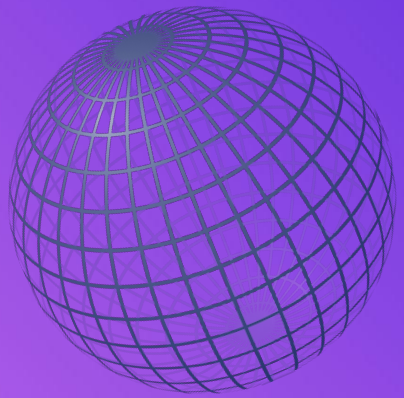
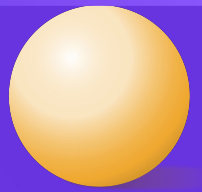


$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) \mathrm{d}\vec{w}'$$

3D Graphics with Three.js

A picture is worth a thousand words

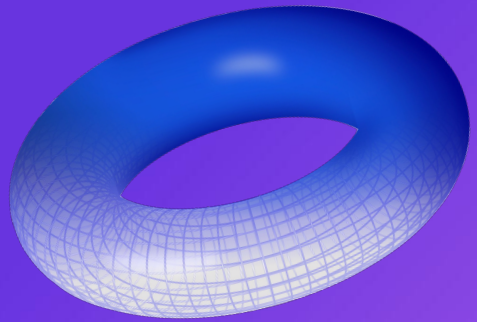
By Lionel Robert

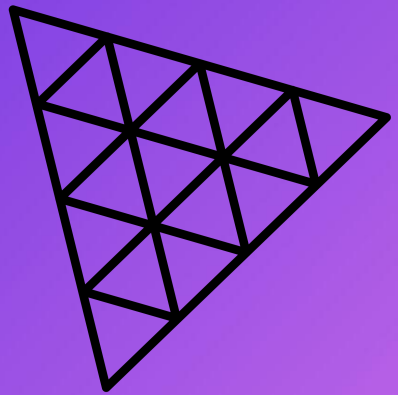


What is Three.js?

Ever seen a website that has beautiful 3D graphics moving around and wondered how that was done? You might have assumed it was too difficult to pull off with vanilla JavaScript or take some advanced work. This is where Three.js comes in.

1. Three.js is a JavaScript library that is available on all browsers that is used to create 3D computer graphics on the web browser
2. It requires very little work to get started and assumes absolutely no foreknowledge of 3D graphics and the complex mathematics that powers it
3. It does this by using WebGL - a low-level graphics API that was created for the web and works in all browsers





Using Three.js

Creating a scene

- A scene is created using the SCENE object along with a CAMERA and finally a renderer
- Without these, nothing would be rendered! And the code for them is pretty simple.

main.js

main.js > normalTexture

You, 2 minutes ago | 1 author (You)

```
1  import "./style.css";
2  import * as THREE from "three";  608.3k (gzipped: 152.7k)
3  import { OrbitControls } from "three/examples/jsm/controls/OrbitControls";
4
5  const scene = new THREE.Scene();
6  const camera = new THREE.PerspectiveCamera(
7      75,
8      window.innerWidth / window.innerHeight,
9      0.1,
10     1000
11 );
12 const renderer = new THREE.WebGLRenderer({
13     canvas: document.querySelector("#bg"),
14 });
15
16 renderer.setPixelRatio(window.devicePixelRatio);
17 renderer.setSize(window.innerWidth, window.innerHeight);
18 camera.position.setZ(30);
19
20 renderer.render(scene, camera);
```

