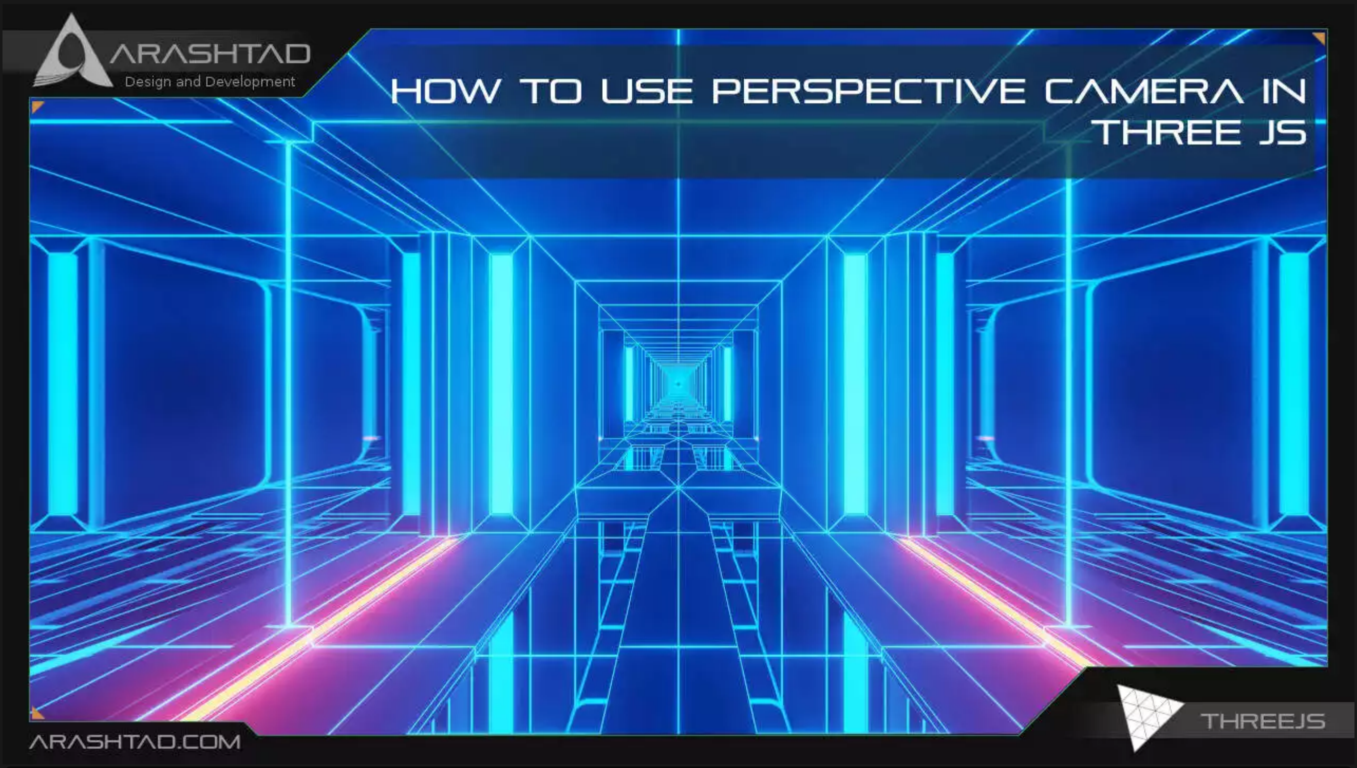
How to Use Perspective Camera in Three JS

No comments



One of the things that many new learners have difficulty with is about the perspective camera in Three js. Like our previous articles about Three js lights and others, we are going to cover a very problematic concept in Three js. This is something that may cause misunderstanding. So, we need to get into it deeply.

