CSE 167 (WI 2021) Homework 0 FAQ/Compilation Notes

Note courtesy Ravi Ramamoorthi, his instruction assistants and students in the past years.

The skeleton code already includes most needed dependencies, and will compile without problems in the vast majority of cases. So **most of you wil not need to read this FAQ at all**.

Also, as a first step, if you are having difficulties, **please update the drivers for your graphics card**.

As a general guideline, we do require support of OpenGL 3.1 and GLSL 330. (If your graphics card only supports OpenGL 2.1 and GLSL 120, see the older skeleton frameworks at the bottom of the assignments and resources page on the CSE167 WI17, which are designed to be backward-compatible to GLSL 120 (OpenGL 2.1)) While almost any machine purchased in the last 5-6 years will support this, some integrated Intel graphics cards might not. The only way to really test is to make sure homework 0 works. In particular, if you get a message (on Windows) like `exception at 0x00000000 in hw0-windows.exe: 0xC0000005: Access violation` it may indicate an issue with your graphics card, and you need to upgrade drivers and/or try a different card/machine, or develop on compatible machines in the CS basement (we have tested most labs in the basement work).

Please ignore compiler warnings, as long as the program compiles and runs. In particular, to allow for execution on the widest variety of older systems, we do use a few backwards-compatible deprecated programming constructs, which may lead to a few warnings on newer machines. The programs will still compile and run, and if they do so, you may safely ignore these warnings.

Contents

0.1 Windows

Visual Studio 2012 (or 2013 and 2015+) can usually be downloaded for free by students at `https://www.dreamspark.com/`. (There may be other UCSD-specific ways of obtaining the code). If this free link does not work for you, we assume you can obtain Visual Studio or an equivalent C++ development environment by other means. The standard download is `http://www.microsoft.com/visualstudio/eng/downloads`. Please note that with VS 2015, you must explicitly select C++ support when installing; by default only C# *etc.* is installed. Failing to include C++ support will lead to lots of issues with missing headers.

Choose the right version for your computer. Please note that the Visual Studio Express 2012 (or 2013 or later) edition should be adequate for this course, and is free for everyone. Some students have mentioned that it is necessary to use Visual Studios Express 2012 (or 2013) for Windows *Desktop* rather than just VS 2012 (2013/later) express for Windows. A more recent version is Visual Studio Community 2015. If you install this version of VS 2015, please ensure that you select the "Common Tools for Visual C++ 2015" feature under "Programming Languages"/"Visual C++". If the above step isn't done correctly, the skeleton code may not be

upgraded properly. If you installed Visual Studio 2015 without this feature, you can re-run the installation program and modify Visual Studio to add the feature. Please note that by default, VS 2015 does not include C++ support. *You must explicitly select C++ support in the installer when installing Visual Studio 2015.* (Otherwise only C# is supported, and you will get a bunch of errors with missing header files when compiling).

(Please note that we use a VS 2012 skeleton for maximum backwards compatibility. If you are using VS 2013 or VS 2015, the skeleton will still work; you only need to click OK in the conversion/upgrade dialog once you open the skeleton code). If Visual Studio note that you are missing the build tools for the project, right click the project under the Solution Explorer tab, go to Properties, and under General, select a Platform Toolset that matches your current version of Visual Studio.

Finally, as noted in the README file, [only] if your GLUT, GLM or GLEW malfunction, go to the "Tools" dropdown on the top of Visual Studio, navigate to "NuGet Package Manager", and select "Manage NuGet Packages for Solution." Browse for and install "glm" for the current project in order to include and use glm headers. If your OpenGL is not set up properly, you may also need to browse for and install `nuopengl.core`. Please note that there are `.redist` packages with the same name. Please install the packages without `.redist` in the name! Hopefully, most of you will not need this step, since we have included glm etc. in the skeleton framework. (Some students have indicated that after using NuGet, they still need to copy the relevant libraries to the appropriate `.\lib` directory so that they were linked properly).

