Building OpenGL/GLUT Programs

on:

- **Linux with Eclipse,**
- **Mac OS X and Xcode, and**
- **Windows with Visual Studio and Cygwin**

Sample code: the sample Makefile and source code build a simple line drawing application and runs on Linux, Mac OS X, and Windows platforms. You can also try out some examples with buffer objects and shaders.

If you're interested in using GLFW instead of GLUT, please refer to the course note *Building OpenGL/GLFW Apps*. Please let me know if you have any correction or addition. Thanks.

To find out how to specify command line options, add to header file search path, and link with libraries such as GLEW, Expat, JPEG, and PNG, see the course note on these topics.

Linux:

Ubuntu 14.04.1 LTS (Trusty Tahr), 12.10, 12.04.5 LTS (Precise Pangolin)
Fedora 15 2.6.43.5-2.fc15.x86_64
Red Hat Enterprise Linux Workstation release 6.3 (Santiago) 2.6.32-279.el6.x86_64
gcc: 4.8.2, 4.7.2, 4.6.3, 4.4.6
Eclipse 3.7.2-1 (Indigo), 3.6.1 (Helios), 3.2.0 (Callisto)

- Installation/verification:

  OpenGL comes with the X11 installation, to verify, look for the following files:

  - libraries:
    /usr/lib/libGL*

    on Ubuntu 14.04.1:
    /usr/lib/x86_64-linux-gnu/libGL*
  - header files:
    /usr/include/GL/{gl,glu}.h

  If GLUT is installed, it will be visible as:

  - library:
    /usr/lib/libglut*

    on Ubuntu 14.04.1:
    /usr/lib/x86_64-linux-gnu/libglut*
  - header file:
    /usr/include/GL/glut.h

  If OpenGL and/or GLUT is not installed on your system, you can install both with:

  ```
  Ubuntu# sudo apt-get install freeglut3-dev
  ```
  or
  ```
  Fedora/RedHat# sudo yum install freeglut-devel
  ```

  which should install all the dependent packages, including OpenGL itself. You must have sudo/administrator privileges.

- In your OpenGL source files, include the following line:

  ```
  #include <GL/glut.h>
  ```

  You don't need to include gl.h and glu.h, as they are already included in glut.h.
  For an example, see the provided sample source code.

- To make a GLUT application on the command line, use the following linker options:

  ```
  -lGL -lGLU -lglut
  ```

- Eclipse 3.7.2 project (graphical step-by-step or older version of Eclipse):

  The figures here are from version 3.6.1, which are more or less the same as for 3.7.2.

- To install Eclipse on Ubuntu:

  ```
  sudo apt-get install eclipse eclipse-cdt
  ```

- Start up Eclipse and choose your workspace, click "OK" (Fig. 1)
- Open the C/C++ perspective: on the main menu select "Window→Open Perspective→C/C++" (Fig. 2). If the C/C++ perspective is not available, choose "Other..." and it should show up in the window that pops up.
- Create a new project:
  - Select "File→New→C++ Project" (Fig. 3)
  - Give the project a name, e.g., "sampleapp". In the panels below, specify project type, "Executable→Empty Project", and select the appropriate toolchain (make, gcc, ld, etc.). Since we're not cross compiling, I choose the "Linux GCC" toolchain. Click "Finish" (Fig. 4)
- Add source files:
  - Right click on "sampleapp" on the first line of the "Project Explorer" tab on the left and select "Import" (Fig. 5)
  - On the "Select" page, choose "General→File System" and click "Next >" (Fig. 6)
  - On the "File system" page, either type in the directory where your source file resides, or click the "Browse" button next to "From directory:" to choose the directory (Fig. 7). In browsing for the directory, remember that you're telling eclipse the directory where your source files are, not the source files themselves.
  - Once the directory is identified, click on all the source files on the right pane you want imported, for example, the provided server.c (or click the directory name on the left pane to import all files in the directory), and click "Finish" (Fig. 8)
- Add libraries:
  - Right click on "sampleapp" on the first line of the "C/C++" pane on the left and select "Properties" to tell the linker which libraries need to be added (Fig. 9)
  - On the "Properties for sampleapp" page, click "C/C++ Build→Settings". Under the "Tool Settings" tab, click "GCC C++ Linker→Libraries" (Fig. 10)
  - On the "Libraries (-l)" pane click the add file ( ) button (Fig. 11)
  - On the popup dialog box type "GL" and click "OK"
  - Repeat the above two steps to add "GLU" and "glut" libraries
  - Next click "Apply" and "OK". Be sure to click "Apply" before clicking "OK" (Fig. 11)
- Build and run the program:
  - On the main menu, select "Project→Build All" (Fig. 13)
  - Under the play button on the second menu bar ( ), click on the drop-down menu and select "Run Configurations..."" (Fig. 14)
  - On the "Create, manage, and run configurations" page, double click on "C/C++ Application" (Fig. 15). It will create a "sampleapp Debug" application.
  - On the "sampleapp Debug" configuration page, check "Enable auto build" and click "Run" (Fig. 16)
  - Click on the "Console" tab on the lower part of the central pane to view the console output of the program, if any. Click on the red square to stop the program. (Fig. 17).

