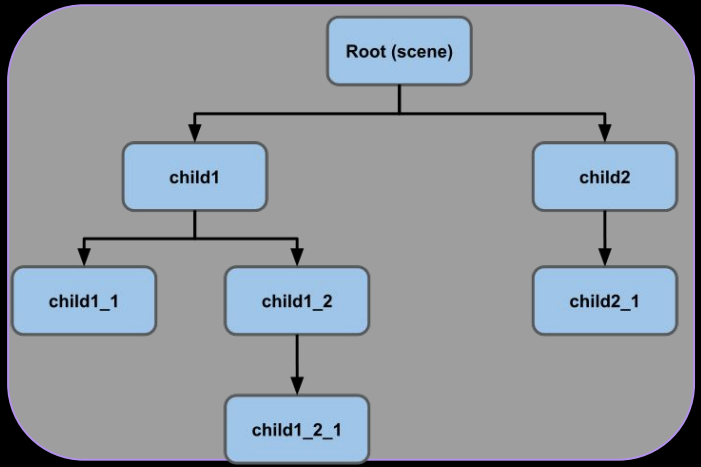


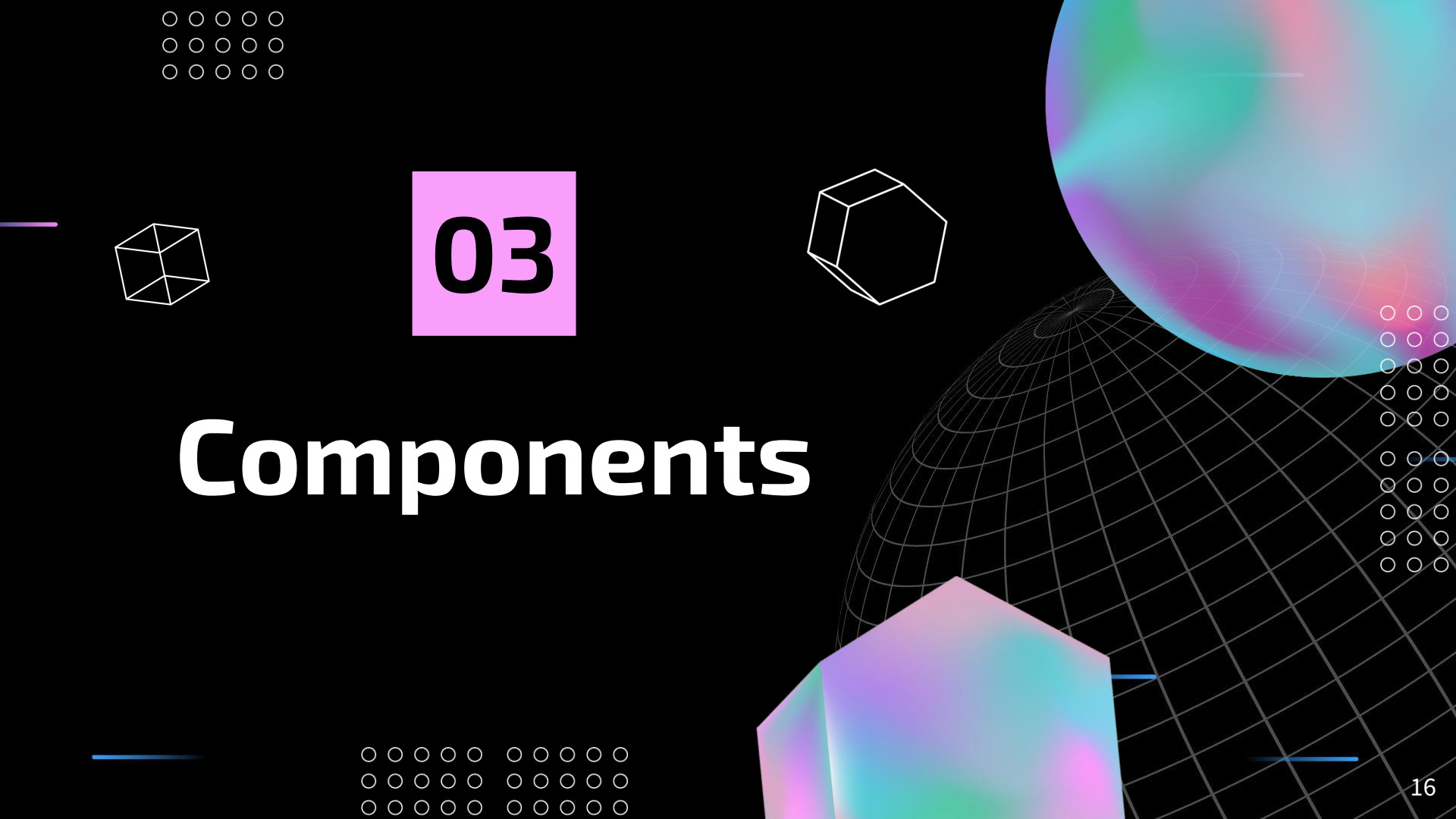
Scene

What will be rendered

```
const scene = new THREE.Scene();
scene.add(/*Elements*/)
```

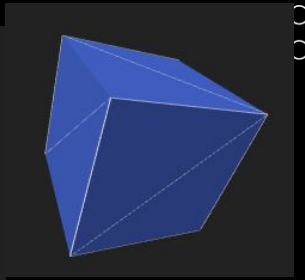
We can add objects or lights

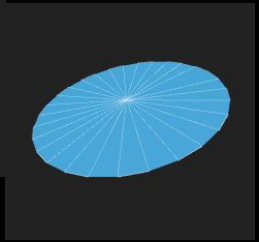



03
Components

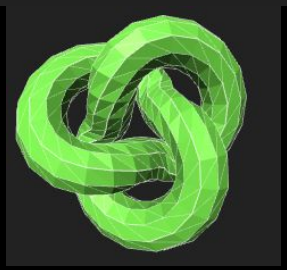
Objects

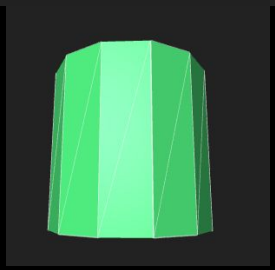
```
BoxGeometry(width, height, depth);
CircleGeometry(radius, segments);
CylinderGeometry(radiusTop, radiusBottom, height, radialSegments);
PlaneGeometry(width, height);
SphereGeometry(radius, widthSegments, heightSegments);
TorusKnotGeometry(radius, tubeRadius, tubularSegments, radialSegments, p, q);
// And many more
```

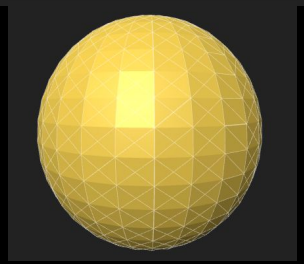


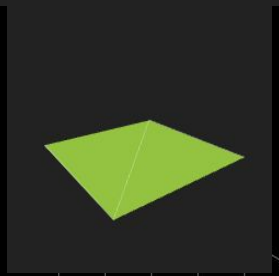


[Ex-2-figures-example.js]