The FreeImage Library `.DLL` is included with the Windows distribution. If you change your build configuration from "Debug" to "Release", you must copy "FreeImage.dll" to the "Release" directory that resides in the project root directory containing the `.SLN` file.

Some students have reported that on older toolsets, you may need to modify `Geometry.h` if the solution does not compile, and in particular complains that the identifier INFINITY is undefined. In that case, include the following lines in `Geometry.h` (this is needed only on older toolsets and should not usually be a problem).

```
#include <limits>
#define INFINITY FLT_MAX
```

If you are having issues taking screenshots, you can either manually crop a print screen (printscreen button) or use Fraps to take a screenshot. Some students have reported that this issue can be fixed simply by changing the code to save the back buffer instead of the front (this is in the third line of the `saveScreenshot()` procedure in `mytest3.cpp`), that is by using

```
    glReadBuffer(GL_BACK);
```

If you are still having issues, you may need to find another machine or OS. This only happens for a very small percentage of Windows systems; most are fine; we have also modified the skeletons recently to eliminate the vast majority of issues.

(Only) if you run into issues with 32/64-bit compilation after re-installing glm, glut, glew as above, make sure your platform target says Win32, rather than x64.

The instructions below apply only to Windows users who cannot get the skeleton code to compile directly; most people will not need them, and they are dated.

Using Visual Studio, download: "Additional_Libraries.zip"

To import the required dependencies, copy from `Additional_Libraries.zip`

```
Dll\glew32.dll              to          %SystemRoot%\system32
```

```
Dll\glut32.dll              to          %SystemRoot%\system32
Lib\glew32.lib              to          {VC Root}\Lib
Lib\glut32.lib              to          {VC Root}\Lib
Include\GL\glew.h           to          {VC Root}\Include\GL
Include\GL\glut.h           to          {VC Root}\Include\GL
%SystemRoot% on Windows 7 x64 is typically C:\Windows
{VC Root} for VS2008 is typically C:\Program Files (x86)\Microsoft Visual Studio 9.0\VC,
{VC Root} for VS2010 is typically C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\,
{VC Root} for VS2012 is typically C:\Program Files (x86)\Windows Kits\8.0\include\um for
     includes
and C:\Program Files (x86)\Windows Kits\8.0\um\x86 for libs.
```

If it still doesn't work, some students have reported that

```
Include\GL\glew.h           to          C:\Program Files (x86)\Microsoft SDKs\Windows\v7
    .0A\Include\GL
Include\GL\glut.h           to          C:\Program Files (x86)\Microsoft SDKs\Windows\v7
    .0A\Include\GL
Lib\glew32.lib              to          C:\Program Files (x86)\Microsoft SDKs\Windows\v7
    .0A\lib
Lib\glut32.lib              to          C:\Program Files (x86)\Microsoft SDKs\Windows\v7
    .0A\lib
Dll\glew32.dll              to          ...hw0\Debug
Dll\glut32.dll              to          ...hw0\Debug
```

Or

```
Dll\glew32.dll              to          C:\Windows\SysWOW64\
Dll\glut32.dll              to          C:\Windows\SysWOW64\
```

help solve issues as well. Finally, open the .sln file.

0.2 Linux

For HW1 or HW2, if you have compile errors from "usleep" not being in scope, try including the system file `unistd.h` in the includes for `grader.h`. Please also make a similar change if necessary for subsequent assignments.

If your source files compile, but you have link errors, such as

```
/usr/bin/ld: final link failed: Nonrepresentable section on output
```

it is likely our libraries, provided for convenience, are outdated, and don't work with your modern Linux distribution. Try removing the `-L./lib/nix` in the Makefile (and possibly the `-I./include` although this is not required). This will ensure the code simply uses the system libraries. If successful, you will need to make this change for all homeworks in the course.

