CS 184: Assignment 0—Account Setup and Compilation

Ravi Ramamoorthi

This assignment gets all the groundwork ready so you can stop worrying about logistical issues and focus on the content in CS 184. In particular, you will set up your account and make sure you know how to compile and run OpenGL programs with shaders. You will also make use of the *submit* program. Please coordinate with the teaching assistants if you run into any issues.

Please also use the newsgroup we have created for the class. It is likely any problem you face will be shared with other students, and any answers and solutions you find are useful to all. Since there is no "test" in this assignment, feel free to post code and other suggestions that are useful for compilation (of course, do not post any code relevant to homework 1 or any future homeworks).

Please note the due dates. The first part, which is the account setup, is due on Jan 26. In addition, you must either submit the full assignment by that date, or your submission to the first part must clearly document what compilation issues you are having (please also get in touch with the teaching staff about solving them). In any event, the full assignment must be submitted by Jan 31.

Account Setup

First, please sign up for a CS 184 class account. Try to choose initials corresponding to your name, rather than randomly (this is not strictly required, but would help a great deal). The account is useful for all class-related activity, submitting assignments, making websites for grading and so on.

In your account, create a world-readable sub-directory called *public_html* and create an HTML page *index.html*. This website should at minimum contain your name, and a link to where you will post optional and required assignment results. Feel free to include any other information and customize your website. Make sure you can load the webpage by going to `http://inst.cs.berkeley.edu/~CS184-xx` (xx should be replaced by your account name). You may also want to set up e-mail forwarding so we can send e-mail to your cs184 account and have it forwarded.

Submission: Your submission for this assignment (besides creating the website above) is done via the *submit* program as documented in `http://inst.eecs.berkeley.edu/cgi-bin/pub.cgi?file=submit.help`. (ask the TA or post on the newsgroup if you need help on this). This first part of the assignment is called *hw0a*, so run *submit hw0a*. Please submit a directory containing the following elements (please follow the instructions exactly).

1. A file called *info.txt* that has separate lines for each of the following: Your name; your primary e-mail for sending announcements; your CS 184 account; your year and major (such as EECS Sophomore); your primary development platform (such as Windows 7, Ubuntu, Mac OS Lion, Python and Lisp on Iphone , etc.) ; any other information of interest.

2. A file called *photo.jpg* which is a JPEG image with a picture of yourself. The image should be recognizable and be of resolution 120×160 (otherwise we will resize it for the class roster). If you have any privacy concerns in this matter, please speak to the professor beforehand.

3. A file called *compilation.txt* that documents what you tried, any useful suggestions and any outstanding issues in completing the second part of the assignment involving compilation. If you have turned that part in, your file need only say "Turned in hw0b", although if you did something unusual we'd still like to know and may (no promises) give you extra credit. In any event, you should contact the teaching staff to make sure you can turn in the second part by the deadline the following week.

Compiling OpenGL programs

Much of the material for the class will be using OpenGL programs. One unit of the course will be on programming OpenGL, but you will be using it for the other assignments in any case. This year, we will also be introducing modern programming with programmable vertex and fragment shaders for the first time, using the OpenGL shading language (GLSL).

In *principle*, OpenGL and related code should be portable across many platforms, and indeed this is its strength. My major development platform is an older (circa 2008) Linux system, and I have modified and compiled the programs on an older Windows machine with Visual Studio 2008. Therefore, I believe they should work on almost any machine you may be using. As such, 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 (more on the latter below). Please speak with the teaching assistants if you need a C/C++ development environment such as Microsoft's Visual Studio.

In *practice*, the above statement is pretty close to true, but you need different installs and tricks to get your machine set up to run the programs for this class properly; this differs on different operating systems. Moreover, the shading language is constantly evolving. All of this means that **you should definitely start as early as possible.** Post to the newsgroup and contact the teaching assistant regarding problems in compilation. If you come up with some interesting tricks to get things to work, let us know that too. As a practical matter, you cannot progress further in the course without being able to compile OpenGL and GLSL programs, so treat this assignment as being of utmost importance.

Assignment: To make sure you can run programs for the course, please download and compile two programs. The first is the skeleton code for assignment 1. The second is provided off our website and is our demo program for the OpenGL assignment. We include both Makefiles for Linux and Visual Studio projects for Win 32. While our versions include the compiled objectives and executable to help you, please try to recompile the source code from scratch to make sure everything works. If you develop on a different platform, you may need to re-create projects and Makefiles from the source code.

A brief description of the two projects. The homework 1 skeleton should show a teapot on a blue background. The solution should allow you to use arrow keys to change your viewpoint and look at the teapot from different directions.

The larger demo program has a textured ground plane with 4 "pillars" and a teapot with lighting that moves. 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.