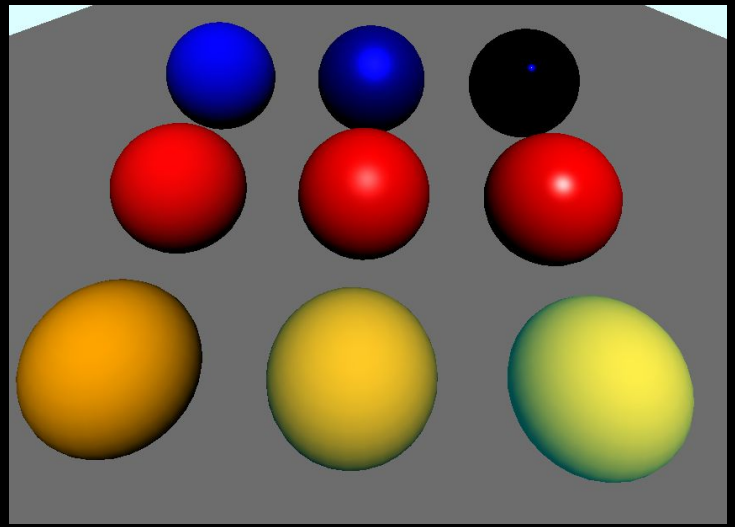


Materials

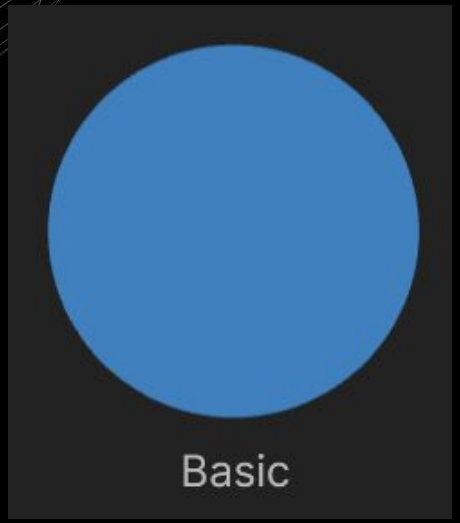
We can change the material of the object to change its colour and other properties like how shiny or rough it is.

- Phong
- Standard
- Lambert

[ex-3-materials-example.js]



How materials work



The MeshBasicMaterial is not affected by lights.

The MeshLambertMaterial computes lighting only at the vertices.

The MeshPhongMaterial computes lighting at every pixel.

Textures

Textures allow us to change even more the appearance of our objects, for example with custom images.
We need to load the textures:

```
const LOADER = new THREE.TextureLoader();
const BRICKS = new LOADER.load('../textures/bricks.jpg');
const TILES = new LOADER.load('../textures/tiles.jpg');
const WOOD = new LOADER.load('../textures/wood.jpg');
const GRASS = new LOADER.load('../textures/grass.jpg');
```

[ex-4-textures-example.js]

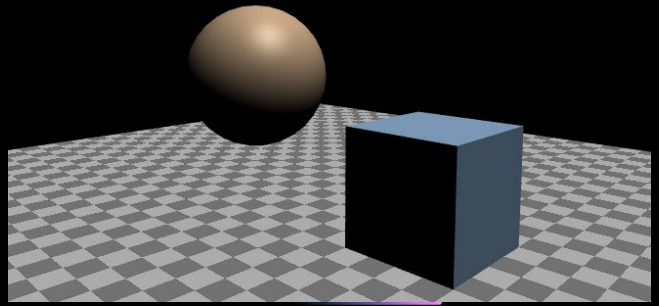
Lighting

Lights are necessary to illuminate our scene, there are various types of lights:

- Ambient light
- Directional light
- Spot light
- Point light
- Hemisphere light

```
new THREE.AmbientLight(AMBIENT_COLOR, AMBIENT_INTENSITY);
new THREE.DirectionalLight(DIRECTIONAL_COLOR, DIRECTIONAL_INTENSITY);
new THREE.SpotLight(SPOT_COLOR, SPOT_INTENSITY, DISTANCE, ANGLE);
new THREE.PointLight(POINT_COLOR, POINT_INTENSITY, DISTANCE, DECAY);
new THREE.HemisphereLight(HEMISPHERE_1_COLOR, HEMISPHERE_2_COLOR,
HEMISPHERE_INTENSITY);
```

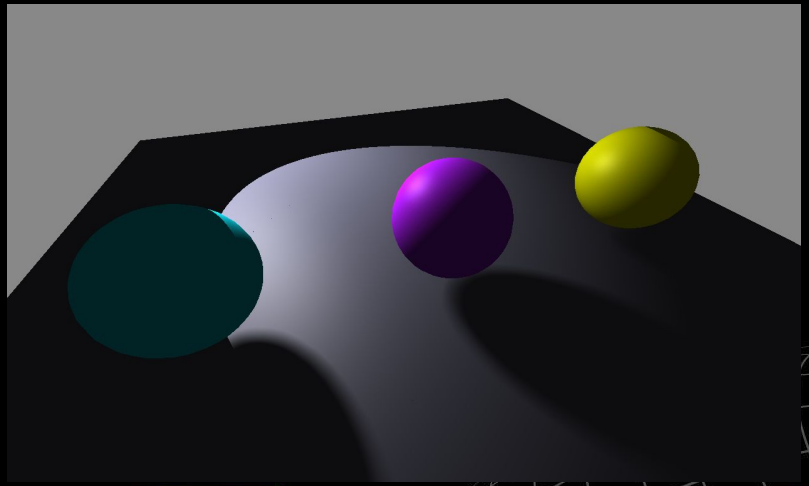
[ex-5-lights.js]



