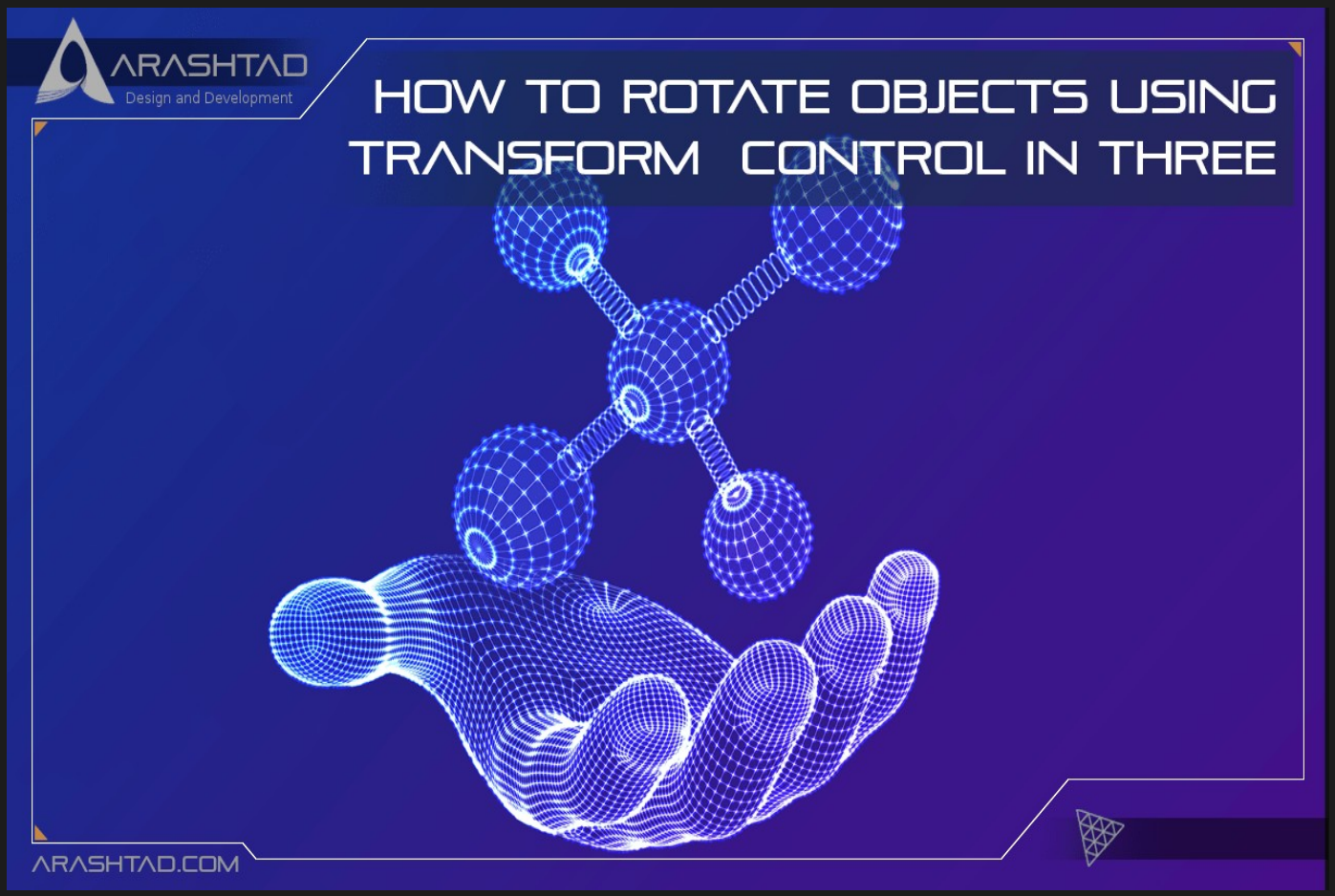
ARASHTAD

Title	HOW TO ROTATE OBJECTS USING TRANSFORM CONTROL IN THREE JS
Description	Designing 3D web pages using Three.js
Date	July 2022
Author	Arashtad
Author URI	https://arashtad.com



