THREE.JS: ILUMINAÇÃO

Motores Gráficos

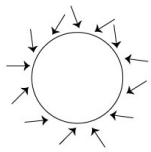
Prof. João Paulo Lima

Universidade Federal Rural de Pernambuco
Departamento de Computação

Luz Ambiental

- Luz presente na cena que não vem de fonte definida
- Incide com mesma intensidade e em todas as direções
- Refletida igualmente em todas as direções
- Three.js: `AmbientLight`

```
color = materialColor * light.color * light.intensity;
```



Luz Ambiental

- 1. Alterar arquivo index.html conforme a seguir

```html
<!DOCTYPE html>
<html>
    <head lang="en">
        <meta charset="utf-8">
        <title>My first three.js app</title>
        <link type="text/css" rel="stylesheet" href="main.css">
        <script async src="https://unpkg.com/es-module-shims@1.6.3/dist/es-module-shims.js"></script>

        <script type="importmap">
          {
            "imports": {
              "three": "https://unpkg.com/three@v0.152.2/build/three.module.js",
              "three/addons/": "https://unpkg.com/three@v0.152.2/examples/jsm/"
            }
          }
        </script>
    </head>
    <body>
        <canvas id="c"></canvas>
        <script type="module" src="/main.js"></script>
    </body>
</html>
```


Luz Ambiental

- 2. Alterar arquivo main.css conforme a seguir

```css
html, body {
  margin: 0;
  height: 100%;
}
#c {
  width: 100%;
  height: 100%;
  display: block;
}
```


Luz Ambiental

- 3. Alterar arquivo main.js conforme a seguir

```javascript
import * as THREE from 'three';

function main() {
  const canvas = document.querySelector('#c');
  const renderer = new THREE.WebGLRenderer({antialias: true, canvas});

  const fov = 45;
  const aspect = 2;
  const near = 0.1;
  const far = 100;
  const camera = new THREE.PerspectiveCamera(fov, aspect, near, far);
  camera.position.set(0, 10, 20);

  const scene = new THREE.Scene();
  scene.background = new THREE.Color('black');
```



```javascript
{
  const planeSize = 40;

  const loader = new THREE.TextureLoader();
  const texture =
loader.load('https://threejs.org/manual/examples/resources/images/checker.png');
  texture.wrapS = THREE.RepeatWrapping;
  texture.wrapT = THREE.RepeatWrapping;
  texture.magFilter = THREE.NearestFilter;
  const repeats = planeSize / 2;
  texture.repeat.set(repeats, repeats);

  const planeGeo = new THREE.PlaneGeometry(planeSize, planeSize);
  const planeMat = new THREE.MeshPhongMaterial({
    map: texture,
    side: THREE.DoubleSide,
  });
  const mesh = new THREE.Mesh(planeGeo, planeMat);
  mesh.rotation.x = Math.PI * -.5;
  scene.add(mesh);
}
{
  const cubeSize = 4;
  const cubeGeo = new THREE.BoxGeometry(cubeSize, cubeSize, cubeSize);
  const cubeMat = new THREE.MeshPhongMaterial({color: '#8AC'});
  const mesh = new THREE.Mesh(cubeGeo, cubeMat);
  mesh.position.set(cubeSize + 1, cubeSize / 2, 0);
  scene.add(mesh);
}
```

7

```
{
  const sphereRadius = 3;
  const sphereWidthDivisions = 32;
  const sphereHeightDivisions = 16;
  const sphereGeo = new THREE.SphereGeometry(sphereRadius, sphereWidthDivisions,
sphereHeightDivisions);
  const sphereMat = new THREE.MeshPhongMaterial({color: '#CA8'});
  const mesh = new THREE.Mesh(sphereGeo, sphereMat);
  mesh.position.set(-sphereRadius - 1, sphereRadius + 2, 0);
  scene.add(mesh);
}

{
  const color = 0xFFFFFF;
  const intensity = 1;
  const light = new THREE.AmbientLight(color, intensity);
  scene.add(light);
}

function resizeRendererToDisplaySize(renderer) {
  const canvas = renderer.domElement;
  const width = canvas.clientWidth;
  const height = canvas.clientHeight;
  const needResize = canvas.width !== width || canvas.height !== height;
  if (needResize) {
    renderer.setSize(width, height, false);
  }
  return needResize;
}
```

8

Luz Ambiental

```
  function render() {

    if (resizeRendererToDisplaySize(renderer)) {
      const canvas = renderer.domElement;
      camera.aspect = canvas.clientWidth / canvas.clientHeight;
      camera.updateProjectionMatrix();
    }

    renderer.render(scene, camera);

    requestAnimationFrame(render);
  }

  requestAnimationFrame(render);
}

main();
```

