Programming OpenGL in C/C++

How To Setup and Get Started

We need a C/C++ compiler, either GCC (GNU Compiler Collection) from MinGW or Cygwin (for Windows), or Visual C/C++ Compiler, or others.

We need the following sets of libraries in programming OpenGL:

1. **Core OpenGL (GL)**: consists of hundreds of functions, which begin with a prefix "gl" (e.g., glColor, glVertex, glTranslate, glRotate). The Core OpenGL models an object via a set of geometric primitives, such as point, line, and polygon.
2. **OpenGL Utility Library (GLU)**: built on-top of the core OpenGL to provide important utilities and more building models (such as qradric surfaces). GLU functions start with a prefix "glu" (e.g., gluLookAt, gluPerspective)
3. **OpenGL Utilities Toolkit (GLUT)**: provides support to interact with the Operating System (such as creating a window, handling key and mouse inputs); and more building models (such as sphere and torus). GLUT functions start with a prefix of "glut" (e.g., glutCreatewindow, glutMouseFunc).   
   Quoting from the [opengl.org](http://www.opengl.org/resources/libraries/glut/): "GLUT is designed for constructing small to medium sized OpenGL programs. While GLUT is well-suited to learning OpenGL and developing simple OpenGL applications, GLUT is not a full-featured toolkit so large applications requiring sophisticated user interfaces are better off using native window system toolkits. GLUT is simple, easy, and small."  
   Alternative of GLUT includes SDL, ....
4. **OpenGL Extension Wrangler Library (GLEW)**: "GLEW is a cross-platform open-source C/C++ extension loading library. GLEW provides efficient run-time mechanisms for determining which OpenGL extensions are supported on the target platform." Source and pre-build binary available at <http://glew.sourceforge.net/>.

Each of the software package consists of:

1. A *header* file: "gl.h" for core OpenGL, "glu.h" for GLU, and "glut.h" (or "freeglut.h") for GLUT, typically kept under "include\GL" directory.
2. A *static library*: for example, in Win32, "libopengl32.a" for core OpenGL, "libglu32.a" for GLU, "libglut32.a" (or "libfreeglut.a" or "glut32.lib") for GLUT, typically kept under "lib" directory.
3. An optional *shared library*: for example, "glut32.dll" (for "freeglut.dll") for GLUT under Win32, typically kept under "bin" or "c:\windows\system32".

It is important to locate the *directory path* and the *actual filename* of these header files and libraries in your operating platform in order to properly setup the OpenGL programming environment.

**1.  Eclipse CDT with Cygwin or MinGW**

**1.1  Installing Eclipse CDT / Cygwin or MinGW, OpenGL, GLU and GLUT**

**Step 1: Setup the Eclipse CDT (C Development Toolkit)**

Read "[How to install Eclipse CDT](https://www3.ntu.edu.sg/home/ehchua/programming/howto/EclipseCpp_HowTo.html)".

**Step 2: Setup a GCC Compiler**

We could use either MinGW or Cygwin.

* **MinGW:** For MinGW, we need to install GLUT separately. Download freeglut (@ <http://freeglut.sourceforge.net/index.php>). I recommend using the pre-package version for MinGW (freeglut 2.8.0 MinGW Package) available at <http://www.transmissionzero.co.uk/software/freeglut-devel/>.  
  Download, unzip and copy header files from "include\GL" to "<MINGW\_HOME>\include\GL"; the libraries from "lib" to "<MINGW\_HOME>\lib", and shared library from "bin" to "<MINGW\_HOME>\bin" (which should be included in the PATH environment variable), where <MINGW\_HOME> is the MinGW installed directory.  
  Take note of the headers and libraries:
  1. Headers: the OpenGL header "gl.h", GLU header "glu.h" and GLUT header "glut.h" (or "freeglut.h") are kept in "<MINGW\_HOME>\include\GL" directory. Since "<MINGW\_HOME>\include" is in the implicit *include-path*. We can include the headers as <GL/glut.h>, <GL/glt.h>, and <GL/gl.h>.
  2. Libraries: the OpenGL library "libopengl32.a", GLU library "libglu32.a" and GLUT library "libfreeglut.a" are kept in "<MINGW\_HOME>\lib" directory. This directory is in the implicit *library-path*.  
     Nonetheless, we need to include these libraries in linking. They shall be referred to as "opengl32", "glu32", "freeglut" without the prefix "lib" and suffix ".a".