Mac OS X:

Mavericks 10.9.4 (13E28), Lion 10.7.2 (11C2002), SnowLeopard 10.6.4 (10FS69), Leopard 10.5.8 (9L30), Tiger 10.4.11 (8S165)
Darwin 13.3.0, 11.2.0, 10.4.0, 9.8.0, 8.11.0
Xcode: 5.1.1, 4.2.1, 3.2.4, 3.1.2, 2.0
gcc: 4.2.1, 4.0.1, 4.0.0
Apple LLVM version 5.1 (clang-503.0.40) (based on LLVM 3.4svn)

- Installation/verification:

  OpenGL and GLUT come with the OS and Xcode installations. To verify, check for:

  ```
  /System/Library/Frameworks/{OpenGL,GLUT}.framework
  ```

  [The installed GLUT is the original GLUT not freeglut.]

- In your OpenGL source files, include the following line:

  ```
  #include <GLUT/glut.h>
  ```

Despite Apple's documentation, you don't need to include gl.h and glu.h, as they are already included in glut.h. For an example, see the provided sample source code.

- To make a GLUT application on the command line, use the following linker options:

  -framework OpenGL -framework GLUT

- Xcode 5.1.1 or 4.2.1 project (graphical step-by-step or older versions of Xcode):

  The figures here are from version 4.2.1, which are more or less the same as for 5.1.1.

  - Create a new project:
    - Select "File→New→New Project" (Fig. 18).
    - Choose a template for your new project under Mac OS X (not iOS): "Application→Command Line Tool" and click "Next" (Fig. 19).
    - For the sample code, specify "Product Name" as "sampleapp", "Company Identifier" as "sample", select "Type" "C++", (for 4.2.1, uncheck "Use Automatic Reference Counting"--it's only useful for Objective C), and click "Next" (Fig. 20).
    - A window pops up to prompt you where you want to put the new project folder. In this example I've decided to put the project folder on my Desktop. Uncheck "Create local git repository for this project" if you don't want it. Then click "Create" (Fig. 21).
    - In the leftmost pane, under "sampleapp" select "main.cpp" and "sampleapp.1" and right click to Delete them (Fig. 22). Click "Delete" or "Move to Trash" on the confirmation dialog box.
  - Add source files:

    Right click on "sampleapp" at the top of the left most pane and select "Add Files to 'sampleapp'" (Fig. 23), select your file, for example, the provided sample.c. You can choose whether to copy added items into the project folder. I've chosen not to make copy of added items but to leave them in the project folder instead, in this example. Click "Add" (Fig. 24).
  - Add libraries:
    - In the middle pane, click the "Build Phases" tab, open the "Link Binary With Libraries" list, and click the '+' sign at the bottom left corner of the list (Fig. 25).
    - A list of frameworks should pop up. Scroll down the list to "OpenGL.framework" and click "Add" (Fig. 26). Repeat for "GLUT.framework".
  - To suppress "Deprecations" warnings for GLUT, in the middle pane, click the "Build Settings" tab. On the second menu of the panel, click on "All". Scroll down (or search for) "Apple LLVM 5.1 - Custom Compiler Flags". Click to the right of "Other C Flags" and type in "-Wno-deprecated" (Fig. 26bis).
  - Click on 

to build and run the program.

