Installing OpenGL

You need to install an environment on your hardware where you can compile and run OpenGL programs. OpenGL is supported on most modern operating systems. Linux and OSX natively supports OpenGL, but on Windows you need to use GLEW to get a recent version of OpenGL.

In addition to installing OpenGL you need to install a wrapper program such as GLUT, GLFW or SDL to interact with the window manager and handle operating system specific functions like opening and closing windows, provide keyboard and mouse interaction, and so on.

My examples use GLUT is the simplest environment for OpenGL, except for examples glfw21, sdl21v1 and sdl21v2 which use OpenGL with GLFW, SDL version 1.2 and SDL version 2, respectively.

Start by installing GLUT as this will also install all the OpenGL libraries. If you want to use GLFW and/or SDL install that only after installing GLUT because these libraries can be used with other graphics engines and may not imply GL.

The OpenGL Extension Wrangler (GLEW) can be used on Windows and OSX to get access to the latest OpenGL features, but is NOT required on Linux. Make sure that you make compiling with GLEW optional using the **-DUSEGLEW** compiler flag. If not this will create compilation problems on my Linux machine which does not need or provide GLEW.

Before asking for help, look at the Troubleshooting hints to localize the problem rather than asking a meaningless question like *It doesn't work*.

Linux

OpenGL support is very easy to enable on current Linux distributions using the standard package managers included with the distribution. Since CU mostly uses Ubuntu this description is Ubuntu specific. For other distributions the procedure is the same, but the packages may be named slightly differently.

Always install GLUT first, whether you plan to use it or not, because installing GLUT will also install OpenGL. Then add GLFW, SDL, or your wrapper of choice. Note that all these libraries will happily coexist. Since you are building programs you need the development versions of these libraries to also get the header files.

On X11 based systems install glxinfo using

```
apt-get install mesa-utils
```

Then run

```
glxinfo|grep 'direct rendering'
```

If the result is YES, then hardware support for OpenGL is working. If it is NO, some things are done in software and you may take a performance hit. Depending on your hardware, you may want to work on your X server. Specifically, the nVidia and AMD/ATI web sites contains updated drivers that result in improved performance over the stock Xorg drivers.

On Ubuntu using the Additional Drivers integrates the vendor drivers with the Ubuntu kernels, which is the recommended procedure to void manual updates when the kernel is upgraded. Be careful to not override the Ubuntu mechanism to maintain the kernel and drivers, as this can leave the system in an unbootable state or break the X server.

The compiz window manager (which is an OpenGL window manager) makes applications with high frame rates run jerky unless you enable VSync. This seems to be an issue especially with newer Ubuntu installs.

GLUT

Since GLUT depends on OpenGL and a number of other libraries, installing GLUT will trigger the dependencies needed to install everything else. The installation command is

```
apt-get install freeglut3-dev
```

To compile and link the program foo.c use

```
gcc -Wall -o foo foo.c -lglut -lGLU -lGL -lm
```

GLFW

The installation command is

```
apt-get install libglfw3-dev
```

To compile and link the program foo.c use

```
gcc -Wall -o foo foo.c -lglfw -lGLU -lGL -lm
```

SDL

There are two versions of the SDL currently in widespread use which are not binary compatible, but can co-exist. To install SDL 1.2 use

```
apt-get install libsdl1.2-dev libsdl-mixer1.2-dev libsdl-image1.2-dev
```

To compile and link the program foo.c use

```
gcc -Wall -o foo foo.c -lSDL -lSDL_mixer -lGLU -lGL -lm
```

To use SDL2 instead, install with

```
apt-get install libsdl2-dev libsdl2-mixer-dev libsdl2-image-dev
```

To compile and link the program foo.c use

```
gcc -Wall -o foo foo.c -lSDL2 -lSDL2_mixer -lGLU -lGL -lm
```

Note that these examples only uses the Mixer library to add sound. You can also use SDL to load images, but that would require adding the SDL_image or SDL2_image libraries.

OS/X

The OS/X Darwin environment is transitioning away from OpenGL. Apple is pushing their propietary Metal environment for graphics, but still supports OpenGL although strictly it has been deprecated. Apple makes it hard to use OpenGL features beyond OpenGL 3.2. OpenGL 4 is supported on Apple hardware but only using the core profile so switching between the fixed and programmable pipelines is not supported.

Start by updating Xcode to the latest release. This is a multi-gigabyte download which installs both OpenGL and GLUT. When installing Xcode be sure to select Command Line Tools as part of the install.

The easiest way to install additional packages such as GLFW, SDL and GLEW is using Homebrew. Install the brew package manager using the command shown on the web site

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

The description below uses brew to install additional packages.

