How to load STL 3d models in Three JS

No comments

The number of objects that you can design in three.js very limited and nearly all of them are some basic geometries like cube, sphere, cylinder, torus, and so on. We all know that there are a tone of various models that can be created using designing software and platforms like Blender. You can use these 3D models to create a web based game or animation or any kind of 3D interactive UX designs. Any of these 3D models have a specific kinds of file format such as STL, OBJ, FBX, PLY, GLTF, BLEND and so on. So, we need to be able to load all of these files in order to use them in Three JS. In This tutorial, we will get familiar with the STL loader in three.js and learn about the details of it. You can find different 3D models for free on websites like CGTrader.com, Thingiverse, and so on. You can also find a tone 3D models of human characters, animals, and all kinds of different objects. A great job you can do is to further modify these models in 3D softwares like Blender and add clothes to the human characters. We will soon have a blog how to create the 3D model of yourself using your 2D image captured by your phone.

A simple example from scratch:

We will get started with the main elements of a Three js scene, including the camera, the renderer, the scene, and the object. Before doing that, we use the Vite plugin to easily create all the folders and files you need to run the Three.js code. First off, create a folder in the directory of your projects by using the following commands: `mkdir STLLoader`
`cd STLLoader` Then, inside of the your project folder, create the necessary files and folders by simply running the Vite plugin command: `npm create vite@latest` Then enter the name of the project. You can write the name of your project as the name. And also the package (the name is arbitrary, and you can choose anything you want). Then select vanilla as the framework and variant. After that, enter the following commands in the terminal. Notice that here STLLoader is the project folder's name, and thus, we have changed the directory to STLLoader. The name depends on the name you enter in the Vite plugin : `cd STLLoader`
`npm install` Afterward, you can enter the JavaScript code you want to write in the main.js file. So, we will enter the base or template code for running every project with an animating object, such as a sphere. Also, do not forget to install the Three.js package library every time you create a project: `npm install three`

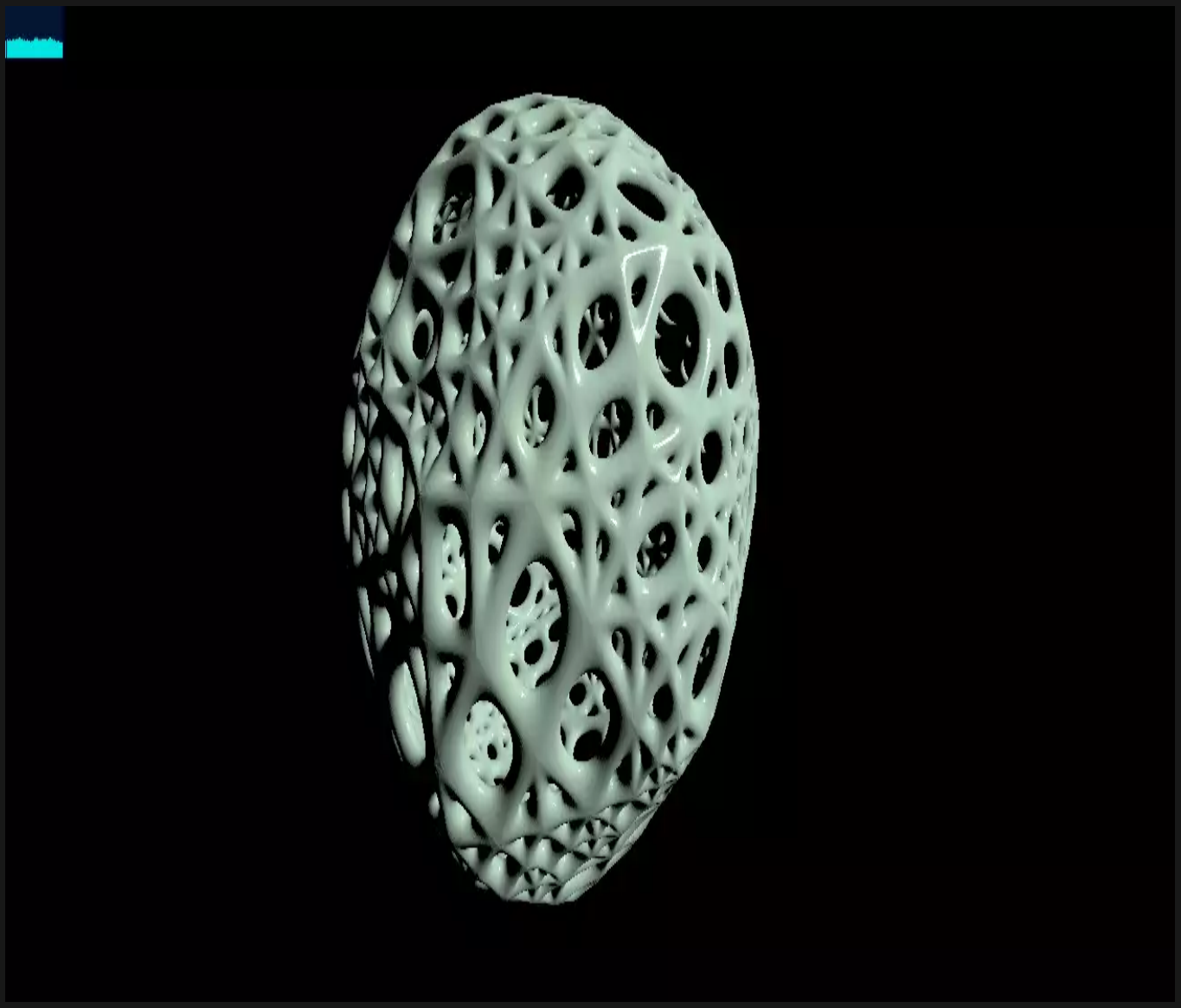
The code:

