Installing OpenGL and Running the Book's Code

We have changed our Windows/MSVS environment significantly from the second edition. The consequent change to code itself, however, is not significant and almost all to a few preprocessor directives at the top of each program. Nevertheless, keep in mind that code from the second edition will not run off the bat in the environment for the third and vice versa.

Our programs were all developed over the past few years on a Microsoft Visual Studio Community (MSVS) 2015 IDE running on Windows 10 on a 64-bit PC with a graphics card supporting OpenGL 4.3. Subsequently, they were each tested to run with MSVS Community 2017 as well.

Described below is how to set the environment up on our development platform of MSVS Commumity 2015/64-bit Windows 10. Keep in mind that, though a graphics card supporting at least OpenGL 4.3 is ideal, a lower version of OpenGL might be fine too – see a note at the end of this guide. These instructions should be applicable to other versions of Windows and Visual Studio and to a 32-bit machine as well, possibly with minor adjustments and the occasional tweak to the code itself; our programs, again possibly with minor changes, should all run on a Linux or Mac OS environment. Incidentally, Linux and Mac programmers might find useful installation suggestions through the link to the second edition at the book's website `http://www.sumantaguha.com`.

- Install Visual Studio

    1. Download Microsoft Visual Studio Community 2015 from `https://go.microsoft.com/fwlink/?LinkId=532606&clcid=0x409` and follow the installation steps.

- Install Helper Libraries

    1. Create a folder called `OpenGLwrappers` in the `C:` drive, so this folder is `C:\OpenGLwrappers`. This particular name and location is so that all our programs in `ExperimenterSource` run out of the box. If for some reason, e.g., not having access to the root drive, you can't place `OpenGLwrappers` as asked, then place it where you can. You will have to change our project properties accordingly and we'll say further on how to do this.

        The version numbers below of the libraries we ask you to download may not be the latest but they are the ones we have installed in our environment. There are sometimes niggling little things that differ with versions. To be sure, we have not yet heard of a problem in upgrading to a newer version that could not be fixed without too much trouble. Still, to avoid any issues at all while you are learning we suggest you install the exact versions below.

2. FreeGLUT: Download and unzip the file `http://files.transmissionzero.co.uk/software/development/GLUT/older/freeglut-MSVC-2.8.1-1.mp.zip` and save the folder `freeglut-MSVC-2.8.1-1.mp` in `OpenGLwrappers`.
3. GLEW: Download and unzip the file `https://sourceforge.net/projects/glew/files/glew/1.10.0/glew-1.10.0-win32.zip/download` and save the folder `glew-1.10.0-win32` in `OpenGLwrappers`.
4. GLM: Download and unzip the file `https://github.com/g-truc/glm/releases/download/0.9.7.5/glm-0.9.7.5.zip` and save the folder `glm-0.9.7.5` in `OpenGLwrappers`.
5. Copy `freeglut.dll` from `C:\OpenGLwrappers\freeglut-MSVC-2.8.1-1.mp\freeglut\bin` to `C:\Windows\SysWOW64`.
6. Copy `glew32.dll` from `C:\OpenGLwrappers\glew-1.10.0-win32\glew-1.10.0\bin\Release\Win32` to `C:\Windows\SysWOW64`.
7. Check if `glu32.dll` is already in `C:\Windows\SysWOW64`. Normally, it should be. If not, search resources on the web to reinstall it there.

Now, if, in fact, you were able to place `OpenGLwrappers` in `C:`, then you should be all set run the book programs so go ahead and test your environment. If you weren't, we'll tell you what to do momentarily.

- Test the Environment

1. Download the files `Experimenter.pdf` and `ExperimenterSource.zip` from the book's website `http://www.sumantaguha.com`. Unzip the latter file and place `ExperimenterSource` in the *same* folder as `Experimenter.pdf`.
2. Open `Experimenter.pdf` with (preferably) Adobe Reader. Go to Chapter 2 and click Windows Project just above Experiment 2.1 to bring up the Square project in MSVS.
3. On the MSVS tool bar click Build → Build Solution to compile the project, which should show 1 succeeded and 0 failed in the Output pane at the bottom.
4. On the tool bar click Debug → Start Debugging to run. This should bring up two windows: an all black C++ window and the OpenGL window with a black square drawn on a white background (Figure 1 is a screenshot of the latter window). Do you see these two windows? Great! You are all set.