GLUT

To compile and link the program foo.c using the Apple SDK requires

```
gcc -Wall -Wno-deprecated-declarations -o foo foo.c -framework GLUT -framework OpenGL
```

Apple considers the OpenGL API deprecated and will throw LOTS of warnings if you do not use the **-Wno-deprecated-declarations** compiler flag.

Note that under OS/X, the GLUT header files are in the subdirectory GLUT rather than the GL subdirectory. The following code works on OSX and Linux

```
#ifdef __APPLE__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif
```

GLFW

Install GLFW using the command

```
brew install glfw
```

Homebrew installs GLEW and GLFW into /usr/local. You can now compile and link GLFW based programs using

```
gcc -Wall -o foo foo.c  -lglfw -framework Cocoa -framework OpenGL -framework IOKit
```

Note the need for the Cocoa and IOKit frameworks since GLFW relies on these. Apple places the OpenGL files in a different location than other systems, so it is important that you make the header file inclusion conditional

```
#ifdef __APPLE__
#include <OpenGL/glu.h>
#include <OpenGL/gl.h>
#else
#include <GL/glu.h>
#include <GL/gl.h>
#endif
```

SDL

There are two versions of the SDL currently in widespread use which are not binary compatible, but can co-exist. To install SDL 1.2 use

```
brew install sdl sdl_image sdl_mixer
```

To compile and link the program foo.c use

```
gcc -Wall -Wno-deprecated-declarations -o foo foo.c -lSDLmain -lSDL -lSDL_mixer -framework Cocoa -framework OpenGL
```

To use SDL2 instead, install with

```
brew install sdl2 sdl2_image sdl2_mixer
```

To compile and link the program foo.c use

```
gcc -Wall -Wno-deprecated-declarations -o foo foo.c -lSDL2main -lSDL2 -lSDL2_mixer -framework Cocoa -framework OpenGL
```

Note that these examples only uses the Mixer library to add sound. You can also use SDL to load images, but that would require adding the SDL_image or SDL2_image libraries.

GLEW

Apple support for OpenGL 3.3 and later requires that you use GLEW. Install GLEW using

```
brew install glew
```

You can now compile and link programs by adding conditional compilation of GLEW using the **-DUSEGLEW** compiler flag and adding **-lglew** to link in the GLEW library. For example, using GLEW with GLUT would use

```
gcc -Wall -Wno-deprecated-declarations -DUSEGLEW -o foo foo.c -lglew -framework GLUT -framework OpenGL
```

or using GLEW with GLFW would be

```
gcc -Wall -Wno-deprecated-declarations -DUSEGLEW -o foo foo.c  -lglfw3 -lglew -framework Cocoa -framework OpenGL -framework IOKit
```

To access features beyond OpenGL 3.2 on OSX when using GLFW, you need to provide the following hints to GLFW. Note that this only allows access to the core profile, not the compatibility profile, which is much stricter about what you can do in OpenGL 4.

```
glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR,3);
glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR,2);
glfwWindowHint(GLFW_OPENGL_PROFILE,GLFW_OPENGL_CORE_PROFILE);
glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT,GL_TRUE);
```

Note that in my example programs, using GLEW is required on some systems and not on others. So I use the compiler flag **-DUSEGLEW** to conditionally compile in GLEW when needed. I use the **-DAPPLE_GL4** flag to include window hints shown above.

Retina Display Issues

The Apple retina displays sometimes cause problems with OpenGL because they report a resolution to GLUT that is half the size of the frame buffer mapped to the window. When using my examples, you can add -DRES=2 on the compile line to correct this, which is enabled by default on Apple hardware.

If you find that only half the image appears on the screen, try using -DRES=1 which reverts this setting. I have not found a definitive test which determines when -DRES=1 is required and when -DRES=2 is required, so you may have to set this specific to your installation.

Windows: MSYS2/MinGW

Various Windows versions support OpenGL natively, but graphics card manufacturers often replace the native Windows OpenGL libraries with their own libraries that support specific features of their hardware.

Since most Windows environments do not contain a native compilation suite, installing the necessary header files and libraries differs depending on the compiler used.

Your simplest option to get a full toolchain on Windows is to use the MSYS2 platform, which provides an easy to use environment for installing and running native Windows software. MSYS2 uses the MinGW toolchain and the pacman package manager to easily install various packages.

Start by downloading and installing MSYS2 from here. Just use the default locations. Only do the basic MSYS2 installation. **Do not** do any of the pacman steps on the MSYS2 install page.

Once installed launch MSYS and update the packages to the latests version using.

```
pacman -Syu
```

This may close the window in some cases. Relaunch the window and rerun the command until it says everything is up to date.

