

WebGL e Three.js

DANILO LESSA

VICTOR SAGAWA

01

WebGL

- PRINCIPAIS CARACTERÍSTICAS
- HISTÓRIA DO WEBGL
- ÁREA DE ATUAÇÃO
- PRINCÍPIOS BÁSICOS DE UTILIZAÇÃO





Principais Características do WebGL

API JAVASCRIPT

Renderização de objetos 2D e 3D com alta performance

CLIENT SIDE RENDER

Reduz a carga de trabalho em cima do servidor

RENDERIZAÇÃO NA GPU

Processamento de imagens muito mais rápido

Cronologia

2006 - PRIMEIRO PROTÓTIPO

A ideia surgiu de experimentos realizados por Vladimir Vukićević, funcionário da Mozilla, criando protótipos de Canvas 3D

2009 - CRIAÇÃO DO WEBGL

O consórcio Khronos Group foi o responsável pela criação do projeto

2011 - LANÇAMENTO OFICIAL DA VERSÃO 1.0

Gigantes da tecnologia, como a Apple, Mozilla e Google estiveram por trás do seu desenvolvimento

2013 - DESENVOLVIMENTO DA VERSÃO 2.0

As primeiras implementações foram realizadas no Firefox 51, Chrome 56 and Opera 43

2017 - LANÇAMENTO OFICIAL DA VERSÃO 2.0

No início de 2022, a Khronos Group anunciou que o WebGL já é compatível com a maioria dos navegadores no mercado.

