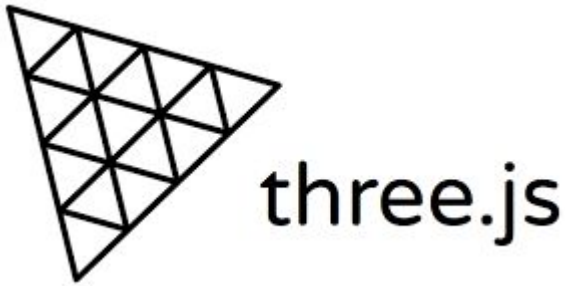
CSE4204
LAB-4 : Intro to Three.js

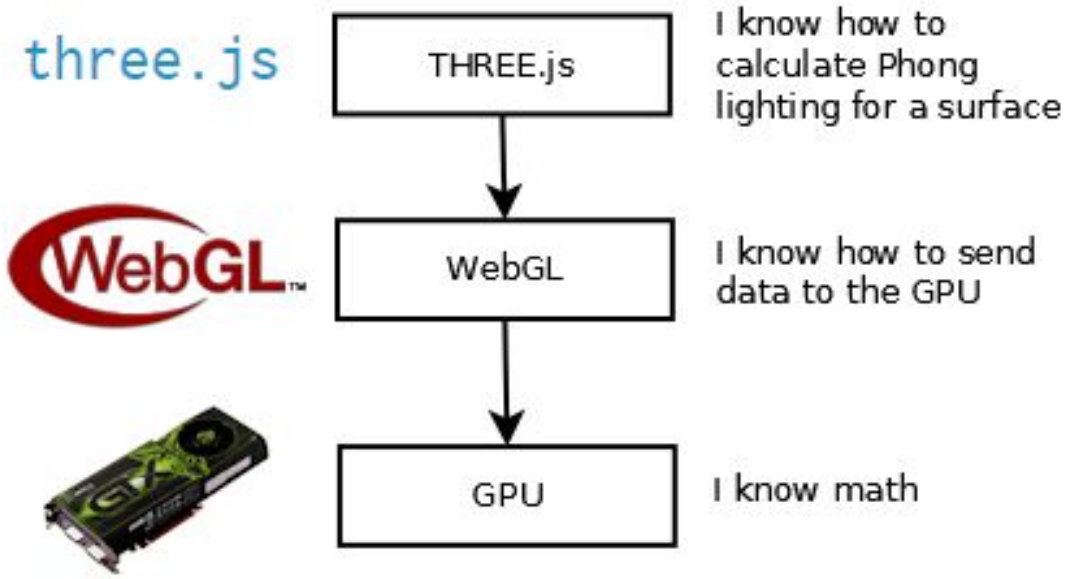
Download the materials at:
shorturl.at/admy9

What is three.js?

- A javascript library
- You can create and display 3D graphics on web browser
- Built on WebGL API
- Easy to use
- https://threejs.org