Perspective and Other Cameras in Three JS

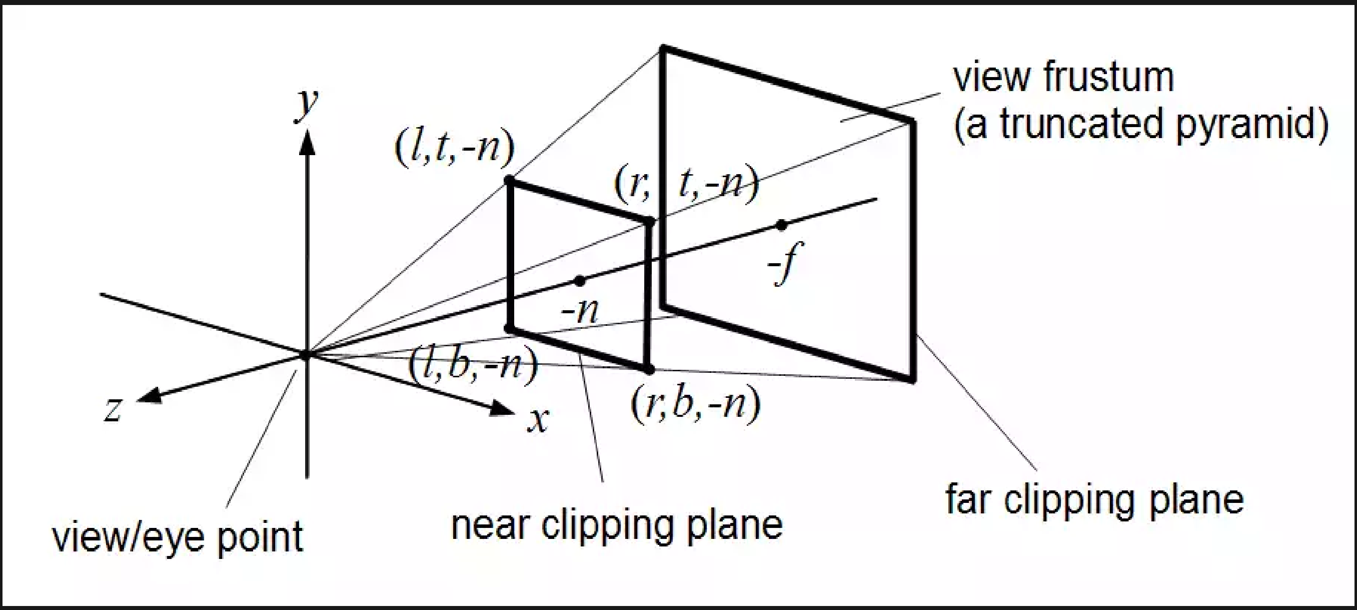
In general, Three js is a library that allows you to visualize the 3D data in the form of meshes and lights. Then, convert that data into a 2D representation for an HTLM canvas. However, a Three js scene does not boil down to meshes and lights. There are 3 main elements before starting to add any item. These elements are the camera, the scene, and the renderer.