Windows:

8.1 Enterprise, 7 Enterprise, Vista x64 EE SP2 (Build 6002), Vista EE SP2 (Build 6002), XP PE SP3 5.1 (Build 2600)
Visual Studio 2013 12.0.30723.00 Update 3, 2012 11.0.51106.01 Update 1, 2010 10.0.420219.1 SP1Rel, 2008 9.0.21022.8.RTM
.NET Framework 4.5.51641, 4.5.50709, 4.0.30319 SP1Rel, 3.5 SP1

- Installation/verification:

  OpenGL comes with the OS and Visual Studio installations, to verify:

  - header files:
    C:\Program Files (x86)\Windows Kits\8.1\Include\um\gl\{GL,GLU}.h
    (Your installation may have something other than um on the path.)

    For earlier versions of Visual Studio:
    C:\Program Files (x86)\Microsoft SDKs\Windows\v7.1A\Include\gl\{GL,GLU}.h
    [without the " (x86)" for 32-bit Windows; VS2010: v7.0A, VS2008: v6.0A]
  - static library (32- and 64-bit respectively):
    C:\Program Files (x86)\Windows Kits\8.1\Lib\winv6.3\um\x86\{OpenGL32,GlU32}.Lib
    C:\Program Files (x86)\Windows Kits\8.1\Lib\winv6.3\um\x64\{OpenGL32,GlU32}.Lib

    Earlier versions of Visual Studio:
    C:\Program Files (x86)\Microsoft SDKs\Windows\v7.1A\Lib\{OpenGL32,GlU32}.Lib

- runtime, dynamically linked (dll), shared libraries:
  C:\Windows\SysWOW64\{opengl,glu}32.dll

  On 32-bit Windows:
  C:\Windows\System32\{opengl,glu}32.dll

Out of the box, Windows doesn't come with GLUT installed. To get both a 32- and 64-bit GLUT library, you can install Nvidia's Cg toolkit. Once Cg is installed, the GLUT files will be in C:\Program Files (x86)\NVIDIA Corporation\Cg folder. You can then distribute the GLUT files as follows:

- header file:
  C:\Program Files (x86)\Microsoft Visual Studio *\VC\include\GL\glut.h
  The '*' matches your version of VS: 12.0 for VS2013, 11.0 for VS2012, 10.0 for VS2010, 9.0 for VS2008. You may have to create the include folder.

  On earlier versions without a VC folder, try instead:
  C:\Program Files\Microsoft SDKs\Windows\v7.0A\Include\GL\glut.h

- library file:
  C:\Program Files (x86)\Microsoft Visual Studio *\VC\lib\glut32.lib

  If you don't have a VC folder, try instead:
  C:\Program Files\Microsoft SDKs\Windows\v7.0A\Lib\glut32.lib

  If you want to build 64-bit apps, put the 64-bit library file in:
  C:\Program Files (x86)\Microsoft Visual Studio *\VC\lib\amd64\glut32.lib

- runtime library:
  C:\Program Files (x86)\Microsoft Visual Studio *\VC\bin\glut32.dll

  If you don't have a VC folder, try instead:
  C:\Windows\system\glut32.dll

  To build 64-bit apps, put the 64-bit runtime library file in:
  C:\Program Files (x86)\Microsoft Visual Studio *\VC\bin\amd64\glut32.dll

It has been reported that the stdlib header file that comes with Visual Studio 2013 has compatibility issue with the header file that comes with the 32-bit glut-3.7.6.zip (web site), in particular with the redefinition of exit(). So you may want to avoid using this library. There is also the freeglut library. However, freeglut on Windows makes duplicate calls to the display callback handler. This means your display handler must be safe to be run multiple times, which could require extra memory copying to save and restore states. It also is a performance hit as each display redrawn must be done twice. I suggest you don't use freeglut on Windows in EECS 487.