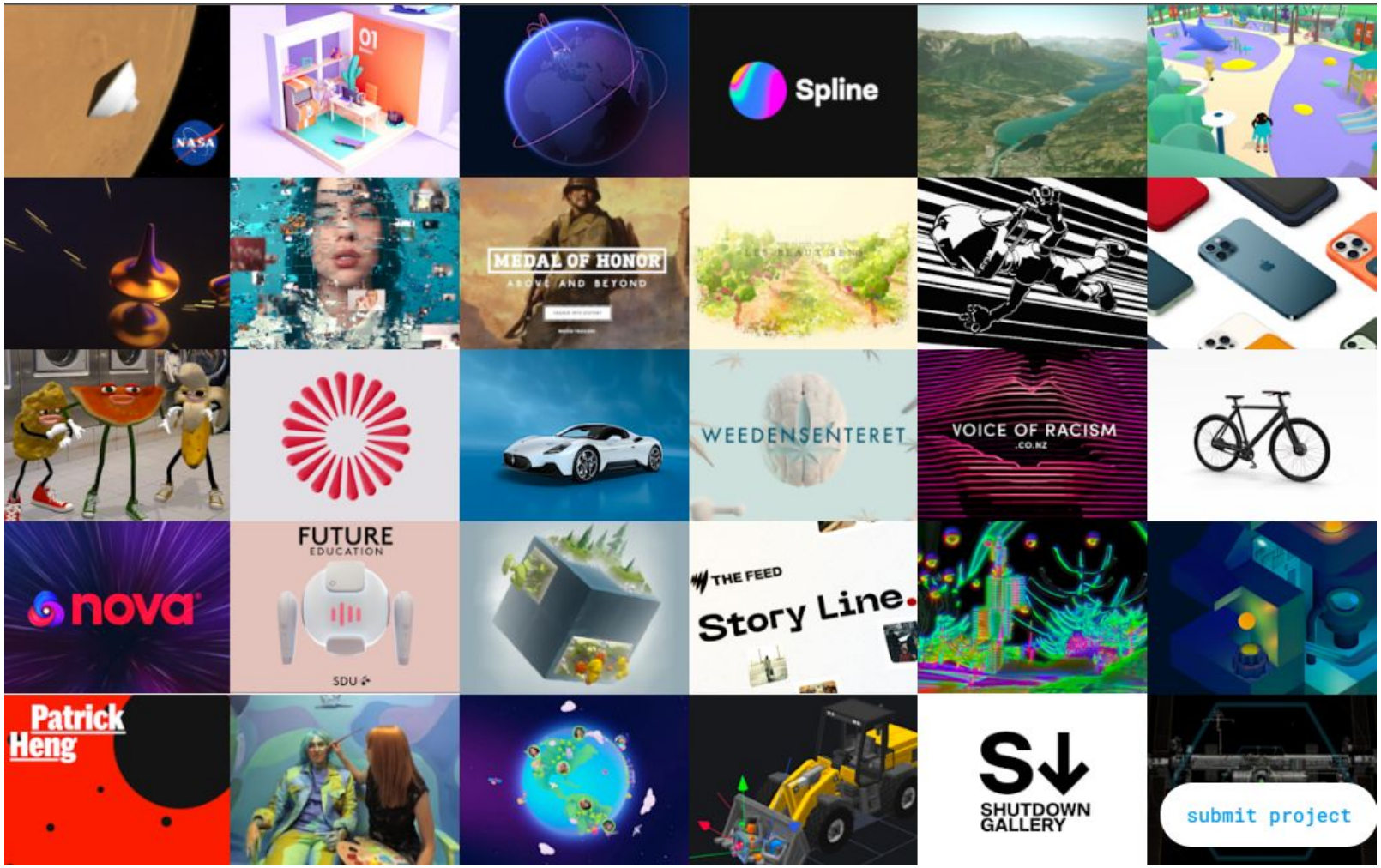


three.js



Source: https://cglearn.codelight.eu/pub/computer-graphics/computer-graphics

What you can do with three.js



Source: https://threejs.org/

Installation

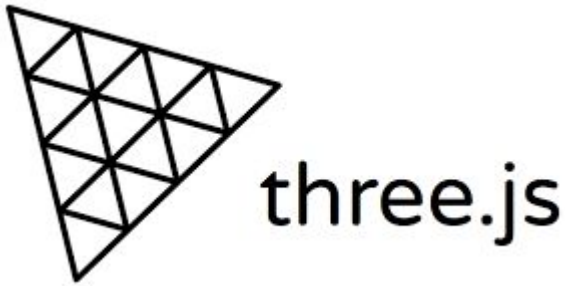
- three.js can be installed with npm
- or can be used with static hosting or a CDN
- npm is the most common approach

Installation with npm

- Download and install the latest version of Node.js
- Open a terminal window in your project folder and run:

```
npm install three
```