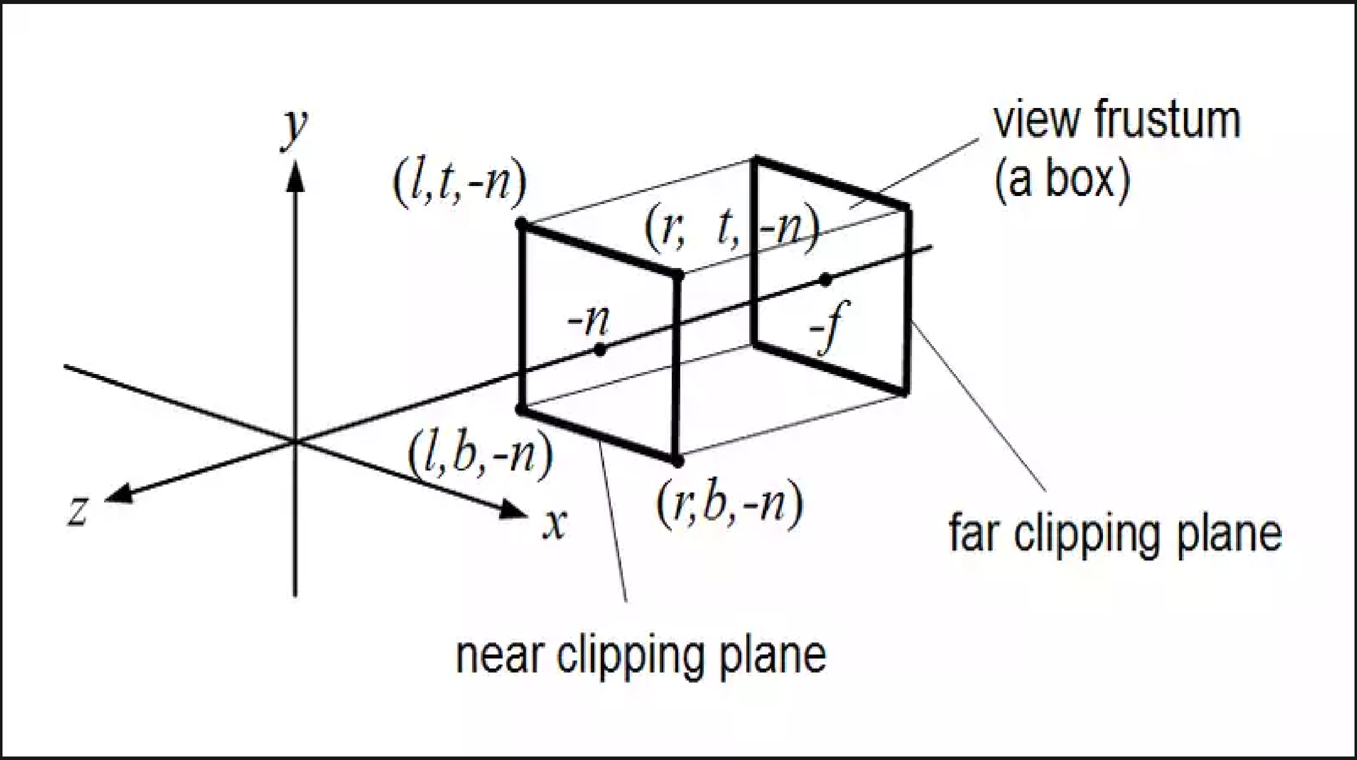
The renderer displays the scene onto an HTML canvas element and uses WebGL by default. The scene, which is created at the beginning of every project, allows you to set up what is to be rendered by Three js and the coordinates of the items. And at last, we have a camera. There are 2 types of cameras in general considering the way they see the object including the Perspective camera and the Orthographic camera. However, there are other types of cameras used in Three js such as array camera, Cube camera, and the Stereo camera. Throughout this article, we will introduce cameras in detail and provide examples for the perspective camera.

Perspective vs Orthographic in Three JS

The perspective camera and the orthographic one are most usually compared due to the differences they have in the perspective view. The perspective camera mimics the way the human eye sees objects whereas the orthographic camera is like a cube in itself where the perspective remains constant as the object gets further meaning that the size remains the same regardless of the distance of the camera from the object.

The 2 photos below show exactly how the perspective camera differs from the orthographic one. The first photo represents the view from the perspective camera, as you can see, as the view plane gets further from the camera (eye point), the space that the camera viewport can cover, gets larger. While, if you look at the second photo, the size of the object won't change as it gets further from the viewpoint.





Stereo Camera's Role in Using Perspective in Three JS

The stereo camera uses dual perspective cameras for effects like 3D anaglyph or parallax barrier. The 3D anaglyph in summary is a kind of effect that creates the perception of a 3-dimensional photo by providing 2 layers of filtered images of opposite colors like red and cyan. The 2 layers of images will be perceived as 3D using certain glasses that have 2 different filters on each glass which is going to reveal only one of the 2 layers for any of them. For instance, the left glass which is red filters one of the 2 layers, and the right one which is cyan will reveal the other layer.

The superposition of the layers filtered with certain glasses will result in the 3-dimensional perception. The stereo camera provides the kind of 3D scene that if the viewer wears the 3D glass will be able to see a 3-dimensional view on the screen. In the following photo, you can see an anaglyph photo. On the other hand, the parallax barrier provides a kind of 3d image perception without the need for the viewer to wear 3D glasses.

