Implementing OrbitControls into your Threejs/Ammojs project

Modules

Orbit controls allow the camera to orbit around a target in your Threejs scene. This guide assumes that you have Threejs downloaded (Ammojs optional) and have an already existing scene set up.

Before following this guide, you must ensure you know whether you are already using JavaScript ES6 modules in your html code or not.

If you are not sure if you are using modules it is very straightforward to check. If any of your script tags contain the statement: type = “module” then you are using modules. This is because to use a script file as a module in html (so import and export statements can be used), the file must be set as type module.

Module Threejs example

```
<script type="module" src="./js/ball.js"></script>
```

And the first line in ball.js says:

```
import * as THREE from './three.module.js';
```

Non-Module Threejs Example

```
<script src="./js/three.js"></script>
```

Once you have figured out if you are using modules or not, you should know that OrbitControls.js has a module version and a non-module version. The module version uses an import statement to import certain Threejs classes whereas the non-module version doesn’t.

Module version - three\examples\jsm\controls\OrbitControls.js
Non-Module version - three\examples\js\controls\OrbitControls.js

Steps if you are not using JavaScript ES6 modules

1. Ensure that you have Threejs and the **example folder** downloaded. You can download the Threejs Master from GitHub here.

2. If you **are not** using javascript modules then you should plan where you want to move the OrbitControls.js file into (eg. with Threejs and Ammojs). Next, you should navigate to the directory specified below and move the OrbitControls.js file to your desired location

   three\examples\js\controls\OrbitControls.js

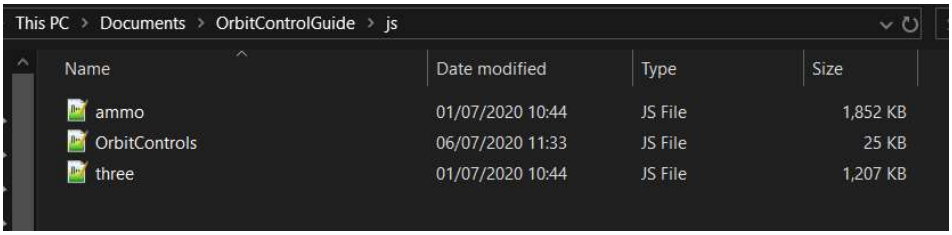

Figure 1 - Example showing where I moved OrbitControls.js to

3. Once you have OrbitControls.js in your desired location, you are ready to implement it into your code. Firstly, call OrbitControls in your html file using script tags. For me, the html file is one folder above the "js" folder, so my script tags looked as shown below:

   <script src="./js/OrbitControls.js"></script>

   You may have to change the relative src of the orbit controls file depending on where you moved it to. The source is relative to the current file, which in this case is the html file.

   ./ means relative to the same folder as the current file
   ../ means relative to the folder above the current file
   ../../ means relative to two folders above the current file

4. Next, decide whether you create a separate method for setting up the orbit controls (eg. setUpOrbitControls()) and add the lines or if you will just add in the couple of lines in with the code where you set up the lighting and renderer, etc. Depending on your decision, either:
   - Add the following two lines of code to the setUpOrbitControls() function you made and call this function **just after** where you call the function containing the renderer
   - Or add the following two lines of code to your script **just after** the code where you set up the renderer

   ```
   controls = new THREE.OrbitControls( camera, renderer.domElement );
   controls.update();
   ```

Steps if you are using JavaScript ES6 modules

1. Ensure that you have Threejs and the **example folder** downloaded. You can download the Threejs Master from GitHub here. In this case, I have all my main JavaScript code in a file called ball.js, instead of writing all the JavaScript code in between script tags. In the html file, in this example, my only use of Script tags is to call the JavaScript file, ball.js. I'll be importing Threejs and OrbitControls classes into the ball.js file.

2. If you **are** using javascript modules then you should plan where you want to move the OrbitControls.js file into (eg. with three.module.js). Next, you should navigate to the directory specified below and move the OrbitControls.js file to your desired location.

   three\examples\jsm\controls\OrbitControls.js

   If you do not already have the Threejs module then you can move it from the directory below

   three\build\three.module.js

   I placed the ball.js, three.module.js, and the OrbitControls.js modules into the same folder

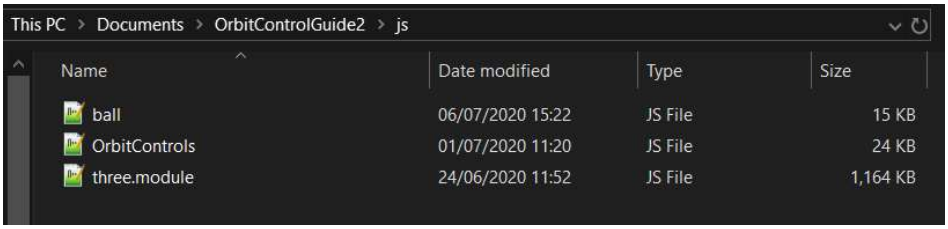

Figure 2 - Example showing where I moved OrbitControls.js and three.module.js to

3. After moving OrbitControls, the import statement (for the Threejs classes) at the top of the OrbitControls.js file will need a quick edit. By default, the import location is as shown below:

```
18 } from "../../../build/three.module.js";
```

   The location of the import must be updated to where the three.module.js file is, relative to the OrbitControls.js file. For example, in figure 2, OrbitControls and three.module.js are moved into the same folder so the new line 18 will be as shown below:

```
18 } from "./three.module.js";
```

   You may have to change the relative src of the import location to something different depending on where you moved it to. The source is relative to the current file, which in this case is the OrbitControls.js file.

./ means start looking from the same folder as the current file
../ means start looking from the folder above the current file
../../ means start looking from two folders above the current file

4. Once you have OrbitControls.js in your desired location and have edited the import statement, you are ready to add some code. Assuming you have imported Threejs on the first line, import OrbitControls into your script by adding the following line just below it:

```
import { OrbitControls } from './OrbitControls.js'
```

You may have to change the relative src of the orbit controls file depending on where you moved it to. The source is relative to the current file, which in this case is the ball.js file.

5. Next, decide whether you create a separate method for setting up the orbit controls (eg. setUpOrbitControls()) and add the lines or if you will just add in the couple of lines in with the code where you set up the lighting and renderer, etc. Depending on your decision, either:
    - Add the following two lines of code to the setUpOrbitControls() function you made and call this function **just after** where you call the function containing the renderer
    - Or add the following two lines of code to your script **just after** the code where you set up the renderer

```
controls = new THREE.OrbitControls( camera, renderer.domElement );
controls.update();
```

Getting the OrbitControls target to adjust automatically

By default, the orbit controls rotate about the point (0, 0, 0). Ideally, the orbit controls should orbit around the vertical center of the desired object. To set it up so that the target of an object is automatically calculated, add the following lines of code **just before you add the object to the scene**. (This is assuming that you have set up the orbit controls earlier in the code)

```
// create your object ( mesh = new THREE...)

var box = new THREE.Box3().setFromObject( mesh );

centerHeight = (box.max.y-box.min.y) / 2;
centerPosition = box.max.y - centerHeight;

controls.target.set(0, centerPosition, 0);
controls.update();

// add object to the scene ( scene.add(mesh) )
```

Tips for OrbitControls

Orbit controls use the up and down arrow keys to zoom in and out slowly, which can be a problem if you want to use the arrow keys for something else.

If you go down to line 76 in OrbitControls.js you can disable the use of zooming with the arrow keys by setting the variable enableKeys to false