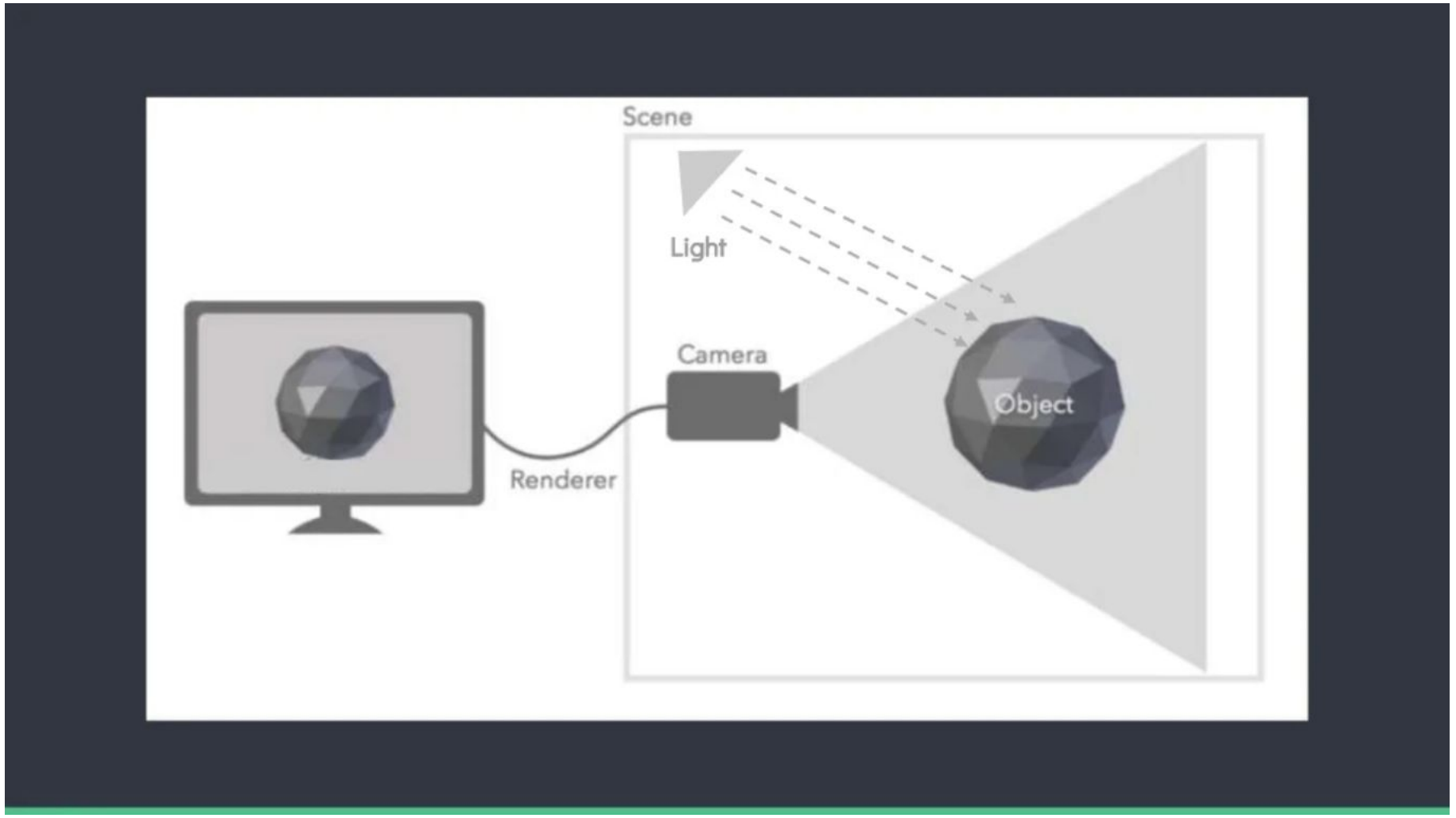
- Package will be downloaded and installed
- Full download will give so many additional files
- You can only use the Three.js source



Basic Setup: Adding the library file

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>My first three.js app</title>
    <style>
        body {
            margin: 0;
        }
    </style>
</head>

<body>
    <script src="js/three.js"></script>
    <script>


    </script>
</body>

</html>
```

Basic Elements of Three.js



Basic Elements: Renderer

- Renderer draws the scene
- Needs to be attached to an HTML element

I. Create a WebGLRenderer

```
const renderer = new THREE.WebGLRenderer();
renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);
```

II. Beautify it

