UWaterloo EngSoc
WebGL Workshops
Fall 2017

Prepared by Deniza Malataeva
Last updated on March 11, 2018

Environment Setup:

For the WebGL Workshop the only three things we will require are Git, Node and a suitable Javascript IDE (program to edit your code). You can use notepad or vim if you're confident, up to you! There will not be any IDE listed for installation, because this is a personal preference. You can easily Google and download popular IDEs for free.

If you have the required setup, you can skip to the next section below!

If you are using a…

Mac:

1. Download and install Git (Select Mac OSX from given options)
   https://git-scm.com/downloads
2. Download and install Node (Select Mac OSX installer)
   https://nodejs.org/en/download/
3. Open Terminal, navigate to your intended folder using 'cd' and 'ls'
4. Enter command 'git clone https://github.com/dmalataeva/uw-workshops'

Windows:

1. Download and install Git (Select Windows from given options)
   https://git-scm.com/downloads
   Make sure you select to install Git bash - this is the command line interface (CLI) we will be using primarily!
2. Download and install Node (Select Windows installer)
   https://nodejs.org/en/download/
3. Open Git Bash, navigate to your intended folder using 'cd' and 'ls'
4. Enter command 'git clone https://github.com/dmalataeva/uw-workshops'
   ALTERNATIVELY, if the repository takes too long to clone, you can download the repository from this Google Drive folder.

Final step:
Try running ‘git status’’ after the repository is cloned, does it say ‘up to date’? If it does, you are good to go!

Repo Build Instructions:

There are two existing webpack commands for this repository:
npm run build - transpiles and outputs your code in *src/js/script.js* into *src/js/webgl.js*. Transpiling your code is necessary because there are a couple of different standards for Javascript syntax, which the JS engines in browsers may or may not understand. Transpiling the script to an older, more stable standard ensures compatibility.
npm run dev - watches for changes and automatically transpiles/builds your code, i.e. looping version of ‘npm run dev’.

First, navigate to ‘../uw-workshops/webgl-workshop’ and run npm install to download all dependencies needed for your project. (if you’re curious, you can find all your dependencies listed in the *package.json* file)

There is also an additional script we will be using once we get to using images:
Run npm run server in ‘../uw-workshops/webgl-workshop’ to run a mini-server which will let webGL use your image files (this is needed to overcome the browser’s major security issue - the CORS policy).

Once you have successfully installed all dependencies, open two different bash windows, navigate to ‘../uw-workshops/webgl-workshop’ and run npm run dev and npm run server in both separately. While these are running, you can keep coding and simply refresh the browser window to see changes. The Terminal (or Git Bash) will give you very useful information if it encounters a simple bug in your code.

You can also view errors in the developer console of your browser.

For **Chrome**, go to View->Developer->Javascript Console.
For **IE**, press F12-> CTRL + 2
For **Firefox**, press CTRL + Shift + K on Windows, or CTRL + Command + K on Mac OSX.

For other browsers, you can easily google how to do it and follow through.

Keep in mind that if your console is open in the browser, you may have to select ‘hard reset’ or ‘clear cache’ every time you refresh your page to see changes.

File Information:

Your */src* folder contains everything that is needed for the project to work. The folder */examples* contains .js files with examples of how to implement various things that we will go over. You can keep this folder as a reference in the future. The */docs* folder contains reading material, such as this document, as well as the presentation used at the event. The */media* folder contains a zip of all the media files we will be using. The */node_modules* folder that has appeared after you added all the needed dependencies can be disregarded. It simply contains packages that are used for managing the Node side of the project. Note that three.js is excluded from the general dependencies. This is for anyone that wants to read the source code and find out how Three.js works. **index.html is already configured to include all the files that are needed**, so you can start writing WebGL code right away.

There are two options to where you can write your code. The **first option** is the file */src/js/script.js*. The webpack transpiler is configured to read that file, and output the transpiled code to /src/js/webgl.js. You can use this option if you want to keep all your scripts separated from your HTML file. It tends to get messy when you add more than just webGL to your webpage. The **second option** is writing directly into the HTML file. You will find a <script> tag in the body that will tell you where to place the code. This option allows you to have everything in one file. If you go with this option, then you do not need to run the webpack transpiler, but you have to ensure you are using the ES2015 standard for things other than the three.js library. For more information on compatibility, refer to this chart [here](#).

At last but not least - your friend in writing good webGL code will be the documentation for https://threejs.org/. It is updated by its creator often enough. Refer to it for examples, specifications and FAQ. Or ask us!