Homework 1 – Preliminary Reading

Step 1

Download and compile the first example project.

a) Download the Homework 1 starter project.

b) Unzip the project, open the Microsoft VisualStudio 2013 solution file (.sln), and compile the program. The assumption here is that you are using Windows operating system (if using other OS such as Linux/Ubuntu/Mac, click here). The Visual Studio versions later than 2013 should work fine after giving a warning message. Once the program compiles and runs (CTRL+F5), you should see a 3D teapot.

Step 2

The terminology below will help you understand the important parts of this example project:

1) **SDL.** Cross-platform development library to provide low level access to audio, keyboard, mouse, joystick, and graphics hardware via OpenGL and Direct3D. Why do we need SDL?

   a) To create graphics with OpenGL we need to create an OpenGL context and an application window to draw in, but these are operating system dependent operations

   b) OpenGL purposefully abstracts these operations, meaning that we have to create a window, define a context, and handle user input by ourselves (hard)

   c) SDL, and other libraries like GLUT or GLFW, exist to perform operating-system specific work and give us a window and OpenGL context to render into

2) **OpenGL.** This is your API to create any virtual environment in this class. OpenGL is somewhat low-level, but it will help you to understand all the important concepts of computer graphics, 3D rendering, and virtual reality. For now, we will focus on drawing simple 3D objects and transforming them. a) Good OpenGL reference: http://learnopengl.com/

3) **GLEW.** Since OpenGL is a standard/specification it is up to the driver manufacturer to implement the specification to a driver that the specific graphics card supports. Since there are many different versions of OpenGL drivers, the function locations are not known at compile-time but at run-time. Therefore, using OpenGL function requires retrieving and storing function pointers before use. This becomes a tedious task to do for every function that hasn't been declared yet. Libraries exist to perform these tasks. We will be using GLEW, which is the most popular and up-to-date library.

```
// Define the function's prototype
typedef void (*GL_GENBUFFERS) (GLsizei, GLuint*);
// Find the function and assign it to a function pointer
GL_GENBUFFERS glGenBuffers  = (GL_GENBUFFERS)wglGetProcAddress("glGenBuffers");
// Function can now be called as normal
GLuint buffer;
glGenBuffers(1, &buffer);
```

Figure 1. Retrieving function pointer in Windows

4) **GLM.** This is your API for all matrix transforms. Use the functions glm:rotate, glm::translate, glm::scale, glm::lookat, and glm::perspective to create and concatenate all your transforms (and potentially invert them). It is a pretty self-explanatory library. There are plenty of examples online of how to use it. Also most of the commands that you will need to call are already in the starter code.