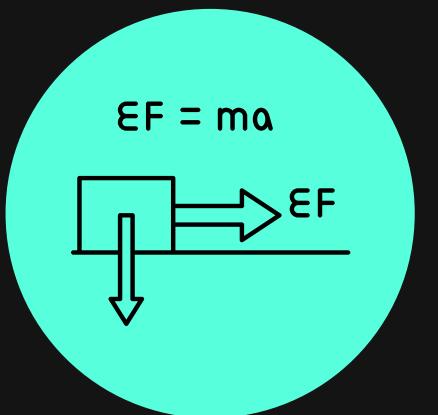
Áreas de atuação



Indústria de Games



Visualização de dados



Simulação física



Web design

Exemplos

- **PISCINA VIRTUAL COM WEBGL**

<https://madebyevan.com/webgl-water/>

- **WEBGL E REALIDADE VIRTUAL**

<https://tonite.dance/>

- **TRAILER INTERATIVO DE UMA PEÇA DE ÓPERA COM TEMÁTICA DE TERROR UTILIZANDO WEBGL**

<https://www.operanorth.co.uk/turn-of-the-screw-immersive-trailer/>

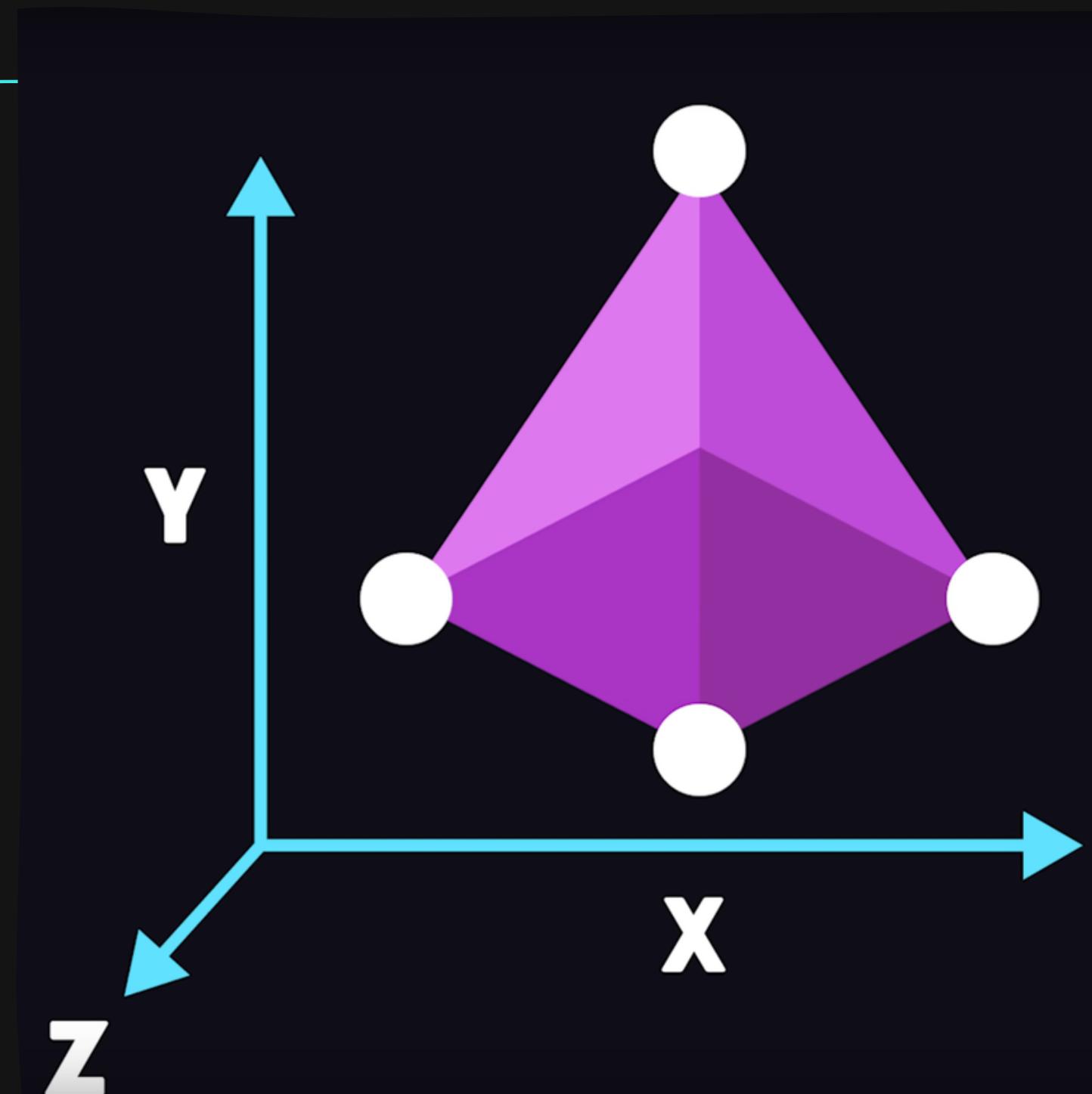


Princípios básicos de utilização

- TEORIA BÁSICA DE OBJETOS 3D
- PROCESSO DE RENDERIZAÇÃO
- SHADERS



Teoria básica de objetos 3D



VÉRTICES

Em um objeto 3D, cada ponto é um Vértice identificado pelas suas coordenadas X, Y e Z