(Alternatively, you could download Nate Robin's original Win32 port of GLUT from @ <http://www.xmission.com/~nate/glut.html>, which has not been updated since 2001. Download, unzip and copy "glut.h" to "<MINGW\_HOME>\include\GL", "glut32.lib" to "<MINGW\_HOME>\lib", and "glut32.dll" to "<MINGW\_HOME>\bin" (which should be included in the PATH))

* **Cygwin:** We need to install "gcc", "g++", "gdb", "make" (under the "Devel" category) and "opengl", "freeglut" (under the "Graphics" category).
  1. Headers: the OpenGL header "gl.h", GLU header "glu.h", and GLUT header "glut.h" are provided in the "<CYGWIN\_HOME>\usr\include\w32api\GL" directory. As "<CYGWIN\_HOME>\usr\include\w32api" is in the implicit *include-path*. We can include the headers as <GL/glut.h>, <GL/glt.h>, and <GL/gl.h>.
  2. Libraries: the OpenGL library "libopengl32.a", GLU library "libglu32.a" and GLUT library "libglut32.a" are provided in the "<CYGWIN\_HOME>\lib\w32api" directory. This directory is in the implicit *library-path*.  
     Nonetheless, we need to include these libraries in linking. They shall be referred to as "opengl32", "glu32", "glut32" without the prefix "lib" and suffix ".a".

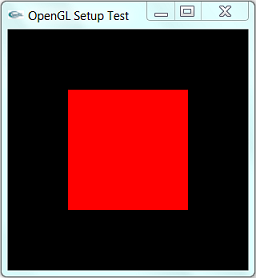
**Step 3: Configuring the Include-Path, Lib-Path and Library:** We can configure on per-project basis by right-click on the project ⇒ Properties ⇒ C/C++ general ⇒ Paths and Symbols ⇒ Use "Includes" panel to configure the Include-Path; "Library Paths" panel for the Lib-Path; and "Libraries" panel for individual libraries. We will do this later.

On command-line (for GCC), we could use option -I<dir> for include-path, -L<dir> for lib-path, and -l<lib> for library.

**1.2  Writing Your First OpenGL Program**

1. Launch Eclipse.
2. Create a new C++ project: Select "File" menu ⇒ New ⇒ Project... ⇒ C/C++ ⇒ C++ Project ⇒ Next.  
   In "Project name", enter "Hello" ⇒ In "Project type", select "Executable", "Empty Project" ⇒ In "Toolchain", select "Cygwin GCC" or "MinGW GCC" (depending on your setup) ⇒ Next ⇒ Finish.
3. Create a new Source file: Right-click on the project node ⇒ New ⇒ Other... ⇒ C/C++ ⇒ Source file ⇒ Next.  
   In "Source file", enter "GL01Hello.cpp" ⇒ Finish.
4. In the editor panel for "GL01Hello.cpp", type the following source codes:  
   NOTE: For Windows, you should include "windows.h" header before the OpenGL headers.
5. /\*
6. \* GL01Hello.cpp: Test OpenGL C/C++ Setup
7. \*/
8. #include <windows.h> // For MS Windows
9. #include <GL/glut.h> // GLUT, includes glu.h and gl.h
11. /\* Handler for window-repaint event. Call back when the window first appears and
12. whenever the window needs to be re-painted. \*/
13. void display() {
14. glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Set background color to black and opaque  
     glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer
16. // Draw a Red 1x1 Square centered at origin
17. glBegin(GL\_QUADS); // Each set of 4 vertices form a quad
18. glColor3f(1.0f, 0.0f, 0.0f); // Red
19. glVertex2f(-0.5f, -0.5f); // x, y
20. glVertex2f( 0.5f, -0.5f);
21. glVertex2f( 0.5f, 0.5f);
22. glVertex2f(-0.5f, 0.5f);
23. glEnd();
25. glFlush(); // Render now
26. }
28. /\* Main function: GLUT runs as a console application starting at main() \*/
29. int main(int argc, char\*\* argv) {
30. glutInit(&argc, argv); // Initialize GLUT
31. glutCreateWindow("OpenGL Setup Test"); // Create a window with the given title
32. glutInitWindowSize(320, 320); // Set the window's initial width & height
33. glutInitWindowPosition(50, 50); // Position the window's initial top-left corner
34. glutDisplayFunc(display); // Register display callback handler for window re-paint
35. glutMainLoop(); // Enter the infinitely event-processing loop
36. return 0;

