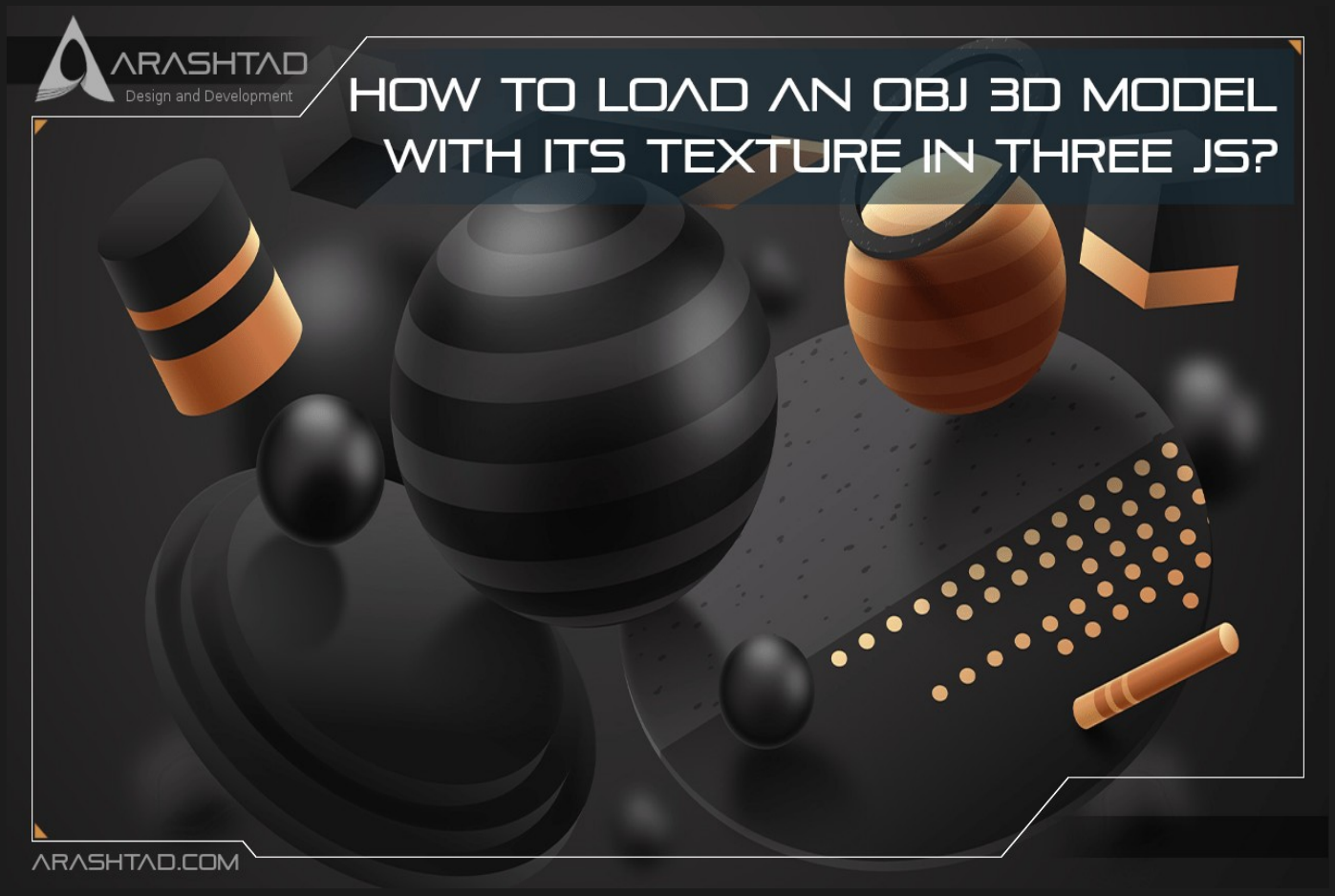
ARASHTAD

Title	HOW TO LOAD AN OBJ 3D MODEL WITH ITS TEXTURE IN THREE JS?
Description	Designing 3D web pages using Three.js
Date	July 2022
Author	Arashtad
Author URI	https://arashtad.com



Most of the 3D models that we want to use in games, such as characters, furniture, closed areas, rooms, cars, etc, are designed in the OBJ format and naturally they have a texture too. The texture is usually exported in the MTL file format. Most of the prepared free models that you will find different on websites that offer 3D models for free like thingiverse, will give zip file containing OBJ and MTL file formats. In this tutorial, we will focus on importing OBJ and MTL files in Three JS. This tutorial, is an important part of designing your game, animation or any other kind of web application because there are already many free 3D designs out there on the internet. You only need to import and use them in your scene. So now, let's get started with the project. Don't forget to find a 3D model of your choice from where ever you like. Notice that the file need to have .obj and .mtl file formats. Some these models have a .png or .jpg photo with them.