Cube Camera

Is a kind of camera that creates 6 cameras that render to a WebGL Cube Render Target. One of the examples shown by the Three js official website is a rotating view over a sphere. The type of camera that is used for the 6 cameras is the perspective type.

Array Camera

Array camera is normally used in VR scenes where a set of cameras are considered to efficiently cover all the viewports. In other words, as we need different cameras to cover different views in VR scenes, an array of cameras are considered to cover a specified part of the 3D scene. The kind of camera that is used for each sub camera, is the perspective type.

Basics of Using Perspective Camera in Three JS

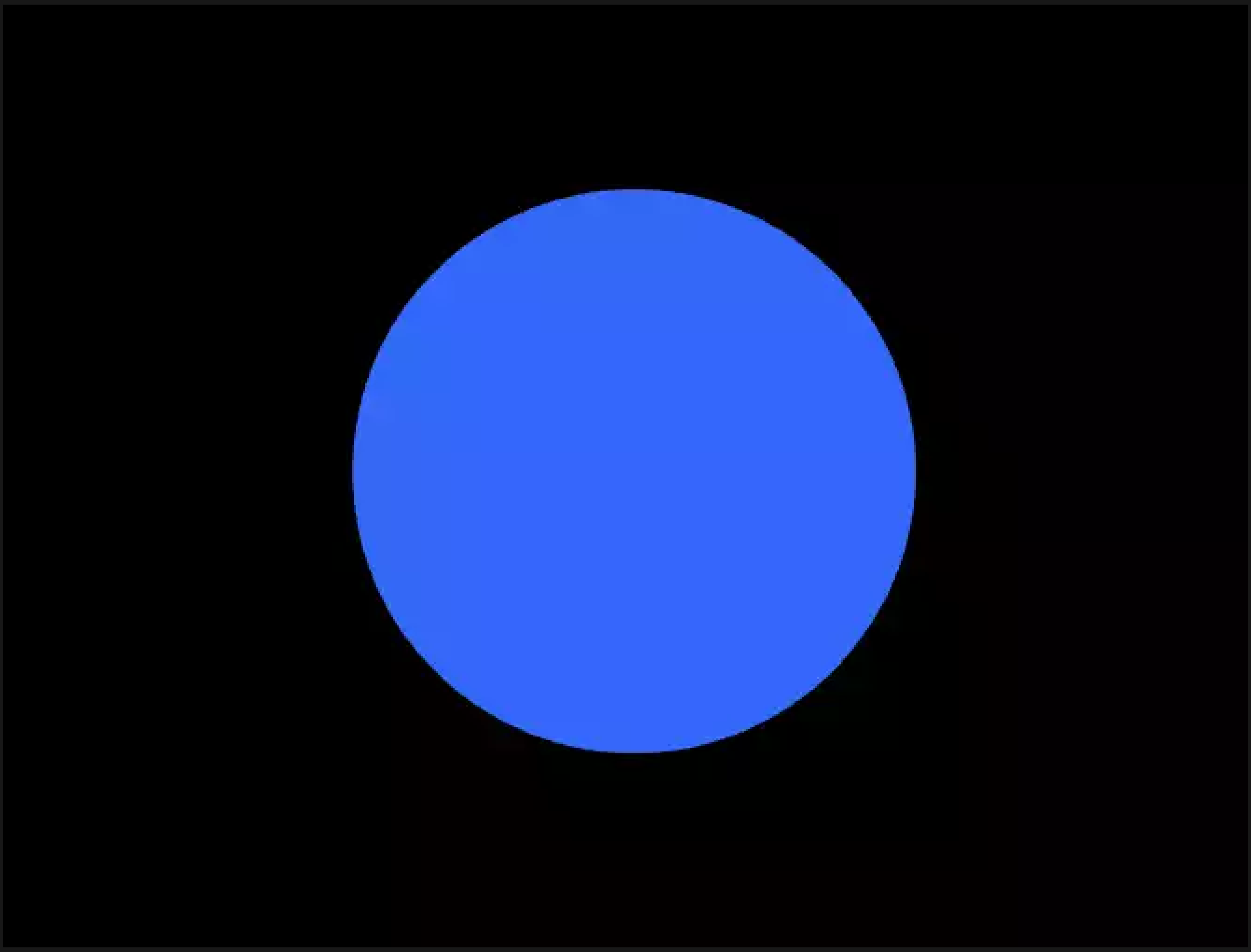
We are going to get started with the simple elements of a Three js scene including the camera, the renderer, the scene, the object, and the light source (if