```
renderer.setClearColor( 0xffffff, 1);
renderer.clear();
```

Output!

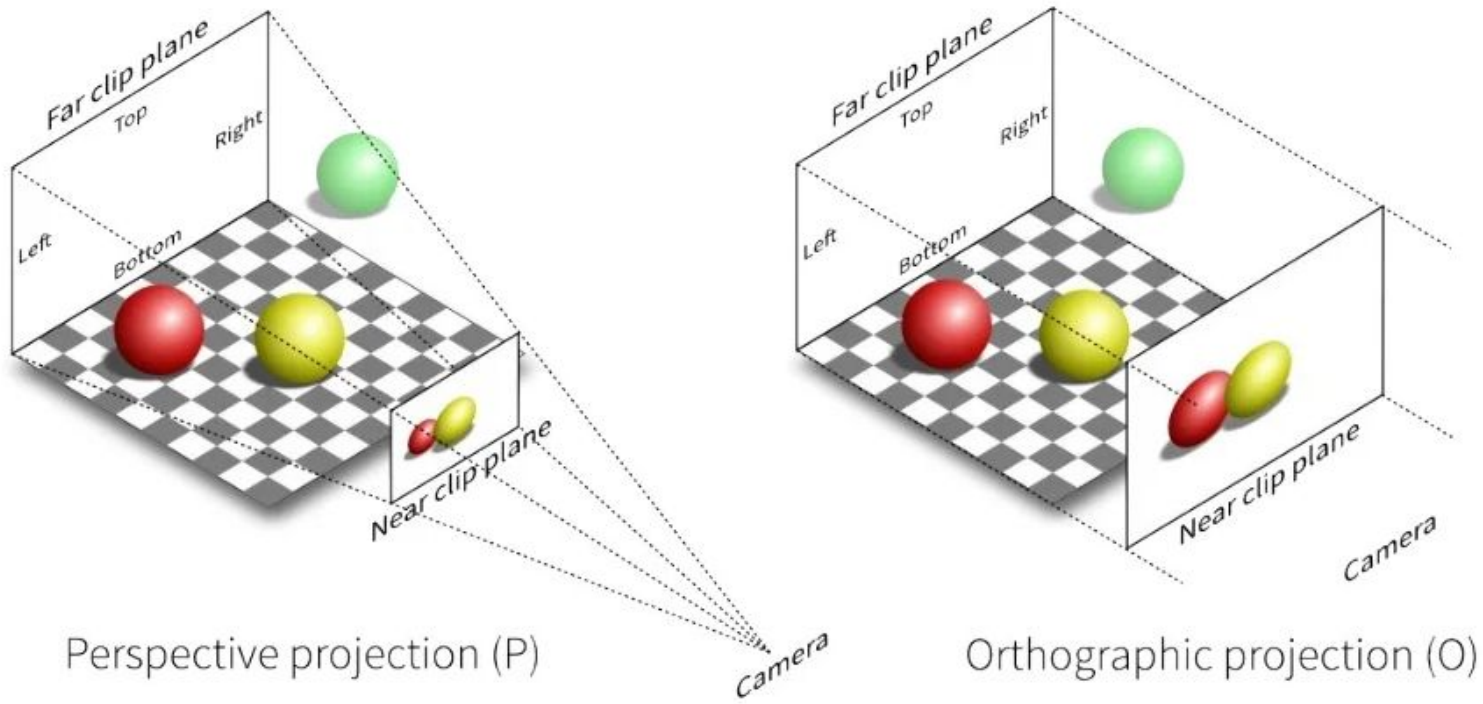
Basic Elements: Camera

- Camera is the view port to look at the objects in a scene
- Camera controls how your object will be seen
- 2 main types of camera: Orthographic and Perspective

III. Creating a Camera

```
//new new THREE.PerspectiveCamera(FOV, viewAspectRatio, zNear, zFar)
const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);
camera.position.z = 300;
```

Basic Elements: Camera



Source: https://www.slideshare.net/GarethMarland/introduction-tothreejs

Basic Elements: Scene

- Scene is where you put your modes, such as car, house or cubes

IV. Create a Scene

```
const scene = new THREE.Scene();
```

V. Render the scene from the camera

```
renderer.render(scene, camera);
```

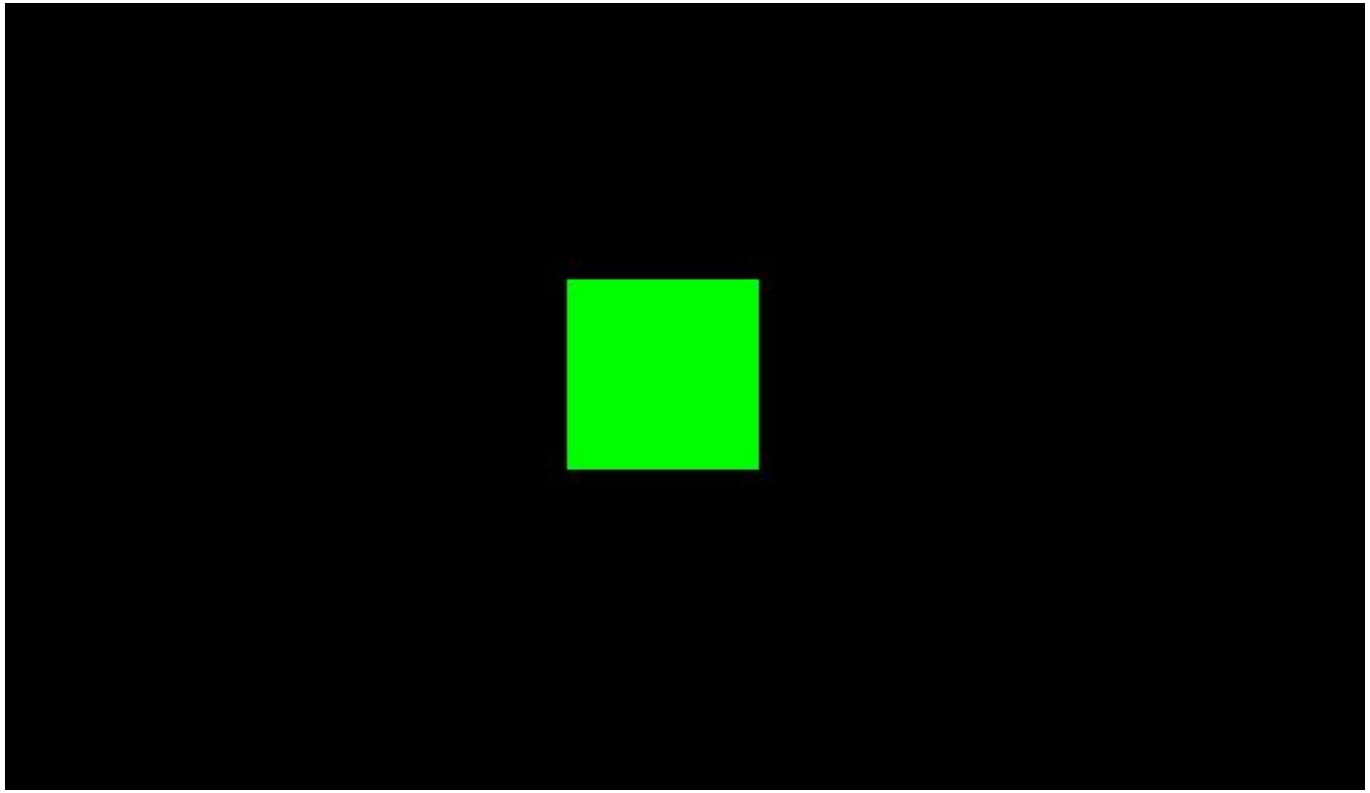
Basic Elements: Geometry (Objects)

- three.js provides some predefined geometry
- You can define your own vertices
- Use an existing model

VI. Create a mesh with geometry and add it to the scene