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 OBJLoader
```

```
cd OBJLoader
```

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 `OBJLoader` is the project folder's name, and thus, we have changed the directory to `OBJLoader`. The name depends on the name you enter in the Vite plugin :

```
cd OBJLoader
```

```
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
```

IMPORTING THE OBJ 3D MODEL:

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

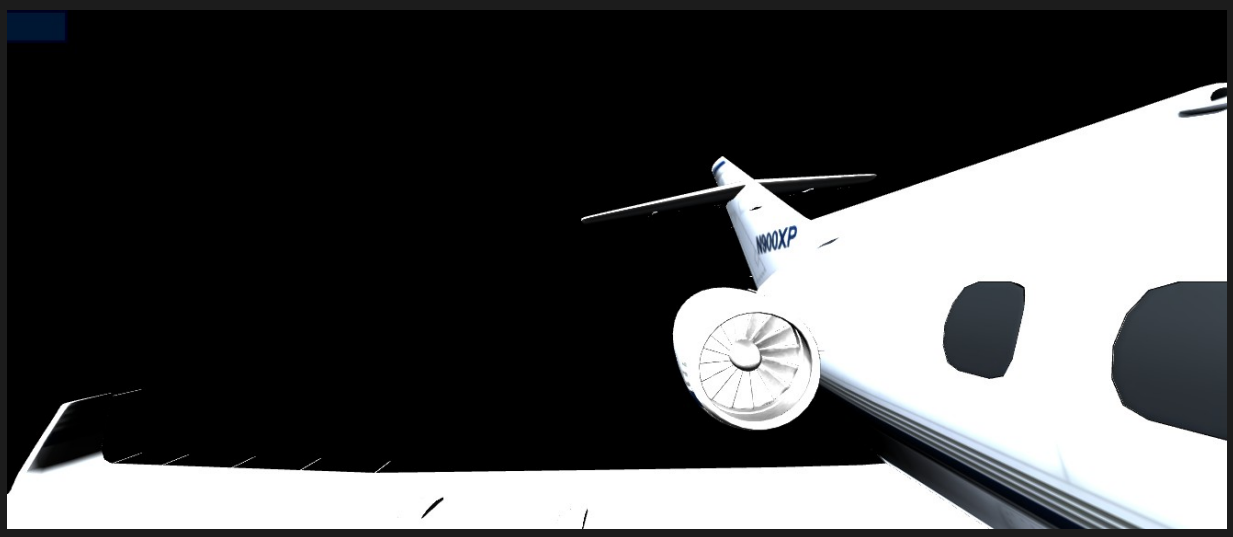
```
import * as THREE from 'three'
import { Mesh } from 'three'
import { OBJLoader } from
    'three/examples/jsm/loaders/OBJLoader'
import { MTLLoader } from 'three/examples/jsm/loaders
    /MTLLoader'
import {OrbitControls} from "/node_modules/three/examples/jsm
    /controls/OrbitControls.js"
import Stats from "/node_modules/three/examples/jsm/libs
    /stats.module.js"

const scene = new THREE.Scene()
scene.add(new THREE.AxesHelper(5))

const pointLight = new THREE.PointLight(0xffffff,10)
pointLight.position.set(0,8000,0)
pointLight.intensity = 5
scene.add(pointLight)
const pointLight2 = new THREE.PointLight(0xffffff,10)
pointLight2.position.set(2000,3000,4000)
pointLight2.intensity = 5
scene.add(pointLight2)

const width = 20;
const height = 20;
const intensity = 5;
const rectLight = new THREE.RectAreaLight( 0xffffff,
        intensity,width, height );
rectLight.position.set( 5000, 5000, 5000 );
scene.add( rectLight )

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)

const controls = new OrbitControls(camera,
renderer.domElement)

//Loading the MTL and OBJ files
var mtlLoader = new MTLLoader();
mtlLoader.load('/Models/plane.mtl',function(materials){
  materials.preload();
  var objLoader = new OBJLoader();
  objLoader.setMaterials(materials);
  objLoader.load('/Models/plane.obj',function(object){
    scene.add(object);
  });
});

camera.position.z = 15

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)
    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:



As you can see, we are observing the smallest details of an airplane with all the textures of the different parts.

EXPLAINING THE CODE:

As always, we added the necessary elements of every scene, such as the scene itself, the camera, the renderer, the orbit controls, and the animation function. Notice that we also imported all the necessary packages for our purpose.

The main part of this code is related to loading the OBJ and MTL files where we first imported the MTL and inside of its function added the OBJ file as well. In other words MTL is the material of the OBJ object. It is like creating an object and adding its material to it.

Notice that most of the objects that you find on the internet, are very large or they could in size. So you should set the lights and the camera according to the dimensions of the given object. Otherwise, you will see nothing or you have to zoom in or zoom out so much in order to see the object clearly. Also, the lighting might not display the object properly. So, you have to set the position and intensity of it, too.

We have placed the models in the Models folder and have imported them according to their names. Make sure you set the names according to your files and you should have the files ready for display!

ARASHTAD

CONCLUSION

In this tutorial, we have managed to import an object and its texture using the OBJ loader and MTL loader in Three JS. Using such a tool, we can import as many 3D models as we want to create the animation, game or the web application of our choice using Three JS. The .mtl file format works like material of the object which is imported in the format of .obj. Hope this tutorial has helped you import the 3D models that you like in your Three.js scenes.

ARASHTAD TEAM

We have a strong 3D modeling and 3D web development team in Arashtad group, ready to design high quality productions in case of 3D websites, 3D games and metaverses. We are also an experienced team in Blockchain development. If you have an idea for a project like this, please don't hesitate to contact us on Arashtad.com. Also, you can see some samples of our previous projects at https://demo.arashtad.com.