necessary). Before we do that, we'd rather use the Vite plugin to be able to easily create all the folders and files that you need to run the Three.js code. First off, create a folder in the directory of your projects by using the following commands: `mkdir Camera` `cd camera` 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. And also the package (the 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 camera
npm install
```

Then, 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` Now, 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 following photo. But to be able to show that, you should write the following command in the terminal: `npm run dev`



Now, we want to change the result in order to test the perspective camera with the wireframe of a cube. So change:

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

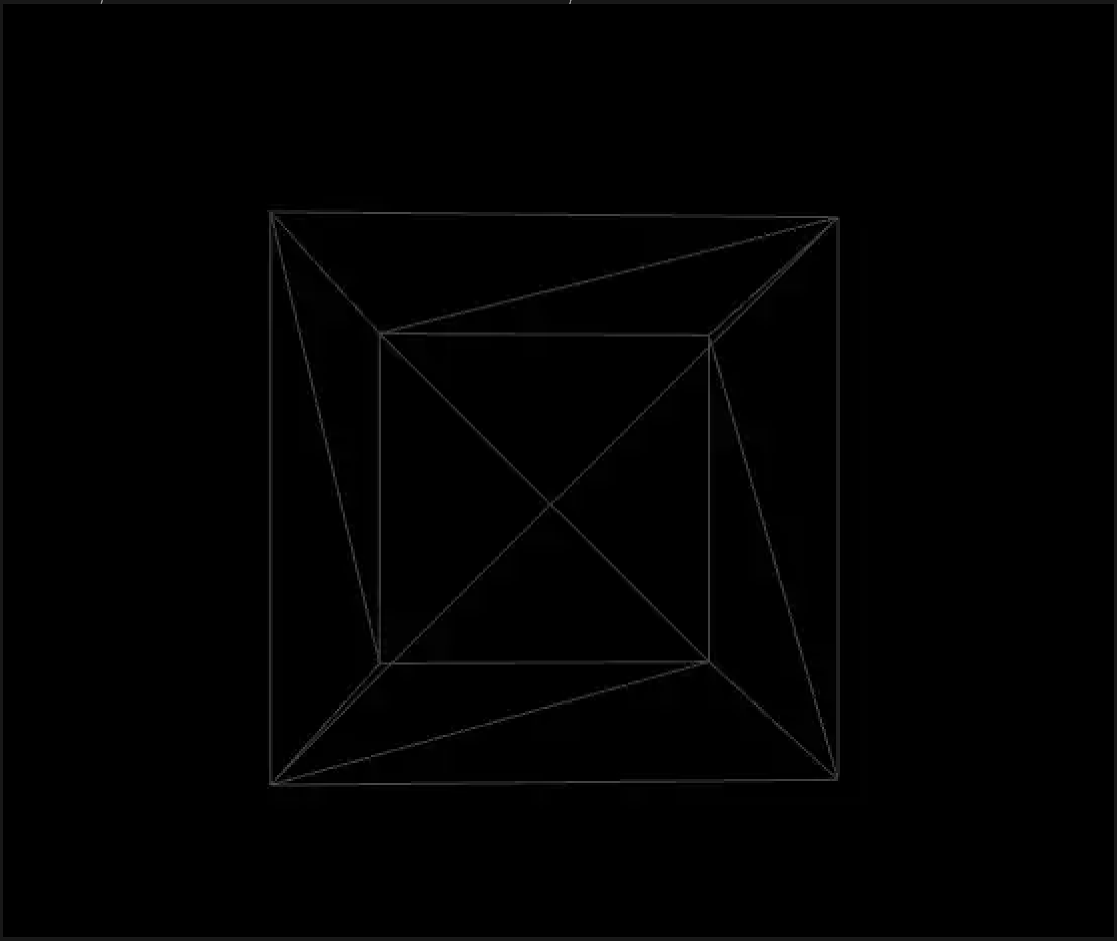
To:

```
const geometry = new THREE.BoxGeometry(8, 8, 8);
const material = new THREE.MeshStandardMaterial();
material.color = new THREE.Color(0xB1E1FF);
material.roughness = 0.2;
material.metalness = 0.7;
material.wireframe = true;
```

Also, add an ambient light to the scene so that you can see the wireframe made with the standard mesh material:

```
const light = new THREE.AmbientLight( 0xffffff);
scene.add(light);
```

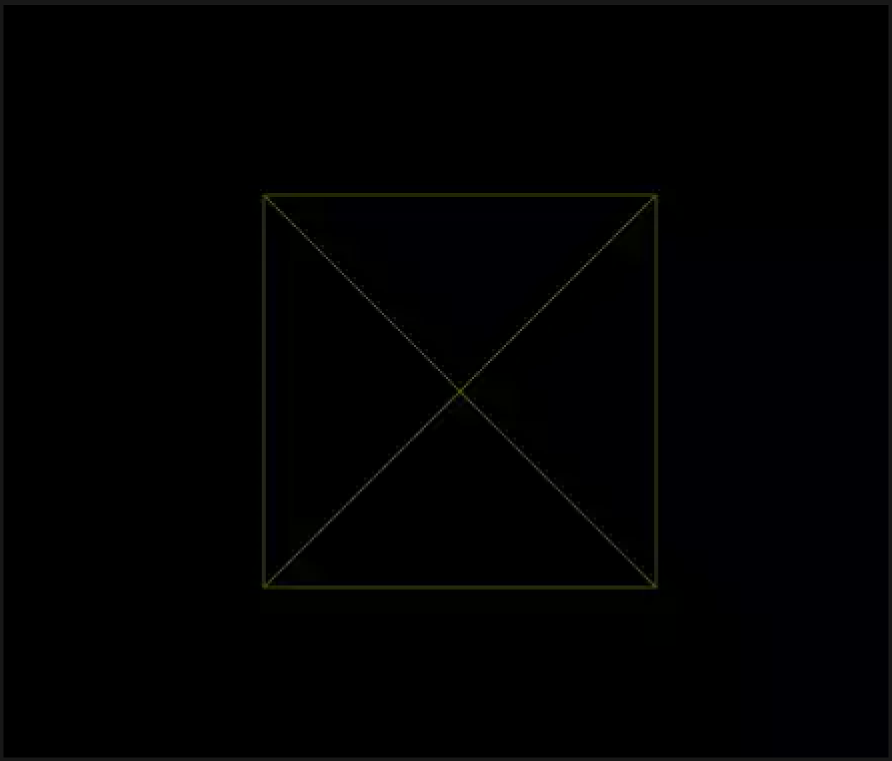
Save the code and you will be able to see a result like this in your browser:



The above result is the representation of the perspective camera. If we change the camera to orthographic by writing:

```
const camera = new THREE.OrthographicCamera( innerWidth / -14
, innerWidth / 14, innerHeight / 14,
     innerHeight / - 14, 1, 100 );
```

The result will be:



Summing Up

In this article, we have got familiar with the concept of the camera in Three js and got familiar with different types of it. Furthermore, we compared the 2 main camera views including the perspective and the orthographic view. Not only that, but we also wrote some Three js scripts to see the differences between the orthographic camera and the perspective one when viewing the wireframes of a cube (box geometry). Moreover, we introduced different cameras from the number of cameras used to view a certain space. These types of cameras were stereo camera, cube camera, and array camera.

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