- In your OpenGL source files, include the following line:

```
#include <GL/glut.h>
```

  You don't need to include gl.h and glu.h, as they are already included in glut.h.
  For an example, see the provided sample source code.

- Command line make: see the instructions for Cygwin below.

- Visual Studio 2013 project (graphical step-by-step or VS2008 version):

  - Create a new project:

    - Select "File→New→Project" (Fig. 27)
    - Choose a template for your new project:
      "Visual C++→Win32→Win32 Console Application" (even if you're building 64-bit app), give the project a name, e.g., "sampleapp", specify the location of the project, and click "OK" (Fig. 28)
    - On the "Welcome to the Win32 Application Wizard" page, click "Next >" (Fig. 29)
    - On the "Application Setting" dialog box, under "Additional options" tick "Empty project", untick "Security Development Lifecycle (SDL) checks", then click "Finish" (Fig. 30)
    - To build a 64-bit app, click on the drop down menu next to the third textbox on the second menu bar at the top of the window ("Solution Platform" should show up when you mouse over it) and choose "x64" instead of "Win32" (Fig. 31). If you don't have "x64" as an option, select "Configuration Manager" and create a new platform definition by simply copying the Win32 configuration when prompted.

  - Add source files:

    Right click on your project, for example, "sampleapp", on the second line of the "Solution Explorer" pane on the left to "Add→Existing Item" (Fig. 32), select your source and header file(s), for example, the provided sample.c, and click "Add" (Fig. 33).

  - Add libraries:

- Right click on "sampleapp" again and select "Properties" (Fig. 34)
- On the "Property Pages", under "Configuration:" specify "All Configurations" (Fig. 35)
- Select "Configuration Properties→Linker→Input" on the left pane. Next to "Additional Dependencies" at the top of the right pane enter:

  opengl32.lib;glu32.lib;glut32.lib;
  [VS2008: with spaces in place of the semicolons]

  Be careful **not** to add a space before or after the semicolon. Hit RETURN and then click "Apply" (Fig. 36).

- Optional:

  To prevent your program from opening a console window, while still on the "Property Pages", select "Configuration Properties→Linker→Command Line" on the left pane. Under "Additional options" at the bottom of the right pane enter:

  /SUBSYSTEM:WINDOWS /ENTRY:mainCRTStartup

  then click "Apply" (Fig. 37). You may not want to disable the console window if you print out messages to the console (see next step).

- Close the "Property Pages" pane by clicking "Ok", then click on the play button on the second menu bar (

) to build and run the program. If you print out messages to the console, run the program using Ctl-F5 instead, to keep the cmd window from exiting after the program exits.

- To distribute your program, include glut32.dll with your distribution. Unfortunately there doesn't seem to be any precompiled statically linked GLUT library for Windows.

Cygwin:

1.7.31-3 (64 bit NT-6.3)
compiler tool chains:
  x86_64-pc-cygwin: gcc 4.8.3 (to generate 64-bit Win32 apps)
  i686-pc-cygwin: gcc 4.8.3 (to generate 32-bit Win32 apps)
  i686-pc-mingw32: gcc 4.7.3 (original MinGW to generate 32-bit Win32 apps)
  i686-w64-mingw32: gcc 4.8.3 (MinGW w64 to generate 32-bit Win32 apps)
  x86_64-w64-mingw32: gcc 4.8.3 (MinGW w64 to generate 64-bit Win64 apps)

Install Cygwin/X11 by downloading and running its setup program. If you're installing 64-bit version, x86_64-pc-cygwin is the compiler toolchain, i.e., compiler, linker, archiver, etc. you get by default. As installed, Cygwin builds X11 (OpenGL/GLX) apps instead of native Windows (OpenGL/WGL) apps. Unfortunately GLX programs on Cygwin that use buffer object segfault on the call to glBindBuffer() and those that use shaders segfault on the call to glCreateProgram(). So we'll build WGL apps instead. The compiler toolchain that comes with Cygwin can build WGL apps (see Building OpenGL/GLFW Apps), but unfortunately there is no GLUT library that works with it. The available freeglut binaries for MinGW makes duplicate calls to the display callback hanlder as noted above. The Nvidia's GLUT binaries were evidently not compiled to support large memory model, resulting in "relocation truncated to fit" error messages from the linker. Fortunately, the MinGW x84_64 compiler toolchain can work with the Nvidia's GLUT libraries, so we'll use this toolchain instead of the native Cygwin toolchain.