Inside every 3D design software, we have the tools for translating the object, meaning we have a button for transferring the object from one point to the other. In addition to that, we have tools for rotating the object and also scaling it. These three are the essential tools for every 3D design application. But imagine a 3D app (software or web application) without orbit controls (having a 360-degree view or rotating around the object); this will result in an annoying user-interactive software everyone hates at first glance. Consequently, every 3D software design app consists of some must-have tools. These essential utilities are, transform controls, and orbit controls, and Three.js is no exception.

In this article, we will start from a boilerplate project using the Vite plugin, and then we will modify the script to have all sorts of controls around the object. Some keys on the keyboard will activate the rotation, translation, and scaling tools, each separately. We will also have orbit controls to see everything clearly from all aspects.

GETTING STARTED WITH THE BASICS:

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 TransformControls

cd TransformControls
```

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 that you want). Then select vanilla as the framework and variant. After that, enter the following commands in the terminal:

```
cd TransformControls

npm install
```

Afterwards, 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
```

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.js";
import {TransformControls} from "/node_modules/three/examples
    /jsm/controls/TransformControls.js";
import Stats from
    "/node_modules/three/examples/jsm/libs/stats.module";

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

const camera = new THREE.PerspectiveCamera(
    75,
    window.innerWidth / window.innerHeight,
    0.1,
    1000
)
camera.position.z = 2;

const renderer = new THREE.WebGLRenderer();
renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);

const geometry = new THREE.BoxGeometry();
const material = new THREE.MeshNormalMaterial({
    transparent : true });

const cube = new THREE.Mesh(geometry, material);
scene.add(cube);
const orbitControls = new OrbitControls(camera,
    renderer.domElement);
const transformControls = new TransformControls(camera,
    renderer.domElement);
transformControls.attach(cube);
transformControls.setMode('rotate');
scene.add(transformControls);
```

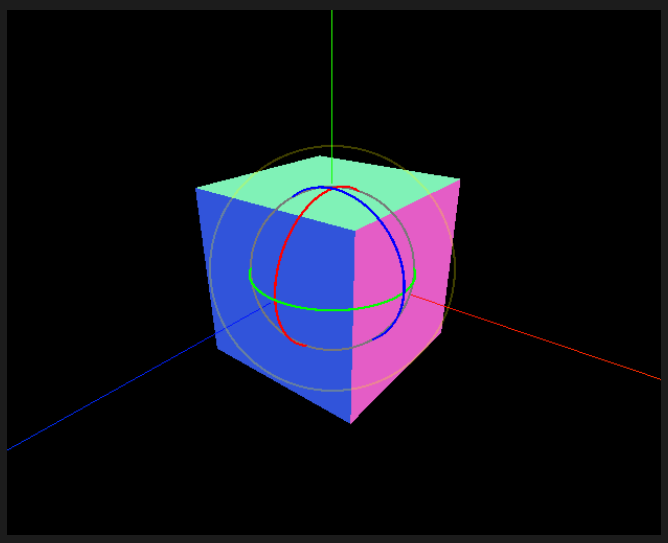
```
transformControls.addEventListener('dragging-changed',
     function (event) {
          orbitControls.enabled = !event.value;
     })
window.addEventListener('keydown', function (event) {
    switch (event.key) {
        case 't':
            transformControls.setMode('translate');
            break
        case 'r':
            transformControls.setMode('rotate');
            break
        case 's':
            transformControls.setMode('scale');
            Break;
     })

