Environment Setup:

For the WebGL Workshop the only two things we will require are Git and Node. If you have these two set up, you can skip to the pink 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, you can just download the repository from this Google Drive: X

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 script commands for this repository:
npm run build - transpiles and outputs your code in *src/script.js* into *src/public/webgl.js*
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 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 browsers’ CORS policy).

Once you have successfully installed all dependencies, run npm run dev and npm run server in two different bash windows. While these are running, you can keep coding and simply refreshing the browser window to see changes, and 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, google ‘how to open console on <YOUR BROWSER>’ and follow steps.
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 the folders /examples and /public, as well as files script.js, three.js, ColladaLoader.js and OrbitControls.js. The last two are written by various contributors to three.js and are used in this workshop to introduce user controls and import Collada files. **index.html is already configured to recognize those two files**, so you can use them right away.

The file *script.js* is where you will be writing all of your code. **All of it**. Again, index.html is **already configured** to use all the necessary dependencies to output graphics. All you have to do is worry about your actual webGL code :) three.js is a file that contains a somewhat minimized version of the Three.js library. Our code cannot work without this library, please do not modify it.

/examples contains various things that we will be going through for our workshop. You can always use its contents as a 'cheat-sheet' for your own projects.
/public contains media files and everything we will use to put our graphics on the web. The only thing that is worth mentioning is the media.js file. It is a simple JSON organization of all the paths to media files in the repository. This is not necessary for webGL, but is highly recommended for projects so you are able to locate media files dynamically.

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