Now install the compiler toolchain using

```
pacman -S --needed base-devel mingw-w64-x86_64-toolchain
```

Just do **all** which installs way more than we need but that's OK.

Now launch **MSYS MinGW 64-bit** from the **Start** window. There may be several MSYS options but only the **MSYS MinGW 64-bit** shell will work for us. You can now test the compiler with the hello world program to make sure all is working.

GLUT

Now install GLUT and GLEW. It is as simple as doing

```
pacman -S mingw-w64-x86_64-freeglut
pacman -S mingw-w64-x86_64-glew
```

You can install any other package you want exacly the same way. To compile a program using GLUT and GLEW do

```
gcc -Wall -DUSEGLEW -o foo foo.c -lfreeglut -lglew32 -lglu32 -lopengl32 -lm
```

Note that in my example programs, using GLEW is required on some systems and not on others. So I use the compiler flag **-DUSEGLEW** to conditionally compile in GLEW when needed. My system does not require or support GLEW, so if you do not make using GLEW conditional the build will fail on my hardware.

GLFW

Install GLFW using

```
pacman -S mingw-w64-x86_64-glfw
```

To compile a program using GLFW and GLEW do

```
gcc -Wall -DUSEGLEW -o foo foo.c -lglfw3 -lglew32 -lglu32 -lopengl32 -lm
```

Note that in my example programs, using GLEW is required on some systems and not on others. So I use the compiler flag **-DUSEGLEW** to conditionally compile in GLEW when needed.

SDL

There are two versions of the SDL currently in widespread use which are not binary compatible, but can co-exist. To install SDL 1.2 use

```
pacman -S mingw-w64-x86_64-SDL mingw-w64-x86_64-SDL_mixer mingw-w64-x86_64-SDL_image
```

To compile and link the program foo.c use

```
gcc -Wall -DUSEGLEW -o foo foo.c -lmingw32 -lSDLmain -lSDL -mwindows -lSDL_mixer -lglew32 -lglu32 -lopengl32 -lm
```

To use SDL2 instead, install with

```
pacman -S mingw-w64-x86_64-SDL2 mingw-w64-x86_64-SDL2_mixer mingw-w64-x86_64-SDL2_image
```

To compile and link the program foo.c use

```
gcc -Wall -DUSEGLEW -o foo foo.c -lmingw32 -lSDL2main -lSDL2 -mwindows -lSDL2_mixer -lglew32 -lglu32 -lopengl32 -lm
```

Note that these examples only uses the Mixer library to add sound. You can also use SDL to load images, but that would require adding the SDL_image or SDL2_image libraries.

Troubleshooting hints

If you are having trouble getting the programs to compile, you may want to consider these things to determine if the problem is in your compiler, header files or linker/libraries.

- Try compiling the classic C program that prints *Hello World!* to the screen to check that your compiler and linker works in the absence of OpenGL. If this fails to compile and run, your compile environment is broken and needs to be fixed before you can work on getting OpenGL to work.
- On Windows, only the **MSYS MinGW 64-bit** shell will find the correct header files and libraries. This is a common problem with compiling and linking.
- When running the program from the command line, you may need to include ./ before the command (e.g. ./ex1) unless you set the PATH to include the current directory.
- Try compiling ex1.c. If you get errors like
  error: GL/gl.h: No such file or directory
  or
  error: 'GL_COLOR_BUFFER_BIT' undeclared
  or
  error: GL/glew.h: No such file or directory
  the compiler cannot find your OpenGL header files. Check that on Windows you are using the **MSYS MinGW 64-bit** shell and that files are in an expected place like **/usr/include/GL** or use the **-I** compiler flag to tell the compiler where it is.
- Try linking ex1.c.
  undefined reference to 'glut... means the GLUT library was not found.
  undefined reference to 'glu... means the GLU library was not found.
  undefined reference to 'gl... means the GL library was not found.
  Make sure the libraries are in an expected place like **/usr/lib** or **/usr/local/lib** or use the **-L** flag to tell the linker where it is.
- If you get **warning: implicit declaration of function 'glWindowPos2i'** make sure you have
  #define GL_GLEXT_PROTOTYPES
  before any includes.
- If you get **undefined reference to glWindowPos2i** you may have an old GL library. Make sure that you define **-DUSEGLEW** on the compile line so that you have a newer version of OpenGL.
- If in compiling the example programs you get errors referencing, for example, **glBindFrameBuffer** or similar functions, you may not have support for OpenGL 3 or later. Adding **-DUSEGLEW** on the compiler flags should fix this problem in most cases.