Processo de renderização

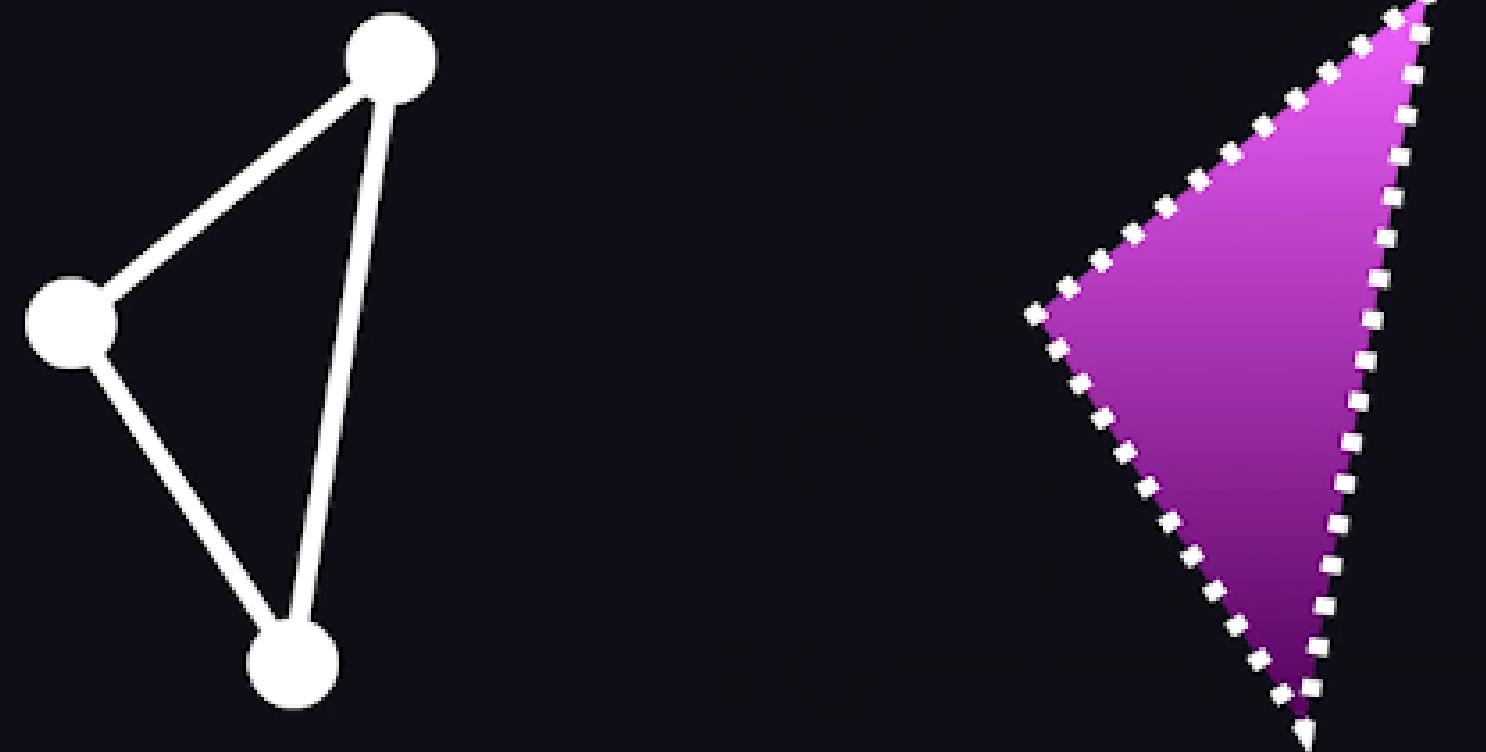


VERTICES → **PRIMITIVES**

VÉRTICES

Esses vértices são então conectados a outros vértices, criando pequenos triângulos conhecidos como Primitivos.

Processo de renderização



PRIMITIVES → **FRAGMENTS**

PRIMITIVOS

O processo de renderização no WebGL agrupa esses Primitivos e simula uma fonte de luz que atinge esse Fragmento de diversos angulos, criando a impressão de sombra e profundidade.