Using Three.js

Adding something interesting

- Once you've got your scene, camera, and renderer all set up, time to add some 3D shapes!
- We start by adding some geometric coordinates, the material (texture color), then we create a torus mesh
- But since we want to actually see it, we add some light to the scene!
- There are two lights; ambient and point lights

main.js

JS main.js > geometry

```
15
16  renderer.setPixelRatio(window.devicePixelRatio);
17  renderer.setSize(window.innerWidth, window.innerHeight);
18  camera.position.setZ(30);
19
20  renderer.render(scene, camera);
21
22  const geometry = new THREE.TorusGeometry(10, 3, 16, 100);      You, 46 minutes
23  const material = new THREE.MeshStandardMaterial({ color: 0xff6347 });
24  const torus = new THREE.Mesh(geometry, material);
25
26  scene.add(torus);
27
28  const pointLight = new THREE.PointLight(0xffffff);
29  pointLight.position.set(20, 20, 20);
30
31  const ambientLight = new THREE.PointLight(0xffffff);
32
33  scene.add(pointLight, ambientLight);
34
35  const lightHelper = new THREE.PointLightHelper(pointLight);
36  const gridHelper = new THREE.GridHelper(200, 50);
37  scene.add(lightHelper, gridHelper);
38
39  const controls = new OrbitControls(camera, renderer.domElement);
```

Using Three.js

Now for some animations!

- Everything should be looking good. The final step will be to make the torus move.
- Some simple animation loop code should do the trick

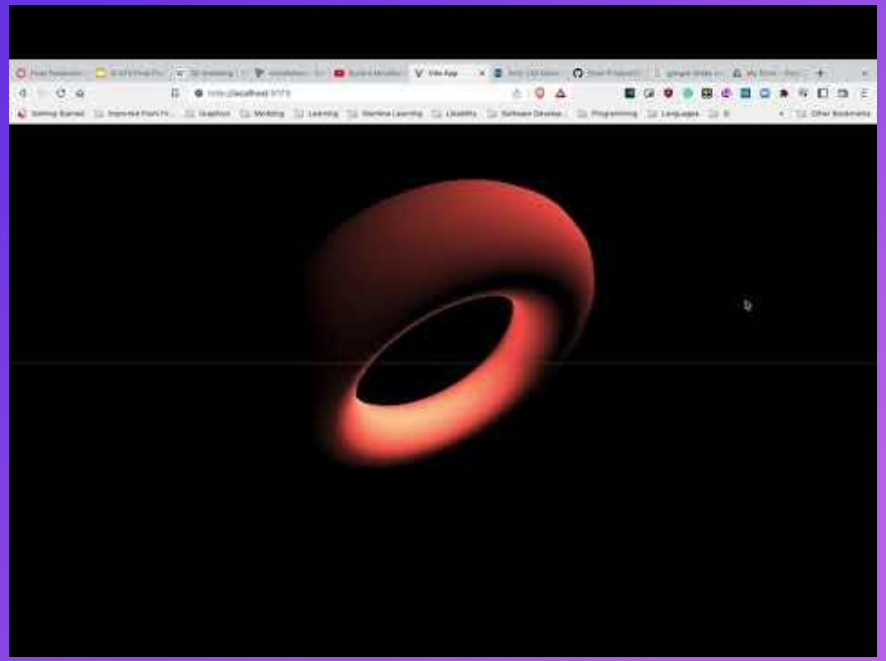
main.js

```
77
78  const animate = () => {
79      requestAnimationFrame(animate);
80
81      torus.rotation.x += 0.01;
82      torus.rotation.y += 0.005;
83      torus.rotation.z += 0.01;
84
85      controls.update();
86
87      renderer.render(scene, camera);
88  };
89
90  animate();
```

Using Three.js

Now for some animations!

- Now look at this incredible torus!!
- If I couldn't get the video to work; imagine the torus spinning around looking cool



Using Three.js

Add some beautiful stars

- Everything should be looking good. The final step will be to make the torus move.
- Some simple animation loop code should do the trick

