Making A Three JS Cube: A Useful Guide

No comments

One of the simplest projects that will help you learn the basics of Three js, is designing a cube, changing the color, light sources, materials, adding textures, and so on. Of course, if you want to just get started with Three js, this example is one of the most idealistic projects that you need in order to learn it. Fewer tutorials will teach you everything about manipulating the simple cube and how you can decorate it in the very first tutorial. But, here we have provided the step-by-step guidelines for getting started with the process.

The things that you will learn throughout this tutorial are creating a Three js using Vite plugin, creating a cube with basic mesh material and also standard mesh material. Afterwards, we will add point light in order to be able to see the cube created with the standard mesh material. The next thing that we will try will be to create a wireframe for the cube. Finally, we will add a texture to the cube to make it look nice and beautiful.

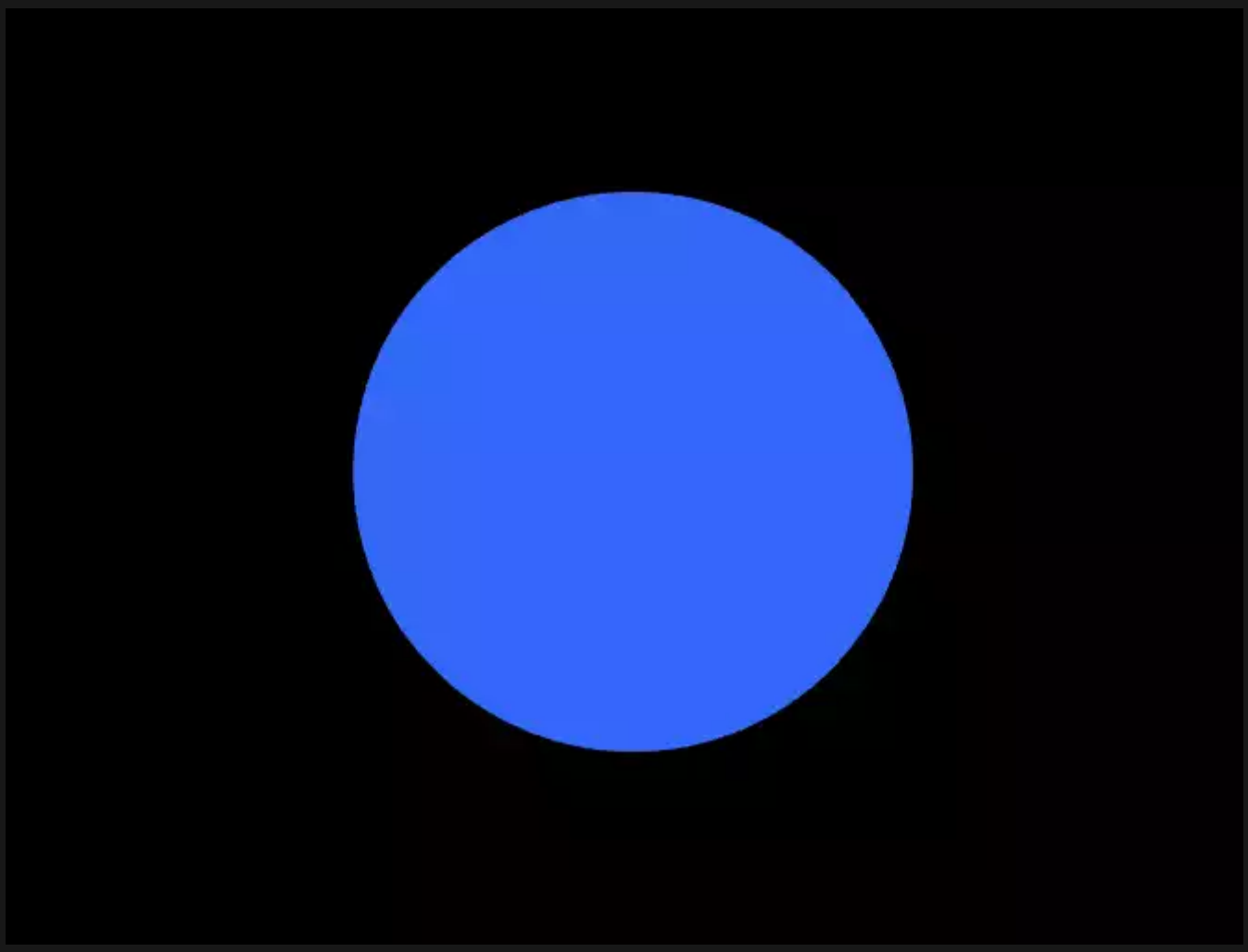
The Basics of Making A Three JS Cube

We are going to get started with the simple elements of a Three js scene including the camera, the renderer, the scene, the object and the light source (if necessary). Before we do that, we'd rather use the Vite plugin to be able to easily