- Installation and verification:

  Use the Cygwin setup program to install the MinGW x86_64 compiler toolchain:

  Devel→mingw64-x86_64-gcc-g++

  The setup program will figure out, download, and install all necessary dependencies. To use WGL instead of GLX under Cygwin, you need to link your program against native Win32 graphics libraries. Let YOUR_TOOLCHAIN be x86_64-w64-mingw32 and replace every occurrences of YOUR_TOOLCHAIN below with x86_64-w64-mingw32 (we'll see the use of other toolchains later). Verify that you have the Win32 version of OpenGL and GLU installed:

  /usr/YOUR_TOOLCHAIN/sys-root/mingw/lib/libopengl32.a
  /usr/YOUR_TOOLCHAIN/sys-root/mingw/lib/libglu32.a

  with the OpenGL include files in:

  /usr/YOUR_TOOLCHAIN/sys-root/mingw/include/GL

  You would then need to add the path /usr/YOUR_TOOLCHAIN/sys_root/mingw/bin and /usr/YOUR_TOOLCHAIN/bin to your search path, i.e., your environment PATH variable, for binaries and dynamically linked libraries (dll). (As such, you can only have one toolchain in use for a given PATH environment variable.)

- To install GLUT, first install Nvidia's Cg toolkit on Windows. Then move the GLUT files from /cygdrive/c/Program\ Files\ (x86)/NVIDIA\ Corporation/Cg as follows:
  - include/GL/glut.h header file to:

    /usr/YOUR_TOOLCHAIN/include/GL/

    You'd have to create the include and GL directories.
  - lib.x64/glut32.lib static library to:

    /usr/YOUR_TOOLCHAIN/lib/libglut32.a

    When linking against a library, unlike Visual Studio, the MinGW/Cygwin linker looks for either lib<pkg>.a or <pkg>.lib, **not** lib<pkg>.lib.
  - bin.x64/glut32.dll runtime dynamically-linked, shared library to:

    /usr/YOUR_TOOLCHAIN/bin/glut32.dll

    Remember to add /usr/YOUR_TOOLCHAIN/bin to your environment PATH variable prior to launching your app, as noted above.

  If you use other libraries, e.g., GLEW, PNG, JPEG, etc. you'll need to link against the Win32 version of those libraries instead of the ones installed through Cygwin's setup program. See the course note on how to add libraries.
- Prepare your source files as you would for Visual Studio above.
- To compile, use the following linker options:

  -lglut32 -lglu32 -lopengl32

  [Some antivirus software such as Avast! may automatically move the binary you build into its virus quarantine folder. You may want to exclude your working folder from automatic virus quarantine.]

On 64-bit Cygwin, you can choose to build either 32- or 64-bit apps. If you use static libraries, you can set up Cygwin to support both. If you want to build 32-bit apps, you have three toolchains to choose from (see above for a brief summary of each). Use the Cygwin setup program to install your toolchain of choice:

Devel→cygwin32-gcc-g++
Devel→mingw64-i686-gcc-g++
Devel→mingw-gcc-g++

use the following labels, respectively, in place of YOUR_TOOLCHAIN above: i686-pc-cygwin, i686-pc-mingw32, or i686-w64-mingw32. For i686-pc-cygwin, the subdirectory under sys-root is called usr instead of mingw.

References not cited:

- Bradford, J., *Using OpenGL & GLUT in Visual Studio .NET 2003*, 2006.
- Nakano, A., *How to Install OpenGL*, 2008.
- Hewgill A. and Bockus, D., *Setting up Microsoft Visual Studio .NET for OpenGL/GLUT and Free Image*, 2005.
- Wong, A., *OpenGL on Cygwin*, 2007.
- LQWiki, *OpenGL*.
- Angel, *Sixth Edition Code*, 2012.

Last updated: Oct. 29, 2014 by Sugih Jamin
Thanks to D. Kohler for corrections.