main.js

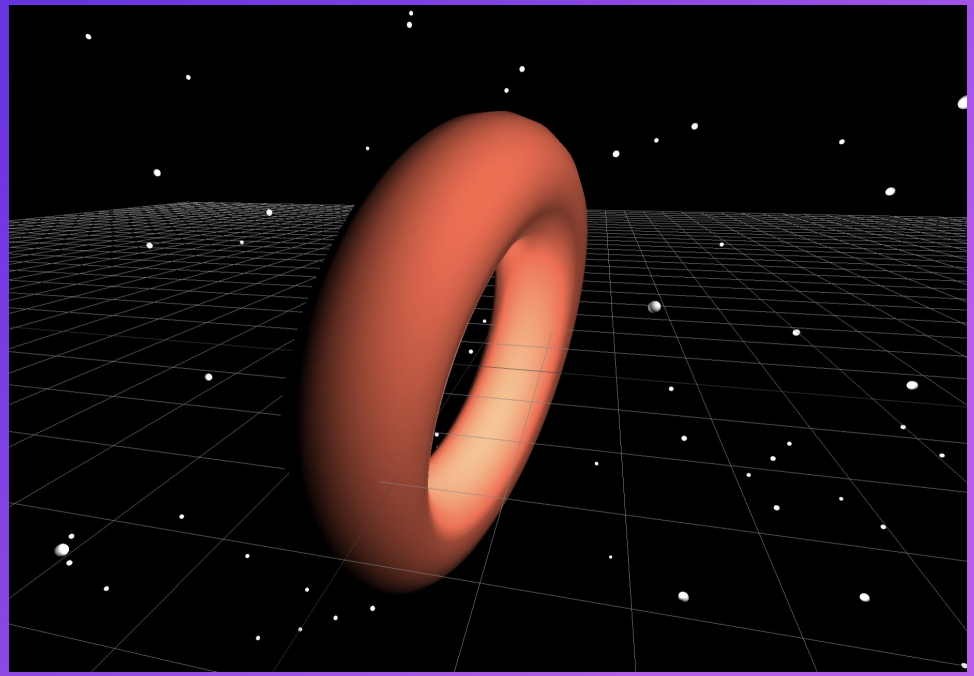
JS main.js > ...

```
34
35  const lightHelper = new THREE.PointLightHelper(pointLight);
36  const gridHelper = new THREE.GridHelper(200, 50);
37  scene.add(lightHelper, gridHelper);
38
39  const controls = new OrbitControls(camera, renderer.domElement);
40
41  const addStar = () => {
42      const geometry = new THREE.SphereGeometry(0.25, 24, 24);
43      const material = new THREE.MeshStandardMaterial({ color: 0xffffff });
44      const star = new THREE.Mesh(geometry, material);
45
46      const [x, y, z] = Array(3)
47          .fill()
48          .map(() => THREE.MathUtils.randFloatSpread(100));
49
50      star.position.set(x, y, z);
51      scene.add(star);
52  };
53
54  Array(200).fill().forEach(addStar);
```

Using Three.js

Look at these beautiful stars!

- Now look these incredibly stars!!
- If I couldn't get the video to work; imagine the torus spinning around looking cool



Using Three.js

Add a background!

- Everything should be looking good. The final step will be to make the torus move.
- Some simple animation loop code should do the trick