Besides compiling and running these programs, to test that it works, I want you to change the color of the red highlight on the teapot to yellow (yellow is made by mixing red and green, which are the first two elements of the color vector: the third is blue). The relevant colors and code are defined in the *display* function of *mytest3.cpp* where it says "add lighting effects."

Note for both programs the *shader* directories that contain glsl shaders. These are files that are loaded and compiled by the OpenGL program at runtime and therefore must exist, but need not be part of the Makefile or project.

Submission: Use the *submit* program to submit *hw0b* with a directory that contains the following files:

1. *hw1.jpg* with a screenshot from the homework 1 skeleton.
2. *demo.jpg* with a screenshot from the demo program (place the teapot somewhere in the center of the screen and zoom in if you want).
3. *demo2.jpg* same as above, now with a yellow instead of red highlight on the teapot.
4. *comments.txt* Comments on what you did to get things to compile, suggestions for us to make this more user-friendly for future years etc. If you have nothing to say, you may leave this file blank.

Compilation Hints: You probably already have **OpenGL** set up on your computer. Search for *gl.h* to see if you have it. If not, go to `www.opengl.org` and follow instructions to download the appropriate files. The website Nehe, at `http://nehe.gamedev.net/lesson.asp?index=01` provides useful OpenGL lessions, especially involving setting up OpenGL in various environments.

You probably don't already have **GLUT** set up. If you want, you can also find the download for that package from `www.opengl.org` (or just google for GLUT to find many sources for it). We'll also provide a link on the assignment website to download the three necessary glut files. The easiest way on Visual Studio for Windows is to simply move the files to the appropriate location, the dll file to the Windows System directory with other dlls, the header file to where the other GL headers are, and the lib file to where other library files are. Chris Wyman's website at `http://www.divms.uiowa.edu/~cwyman/classes/common/howto/winGLUT.html` has a good step-by-step tutorial. Fortunately the internet is your friend, and searching for installing glut for your system quickly brings up a lot of information. Note also that the newer version of GLUT is freeglut, available at `http://freeglut.sourceforge.net`. We will not be using any very complex GLUT functionality so any version should work.

As you can see from the source code for the demo program, once OpenGL and GLUT are properly set up, on my **Linux** machine, I just needed to add an include line to the main file, *#include GL/glut.h*. The compile line then just needs to link against *-lglut* and possibly *-lGL*. Importantly, you should define *GL_GEXT_PROTOTYPES* either in the source code, or as I've done as a compile-time option. It is possible that if you are running one of the newest machines and OpenGL versions, you may need to turn on compatibility profiles. If this is an issue, please speak to the teaching assistant.

On **Windows**, an excellent resource for compiling OpenGL programs with shaders is maintained by Chris Wyman at `http://www.divms.uiowa.edu/~cwyman/classes/common/howto/winGLUT.html` He discusses the various system paths to install the GLUT files. You also need to set your Visual Studio project to include additional linker dependencies such as *glew32.lib glut32.lib glu32.lib opengl32.lib* (modify as needed for 64 bit, or make sure you compile to a 32 bit executable). The visual studio project you download has this done already. The headers should also *#include GL/glut.h* as in Linux, but the order of includes does seem to matter; see what I did in the Windows version.

When using **GLSL on Windows**, there is also an issue with extensions and the shading functions. There are a variety of ways to address this (just search online), but the preferred option is to install *GLEW*, that stands for *The OpenGL Extension Wrangler Library* and is found at `http://glew.sourceforge.net`. The library is cross-platform and may be useful on other machines as well. Follow the same instructions as for installing GLUT. The dll files go in the same directory as the glut dll files, same for lib files and header files. You need to *#include GL/glew.h* before you include GLUT. Crucially, you need to call *glewInit() ;* in the main loop of your program, *after* you have created the GLUT window. The visual studio project and Windows source code includes all of these tweaks.

The skeleton code for assignment 1 includes the **GLM library** which is available for you to download at `http://glm.g-truc.net/`. This library is header-only, so should be very portable and includes many useful matrix-vector operations, and replacements for many deprecated functions. I have included the zip file and unpacked versions already in the skeleton. You just need to make sure you either move the files to your standard include location, or include the GLM libraries in your include path (this should already be done in the Linux Makefile and Visual Studio project).

I have not personally tested the code on other platforms, but some combination of these ideas should certainly work, and there is a wealth of information available online. As always, the teaching assistants and instructors are able to help. Good luck!

Acknowledgments: The first part of this assignment draws largely from Prof. O'Brien's. The compilation with glsl shaders is brand new in CS 184 this year, and requires extra work to make sure you have everything working. *So, please start early!*