Notes on THREE.js webGL interface

So now you want to learn how to create cool 3D objects in a web browser? Well, you can always use webGL directly. But that’s a nightmare and would take you hundreds of lines for one piece of cube. You aren’t going to go and create an API yourself, are you? This is where THREE comes in. THREE is an API for webGL that allows for easy creation of 3d geometry. So, let’s get started:

Script

Everything goes in <script></script> headings when working in THREE.

Get Started

You need three things to display things in THREE:

1. Scene
2. Camera
3. Renderer

var scene = new THREE.Scene();

var camera = new THREE.PerspectiveCamera( 75, window.innerWidth / window.innerHeight, 0.1, 1000 );

var renderer = new THREE.WebGLRenderer();

renderer.setSize( window.innerWidth, window.innerHeight ); document.body.appendChild( renderer.domElement );

Camera

There are three types of cameras:

1. Perspective Camera: The first attribute is the Field of View. The second attribute is the aspect ratio. You almost AWLAYS want width of the element divided by height. Next two are near/far clipping.

Renderer

There are a few renderers available for THREE:

1. WebGLRenderer: You need to set the size at which to render at. If you render at the width/height of your browser, its crips and clean. If you render it smaller than that, it will look blurry but be less processor intensive.

Mesh

A mesh is what the user will see, and it is what is rendered. Alone, a mesh does nothing, it needs two things along with it to work: A geometry and a material. The mesh will take the material and apply it to the geometry and then allow it to be added to the scene.

Geometry

There are a bunch of geometries in THREE. All of them take their x,y,z size as constructor values.

1. Cube Geometry

Adding to a scene

Use scene.add() to add something to your scene. They start at position 0,0,0, so you might want to move some of them around after adding them.

Rendering the scene

To render a scene, you need a RENDER LOOP.

function render() {

requestAnimationFrame(render);

renderer.render(scene, camera);

}

render();

Moving things

Anything that you want to be moved has to go IN your render loop!

Each mesh has a bunch of methods for moving things, check the mesh docs for what methods there are to move around with.