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();
```

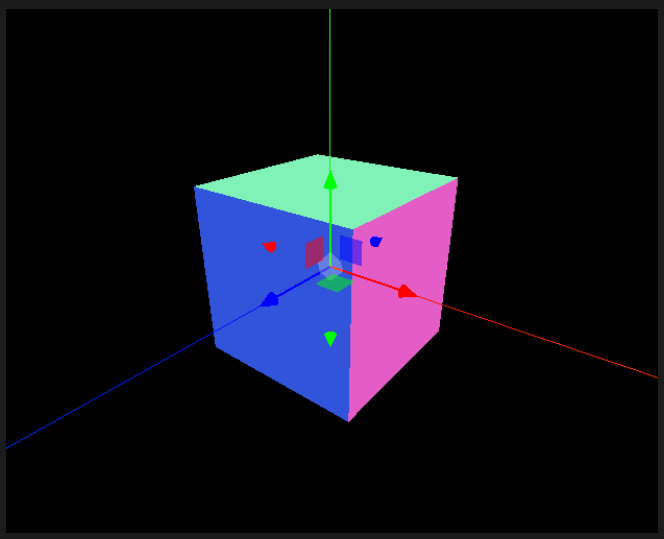
Now if we save the code, and enter the following command in the terminal:

```
npm run dev
```

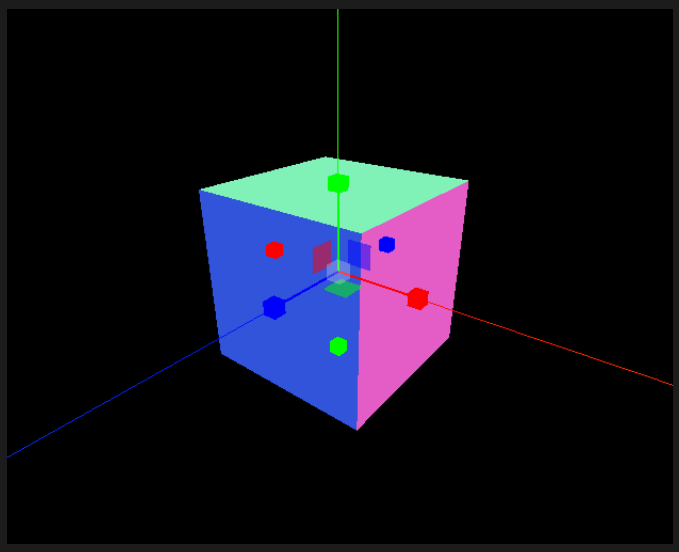
We should see the following result:



You can rotate the cube if you click on the blue, red or green rings and move them. If you press the “t” on the keyboard, you will see:



You can move the object using the arrows given. And if you press the “s” button, you will be able to see:



You can scale the object in all the directions. Notice that in all three states you can move around the object and have orbit controls, Which will give you a perfect view.

EXPLAINING THE CODE:

The first thing we did at the beginning was to import the necessary libraries for the orbit controls, and the transform controls in addition to other dependency libraries. Then, we defined the scene, camera, renderer, geometry, and material just as we did in our recent tutorials on Three JS. Afterward, we defined the transform control and added the cube to it to be able to apply them to the object. We also defined the orbit controls. To activate each one of the transform controls (rotation, translation, and scaling), we added an event listener and used the switch case statement for each of the transform controls. Moreover, we added another event listener for the orbit controls so that the user can easily interact with the scene. In the end, we animated the object using the animation function.

CONCLUSION

In this tutorial, we implemented the transform controls, including the rotation, translation, and scaling, in addition to orbit controls which give the user the ability to move around the object and see it from all aspects. Using such a script, you can easily create a user-interactive 3D design application or even a game. As you know, one of the essential tools for every 3D user-interactive app or game is orbit controls and transform controls. Transform control lets the user rotate objects, and move and scale them in Three JS.

We hope this tutorial has helped you solve the challenge that you have had in Three JS. We have covered this tutorial in a video so that you can understand this article more deeply.

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.