}



1. Configuring the "include-paths", "library-paths" and "libraries":  
   Right-click on the project ⇒ Property ⇒ C/C++ general ⇒ Paths and Symbols.  
   Open the "Libraries" tab ⇒ Add ⇒ Enter "glut32" (Cygwin) or "freeglut" (MinGW with freeglut) ⇒ Add ⇒ Enter "glu32" ⇒ Add ⇒ Enter "opengl32".  
   There is no need to configure the "include-paths" and "library-paths", as they are implicitly defined.
2. Build (right-click on the project node ⇒ "Build Project") and Run (right-click on the project node ⇒ Run As ⇒ Local C/C++ Application).

**1.3  GCC Compilation in Command-line**

Observe the Eclipse's console. It uses the following commands to build the project:

> **g++ -O0 -g3 -Wall -c -fmessage-length=0 -o GL01Hello.o "..\\GL01Hello.cpp"**

// -c indicates compile-only

// -o <*filename*>: specifies the output filename

// -Wall shows all warning messages

// -O0 specifies the optimization level

// -g3 specifies the debugging information level, level 3 with extra information

// -fmessage-length=0 specifies the formatted error message length, 0 for no line wrapping

> **g++ -o GL01Hello.exe GL01Hello.o -lglu32 -lopengl32 -lfreeglut**

// -o <*filename*>: specifies the output filename

// -l<*lib*> specifies the library to be linked, which should be in the lib-path

The most important option is -l<lib>, which specifies the "library" to be linked. You might also need -I<dir> to specify the "include-path" and -L<dir> to specify the "lib-path", if the header files and libraries are not kept in implicit directories used by the compiler.

You can find the commands used in Eclipse CDT for build the project at ".metadata\.plugins\org.eclipse.cdt.ui\global-build.log".

**2.   CodeBlocks / MinGW**

**2.1  Installing CodeBlocks, MinGW and GLUT**

1. Install CodeBlocks and MinGW: Read "[How to install CodeBlocks](https://www3.ntu.edu.sg/home/ehchua/programming/howto/CodeBlocks_HowTo.html)".
2. Install GLUT: Refer to the above ["step](https://www3.ntu.edu.sg/home/ehchua/programming/opengl/HowTo_OpenGL_C.html#mingw_glut)" in Eclipse/MinGW setup.

**2.2  Writing Your First OpenGL Program**

1. Create a new project: File ⇒ New ⇒ Project... ⇒ Console Application ⇒ GO ⇒ C++ ⇒ In "Project title", enter "hello" ⇒ Next ⇒ Finish.
2. Open "main.cpp", and replace with the code, as shown in the previous section.
3. Configure the libraries: Right-click on the project ⇒ Build Option... ⇒ Linker Settings ⇒ In "Link Libraries" ⇒ Add ⇒ enter "freeglut" ⇒ Add ⇒ enter "glu32" ⇒ Add ⇒ enter "opengl32".  
   There is no need to configure the "include-paths" and the "library-paths", as they are implicitly defined.
4. Build (right-click on the project ⇒ Build) and Run (Select "Build" menu ⇒ Run).

**2.3  GCC Compilation in Command-line**

Refer to the above "[section](https://www3.ntu.edu.sg/home/ehchua/programming/opengl/HowTo_OpenGL_C.html#gcc_command_line)" on "Eclipse with GCC".