If you are using a synaptic-based distribution, like Ubuntu, run:

```
sudo apt-get install freeglut3-dev glew-utils libglew1.6-dev libfreeimage-dev
```

to install the required dependencies for our OpenGL assignments. If `libglew1.6-dev` cannot be found, try installing an older version, such as `libglew1.5-dev`. (Some students have reported simply using libglew-dev is simpler). Also make sure your package manager is up to date.

To compile, finally run make inside the assignment directory. If you do not have `make` or `g++`, run:

```
sudo apt-get install build-essential
```

on your synaptic-based Linux to install them.

If you get an error with respect to a missing GLX extension, some students have recommended

```
sudo apt-get install libva-glx1
```

Others have indicated they needed to install the NVIDIA driver rather than native driver, and update their system (initramfs) to make sure the NVIDIA driver is loaded upon boot.

If you get a segmentation fault (this is often in line 437 of `mytest3.cpp` `glGenVertexArrays`), it usually indicates that the glew extension wrangler is not set up properly, and there is a simple fix, by adding a single line above line 477 (please make this change for other homeworks too, and also see the discussion at the bottom of the Linux section on MESA support)

```
#ifndef __APPLE__
        glewExperimental = GL_TRUE;  // New Line to Add
  GLenum err = glewInit() ;    // Line 477
        if (GLEW_OK != err) {
          std:cerr << "Error: " << glewGetString(err) << std::endl;
        }
#endif
```

Further tips for those using MESA are provided from 2017 fall MOOC student Giordano-Lipari, if you get an error even after applying the segmentation fault fix above (that we detected OpenGL version 3.0, at least version 3.1 is required). He recommends launching
`MESA_GL_VERSION_OVERRIDE=3.1 ./mytest3`
instead of simply
`./mytest3`.
It is interesting that `MESA_GL_VERSION_OVERRIDE=3.0 ./mytest3` works too. It comes with an extra message about selecting terminal and pressing Enter. Do not select anything and just hit Enter.

In Fedora, some students have reported needing to use the following to install freeimage

```
sudo dnf install freeimage freeimage-devel
```

Some students recommend deleting the "lib" and "include" directories in the homework 0 directory, as you don't need them, provided you have appropriate system libraries installed (in some cases, the precompiled libraries included with the skeleton may also be incompatible with your system, in which case please go ahead and delete them in favor of the system libraries). Some students have noted that on ArchLinux, they need to downgrade to FreeGLUT 2.8.1 instead of 3.0.0.

Please see the following advice on FreeBSD if you are having a problem: Just change the two `/usr/X11R6/` to `/usr/local/` in the Makefile, and run `gmake`. Install freeglut, freeimage and glew in `/usr/ports/graphics`, and remove the lib directory in the homework.

If you cannot get the pillars to display (the teapot works fine) in homework 0, and you've tried everything else (update the drivers etc.), a student "pafnuty" from the Fall 2012 EdX class has suggested this associated patch. Essentially, replace the pillar drawing routine with old-style OpenGL. Please also test homeworks 1 and 2. You may have to end up using the old versions of these homeworks from the bottom of the assignments/resources page on the official website before applying this patch.

If you get a segmentation fault when saving the image, try adding
`glPixelStorei(GL_PACK_ALIGNMENT, 1);`
right before the `glReadPixels` call in `saveScreenshot()`.

