CPSC 314 **Assignment 1** Due Fri Jan. 19, 2018

Introduction to Three.js, and GLSL shaders (5% of final grade)

In this assignment you will gain experience with Three.js and GLSL shaders. Specific outcomes include the ability to:
- run WebGL code based on Three.js
- debug basic javascript errors and shader errors
- modify and build basic 3D scenes using existing object classes in Three.js
- define your own 3D object from vertices and faces
- make basic modifications to a fragment shader

Copy and expand the zip file in a local "cs314" directory for this assignment:

```
cp a1.zip ~/cs314/a1.zip
cd ~/cs314
unzip a1.zip
cd a1
```

View your local version of `a1.html` to ensure that your web browser is properly enabled to run Javascript and WebGL. It should be an up-to-date HTML5 browser. Chrome, Firefox, and Safari all support WebGL. If you get a blank image when launching `a1.html`, the problem is likely that your browswer has not been enabled to access local files. This is unlikely to be a problem in the ICCS x005 lab. For your own machine, the solution depends on your OS and your browser. See here for the details of how to address this access problem: `https://threejs.org/docs/index.html#manual/introduction/How-to-run-thing-locally`. The solutions listed under "Change local files security policy" are perhaps the easiest to work with.
Now walk through the following steps, and implement the requested changes. After each edit to `a1.html`, `a1.js`, or to a GLSL shader file, you will want to reload `a1.html` in order to see the changed result. When introducting errors, fix the error before moving on to the next step. All other changes can be made in a cumulative fashion.

1. Introduction to Three.js and Shaders (12 points total)

   Parts (a)-(h), (m), (n), are worth 3 points taken altogether. There is nothing to hand-in here, but you should be prepared to discuss these with your TA.

   (a) Run the code by launching `a1.html`. Press Ctrl+Shift+J (Windows / Linux) or Cmd+Opt+J (Mac). This should open the console window. Look for the "hello world" message. Change the code so that the console message reads "Assignment 1 (FOO)", where FOO is your name.

   (b) Attempt to print the result of a division by zero. What happens?

   (c) Attempt to use a variable name that does not exist yet. What happens?

   (d) Add a new variable using "var foo;" and then print it's value without first initializing it. What happens?

(e) Remove the terminating semicolon from one of the early lines in a1.js. What happens? Why? Do an internet search on "use of semicolon in javascript" to understand why you should still use terminating semicolons.

(f) Insert the following code at the beginning of a1.js. What are the resulting values of a and b? Why?

```
a=4;
b=5;
function go() {
  var a = 14;
  b = 15;
}
go();
console.log('a=',a,'b=',b);
```

(g) Change the background colour to be a light-blue "sky" colour.

(h) Know how to orbit, pan, and zoom in the scene. This is done via left-mouse-drag, right-mouse-drag, and mouse-wheel. On a Mac trackpad: click-drag, two-finger click-drag, two-finger drag, respectively.

(i) (1 point) Change the custom object, which is currently a white square, to be an orange pyramid with a square base that is parallel to the ground plane. Modify the properties of the material as needed.

(j) (1 point) Create an instance of a second torus. Change the orientation so that it is parallel to the ground. Change the position so that the first torus and second torus are linked together, like links in a chain.

(k) (1 point) Build a twisting stack of three cubes, coloured while, green, and yellow (from bottom to top).

(l) (1 point) The light source, shown as a red sphere, can currently be moved using the W,A,S,D keys. Change the code so that the movement of the light source is bounded to $x, y \in [-5, 5]$. Also make the light source yellow.

(m) The Armadillo has its own vertex shader and fragment shader, stored in `glsl/armadillo.vs.glsl` and `glsl/armadillo.fs.glsl`, respectively. Here, we will only be experimenting with making a few small changes to the fragment shader. First, remove a semicolon from one of the fragment shader statements. What happens? Look at the start of the console log for any useful information. Add the semicolon back in.

(n) Save a copy of the key line in the current fragment shader, i.e., `gl_FragColor = ...`. Now change the fragment shader to shade all pixels of the Armadillo green. Preserve this change as a comment in your shader.

(o) (2 points) Now we will change the fragment shader to compute the lighting according to light coming from a fixed direction. In the shader, define a lighting direction, L, that points towards the light, as a vec3 having a value of (0.0,0.0,-1.0). Note that that the decimal values are important here, because without them, the numbers 0 and 1 would be interpreted as integers by the GLSL compiler. Now define a float, i, that will model the intensity or brightness of the surface as the dot product between N and L. This defines a basic diffuse lighting model. Note that GLSL provides a function `dot(`a`,`b`)` that computes the dot product between two vectors a and b. Assign the computed intensity, i, to each of the red, green, and blue channels of the final fragment. This should produce a grey-shaded armadillo.

(p) **(3 points) Creative Component:** Make several further changes and additions to your scene to make it interesting. The most compelling scenes will be shown in class. You are free to experiment with a variety of features. Give appropriate attribution if you borrow from example three.js code online.

Submit your assignment by the deadline using:

```
handin cs314 a1
```

Show your demo to a TA in your lab section, or during the extra lab hours (to be arranged). You should be able to answer questions related to the various experiments you performed above.