**3.   Visual C++ 2010 Express**

**3.1  Installing VC++, OpenGL, GLU and GLUT**

You need to install:

1. **Visual C++ Express 2010**: Read "[How to install Visual C++ Express](https://www3.ntu.edu.sg/home/ehchua/programming/howto/Vcpp_HowTo.html#vcpp-install)". VC++ would be installed in "C:\Program Files\Microsoft Visual Studio 10.0\VC", with headers in sub-directory "include" and libraries in "lib".
2. **Windows SDK which includes OpenGL and GLU (OpenGL Utility)**. The Visual C++ 2010 Express bundles the Microsoft Windows SDK, which would be installed in "C:\Program Files\Microsoft SDKs\Windows\v7.0A". (Otherwise, you need to download and install the Windows SDK separately).  
   The followings are used from Windows SDK:
   * gl.h, glu.h: header for OpenGL and GLU in directory "C:\Program Files\Microsoft SDKs\Windows\v7.0A\include\gl".
   * opengl32.lib, glu32.lib: libraries for OpenGL and GLU in directory "C:\Program Files\Microsoft SDKs\Windows\v7.0A\lib".
   * opengl32.dll, glu32.dll: dynamic link libraries for OpenGL and GLU in directory "C:\Windows\System32". This directory is to be included in PATH environment variable.

If you use the VC++ IDE, the include-path and lib-path would have been set correctly. If you use the CMD shell, you need to run the batch file "vcvarsall.bat" (in "C:\Program Files\Microsoft Visual Studio 10.0\VC\bin"), or "vcvars32.bat" in the earlier version, to set the environment variables.

1. **GLUT (OpenGL Utility Toolkit)**: Download Nate Robin's original Win32 port of GLUT from @ <http://www.xmission.com/~nate/glut.html> (or freeglut @ [http://freeglut.sourceforge.net](http://freeglut.sourceforge.net/)). Unzip and copy "glut.h" to "C:\Program Files\Microsoft SDKs\Windows\v7.0A\include\gl", "glut32.lib" to "C:\Program Files\Microsoft SDKs\Windows\v7.0A\lib", and "glut32.dll" to "C:\Windows\System32" (that is, the same locations as OpenGL and GLU).

**3.2  Writing Your First OpenGL Program**

To Write your first OpenGL program with GLUT:

1. Launch Visual C++ 2010 Express.
2. Create a new "Win32 Console Application" project: Select "File" menu ⇒ New ⇒ Project... ⇒ In "Project Types", select "Visual C++", "Win32". In "Templates", select "Win32 Console Application". In "Location", set your working directory. In "Name", enter "hello" ⇒ Next ⇒ Check "Empty Project" ⇒ Finish.
3. Create a new Source file: Right-click on the "Source Files" of the project name ⇒ Add ⇒ New Item... ⇒ In "Categories", select "Visual C++", "Code". In "Templates", select "C++ File (.cpp)". In "Name", type "GL01Hello.cpp" ⇒ Add.
4. In the editor panel for "GL01Hello.cpp", replace with the above code (in the previous section).
5. Build the solution ("Build" menu ⇒ Build Solution) and run the program ("Debug" menu ⇒ "Start Without Debugging").

**3.3  Common OpenGL Programming Errors (for VC++)**

* Linkage Error: Right-click on the project name ⇒ In "Configuration" drop-down menu, select "All Configurations" (i.e., release and debug) ⇒ Expand the "Configuration Properties" node. Expand the "Linker" sub-node ⇒ Select "Input" ⇒ In "Additional Dependencies", type "opengl32.lib glu32.lib glut32.lib".  
  Alternatively, you could also use pre-processor directives to instruct compiler about the libraries used (I prefer this approach, which avoids manual configuration for every project):
* #ifdef \_MSC\_VER // Check if MS Visual C compiler
* # pragma comment(lib, "opengl32.lib") // Compiler-specific directive to avoid manually configuration
* # pragma comment(lib, "glu32.lib") // Link libraries