If the saved images are black, some students have reported that this issue can be fixed simply by changing the code to save the back buffer instead of the front (this is in the third line of the `saveScreenshot()` procedure in `mytest3.cpp`), that is by using

```
glReadBuffer(GL_BACK);
```

Some useful hints on working with the Linux Slackware64 distro, courtesy of Spring 2013 EdX student "jcoppens". First, please include `unistd.h` as noted at the top of the Linux FAQ section. Next, note that many 64 bit libraries (including X11) are merged in `/usr/lib64` While symlinks should have been added, the Makefile may still need changes. He recommends making the following edit near the beginning:

```
else CFLAGS = -g -DGL_GLEXT_PROTOTYPES INCFLAGS = -I./glm-0.9.2.7 -I./include/ -I/usr/
    include -I/sw/include \ -I/usr/sww/include -I/usr/sww/pkg/Mesa/include

LDFLAGS = -L/usr/lib64 -L/sw/lib -L/usr/sww/lib \ -L/usr/sww/bin -L/usr/sww/pkg/Mesa/lib
     -lglut -lGLU -lGL -lX11 -lfreeimage
```

In slackware64, freeimage does not come with the distribution, and may need to be compiled from source. To compile freeimage, "jcoppens" reports that he modified line 9 in the `Makefile.gnu` to

```
INSTALLDIR ?= $(DESTDIR)/usr/lib64
```

and that he needed to add `#include <string.h>` in the file
`Source/OpenEXR/llmlmf/lmfAutoArray.h`
(just before the first other `#include`).

In Ubuntu, some students have reported issues with NVIDIA drivers. If you get an error saying something like

```
Inconsistency detected by ld.so: dl-version.c: 224: _dl_check_map_versions: Assertion `
    needed != ((void *)0)' failed!
```

It is suggested to replace the Makefile line that says

```
-L/usr/sww/pkg/Mesa/lib
```

to

```
-L/usr/lib/nvidia-304-updates/
```

or to the path to your driver-supplied libraries. Another student reported he fixed this by adding `-fuse-ld=gold` to `LDFLAGS` in the Makefile. It is apparently a very old bug on debian and debian derived distros with NVIDIA cards.

If you get an error [in Ubuntu] similar to

```
relocation_R_X86_64_32S against '.rodata' cannot be used when making a
  PIE object
```

that mentions a "PIE object" add the "`-no-pie`" flag to the flags in the Makefile.

For some students, the MESA software renderer may be a good option in Linux, especially if nothing else works. Besides following the instructions above, here are some tricks courtesy of community TA lsorense. In particular, Mesa defaults to using a compatibility profile of OpenGL 3.0. To get it to use OpenGL 3.1, add the following single line above line 470 in `mytest3.cpp` (the line in question is highlighted). You will also likely need to make a similar change in the main routine for homeworks 1 and 2. Please also see the comment above about issues with glew if you still get segmentation faults; you might need to add
`glewExperimental = GL_TRUE ;`
above the `glewInit()` line [line 477 in the original code]. In some cases, you may also want

to change the include `GL/glut.h` to `GL/freeglut.h` (In addition, if the skeleton framework libraries like `libGLEW.a` are not linkable, use the distribution libraries and modify the Makefile accordingly. This was tested by "lsorense" with installed packages `freeglut3-dev libfreeimage-dev mesa-common-dev` His Makefile may be of use.)

```
#ifdef __APPLE__
  glutInitDisplayMode (GLUT_3_2_CORE_PROFILE | GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
#else
        glutInitContextVersion(3,1);  // New Line to Add
  glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH); // Line 470
#endif
```

For those using virtual machines (and perhaps others), if the above approach does not work, it may be because you have graphics acceleration turned on which does not support OpenGL 3.1. In this case, please disable your Virtual Machine's 3D acceleration. For VirtualBox, this corresponds to unchecking the Enable 3D Acceleration box in the Display settings for your guest VM before running the VM.

0.3 OSX

Download and install Xcode from the App Store, or from `http://developer.apple.com/xcode/` You may need an older version if you're on an older OSX version.

If using make, just run make in the assignment directory to compile. By default, neither make nor g++ are installed on OSX. To install these command line tools if you haven't already, open 'Terminal' (if you've never done this before, type Terminal into Spotlight on the upper right). Type `xcode-select --install`. An alert window will pop up. Click on 'Install' to continue installation.