create all the folders and file that 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 cube` `cd cube` Then, inside of the glowing sphere 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 `glowingSphere` as the name. Also, the package name is arbitrary and you can choose anything that you want. Then, select vanilla as the framework and variant. After that, enter the following commands in the terminal: `cd cube` `npm install` Afterward, you can enter the javascript code that 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 an sphere. Also do not forget to install Three js package library every time create a project: `npm install three` Afterward, you can enter the following script in the `main.js` file:

```
import * as THREE from 'three';
import { Mesh } from 'three';
const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera(75
, innerWidth / innerHeight , 0.1, 1000);
const renderer = new THREE.WebGLRenderer({
     antialias : true
})
renderer.setSize(innerWidth, innerHeight);
document.body.appendChild(renderer.domElement);
//creating a sphere
const geometry = new THREE.SphereGeometry(5, 50, 50);
const material = new THREE.MeshBasicMaterial({
     color:0x3268F8
})
const sphere = new THREE.Mesh(geometry,material);
scene.add(sphere);
camera.position.z = 15;
function animate(){
     requestAnimationFrame(animate);
     renderer.render(scene,camera);
     sphere.rotation.y += 0.003;
}
animate();
```

The above code can be used as a boilerplate for later projects. The output of this code will be blue sphere like the below photo. But to be able to show that, you should write the following command in the terminal: `npm run dev`



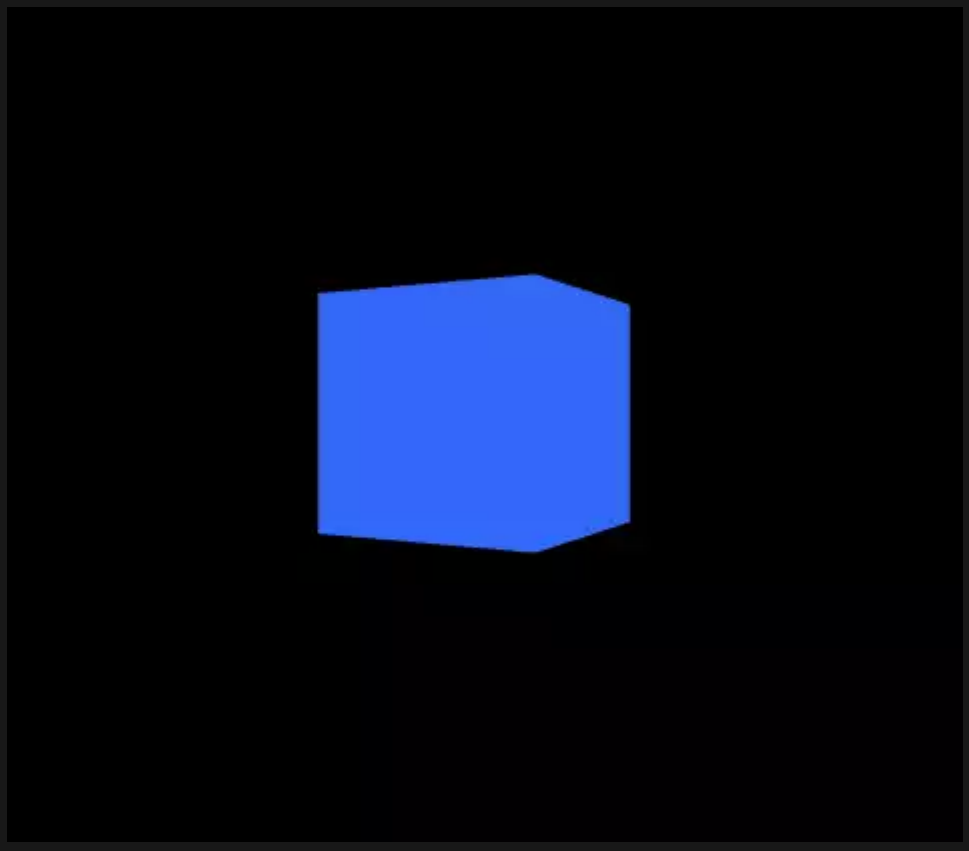
If you want to change the sphere to cube change the following script from:

```
const geometry = new THREE.SphereGeometry(5, 50, 50);
```

To:

```
const geometry = new THREE.BoxGeometry(4, 4, 4);
```

If you save the code, the result will be a rotating cube like this:



And if you want to change the material from basic mesh to standard, change the code from:

```
const material = new THREE.MeshBasicMaterial({
     color:0x3268F8
})
```

To:

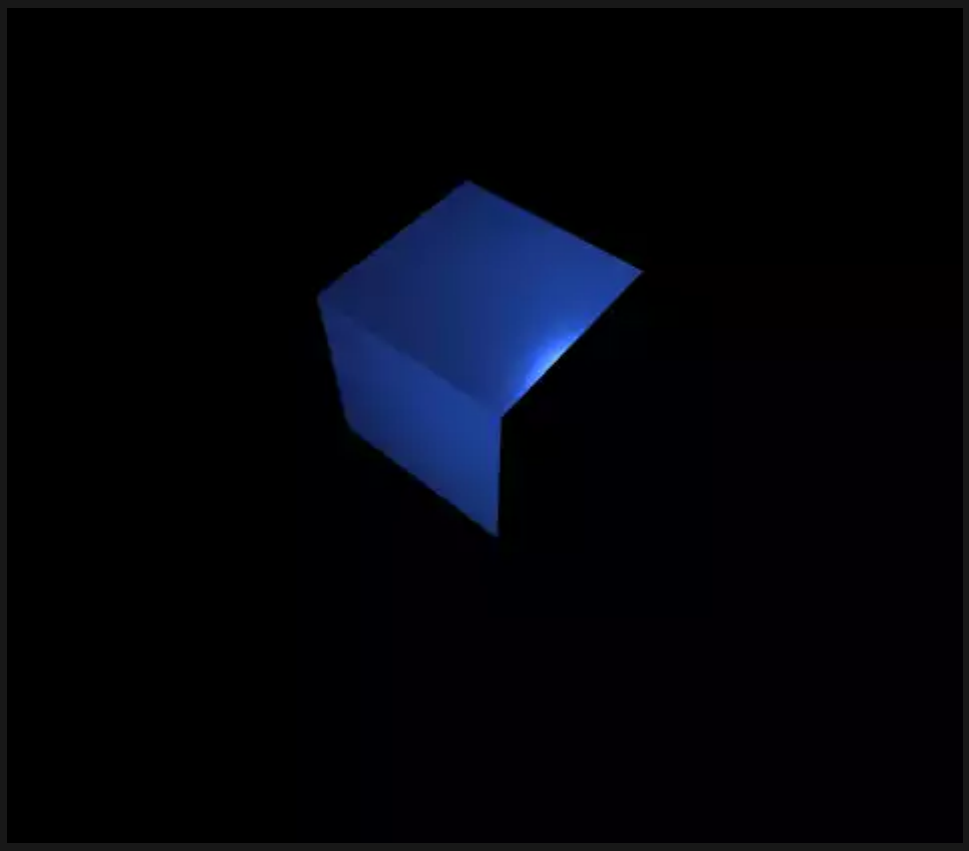
```
const geometry = new THREE.BoxGeometry(4, 4, 4);
const material = new THREE.MeshStandardMaterial();
material.color = new THREE.Color(0x3268F8);
material.roughness = 0.2;
material.metalness = 0.7;
```

Besides, add 2 point lights to the scene so that you can see the cube made with the standard mesh material. To do so, enter the below script twice with 2 different names:

```
const pointLight = new THREE.PointLight(0xffffff, 0.1);
pointLight.position.x = 2;
pointLight.position.y = 3;
pointLight.position.z = 4;
pointLight.intensity = 2;
```

```
scene.add(pointLight);
```