#endif

* "error C2664: 'xxxx' : cannot convert parameter 2 from 'const char [xx]' to 'LPCWSTR'": Right-click on the project name ⇒ In "Configuration" drop-down menu, select "All Configurations" (i.e., release and debug) ⇒ Expand the "Configuration Properties" node ⇒ General ⇒ Character Set ⇒ Set to "Use Multi-Byte Character Set".  
  Alternative, you can use preprocessor directives:
* #ifdef \_MSC\_VER // Check if MS Visual C compiler
* # ifndef \_MBCS
* # define \_MBCS // Uses Multi-byte character set
* # endif
* # ifdef \_UNICODE // Not using Unicode character set
* # undef \_UNICODE
* # endif
* # ifdef UNICODE
* # undef UNICODE
* # endif

#endif

* Cannot find "glaux.h": Try remove "#include <gl/glaux.h>" from the source code. (Nehe's tutorial codes included a header "gl\glaux.h" for reading BMP file from Lesson #6 onwards. Lesson #1 to #5 do not require this header. GLUAX is obsolete and excluded from Windows SDK v6.0a.) If that does not work, you may have to download and install GLUAX package.

**3.4  VC Compilation in Command-line**

You can use the following command-line options to compile and link an OpenGL program. (You should study the command-line options produced by VC++. Right-click on the project name ⇒ Properties ⇒ Expand "Configuration Properties" node ⇒ Expand "C/C++" or "Linker" sub-node ⇒ Command-line.) Assume that the headers (gl.h, glu.h, glut.h) path is set in the INCLUDE environment variable, and the link libraries (opengl32.lib, glu32.lib, glut32.lib) path is set in the LIB environment variable:

> **cl /D "\_MBCS" /c FirstOpenGLProgram.cpp**

/c option for compile only without linking,

/D "\_MBCS" for multi-byte character set,

/D "\_UNICODE" for Unicode character set.

> **link /link opengl32.lib glu32.lib glut32.lib kernel32.lib user32.lib gdi32.lib FirstOpenGLProgram.obj**

/link option specifies the link libraries

**4.  Nehe's OpenGL Tutorial**

Nehe's production maintains an excellent OpenGL tutorial @ [http://nehe.gamedev.net](http://nehe.gamedev.net/).

OpenGL is language neutral, i.e., you can write your OpenGL programs in Windows, Java, Linux, Mac, etc. Under Windows, there are again many ways to write a OpenGL program. e.g., Win32 API directly, GLUT or FreeGLUT, SDL (Single DirectMedia Layer), Allegro, etc.

Nehe's lessons are explained in Win32 API, which is complex, cumbersome and hard and to understand. Nonetheless, the solutions are also ported to all the other platforms. I strongly suggest that you follow the solution using GLUT (which greatly simplifies interaction to Windows).

[**Link to OpenGL/Computer Graphics References and Resources**](https://www3.ntu.edu.sg/home/ehchua/programming/opengl/References_OpenGL.html)

**Introduction**

OpenGL is a popular and widely implemented graphics specification, and is often a good choice when you need to write programs featuring 2D or 3D graphics. However, OpenGL defines how graphics should be drawn, and it is up to the programmer to handle things such as window definition and input from the mouse and keyboard. This can be performed using API specific to the windowing system, which of course requires that the programmer have a good knowledge of that API. Alternatively, libraries such as GLUT or SDL may be used. These provide an abstraction layer from the windowing system, with the added advantage that programs may be written which are independent of architecture and operating system (assuming of course that appropriate libraries are available on that platform).

This tutorial focuses on GLUT, and explains how to develop OpenGL applications for Windows using the C or C++ languages with [MinGW](http://www.mingw.org/). MinGW was chosen because it is freely available for download, and because its usage will be familiar to anyone who has used GCC.

I will cover the setting up of both freeglut and GLUT for Win32 in this tutorial, as these are the two most widely used versions of GLUT (there is OpenGLUT too, but this isn’t actively developed). I’ve created packages of both freeglut and GLUT for Win32 which you can download and use with MinGW.