9

Luz Hemisfério

- Cor do céu + cor do chão
- Multiplica cor do material por uma dessas duas cores
  - Cor do céu: se superfície do objeto aponta para cima
  - Cor do chão: se superfície do objeto aponta para baixo
- Three.js: `HemisphereLight`

10

Luz Hemisfério

- Em main.js, trocar código da esquerda pelo da direita

```
const color = 0xFFFFFF;
```

→

```
const skyColor = 0xB1E1FF;
const groundColor = 0xB97A20;
```

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

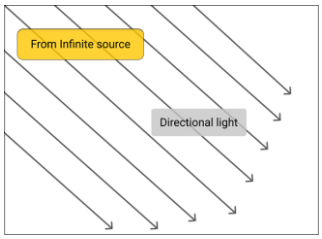
→

```
const light = new
THREE.HemisphereLight(skyColor,
groundColor, intensity);
```

11

Luz Direcional

- Comumente usada para representar o sol
- Three.js: `DirectionalLight`



12

Luz Direcional

- Em main.js, trocar código da esquerda pelo da direita

```
const skyColor = 0xB1E1FF;
const groundColor = 0xB97A20;
const intensity = 1;
const light = new
THREE.HemisphereLight(skyColor,
groundColor, intensity);
scene.add(light);
```

→

```
const color = 0xFFFFFF;
const intensity = 1;
const light = new
THREE.DirectionalLight(color, intensity);
light.position.set(0, 10, 0);
light.target.position.set(-5, 0, 0);
scene.add(light);
scene.add(light.target);
```

Luz Direcional

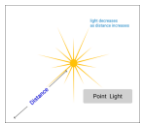
- Em main.js, adicionar código de baixo logo depois do código de cima

```
const color = 0xFFFFFF;
const intensity = 1;
const light = new
THREE.DirectionalLight(color, intensity);
light.position.set(0, 10, 0);
light.target.position.set(-5, 0, 0);
scene.add(light);
scene.add(light.target);
```

```
const helper = new THREE.DirectionalLightHelper(light);
scene.add(helper);
```


Luz Pontual

- Localizada em um ponto
- Emite luz em todas as direções a partir deste ponto
- Three.js: `PointLight`
  - `distance`: alcance máximo da luz, padrão é zero (sem limite)



Luz Pontual

- Em main.js, trocar código da esquerda pelo da direita

```
const light = new
THREE.DirectionalLight
(color, intensity);
```
→
```
const light = new
THREE.PointLight(color, intensity);
```

```
const helper = new
THREE.DirectionalLightHelper
(light);
```
→
```
const helper = new
THREE.PointLightHelper(light);
```

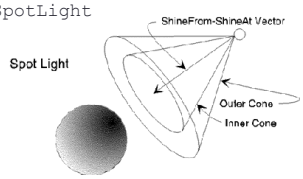
- Em main.js, remover as linhas abaixo

```
light.target.position.set(-5, 0, 0);
```

```
scene.add(light.target);
```

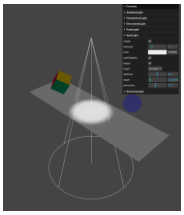

Holofote

- Luz pontual com cone
- Luz só ilumina dentro do cone
  - 2 cones: interno e externo
  - Cone interno: intensidade máxima
  - Entre cone interno e externo, vai de intensidade máxima para zero
- Three.js: `SpotLight`



Holofote

- Three.js: `SpotLight`
  - `angle`: ângulo do cone externo, padrão é `Math.PI/3`
  - `penumbra`: percentual do cone externo não ocupado pelo interno, padrão é zero (cone interno igual a cone externo, sem penumbra)



Holofote

- Em main.js, trocar código da esquerda pelo da direita

```
const light = new
THREE.PointLight(color, intensity);
light.position.set(0, 10, 0);
scene.add(light);
```

```
const light = new THREE.SpotLight(color, intensity);
light.position.set(0, 10, 0);
light.target.position.set(-5, 0, 0);
scene.add(light);
scene.add(light.target);
```

```
const helper = new
THREE.PointLightHelper(light);
```
→
```
const helper = new
THREE.SpotLightHelper(light);
```

Luz com Área

- Área retangular de luz (ex: lâmpada fluorescente)
- Só funciona com `MeshStandardMaterial` e `MeshPhysicalMaterial`
- Three.js: `RectAreaLight`

Luz com Área

- Em main.js, trocar código da esquerda pelo da direita

```
import * as THREE from 'three';
```

→

```
import * as THREE from 'three';
import {RectAreaLightUniformsLib} from
'three/addons/lights/RectAreaLightUniformsLib.js';
import {RectAreaLightHelper} from
'three/addons/helpers/RectAreaLightHelper.js';
```

```
const renderer = new
THREE.WebGLRenderer({antialias:
true, canvas});
```