Processo de renderização



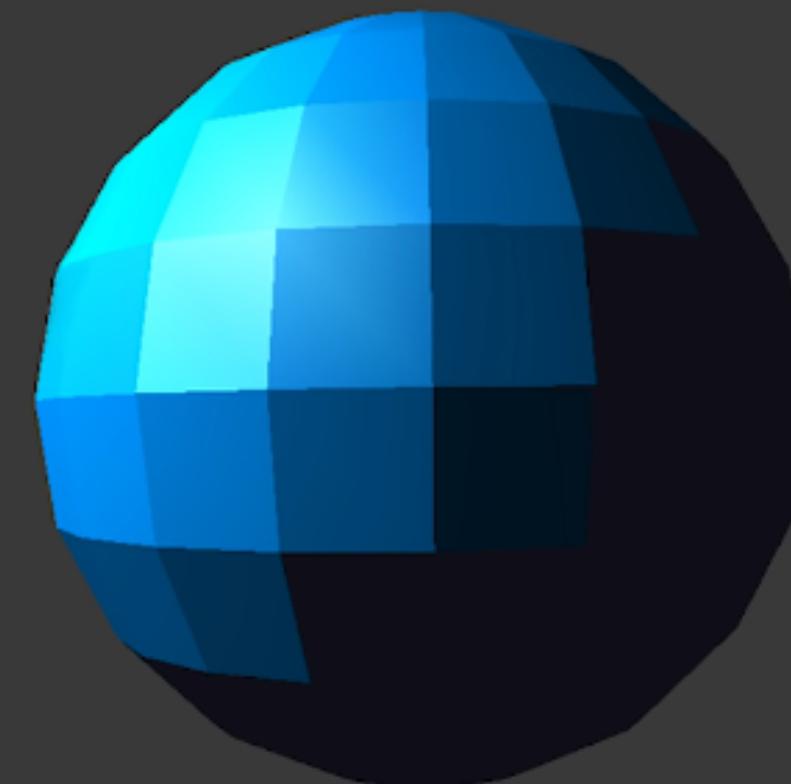
FRAGMENTS → **PIXELS**

PIXELS

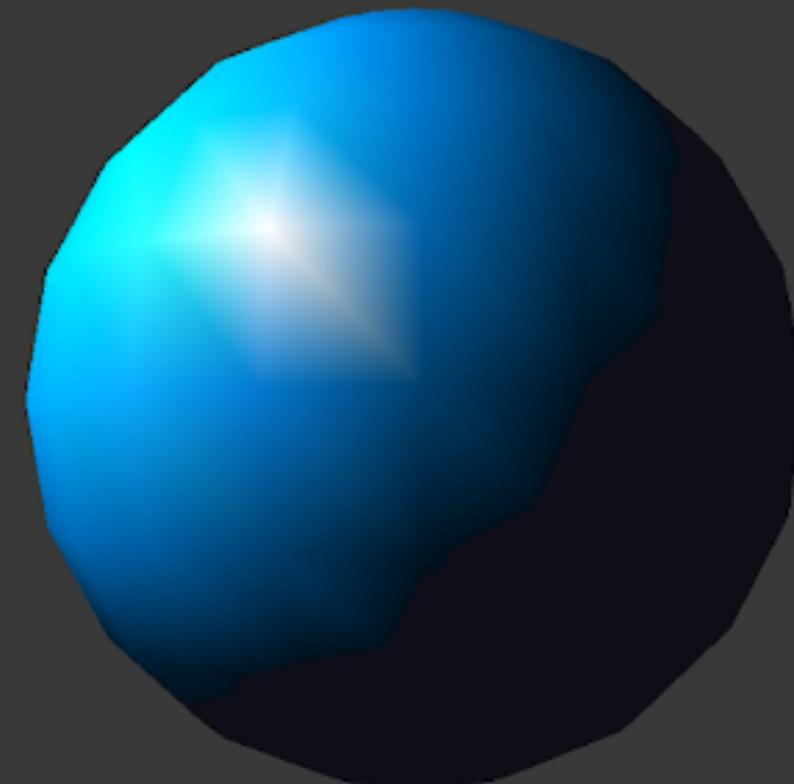
Esses Fragmentos, ou imagens vetoriais, passam então por um processo chamado **Rasterização**, que converte essas imagens tridimensionais em uma projeção 2D pixelada.

Shaders

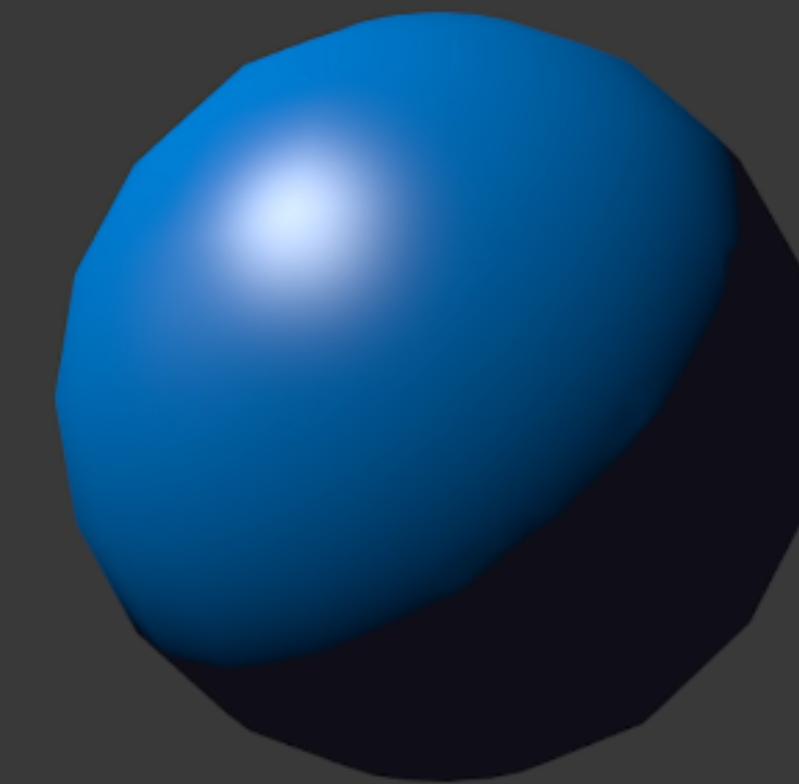
Shaders são basicamente funções que descrevem para a máquina como desenhar esses pixels rasterizados na tela.