On older MacOS systems, to install the command-line tools, first open Xcode. From the menu bar, navigate to "Xcode > Preferences". Next, choose the "Downloads" tab. Within Downloads, choose the "Components" tab. Click the "Install" button to the right of "Command Line Tools". This will install the necessary build tools, such as make and g++.

Some students have mentioned that they may get link errors or warnings like
`Undefined symbol for architecture x86_64`
If it is only a warning, it is probably safe to ignore. Otherwise, one student reports "Looks like the skeleton code did not play well with mp-gcc48. I had to execute sudo port select gcc none to ensure Apple's Clang compiler is used which works fine now." [Of course, please try this only if absolutely necessary].

You should not need to compile FreeImage for OSX; we provide it with the skeleton code. If you must install it on OSX, it would be simpler to use MacPorts or Homebrew to install the library than to compile it from source.

If you are having the teapot strangely offset in the window after you program in your solution, this may be an issue with retina displays on some Macs. One student suggests adding
`glutInitWindowSize(w,h) ;`
before `glutCreateWindow`. (You may also need to do this on future homeworks, and there may still be issues with resizing windows.) Finally, on MacOS Mojave, some students have noted that the program window is initially black; you may need to move the window around, or minimize and reopen it, which should fix the problem.

If you wish to use Xcode's IDE, add GLUT framework (and maybe OpenGL framework also), and change include headers from `GL/glut.h` to `GLUT/glut.h` in `main.cpp` (or anywhere it appears). You may also need to change `GL/glext.h` to `OpenGL/glext.h`

Please note that we do not support IDEs other than Visual Studio for Windows (we recommend the command line in OSX/Linux), so we will have limited ability to answer compilation questions for the XCode IDE, but you should be able to figure it out yourself, and with the help of other students. We provide the following brief tutorial on getting the projects to work in XCode, courtesy of edX community TA Stephan_G, but it is not watertight (and we still recommend using the command line on Mac OS). He has also made a git repository that should work better out of the box:

`git clone https://github.com/Stephan-G/CSE167x.git`

Note however, that it contains the old versions of the skeleton frameworks, and that you must still set up a custom working directory (see below especially item 5) or you won't be able to find the shaders sub-directory, among other things.

1. Make a new project for your assignment. In Xcode select 'File -> New -> Project'. As template choose 'OS X -> Application -> Command Line Tool'. Type or Language should be set to C++.
2. Remove 'main.cpp' from the project folder in Xcode. Drag all files from the sample project, except the Makefile into the group folder in Xcode.
3. In project settings, 'Build Settings' in the section 'Search Paths' add
   `$(SRCROOT)/YOUR-PROJECT-NAME/include`
   (it has to be recursive) to 'User Header Search Paths'. Also set the setting 'Always search user paths' to 'Yes'. For later homeworks, you may also need to add
   `$(SRCROOT)/YOUR-PROJECT-NAME/`
   to the user search paths; otherwise `glm.hpp` won't be found.
4. In project settings, 'Build Phases' in the section 'Link Binaries with Libraries', add 'GLUT.framework' and 'OpenGL.framework' via the selection menu. Add 'libfreeimage.a' and 'libGLEW.a' via the 'Add other...' button (you have to navigate to the file location. They're in the 'lib/mac').
5. Edit the Scheme in the 'Options' tab, check 'Use custom working directory' and set it to the project folder (the folder containing 'wood.ppm' and the 'shader' folder)
6. The project should now be able to build and run.
7. In previous versions of the class, some students have suggested the following fixes if it still doesn't work: At the top of `shaders.cpp`, delete the line that says `#include <GLUT/glext.h>` Some students have also reported that the default compiler may need to be changed from `Apple LLVM Compiler 4.2` to `LLVM GCC 4.2` (in Build Options: look for Compiler C/C++). One student also reported changing from `#include <FreeImage.h>` to `#include "FreeImage.h"` worked.
8. Some users have also reported that grader.cpp must be manually added in OSX and Xcode, especially for later homeworks. To do so, go to the project settings, 'Build Phases' then 'Compile Sources' and add grader.cpp to the list, or make sure it is there already (you will need to do this for HW 1,2 as well). Other users have reported they needed to manually add all the source files, so please try to do so if it doesn't work otherwise.