→

```
const renderer = new
THREE.WebGLRenderer({antialias: true, canvas});
RectAreaLightUniformsLib.init();
```

```
const planeMat = new
THREE.MeshPhongMaterial({
```

→

```
const planeMat = new
THREE.MeshStandardMaterial({
```

```
const cubeMat = new
THREE.MeshPhongMaterial({color: '#8AC'});
```

→

```
const cubeMat = new
THREE.MeshStandardMaterial({color: '#8AC'});
```

```
const sphereMat = new
THREE.MeshPhongMaterial({color: '#CA8'});
```

→

```
const sphereMat = new
THREE.MeshStandardMaterial({color: '#CA8'});
```


Luz com Área

- Em main.js, trocar código da esquerda pelo da direita

```
const color = 0xFFFFFF;
const intensity = 1;
const light = new
THREE.SpotLight(color, intensity);
light.position.set(0, 10, 0);
light.target.position.set(-5, 0, 0);
scene.add(light);
scene.add(light.target);

const helper = new
THREE.SpotLightHelper(light);
scene.add(helper);
```

→

```
const color = 0xFFFFFF;
const intensity = 5;
const width = 12;
const height = 4;
const light = new
THREE.RectAreaLight(color,
intensity, width, height);
light.position.set(0, 10, 0);
light.rotation.x =
THREE.MathUtils.degToRad(-90);
scene.add(light);

const helper = new
RectAreaLightHelper(light);
light.add(helper);
```


Sombras

- Three.js usa mapeamento de sombra por padrão
- 3 tipos de luz podem causar sombras:
  - `DirectionalLight`
  - `PointLight`
  - `SpotLight`

Sombras

- Em main.js, trocar código da esquerda pelo da direita

```
const renderer = new
THREE.WebGLRenderer({antialias:
true, canvas});
```

→

```
const renderer = new
THREE.WebGLRenderer({antialias: true, canvas});
renderer.shadowMap.enabled = true;
```

```
camera.position.set(0, 10, 20);
```

→

```
camera.position.set(0, 10, 40);
```

```
const mesh = new
THREE.Mesh(planeGeo, planeMat);
```

→

```
const mesh = new THREE.Mesh(planeGeo, planeMat);
mesh.receiveShadow = true;
```

```
const mesh = new
THREE.Mesh(cubeGeo, cubeMat);
```

→

```
const mesh = new THREE.Mesh(cubeGeo, cubeMat);
mesh.castShadow = true;
mesh.receiveShadow = true;
```

```
const mesh = new
THREE.Mesh(sphereGeo, sphereMat);
```

→

```
const mesh = new THREE.Mesh(sphereGeo, sphereMat);
mesh.castShadow = true;
mesh.receiveShadow = true;
```

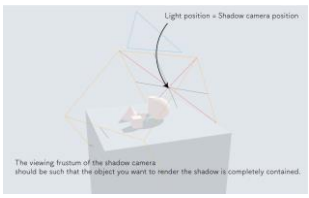
```
const light = new
THREE.DirectionalLight(color, intensity);
```

→

```
const light = new
THREE.DirectionalLight(color, intensity);
light.castShadow = true;
```


Sombras

- Mapas de sombra são criados renderizando a cena do ponto de vista da luz
- Câmera de sombra da luz define área dentro da qual sombras são renderizadas



Sombras

- Em main.js, adicionar código a seguir após a linha `scene.add(light.target);`

```
const cameraHelper = new THREE.CameraHelper(light.shadow.camera);
scene.add(cameraHelper);
```


Sombras

- Em main.js, trocar código da esquerda pelo da direita

```
const light = new
THREE.DirectionalLight
(color, intensity);
```
→
```
const light = new
THREE.SpotLight(color, intensity);
```

```
const helper = new
THREE.DirectionalLightHelper(light);
```
→
```
const helper = new
THREE.SpotLightHelper(light);
```


Sombras

- Em main.js, trocar código da esquerda pelo da direita

```
const light = new
THREE.SpotLight(color, intensity);
light.castShadow = true;
light.position.set(0, 10, 0);
light.target.position.set(-5, 0, 0);
scene.add(light);
scene.add(light.target);
```
→
```
const light = new
THREE.PointLight(color, intensity);
light.castShadow = true;
light.position.set(0, 10, 0);
scene.add(light);
```

```
const helper = new
THREE.SpotLightHelper(light);
scene.add(helper);
```
→
```
const helper = new
THREE.PointLightHelper(light);
scene.add(helper);
```


Exercício

- Criar cena de discoteca
- Fontes de luz posicionais girando ao redor de objetos
- Mudar cor de cada fonte de luz a cada 1 segundo
- Dicas:
  - `Math.random()` retorna número aleatório entre 0 e 1
  - `THREE.Clock()` tem método `getDelta()` que retorna tempo em segundos após última vez que método foi chamado