```
const geometry = new THREE.BoxGeometry(1, 1, 1);
const material = new THREE.MeshBasicMaterial({
    color: 0x00ff00
});
const cube = new THREE.Mesh(geometry, material);
scene.add(cube);
```

cube.html



Recap

- Create a Renderer

  new THREE.WebGLRenderer()
- Create a camera

  new THREE.PerspectiveCamera(FOV, viewAspectRatio, zNear, zFar)
- Create a Scene

  new THREE.Scene()
- Render the scene from the camera

  renderer(scene, camera)
- Create a mesh with geometry and add it to the scene

  const cube = new THREE.Mesh(geometry, material);
  scene.add(cube);

Lighting

- Light globally illuminates the objects
- three.js provides some lighting models
- Common lighting model: Ambient Light, Point Light, Directional Light

Lighting

Just create the light and add it to the scene

Ambient Light

```
const color = 0x00FFFF;
const intensity = 10;
const distance = 100;

const light = new THREE.AmbientLight(color, intensity);
```

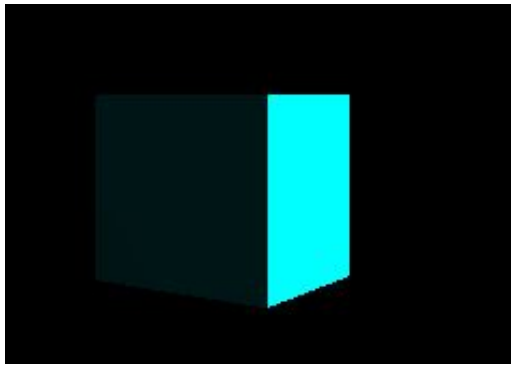
Point Light

