From: **Jason Hansen** jason_hansen@outlook.com
Subject: Re: Github Repository - ThreeJS
Date: June 4, 2019 at 3:12 PM
To: jason brown jason@pendulumswingmedia.com

Hello senior fellow Secrist alumni!

I did lots of damage! Check out the "webpack" branch:

https://github.com/pendulumswing/ThreeJS_Experiment/tree/webpack

You can also see a version of it running in production here:

https://threejsexperiment.jchansen.now.sh/

To run the project, do these steps:

1. Check out the branch
2. Delete node_modules (rm -rf node_modules)
3. Run "npm install"
4. Run "npm start"

This will make the project available on localhost:300 in the browser.

To deploy it, run "npm run deploy:production". That will build the project, place the compiled assets in a "dist" folder at the root of the project, and then attempt to deploy that folder to Now (now.sh). If you try to run it, it will likely ask you to log in or create an account or something, but once you do, it'll end up deploying the project to a URL like "https://threejsexperiment.pendulumswing.now.sh".

Quick notes about what I did:

1. I added webpack, and broke up the project into a bunch of smaller files that can be imported

2. I'm using the ES6 syntax (import * as THREE from 'three') and not ES5 syntax (const THREE = require('three'))

3. You'll notice the index.html file at the project root has no CSS or JS files explicitly imported. They're inserted dynamically (at build time) by Webpack. The process also allows them to get unique hashes in their names which serve as a cache-breaking strategy (preventing browsers from loading old files when you update the code)

4. The index.js file at the root is the entry point for the application. It's the file that gets loaded first, and then it decides what happens next. In this case, I'm importing the styles

loaded first, and then it decides what happens next. In this case, I'm importing the styles, optionally performing a WebGL check, and then importing "src/main.js" which is your main rendering code.

5. When all the code was together, it was easy for classes (Spring and Base) and some or the helper functions (makeInstance) to reference variables (scene, objects) outside their scope (meaning they were passes as values to function/constructor). In some cases (like with Base) I passed some of those values as an argument to the constructor (bond_Geo, bond_Color) but in other instances (scene, objects) I got lazy and just attached them to window (window.scene = scene) which allows them to be access as a global variable. It's generally frowned as a coding practice, but it's a legit tactic you can use when refactoring code to keep it working while changing code structure and interfaces.

Anywho, let's figure out a day/time to tag up and chat. I'm sure parts of the setup are going to be confusing. Webpack is not at all easy to comprehend, but once you figure it out once (or just steal someone else's configuration file) you rarely have to change it. So I figured out how I wanted to use it maybe ~2 years ago and have rarely had to edit my configuration.

Cheers,
Jason

From: jason brown <jason@pendulumswingmedia.com>
Sent: Monday, May 27, 2019 9:57 AM
To: Jason Hansen
Subject: Github Repository - ThreeJS

Helloi fellow Secrist alumni,

First of all, thanks again for meeting up on Saturday. I enjoyed the conversation we had. The wide world of web dev has always seemed so large and fragmented that I'm usually just intimidated
by all the "potential". However, it feels like I'm slowly peeling back the layers and starting to get an understanding of (at least some) things.

I set up a new repository with the ThreeJS project we looked at on Saturday. I've added you as a contributor. Here is a link just in case:

https://github.com/pendulumswing/ThreeJS_Experiment/invitations

The repository is just a copy of my local file so feel free to do as much damage as you like:) I think the only "real" dependencies in there right now are:

Index.html
css/style.css
js/script.js
build/three.min.js

The rest is imported examples and npm modules that may or may not be useful. We could ditch those until needed.

In any case, don't spend too much time with this. I think doing a few practice commits, branching, merging, etc to see what it's like to modify a project team-style would be helpful. I appreciate any help you can give with this.