FLAT SHADING



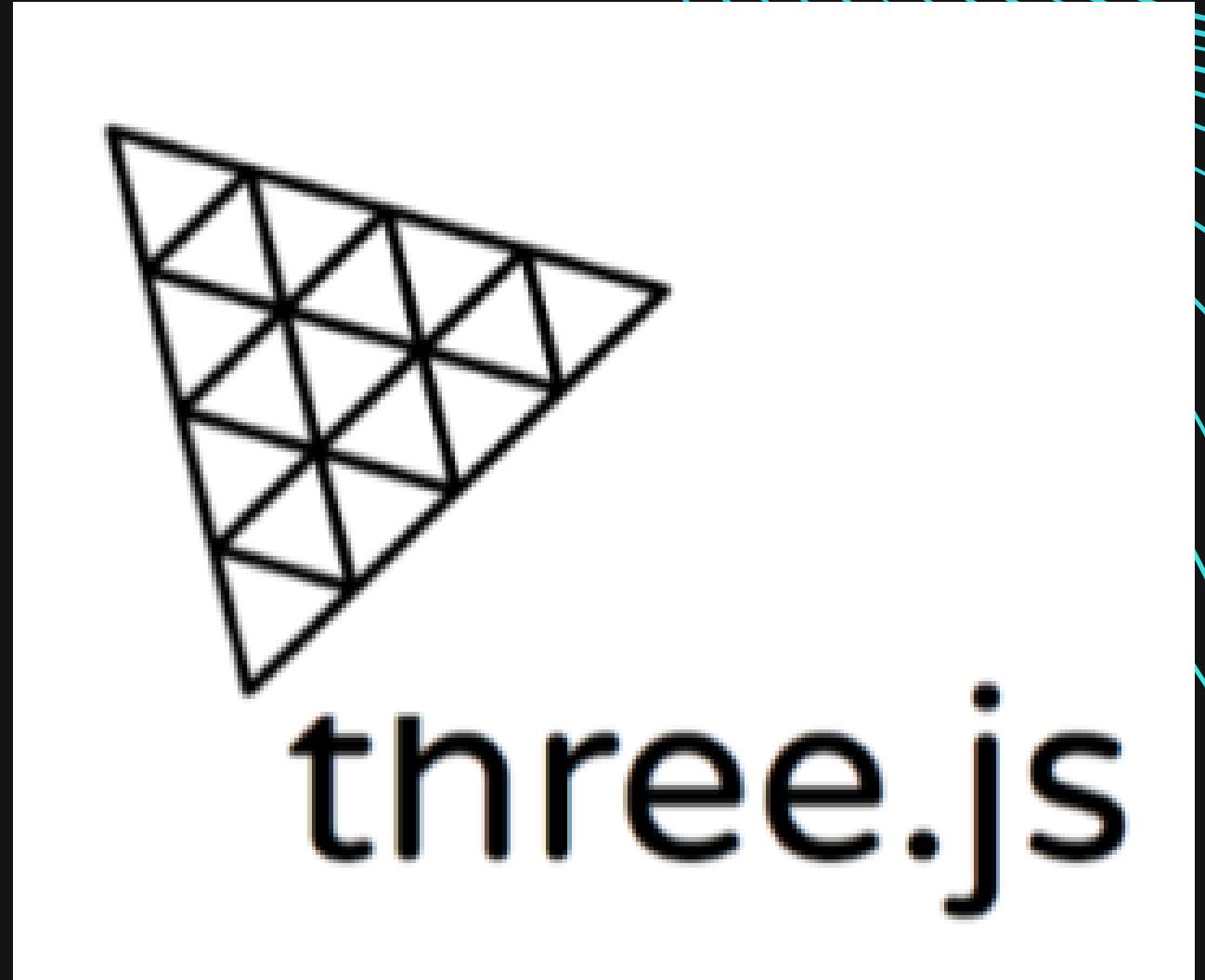
GOURAUD SHADING

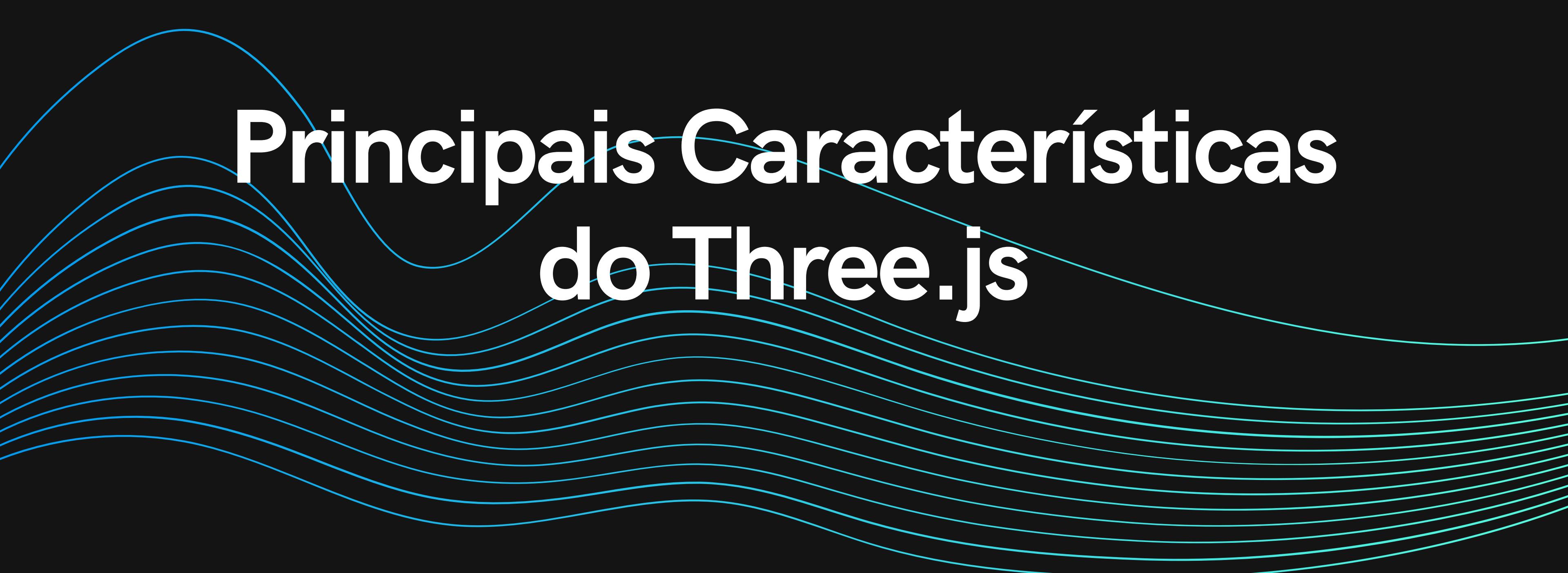


PHONG SHADING

Three.js

- PRINCIPAIS CARACTERÍSTICAS
- HISTÓRIA
- COMPARAÇÃO
- INSTALAÇÃO/INICIALIZAÇÃO





Principais Características do Three.js

FRAMEWORK WEBGL

Muito mais facilidade e dinamismo com WebGL

FÁCIL INSTALAÇÃO

A facilidade de instalação da ferramenta; código mais limpo

APRENDIZADO

Simples utilização e aprendizado

História

- **GITHUB(2010)**

<https://github.com/mrdoob/three.js/>

- **MR. DOOB(RICARDO CABELLO)**

<https://github.com/mrdoob>



Three.js X WebGL

```
const positions = [  
    // Front face  
    -1.0, -1.0, 1.0,  
    1.0, -1.0, 1.0,  
    1.0, 1.0, 1.0,  
    -1.0, 1.0, 1.0,  
  
    // Back face  
    -1.0, -1.0, -1.0,  
    -1.0, 1.0, -1.0,  
    1.0, 1.0, -1.0,  
    1.0, -1.0, -1.0,  
  
    // Top face  
    -1.0, 1.0, -1.0,  
    -1.0, 1.0, 1.0,  
    1.0, 1.0, 1.0,  
    1.0, 1.0, -1.0,  
  
    // Bottom face  
    -1.0, -1.0, -1.0,  
    1.0, -1.0, -1.0,  
    1.0, -1.0, 1.0,  
    -1.0, -1.0, 1.0,  
  
    // Right face  
    1.0, -1.0, -1.0,  
    1.0, 1.0, -1.0,  