**Setting Up MinGW**

It is assumed that you already have MinGW installed on your PC. If not, this is covered very well in the [MinGW “Getting Started” Wiki](http://www.mingw.org/wiki/Getting_Started). When setting up MinGW, the C compiler will be installed by default. No other MinGW components are needed in order to build freeglut applications, but you’ll probably want to install the C++ compiler as well if you intend to use C++.

**Setting Up freeglut With MinGW**

freeglut is intended to be a 100% compatible replacement for the original GLUT libraries. Unlike GLUT for Win32, freeglut is an up to date and actively maintained project, so I would recommend using freeglut if you have the choice. You can download my [freeglut MinGW package](https://www.transmissionzero.co.uk/files/software/development/GLUT/freeglut-MinGW.zip) (with [PGP signature](https://www.transmissionzero.co.uk/files/software/development/GLUT/freeglut-MinGW.zip.asc) and [PGP key](https://www.transmissionzero.co.uk/files/transmissionzero-pgp.asc)), which contains import libraries, headers, and a Windows DLL. Using this, you can either dynamically link against the freeglut DLL, or statically link the library into your application.

Once you have downloaded the freeglut MinGW package, create a folder on your PC which is readable by all users, for example “C:\Program Files\Common Files\MinGW\freeglut\”. Copy the “lib\” and “include\” folders from the zip archive to that location.

The freeglut DLL should either be placed in the same folder as your application, or can be installed in a system-wide folder which appears in your %PATH% environment variable. On a 32 bit Windows system this is typically “C:\Windows\System32\”, and on a 64 bit Windows system this is typically “C:\Windows\SysWOW64\”.

**Compiling freeglut Applications With MinGW**

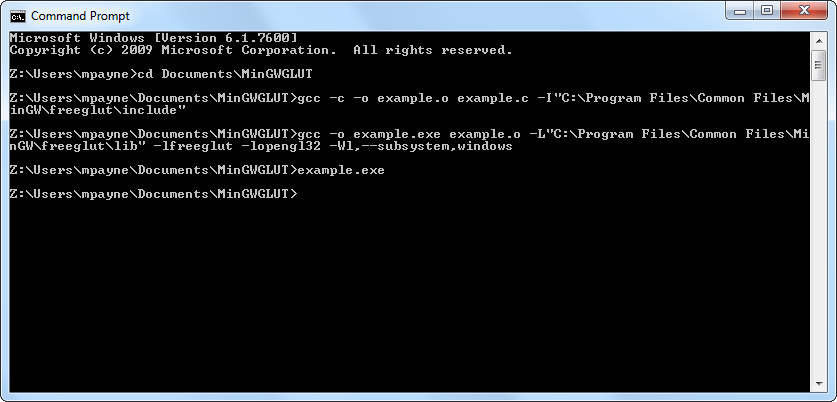
To keep your applications compatible with GLUT, you should “#include <GL/glut.h>”. If you want to use freeglut specific extensions, you can “#include <GL/freeglut.h>” instead (bear in mind that if you’re using freeglut specific extensions, your code will likely need to be modified in order to build with a different GLUT implementation).

When compiling a freeglut application, it’s necessary to pass the path of the “include\” folder you created to the compiler using the “-I” option. Similarly when linking, it’s necessary to pass the path of the “lib\” folder to the linker using the “-L” option. In order to link to the freeglut and OpenGL libraries, you should additionally specify “-lfreeglut -lopengl32” during the link stage.

Given a source file “example.c”, which you want to compile to an application “example.exe”, you can compile and link it with the following commands (replacing the include and lib paths with the ones you created above if necessary):

**gcc -c -o example.o example.c -I"C:\Program Files\Common Files\MinGW\freeglut\include"**

**gcc -o example.exe example.o -L"C:\Program Files\Common Files\MinGW\freeglut\lib" -lfreeglut -lopengl32 -Wl,--subsystem,windows**



The “-Wl,--subsystem,windows” on the command line ensures the executable runs as a Windows GUI application rather than a console application. If you’re using GLU functions, you’ll also need to additionally specify “-lglu32” on the linker command line.

