CSE 167 (WI 2021) Homework 0

Homework 0 is here to make sure that you can compile and run modern OpenGL (required for HWs 1 and 2).

0.1 Compiling OpenGL Programs

In principle, OpenGL and related code should be portable across many platforms. In the past CSE 167, there was almost nobody having difficulty getting a working setup across a wide range of machines. So you may use any computer and platform you choose, but will need some kind of C++ development framework and an installation of OpenGL and GLUT.

In practice, there might still be different installation tricks to get your machine set up to run the programs properly depending on the operating systems. Fortunately we have a detailed FAQ for a variety of different systems.

Background

OpenGL is a specification of a list of functions (and constants) that can command your graphics card. That is, OpenGL an abstract graphics application programming interface (API). The manufacturer of the graphics card in your machine has implemented these functions, whose binaries (as the graphics card driver) are installed somewhere in your machine. (In particular OpenGL has nothing to do with open source.)

In order to access the OpenGL functions, we need the pointers to those functions. This task would be difficult and platform-dependent. Fortunately, there are plenty of *OpenGL loaders*. An OpenGL loader is a library that loads pointers to OpenGL functions at runtime. **GLEW** (OpenGL Extension Wrangler) is one of them. (GL3W is another commonly seen OpenGL loader.) Basically these loaders are just some giant header that replaces all the OpenGL function calls in your code by their automatic function pointer finder.

While the core OpenGL consists of functions about manipulating the state of the rendering pipeline (read/write the memory on the graphics card or use customized programs running on the GPU), a companion **GLUT** (OpenGL Utility Toolkit) consists of functions that perform input/output communications with the host operating system. For example GLUT has functions monitoring keyboard and mouse activities as well as creating windows in a cross-platform fashion. You can imagine GLUT is also quite essential for writing an OpenGL program just to visualize anything. (GLUT is also not open source. The original developer Mark Kilgard maintains the copyright.)

GLUT includes the GLU library, which includes useful helpers such as producing the transformation and projection matrix given camera positions etc. Since learning how to produce these matrices is one of the learning goal, for our purpose many of these GLU functions are just for comparing results (between our implementations and the GLU ones).

GLM (OpenGL Mathematics), on the other hand, is just a standalone open source header-only library. It defines useful matrix-vector operations. We have included the zip file and unpacked versions in the skeletons.

FreeImage is a library for writing image files (jpg, png,...).

0.2 Assignment

Exercise 0.1 Download the skeleton code. The program has a textured ground plane with 4 pillars and a teapot with lighting that moves.

- One Windows, you can run the program by opening the `.sln` file and pressing F5. (Please use your C++ development environment for running, by pressing F5 as mentioned, rather than double-clicking the executable in the folder.)
- On Mac OSX or Linux, you should use the command line (*e.g.* via terminal) to `cd` to the directory containing the files, type `make` and then `./mytest3`. (Please do not double click on the executable file using finder in MacOS; that will lead to errors, since the paths would not be set up properly.)

Please also note that the correct way to exit is to hit the esc key in the program.

The mouse can be used to zoom in. Look at the *keyboard* function in `mytest3.cpp` to see the keys you can press. The p key will start and stop the animation of the teapot. ▪

Exercise 0.2 Once you have the program running successfully, press i to move the teapot into the correct position. Next, press o to output the screenshot to the directory.

Rename it to "`screenshot1.png`" so it isn't overwritten by a subsequent screenshot. Please make sure you do not zoom in or out prior to pressing i or o (if you have earlier played with the mouse, it is best to restart the program). ▪

Exercise 0.3 Next, change the color of the red-orange highlight (RGBA=(1,0.5,0,1)) on the teapot to yellow (RGBA=(1,1,0,1)).[a] The relevant colors and code are defined in the "`display`" function of "`mytest3.cpp`" where it says "add lighting effects."

Once you have changed the color of the highlight from red-orange to yellow, recompile, run and press i then o. Rename this screenshot to "`screenshot2.png`".

[4] RGBA stands for red, green, blue and alpha. Alpha is the opacity. In the additive color of light, yellow is made by mixing red and green, and thus RGBA=(1,1,0,1).

Exercise 0.4 Upload the two screenshots to Gradescope HW0 to show us that everything is working.