```
const light = new THREE.PointLight(color, intensity, distance);
light.position.set(50, 50, 50);
```

Directional Light

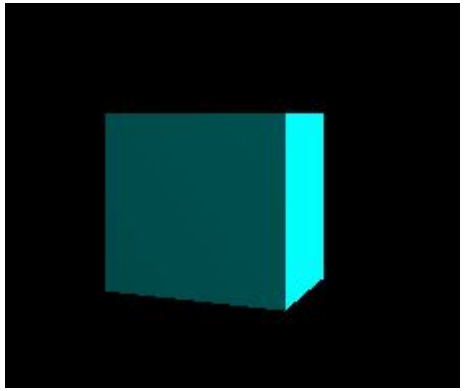
```
const light = new THREE.DirectionalLight(color, intensity, distance);
scene.add(light);
```

lighting.html



Ambient Light

Point Light



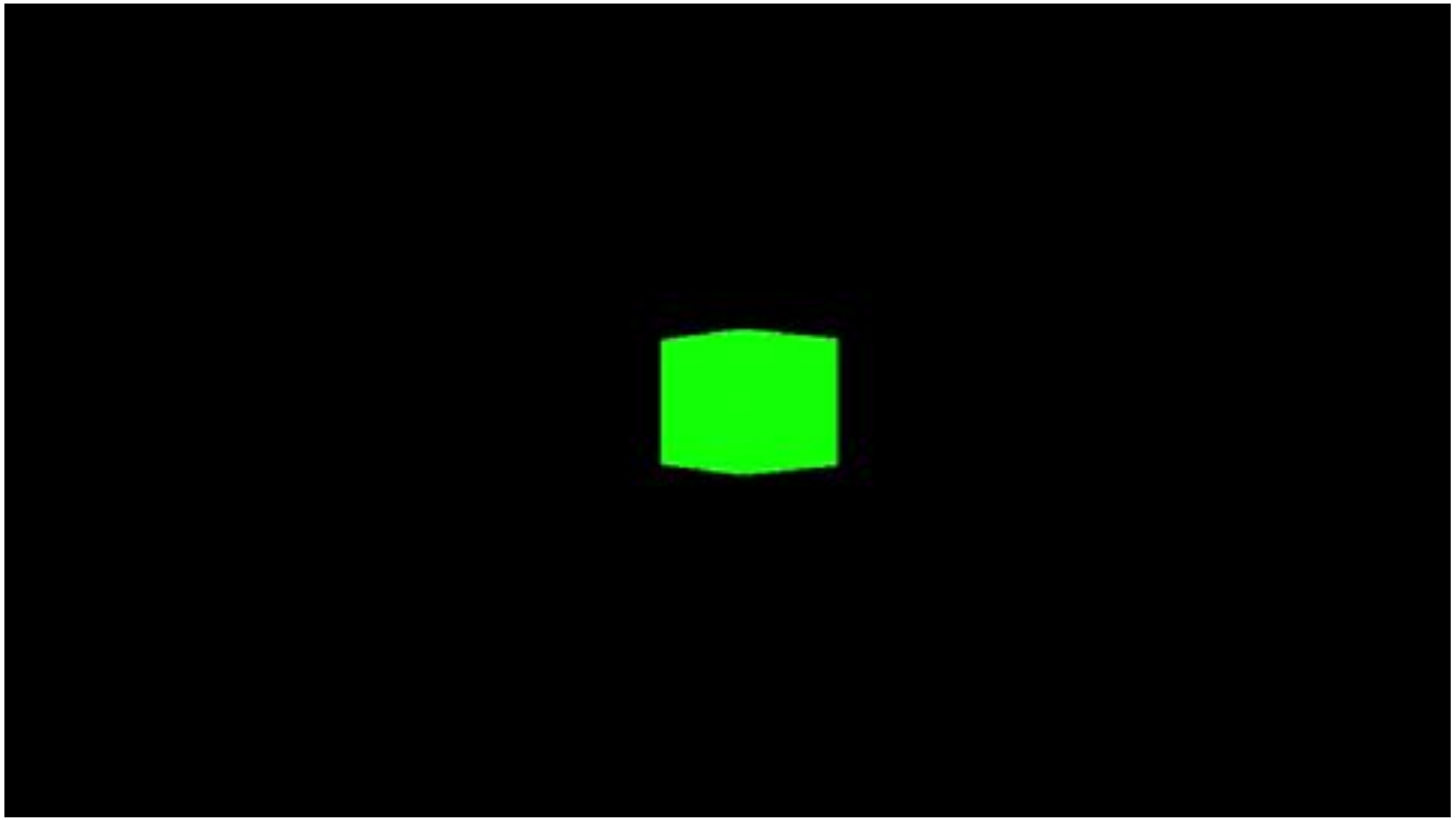
Directional Light

Animation

- requestAnimationFrame creates a loop
- It causes renderer to draw a new scene
- Generally refresh rate is 60FPS

```
function animate() {
    requestAnimationFrame(animate);

    cube.rotation.y += 0.01;

    renderer.render(scene, camera);
};