If HW0 won't compile (particularly relevant to OSX 10.8 [Mountain Lion], giving an error like this (with dozens of lines of stuff after):

```
g++ -g -DGL_GLEXT_PROTOTYPES -I./include/ -I/usr/X11/include -DOSX -c mytest3.cpp
mytest3.cpp:15:21: error: GL/glut.h: No such file or directory
mytest3.cpp:21: error: 'GLdouble' does not name a type
mytest3.cpp:22: error: 'GLfloat' does not name a type
mytest3.cpp:23: error: 'GLfloat' does not name a type
....
```

Try installing XQuartz (download here). This is necessary because Apple stopped including X11 in Mountain Lion. Some students have reported that this step is not needed, and you can compile simply by changing the include `GL/glut.h` to `GLUT/glut.h` (some students also recommend changing `GL/glext.h` to `OpenGL/glext.h`).

If the program has a problem finding the shaders directory, it's possible there is an issue with relative paths in Mountain Lion; try changing the path to the shaders to be the absolute path name within the program. (You will need to look in the skeleton to see where the path to the shaders is specified).

0.3.1 Eclipse

If you encounter an error with

```
enum {FLOOR, CUBE} ;
```

It should be changed to:

```
const unsigned int FLOOR = 0 ;
const unsigned int CUBE = 1 ;
```

0.4 General

If the skeleton code does not render correctly, you may want to update your drivers. Based on past posts in the discussion forum, this is probably the single most common (and easily fixable) issue.

The code frameworks do require that your graphics card (GPU) support at least GLSL version 330 (OpenGL 3.1). [The older skeleton frameworks at the bottom of the assignments and resources page on the CSE167 WI17 are designed to be backward-compatible to GLSL 120 (OpenGL 2.1)]. While almost any NVIDIA or ATI graphics card will be suitable, some old integrated graphics cards (such as Intel GMA950 or similar) may not run the homeworks properly. Given the variety of systems, this should be taken as a guideline rather than absolute; the real test is if homework 0 does compile and work. If not, you may need to upgrade or find a different machine. In particular, if you get a message (on Windows) like exception at `0x00000000 in hw0-windows.exe: 0xC0000005: Access violation` it may indicate an issue with your graphics card, and you need to upgrade drivers and/or try a different card/machine.

Different systems produce slightly different images. We have set our thresholds in the autograder accordingly. Don't worry about a few pixels being off, as long as you pass the test. If there's still a problem, please turn off all antialiasing options (see the top of the page).

Ignore compiler warnings if the program runs.

Be sure to activate floating behavior if you use a tiling WM (eg. DWM). If you don't, the screenshots will be skewed.

We do not directly offer support for IDEs other than Visual Studio on Windows. If you use XCode or Eclipse, you must generate the project yourself, and make sure it will still compile with the provided Makefiles (on OSX/Linux) or the Sln (on Windows), or create your own Makefile.

If you are getting the error (this is in regards to the older code frameworks, not the newer 330 GLSL):

```
ERROR: 0:9: error(#137) 'attribute' supported in vertex shaders only
ERROR: 0:10: error(#137) 'attribute' supported in vertex shaders only
ERROR: 0:11: error(#137) 'attribute' supported in vertex shaders only
```

Please replace the following lines in shaders/light.frag.glsl (lines 9 through 11):

```
attribute vec4 color ;
attribute vec3 mynormal ;
attribute vec4 myvertex ;
```

With:

```
varying vec4 color ;
varying vec3 mynormal ;
varying vec4 myvertex ;
```