Shadows

Three.js uses shadow maps, a technique that consists of rendering a depth map of the scene from the perspective of a light source, which is then used to determine which objects are shadowed and which are not. We have four types of shadow maps that define the WebGL shadowMap.type

- BasicShadowMap
- PCFShadowMap
- PCFSoftShadowMap
- VSMShadowMap

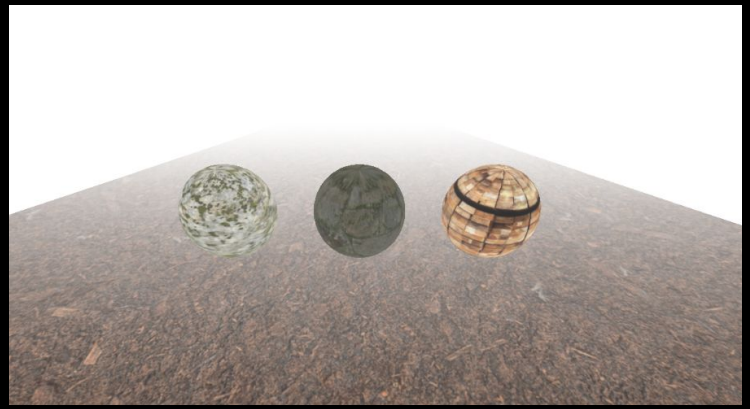


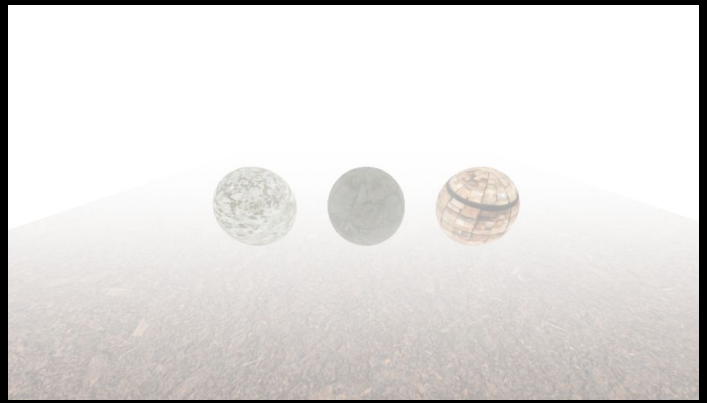
ex-6-shadows.js

Fog

Fog in a 3D engine is generally a way of fading to a specific color based on the distance from the camera. In Three.js we have two types of fog, while one is closer to reality the other type is more commonly used since you can choose where to apply the fog.

