Assignment 0 – Graphics Warmup

Due: September 17th (4:00pm)
Check-in: September 12th

Overview: This assignment is intended to be a hands-on introduction to real-time graphics programming. The goal is to write a very simple application that uses many of the basic features of the OpenGL API, and some simple interaction with what is drawn on the screen. The application will draw a single, textured square in the center of a window, and should allow a user to resize, move and rotate this square.

Check-in: This assignment has a mandatory mid-way check-in. For this check-in, you must turn in a webpage with an image of the working, compiled starter code and showcase the assignment's "basic features" working.

Basic Features (60 points):

- Installing all libraries, and compiling the starter code.
- Drawing a square that moves when the user clicks and drags their mouse.

Required Features (35 points):

(5) Translating only when the user drags on the square's interior (but not on the edge or outside the square)
(5) Scaling the square when the user drags on the edges
(10) Rotating the square when the user drags on the corners
(10) Texture the square with an image loaded from a PPM file
(5) Brighten or darken the image displayed on the texture

Extra Features:

(5) Changing something visually when the user presses the keyboard
(5) Supporting a triangle (in addition to a square)
(5) Graphical indicator of what motion is being performed.
(5) Animating the square (without needing user input)
(5) Multiple squares
(5) Maintaining the square's shape when the window aspect ratio changes

Submission Details:

Submission should be in the form of a link to a webpage. The web page must contain an image of your program in progress. A short description (~1 paragraph) of any difficulties you encountered. A zip file with all the code needed to compile your project and a working executable (note which OS it is for).

Optionally, you may submit an image of an artistically interesting image resulting from your work. Sufficiently "interesting" pictures will be awarded 5 points, the best in the class 10 additional points.

Points over 100 will be applied, at a discount, only if the required features work.

Notes & Tips:

- I understand this assignment is somewhat open-ended or vague at times. My main goal is that you get some experience dealing with the issues involved in setting up a graphics project, while starting to think about how to incorporate creativity with programming. For all your assignments in this class, I hope you focus as much on making the results fun and interesting, as on satisfying the exact details of the specification.

- Rotation is much more difficult than the rest of the assignment. Get everything else working first, ignoring rotation! Rotation will complicate both how you draw the square and how you detect which operation to perform. It is better to turn in the code with everything working without rotation than to get stressed out or hung up trying to make rotation work and not get to the rest of the assignment.

- I know it might have been a while since you last did file input and output in C++. Here is a simple example that opens the PPM file, and prints out all the information:

```
ifstream ppmFile;
ppmFile.open("test.ppm");
string ppmType;
int w, h, maxVal, red, green, blue;
ppmFile >> ppmType;
ppmFile >> w >> h >> maxVal;
printf("PPM Type: %s. Image size %d x %d. Max Val: %d.\n",
       ppmType.c_str(), w, h, maxVal);
int pixelNum = 0;
while (ppmFile >> red >> green >> blue){
       printf("Pixel# %5d, R: %d  G: %d  B: %d \n",
              pixelNum, red, green, blue);
       pixelNum++;
}
```

This is just one example. It's important to remember that there are many ways of doing file IO in C++, feel free to use a different approach if you are more comfortable with that.

-You can convert your own images to PPMs, just make sure they use the same format we discussed in class (or make sure you update your parser to handle the other formats!). In Linux and OS X I recommend installing ImageMagick, you can then covert any image to a PPM using the following command:

```
> convert -compress none example.jpg example.ppm
```

Getting Started:
You will need the starter code, which is available from the course webpage. To complete this assignment, you will also need to install development libraries for SDL2 and OpenGL, and should update all your relevant graphics and display drivers.

Note--these instructions may have issues with any given machine. Please be sure to check the class forums or email the course staff if you get stuck.

Installing SDL2, OpenGL:

OS X:
1. Download Xcode to get GCC/LLVM and other development tools

2. Download and install SDL2: https://www.libsdl.org/download-2.0.php
(e.g., SDL2-2.0.8.dmg)
Note: I placed the file in `/Library/Frameworks`, you can put it on other directories, just make sure it matches the –F parameter when you compile.

3. OpenGL should be already be installed as part of the OS. However, please make sure you have updated to the latest OS release that your computer supports. See: https://www.khronos.org/opengl/wiki/Getting_Started

Linux:
1. GCC is likely already installed.

2. You will need to install the SDL2 development libraries package: `libsdl2-dev`
e.g., `sudo apt-get install libsdl2-dev`

3. Parts of OpenGL likely come with your distribution, but you'll often need to installed additional development packges (e.g., with apt-get). Typically:
libgl1-mesa-dev and mesa-common-dev.
e.g., `sudo apt-get install libgl1-mesa-dev mesa-common-dev`

Windows:
1. You should install MS Visual Studio.

2. Download and install the SDL2 development libraries from:
https://www.libsdl.org/download-2.0.php
Remember to ***get the development libraries***, which will include the header files and the .lib files.

3. OpenGL is mostly included in windows, but be sure to install the latest drivers for your graphics card (see https://www.khronos.org/opengl/wiki/Getting_Started).

Compiling and running the starter code:
On OSX or Linux, be sure to specify that you need to link against SDL2 and OpenGL. You need to include both the code for Square.cpp *and the code for glad.c* when compiling.

OS x:

```
g++ Square.cpp glad/glad.c -F/Library/Frameworks -framework SDL2
-framework OpenGL -o square
```

Linux:

```
g++ Square.cpp glad/glad.c -lSDL2 -lSDL2main -lGL -ldl
  -I/usr/include/SDL2/ -o square
```

Your actual command line call may vary. Typing `locate SDL.h` will tell you where the SDL.h file is located, and you can update the directory after –I to match.

Getting comfortable with Make (or CMake) can help support you as you move to compiling large projects with multiple files, libraries and dependencies.

Windows:
Be sure to include both `SDL2` and `SDL2main` libraries when linking. Be sure to include both `Square.cpp` and `glad/glad.c` when compiling.