    1.0, 1.0, 1.0,  
    1.0, -1.0, 1.0,  
  
    // Left face  
    -1.0, -1.0, -1.0,  
    -1.0, -1.0, 1.0,  
    -1.0, 1.0, 1.0,  
    -1.0, 1.0, -1.0,  
];
```

```
const faceColors = [  
    [1.0, 1.0, 1.0, 1.0], // Front face: white  
    [1.0, 0.0, 0.0, 1.0], // Back face: red  
    [0.0, 1.0, 0.0, 1.0], // Top face: green  
    [0.0, 0.0, 1.0, 1.0], // Bottom face: blue  
    [1.0, 1.0, 0.0, 1.0], // Right face: yellow  
    [1.0, 0.0, 1.0, 1.0], // Left face: purple  
];  
  
// Convert the array of colors into a table for all the vertices.  
  
var colors = [];  
  
for (var j = 0; j < faceColors.length; ++j) {  
    const c = faceColors[j];  
    // Repeat each color four times for the four vertices of the face  
    colors = colors.concat(c, c, c, c);  
}  
  
const colorBuffer = gl.createBuffer();  
gl.bindBuffer(gl.ARRAY_BUFFER, colorBuffer);  
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(colors), gl.STATIC_DRAW);
```

Three.js X WebGL

```
var geometry = new THREE.BoxGeometry( 0.5, 1, 1 );
var material = new THREE.MeshBasicMaterial( { color: 0x0000FF } );
var cube = new THREE.Mesh( geometry, material );
scene.add( cube );
```

Instalação

- NPM
- CDN(STATIC HOST)

```
npm install three
```

```
<script src="https://threejs.org/build/three.js"></script>
```

```
<script type="module">  
  import * as THREE from 'three';  
  const scene = new THREE.Scene();  
</script>
```

Scene, Camera, Render

Para realmente ser capaz de exibir algum conteúdo com o three.js, nós precisamos de três coisas:

```
const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera( 75, window.innerWidth /
window.innerHeight, 0.1, 1000 );

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

Cubo

"Tudo bem, mas onde está aquele cubo que você prometeu?". Vamos adicioná-lo agora.

```
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 );

camera.position.z = 5;
```

Renderizando

Hora do show!

```
function animate() {  
    requestAnimationFrame( animate );  
    renderer.render( scene, camera );  
}  
animate();
```

Animando

```
cube.rotation.x += 0.01;  
cube.rotation.y += 0.01;
```

Resultado

<https://jsfiddle.net/fxurzeb4/>

Exemplos

- [EXEMPLOS THREE.JS](#)
- [EXPERIMENTO ÁUDIO-VISUAL](#)
- [PORTFOLIO BRUNO SIMON](#)

Obrigado!

*graphic design is
my passion.*