When distributing your application, don’t forget to include the freeglut DLL, or at least provide your users with some method of obtaining a copy.

**Statically Linking freeglut**

It’s possible statically link freeglut into your applications, instead of dynamically linking against the DLL as detailed above. The main disadvantage of static linking is that when an updated version of freeglut becomes available, the application must be recompiled in order to use the newer version. This is much more effort than when using dynamic linking, where it’s only necessary to deploy the newer version of the DLL—if your user has freeglut knowledge, they could even do this for their self! In any case, if you do want to use static linking it’s fairly simple.

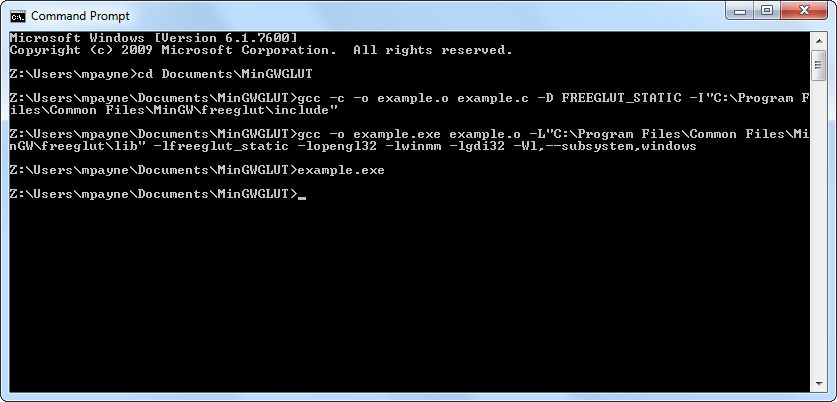
The compilation step is almost the same as before, except that you need to define “FREEGLUT\_STATIC” before any freeglut headers are included. This is best done by adding “-D FREEGLUT\_STATIC” to the compiler command line. The linker step is also slightly different, as you must specify the static version of the freeglut library rather than the dynamic import library. It’s additionally necessary to link against the Windows multimedia library and GDI libraries, as freeglut uses functions from both of these libraries.

Given a source file “example.c”, which you want to compile to an application “example.exe”, you can compile and link it with the static freeglut library using following commands:

**gcc -c -o example.o example.c -D FREEGLUT\_STATIC -I"C:\Program Files\Common Files\MinGW\freeglut\include"**

**gcc -o example.exe example.o -L"C:\Program Files\Common Files\MinGW\freeglut\lib" -lfreeglut\_static -lopengl32 -lwinmm ^**

**-lgdi32 -Wl,--subsystem,windows**



If you get undefined references to functions when trying to statically link freeglut into your application, check your preprocessor definition and linker flags—static linking will fail if you forget to define “*FREEGLUT\_STATIC*”, or if you have defined it but are linking against wrong libraries.

**Setting Up GLUT for Win32 With MinGW**

GLUT for Win32 is a Windows port of the original GLUT library. It’s no longer maintained or supported, but it’s still a very popular GLUT package. The MinGW “*w32api*” package already comes with two GLUT import libraries, “*libglut.a*” and “*libglut32.a*”, but doesn’t come with a glut header file. If you’ve ever downloaded a GLUT header from the internet, and attempted to link an application against either of these import libraries, you likely would have seen it fail with various undefined references. This is because the import libraries are from an older version of GLUT for Win32, which doesn’t contain the “atexit hack” functions which appear in the newer header file. I have created an up to date [GLUT MinGW package](https://www.transmissionzero.co.uk/files/software/development/GLUT/GLUT-MinGW.zip) (with [PGP signature](https://www.transmissionzero.co.uk/files/software/development/GLUT/GLUT-MinGW.zip.asc) and [PGP key](https://www.transmissionzero.co.uk/files/transmissionzero-pgp.asc)) to fix all of these issues.

Once you have downloaded the GLUT MinGW package, create a folder on your PC which is readable by all users, for example “C:\Program Files\Common Files\MinGW\GLUT\”. Copy the “lib\” and “include\” folders from the zip archive to that location.