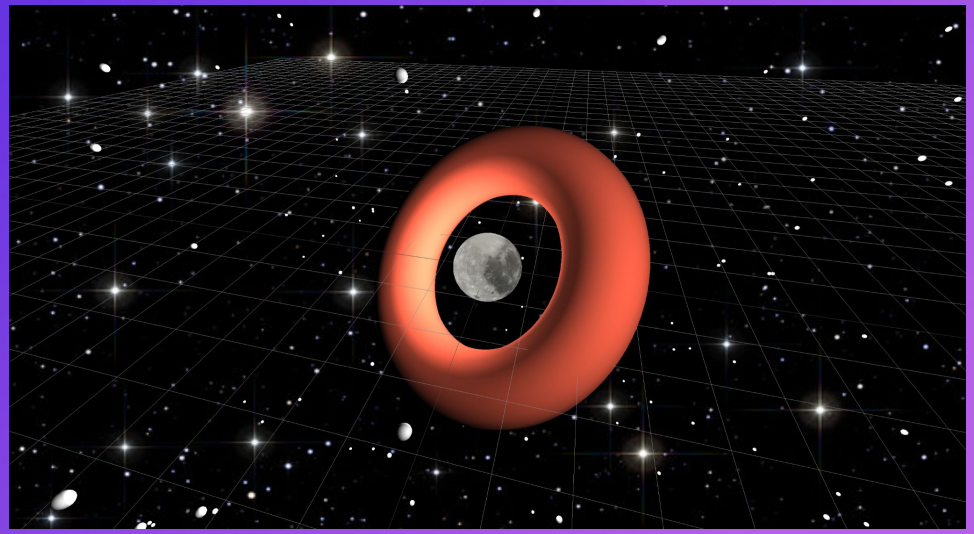
main.js

```
55
56  const spaceTexture = new THREE.TextureLoader().load("space2.jpeg");
57  scene.background = spaceTexture;
58
59  const spongebobTexture = new THREE.TextureLoader().load("spongebob.jpeg");
60
61  const spongebob = new THREE.Mesh(
62      new THREE.BoxGeometry(3, 3, 3),
63      new THREE.MeshBasicMaterial({ map: spongebobTexture })
64  );
65
66  scene.add(spongebob);
67      You, 55 minutes ago • add spongebob and moon texture and normal map …
68  const moonTexture = new THREE.TextureLoader().load("moon.jpeg");
69  const normalTexture = new THREE.TextureLoader().load("moon_normal.jpg");
70
71  const moon = new THREE.Mesh(
72      new THREE.SphereGeometry(3, 32, 32),
73      new THREE.MeshBasicMaterial({ map: moonTexture, normalMap: normalTexture })
74  );
75
76  scene.add(moon);
```

Using Three.js

Look at it!!

- Web development literally does not get better than this. I promise



Using Three.js

Look at it!!

- Web development literally does not get better than this. I promise

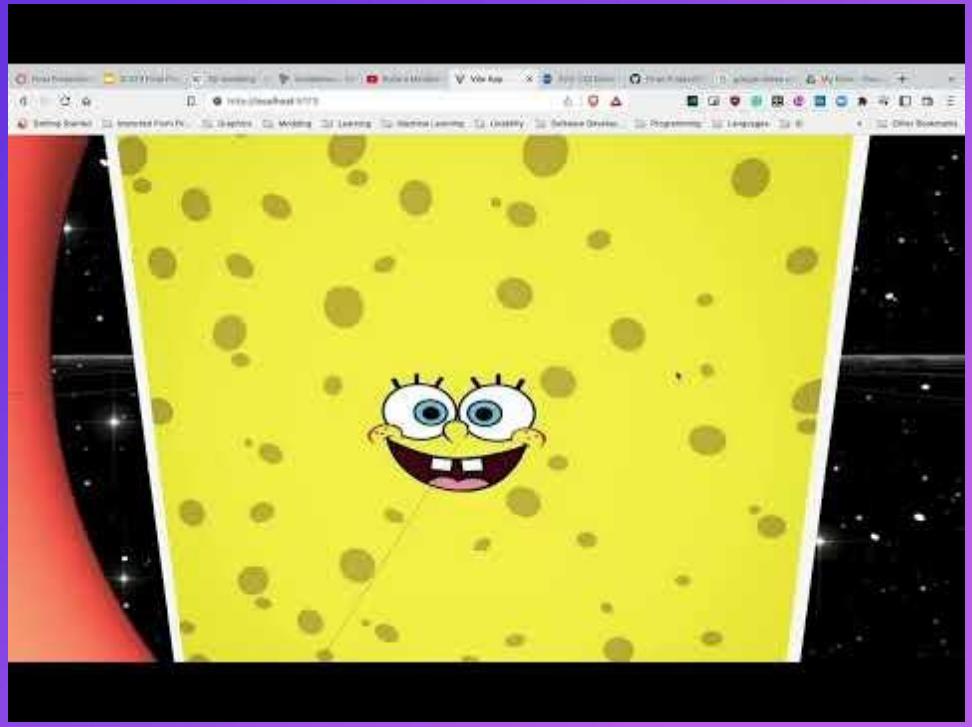


Table of contents