Now, enter the following script in the main.js file:

```
import * as THREE from 'three';
import { OrbitControls } from
'/node_modules/three/examples/jsm/controls/OrbitControls';
import { STLLoader } from
'/node_modules/three/examples/jsm/loaders/STLLoader';
import Stats from '/node_modules/three/examples/jsm/libs/stats.module'
;
const scene = new THREE.Scene();
const light = new THREE.SpotLight();
light.position.set(20, 20, 20);
scene.add(light);
const camera = new THREE.PerspectiveCamera(
     75,
     window.innerWidth / window.innerHeight,
     0.1,
     1000
);
camera.position.z = 3;
const renderer = new THREE.WebGLRenderer();
renderer.outputEncoding = THREE.sRGBEncoding;
renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);
const controls = new OrbitControls(camera,renderer.domElement);
const material = new THREE.MeshStandardMaterial({
     color: 0xffffff,
     metalness: 0.35,
```

```
        roughness: 0.1,
        opacity: 1.0,
        transparent: true,
        transmission: 0.99,
        clearcoat: 1.0,
        clearcoatRoughness: 0.25
});
const loader = new STLLoader();
loader.load(
        'models/3DModel.stl',
        function (geometry) {
                const mesh = new THREE.Mesh(geometry, material);
                scene.add(mesh);
        },
        (xhr) => {
                console.log((xhr.loaded / xhr.total) * 100 + '% loaded');
        },
        (error) => {
                console.log(error);
        }
);

window.addEventListener('resize', onWindowResize, false);

function onWindowResize() {
        camera.aspect = window.innerWidth / window.innerHeight;
        camera.updateProjectionMatrix();
        renderer.setSize(window.innerWidth, window.innerHeight);
        render();
}

const stats = Stats();
document.body.appendChild(stats.dom);

function animate() {
        requestAnimationFrame(animate);
        controls.update();
        render();
        stats.update();
};

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

animate();
```

Now if we save the code, and enter the following command in the terminal: `npm run dev` The above script will give the following result:



Explaining the code:

Before explaining the code, you might ask how to design the above 3D model. Or where should I get it from? First of all, the above 3D model has been designed in Blender by the Arashtad team, and we have provided the guidelines for creating such a lattice structure in one of the blog tutorials called “Different Kinds of Lattice Structure Using Blender”. Secondly, you can download any 3D model you want from the Thingiverse.com website for free, and here, we only want to load a .stl 3D model and visualize it in Three.js. The above code is like any Three JS boilerplate script. At first, we declared the scene, the camera, and the renderer and did all the routine stuff on them. Afterward, we defined the material with all its properties. Next, we loaded the STL file from the models folder (Where we had already pasted our STL 3D model). Finally, we added the animation and render function. Notice that you should either rename the name of your 3D model to 3DModel or enter its name of it in the loader section of the code.

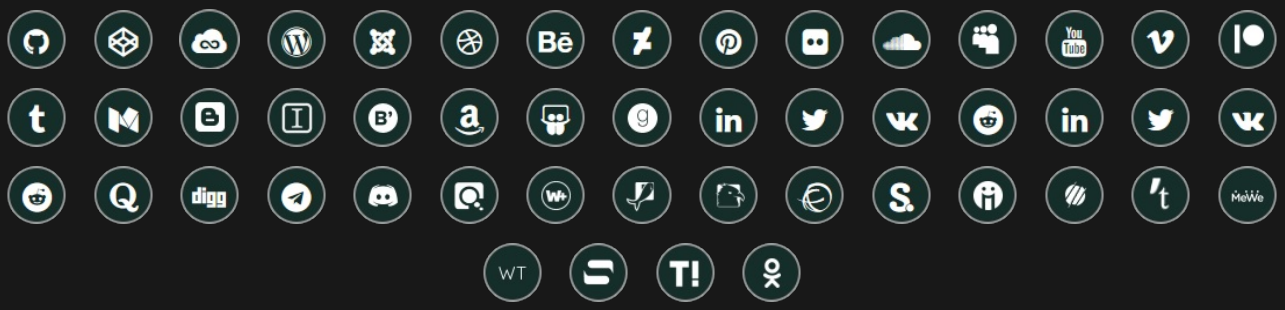
Conclusion

In this tutorial, we learned how to load an STL model in Three JS and visualize it in a fantastic way. You can read our articles about Blender on our blog and learn how to design 3D models so that you can later visualize beautifully in three.js. Moreover, you can download your preferred 3D models from different websites that offer these models for free. There are plenty of file formats that you can store a 3D model. And in Three JS, we can import nearly all of these 3D model file formats. We will cover more detailed files like OBJ with their textures, and also GLTF file format in the future tutorials. Hope you have enjoyed this tutorial!

Join Arashtad Community

Follow Arashtad on Social Media

We provide variety of content, products, services, tools, tutorials, etc. Each social profile according to its features and purpose can cover only one or few parts of our updates. We can not upload our videos on SoundCloud or provide our eBooks on Youtube. So, for not missing any high quality original content that we provide on various social networks, make sure you follow us on as many social networks as you're active in. You can find out Arashtad's profiles on different social media services.



Get Even Closer!

Did you know that only one universal Arashtad account makes you able to log into all Arashtad network at once? Creating an Arashtad account is free. Why not to try it? Also, we have regular updates on our newsletter and feed entries. Use all these benefitial free features to get more involved with the community and enjoy the many products, services, tools, tutorials, etc. that we provide frequently.

SIGN UP | NEWSLETTER | RSS FEED