The GLUT DLL should either be placed in the same folder as your application, or can be installed in a system-wide folder which appears in your %PATH% environment variable. On a 32 bit Windows system this is typically “C:\Windows\System32\”, and on a 64 bit Windows system this is typically “C:\Windows\SysWOW64\”.

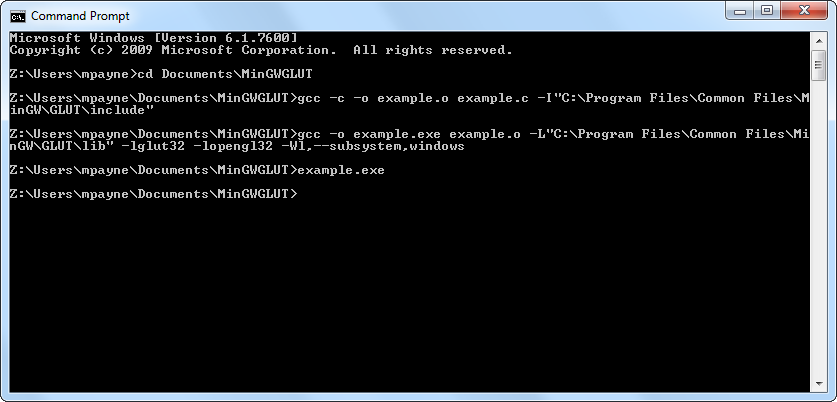
**Compiling GLUT for Win32 Applications With MinGW**

To use functions from the GLUT for Win32 library, you should “#include <GL/glut.h>” in your source code.

As when compiling GLUT applications, it’s necessary to pass the path of the “include\” folder you created to the compiler using the “-I” option, and the path of the “lib\” folder to the linker using the “-L” option. When linking a GLUT application, you should link against the GLUT for Win32 and OpenGL libraries with the flags “-lglut32 -lopengl32”. Again, adding the flag “-Wl,--subsystem,windows” will ensure it compiles as a Windows GUI application rather than a console application. The commands to compile and link a typical GLUT application would be:

**gcc -c -o example.o example.c -I"C:\Program Files\Common Files\MinGW\GLUT\include"**

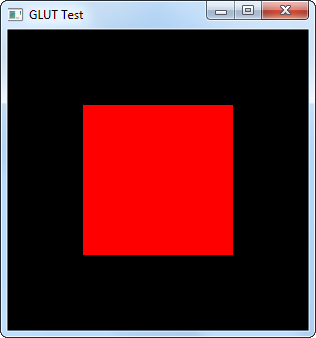
**gcc -o example.exe example.o -L"C:\Program Files\Common Files\MinGW\GLUT\lib" -lglut32 -lopengl32 -Wl,--subsystem,windows**



As with freeglut, it’s necessary to redistribute the GLUT DLL with your application, or at least provide some method for your users to obtain a copy.

**Example GLUT Application**

For an example GLUT application, take a look at the [Hello GLUT example on GitHub](https://github.com/TransmissionZero/Hello-GLUT). You can either git clone the repository or [download a Hello GLUT release](https://github.com/TransmissionZero/Hello-GLUT/releases). The application can then be built with MinGW and when executed should look like the following:



**Cross-Compiling GLUT Applications**

Although this tutorial focused on compiling GLUT applications with the Windows version of MinGW, you can of course cross-compile them on non-Windows versions. Many Linux distributions have MinGW packages available, and you can build your applications using the instructions above—the only difference being that you’ll have different library and include paths.

Under Fedora 14, I placed the libraries and headers under “/usr/local/share/MinGW/freeglut/” and built the application like this:

[mpayne@martpc MinGWGLUT]$ **i686-pc-mingw32-gcc -c -o example.o example.c -I/usr/local/share/MinGW/freeglut/include/**

[mpayne@martpc MinGWGLUT]$ **i686-pc-mingw32-gcc -o example.exe example.o -L/usr/local/share/MinGW/freeglut/lib/ \**

**-lfreeglut -lopengl32 -Wl,--subsystem,windows**