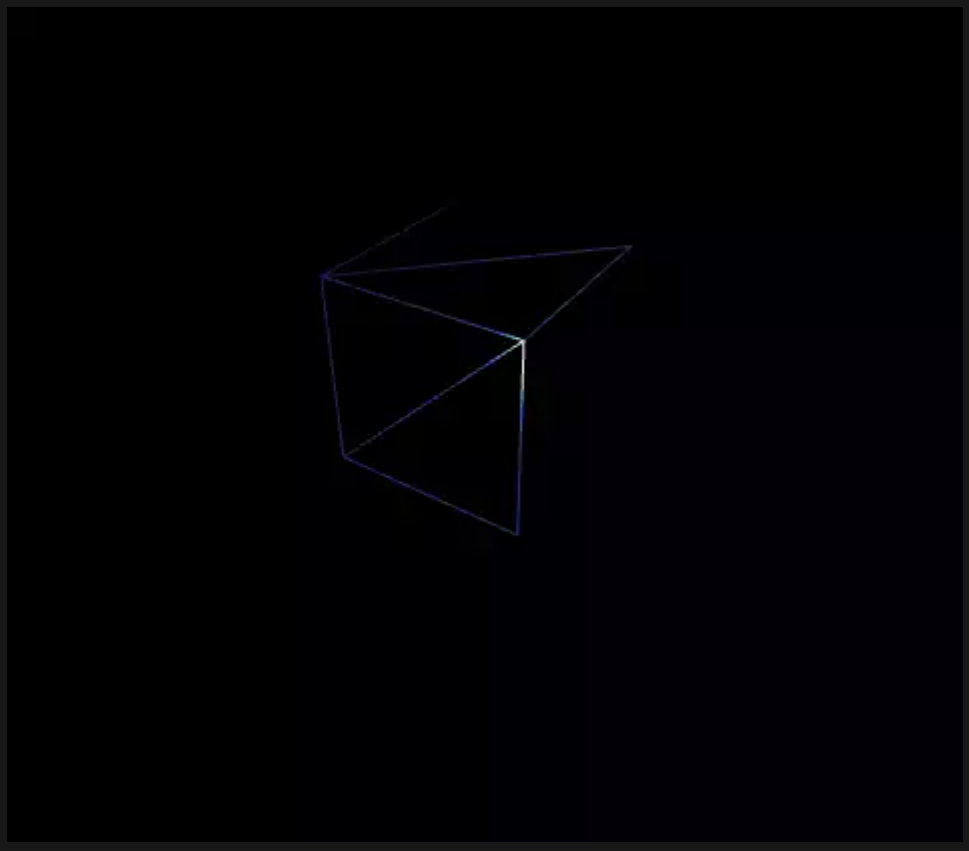
Result:



If you want to turn the cube to a wireframe, you can set the wireframe property as true. Then, for adding texture, set it to false:

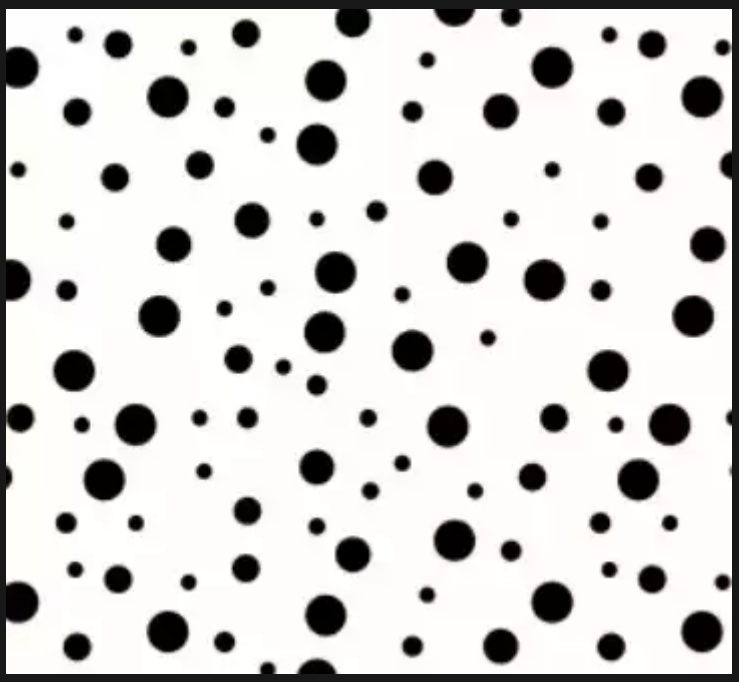
```
material.wireframe = true;
```

Save the script and the result will be:



Adding Texture to the Cube in Three JS

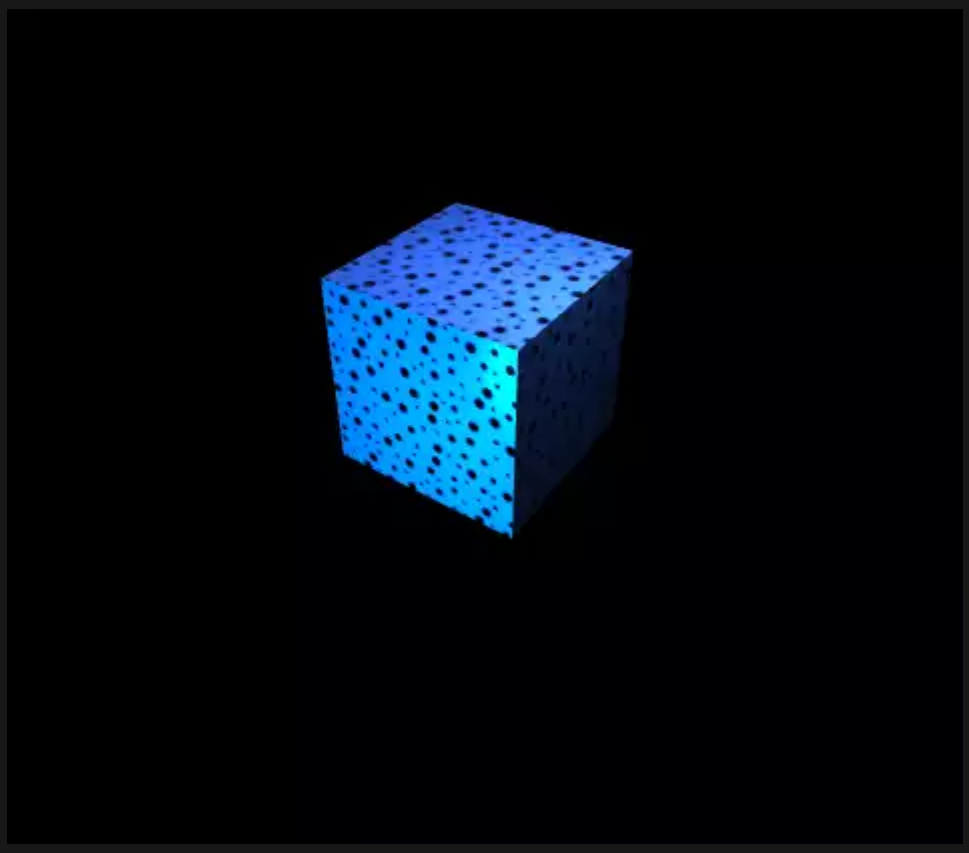
One of the important things that makes your objects more beautiful and decorated, is adding texture to them. The process is so easy, you can google any texture that you want and download it. Then use the map property of the material and load the photo of the texture of the surface of the object by giving the address of the image. For instance, we want a Polka dot texture on our cube. We google it and find this photo:



Now, we will create a new folder called `img` in the directory of the project. And, we add the above texture image to that folder. Then, we will add the below code under the material and save the `main.js`.

```
material.map = new ;
THREE.TextureLoader().load('./img/polkadot.jpg');
```

The result will be exciting:



You can add any other texture of your choice and taste to the surface of the cube. Notice that some textures need precision in the placement. For instance, if you want to design a dice, you will need to map any of the numbers to a certain side of the cube. And some tricks will be needed to have accurate mapping. Doing this task is not the subject of this article, but we will hopefully cover it in our future tutorials.

Final Decision

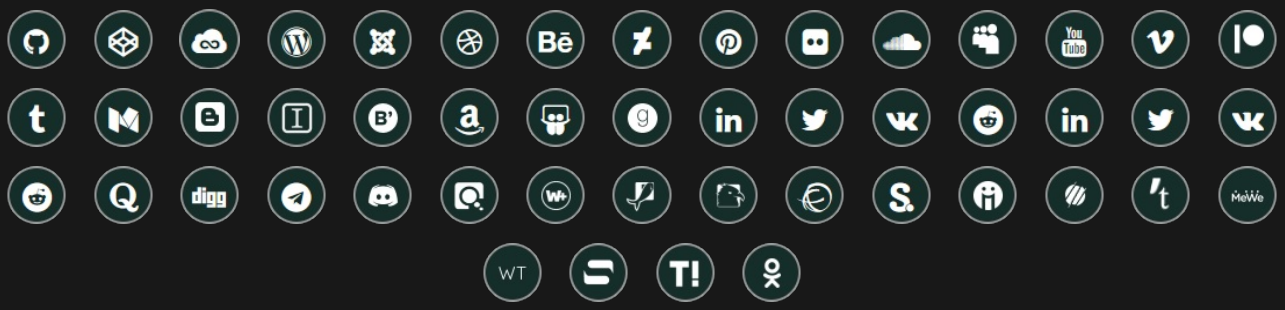
In this tutorial, we managed to create a Three js project from scratch and started with a boilerplate code. Then, we added a cube instead of a sphere to the scene. Afterward, we changed the material from basic to standard, and to be able to see that, we added 2 point lights with high intensities. The next thing that we did was to create the wireframe of the cube and set it to true.

After that, we set it to false again, in order to add some textures to the cube. The most interesting part of the tutorial was when we added the texture. To do so, we first googled the texture of our choice which was a polka dot in our case and added it to a folder inside the directory of the project, and finally added the map property to the material to see the texture on the cube.

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