animate();
```

cubeRotation.html



User Interaction

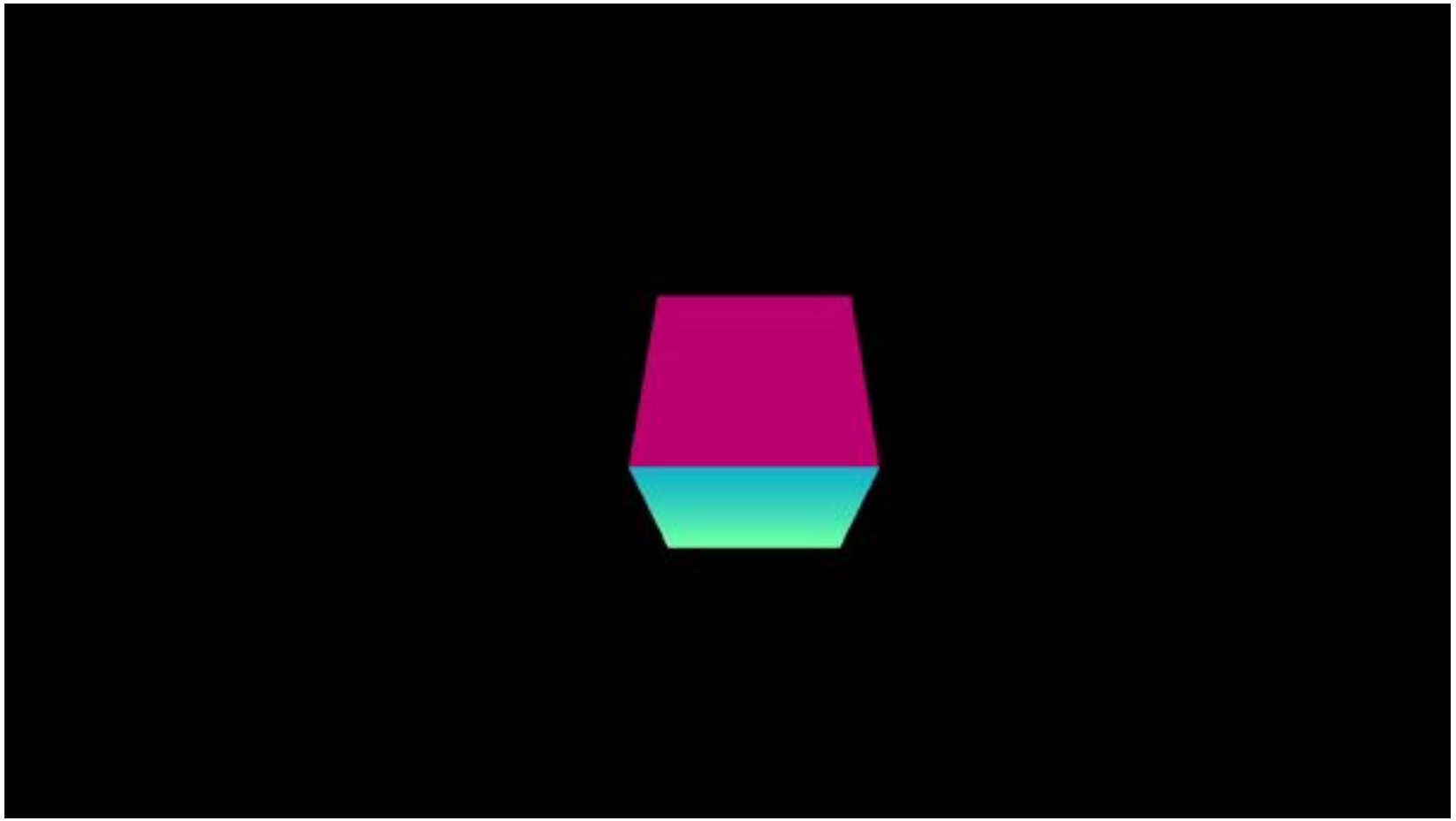
- User interaction can be done with mouse clicking or moving or pressing a key on the keyboard

Create the action method and add it to the event listener

```
let onmousedown = function(){
    cube.rotation.y += 5;
    renderer.render(scene, camera);
}
```

```
document.addEventListener("click", onmousedown, false);
```

mouseClick.html



Shaders

- A shader is a small program written in GLSL
- It runs on the GPU
- Two types of shader: Vertex Shader and Fragment Shader
- Vertex shader is applied on every vertex
- Fragment shader is applied on every fragment/pixel
- In three.js ShaderMaterial allows a material to be rendered with custom shaders

Shaders

Create a vertex shader as a script

```
<script id="vertexShader" type="x-shader/x-vertex">
    // projectionMatrix, modelViewMatrix, position -> passed in from Three.js
    varying vec3 v_color;
    void main() {
        gl_Position = projectionMatrix * modelViewMatrix * vec4(position, 1.0);
        v_color = position;
    }
</script>
```

Create a fragment shader as a script