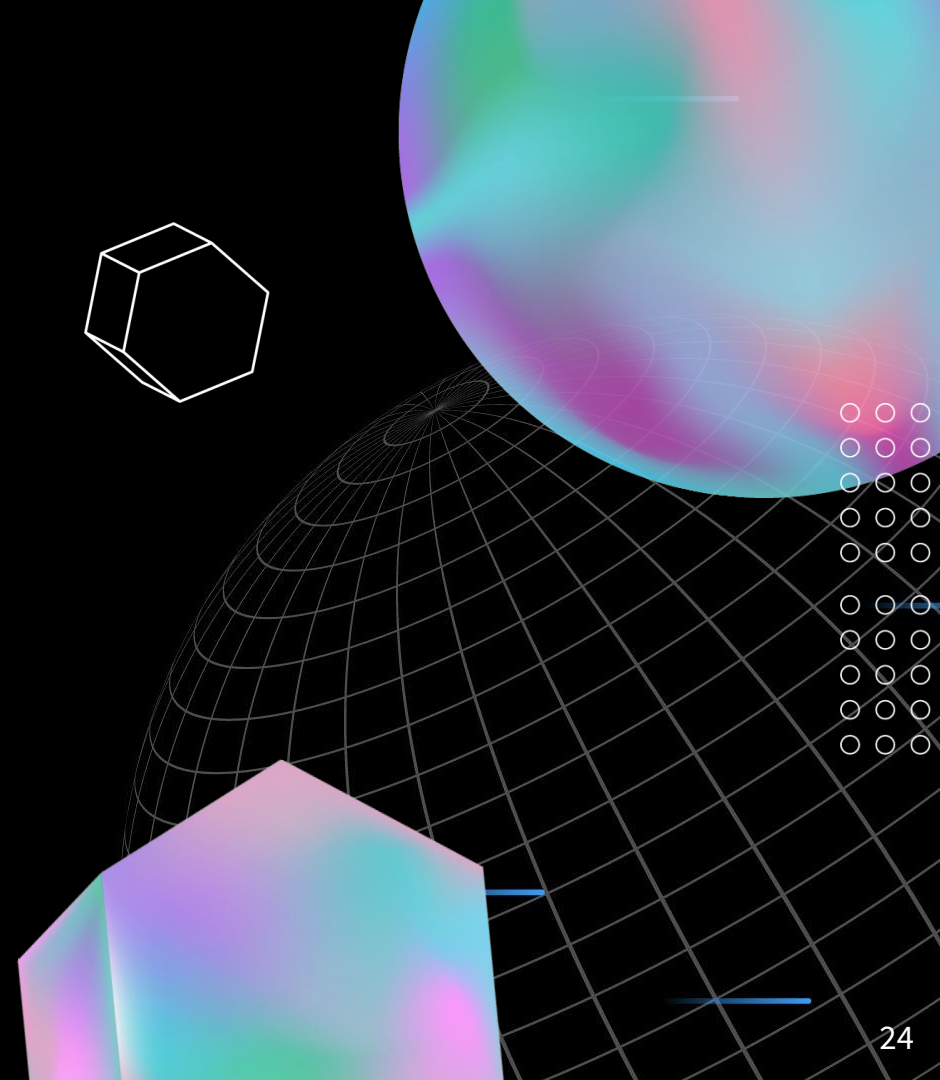
ex-7-fog.js





04

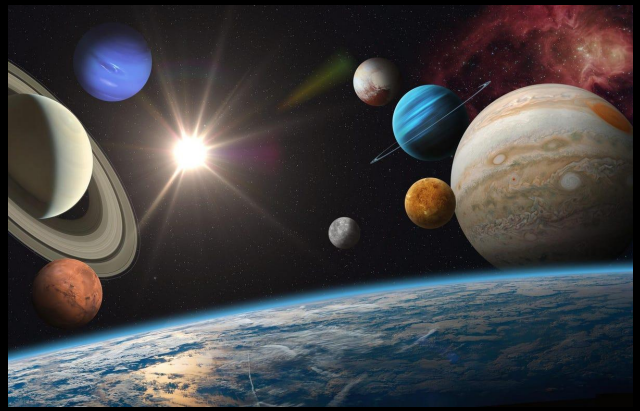
Extra



Background

We can add two types of backgrounds to our scene:

- Static
- 360 degrees



3D models

ex-3d-models.js

Even though we can create 3D objects using Three.js it's not the best way, there are dedicated programs for the purpose of making a complex 3d model like Blender.
We can import the model, which can either be a .obj or a .gltf object. The main difference is that the .gltf object has the texture information embedded.

```
import { OBJLoader } from 'three/addons/loaders/OBJLoader.js';

import { GLTFLoader } from 'three/addons/loaders/GLTFLoader.js';
```

Thanks!

Do you have any questions?

Julio Iván Carrasco Armas
alu0101110639@ull.edu.es

Francisco Marqués Armas
alu0101438412@ull.edu.es

References

Three js: https://threejs.org/manual

Introduction to three.js: https://riptutorial.com/three-js

Intro to Three.js with WebGL: https://davidlyons.dev/threejs-intro

Three.js cookbook: https://github.com/josdirksen/threejs-cookbook

Three.js tutorial by Suboptimal Engineer:
https://github.com/SuboptimalEng/three-js-tutorials