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, but how to compile and link programs varies greatly. The description below assumes that you already have a compilation environment set up.
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. For distributions derived from RedHat Linux, the libraries and header files are installed using the command
yum install freeglut-devel
Since GLUT depends on OpenGL and a number of other libraries, installing GLUT will trigger the dependencies needed to install everything else. For distributions derived from Debian such as Ubuntu, the installation command is
apt-get install freeglut3-dev
To compile and link your program on Ubuntu 14 based distros you need to explicitly grab every library using
gcc -o foo foo.c -lglut -lGLU -lGL -lm
I recommend that you use this full version even if not required on your system. Older distributions may put the files in /usr/X11R6, in which case you need to add -I and -L flags to pick up the header files and libraries.
Once installed, run the **glxinfo** program and look for *direct rendering* in the output. 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.
The compiz window manager (which is an OpenGL window manager) makes applications which use glutIdleFunc() run jerky unless you enable VSync. This seems to be an issue especially with newer Ubuntu installs.

OS/X

The OS/X Darwin environment is based on OpenGL. Therefore any compilation environment for OS/X should already support compiling OpenGL programs.
To compile and link your program using the Apple SDK requires
gcc -o foo foo.c -framework GLUT -framework OpenGL
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
```

Windows

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. You may also need to install GLUT from Nate Robins.

Windows: MinGW

If you don't have a compiliation environment on Windows, the path of least resistance is to install MinGW which provides the gcc and g++ compilers for Windows.
Download and install MinGW. Accept the default install location *C:\MinGW* and install **mingw32-base**, **mingw32-gcc-g++** and **msys-base**. After the installation is completed, change directories to *C:\MinGW\bin* and rename **mingw32-make.exe** to **make.exe** to avoid having to type mingw32-make every time you want to use make. You should also add *C:\MinGW\bin* to the beginning of your PATH.
Finally, install GLUT by unzipping my version of GLUT into *C:\MinGW*. It will provide **include\GL\glut.h** and **lib\libglut32cu.a** needed to use GLUT. It also provides **bin\glut32.dll**. To find **glut32.dll** at run time it must be in the PATH, so including *C:\MinGW\bin* in the PATH is critical.
Note that my version of GLUT has been patched to provide glWindowPos2i which otherwise would not be available on Windows systems which support OpenGL version 1.1.
To compile and link foo.c under MinGW you need
gcc -Wall -ofoo foo.c -lglut32cu -lglu32 -lopengl32
Note that my makefiles assumes that you have the MinGW environment.

Windows: MinGW and GLEW

In order to use more advanced features in OpenGL like shaders and frame buffers, you need to get around the libraries provided by Microsoft, and use the vendor supplied libraries instead. This is provided by the OpenGL Extension Wrangler (GLEW). However, the GLEW libraries provided on the GLEW web site are for the Microsoft compiler and is not compatible with MinGW. So for MinGW you must build the libraries from source.

1. Make sure that you have a working MinGW OpenGL environment as described above. These instructions assume you installed it in the default location **C:\MinGW\lib**.
2. Download the GLEW source from here and unzip it.

3. In order to build GLEW, you need a Unix shell. If you installed MinGW as described above, it would have installed MSYS. To temporarily use MSYS add it to your path using the command**PATH=C:\MinGW\msys\1.0\bin;%PATH%**. Then start a shell by typing **bash**.
4. Use the **cd** command to change the current directory to where you unpacked GLEW. Note the folder **C:\foo\bar** is called **/c/foo/bar** in MSYS.
5. Compile GLEW using the command **make**. The makefile should automatically detect that you are using MinGW.
6. Install GLEW using the command **make GLEW_DEST=/c/mingw install**. This will put the header files in **C:\MinGW\include** and the libraries in **C:\MinGW\lib**.
7. You can now close the window. GLEW will be available from a regular command prompt.

To compile and link foo.c using MinGW and GLEW you need
gcc -DUSEGLEW -Wall -ofoo foo.c -lglew32 -lglut32cu -lglu32 -lopengl32
The -DUSEGLEW compiler flag defines the USEGLEW variable which my codes use to conditionally compile in GLEW.

Windows: DevC++

Jeremy Salter provided this writup for installing OpenGL with DevC++.

Windows: Other

Your Windows system may provide OpenGL 1.1, which dates from 1995. This version will not contain the glWindowPos2i function which I use a lot. You can provide the same functionality in software by including this code in your build. DO NOT include this code in what you submit, as it will result in a conflict in Linux or OSX systems. You do not need to install this code if you use the **glut32cu** library described above as this library already adds this code.

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.
- Try compiling ex5.c. If you get errors like
- **error: GL/glut.h: No such file or directory**
- or
- **error: 'GL_COLOR_BUFFER_BIT' undeclared**
- the compiler cannot find yout OpenGL header files. Check that the 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 ex5.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/lib64** or use the **-L** flag to tell the linker where it is.
- If you get **warning: implicit declaration of function 'glWindowPos2i'** try adding
- **#define GL_GLEXT_PROTOTYPES**
- before any includes.
- If you get **undefined reference to glWindowPos2i** you may have an old GL library. Link in my glWindowPos code or use **glut32cu.a** with MinGW.