```
<script id="fragmentShader" type="x-shader/x-fragment">

    uniform float u_time;
    varying vec3 v_color;
    void main() {

        gl_FragColor = vec4(abs(cos(v_color+u_time)), 1.0);
    }
```

Shaders

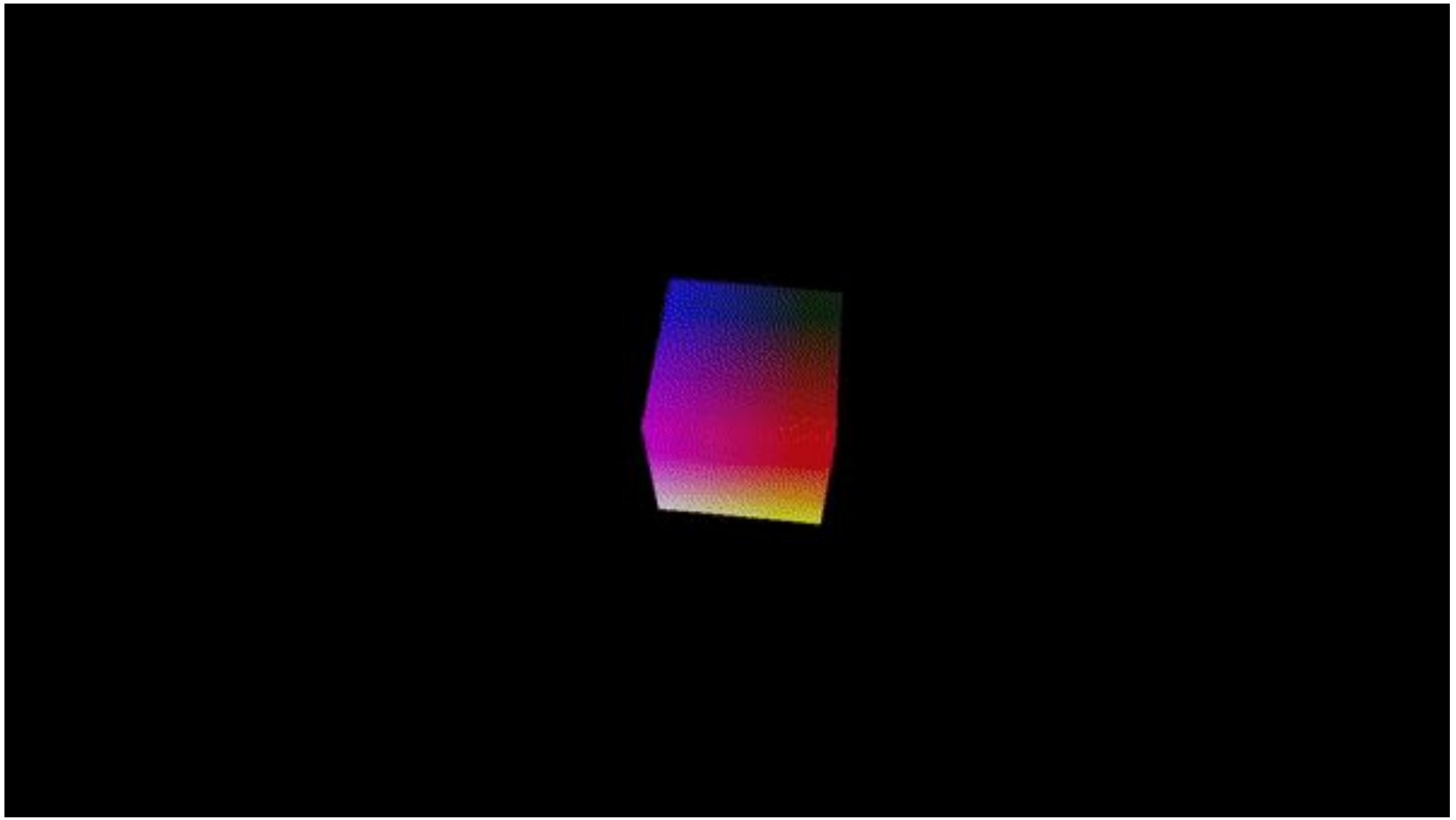
Define some uniform variables if required in dictionary

```
var start = Date.now();
uniforms = {
    u_time: { type: "f", value: (Date.now() - start)/1000 }
};
```

Create a ShaderMaterial and assign your shaders to it

```
material = new THREE.ShaderMaterial({
    uniforms: uniforms,
    vertexShader: document.getElementById('vertexShader').textContent,
    fragmentShader: document.getElementById('fragmentShader').textContent

});
```

shaderMaterial.html



Texture Mapping

- We can give realistic appearance to our objects by applying texture mapping
- A texture is an image that would wrap the object
- In order to apply texture mapping, we need to load the texture
- Texture is loaded as a texture object

Texture Mapping

Load the texture object and map it onto the object

```
//width', 'height', and 'depth'
const geometry = new THREE.BoxGeometry(2, 2, 2);
const texture = new THREE.TextureLoader().load('./images/crate.jpg');
const material = new THREE.MeshBasicMaterial({
    map: texture
});

const cube = new THREE.Mesh(geometry, material);

scene.add(cube);
```

texture.html