Of course, you can run the programs directly in `ExperimenterSource` without going through `Experimenter.pdf`. However, `Experimenter.pdf` has the programs in the same sequence that they appear in the book so the two work nicely together.

Next, we'll show you how to make your own OpenGL project as surely you will be wanting to soon enough. First, though, we recommend creating an OpenGL project template which can then be used for all your future projects.

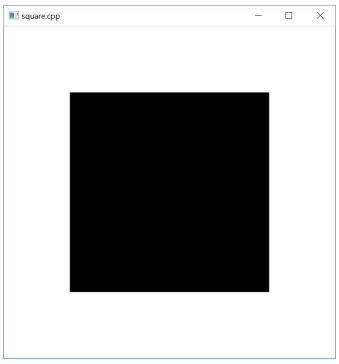

Figure 1: Screenshot of `square.cpp`, in particular, the OpenGL window.

- Create and Use an OpenGL Project Template

  1. Open Visual Studio 2015 from the Start Menu to bring up the MSVS Start Page. Click New Project and in the popup dialog box select Visual C++ → Win32 Console Application (in the middle pane). For Name enter `OpenGLProjectTemplate` and for Location any convenient folder. Leave the boxes "Create directory for solution" and "Add to Source Control" unchecked. Click OK.
  2. The Win32 Application Wizard comes up. Click Next. The radio button Console application should be selected; uncheck the boxes "Precompiled header" and "Security Development Lifecycle (SDL) checks" and check the box "Empty project". Click Finish.
  3. The MSVS page of `OpenGLProjectTemplate` comes up. On the tool bar click View → Solution Explorer to open the Solution Explorer pane. Right click Source Files → Add → New Item. Select Visual C++ → C++ File (.cpp), name the new file `test.cpp` and click Add to see `test.cpp` in a pane.
  4. Open the file `square.cpp` in the folder `ExperimenterSource\Chapter2\Square` in a text editor and copy its contents into `test.cpp`. Ignore the multiple red lines in `test.cpp` for now.
  5. On the tool bar of the MSVS page click Project → OpenGLProjectTemplate Properties to bring up the Property Pages. Select All Configurations from the dropdown menu to the right of "Configuration:".
  6. Expand Configuration Properties and click Configuration Properties → C/C++ → General → Additional Include Directories → ⟨Edit...⟩ (in the drop-down menu) to open the Additional Include Directories dialog box. Click the New Line icon successively to create a new line to add each of the following folders (order doesn't matter; you can click the box with "..." at the right of the empty new line to navigate to the folder):
     If `OpenGLwrappers` *isn't in* `C:`*, then navigate to where it is.*
     (a) `C:\OpenGLwrappers\freeglut-MSVC-2.8.1-1.mp\freeglut\include`
     (b) `C:\OpenGLwrappers\glew-1.10.0-win32\glew-1.10.0\include`
     (c) `C:\OpenGLwrappers\glm-0.9.7.5\glm`

     Click OK to confirm.
  7. Click Configuration Properties → Linker → General → Additional Library Directories → ⟨Edit...⟩ (in the drop-down menu) to open the Additional Library Directories dialog box. Click the New Line icon successively to create a new line to add each of the following folders

(order doesn't matter; you can click the box with "..." at the right of the empty new line to navigate to the folder):
If `OpenGLwrappers` *isn't in* `C:`*, then navigate to where it is.*

(a) `C:\OpenGLwrappers\freeglut-MSVC-2.8.1-1.mp\freeglut\lib`

(b) `C:\OpenGLwrappers\glew-1.10.0-win32\glew-1.10.0\lib\Release\Win32`

Click OK to confirm.

8. Click Configuration Properties → Linker → Input → Additional Dependencies → ⟨Edit...⟩ (in the drop-down menu) to open the Additional Dependencies dialog box. Manually enter the following file names one per line in the window at the top (order doesn't matter):

   (a) `glew32.lib`

   (b) `opengl32.lib`

   Click OK to confirm.

9. Click OK to dismiss the Property Pages. All the red lines in the source file `test.cpp` should now be gone. Now, repeat steps (3) and (4) of Test the Environment. Again, you should see a black C++ window and an OpenGL window as in Figure 1.

10. Expand Source Files in the Solution Explorer pane, right click `test.cpp` and Remove → Delete as it's no longer needed.

11. On the MSVS toolbar click File → Save All. Next click File → Export Template. In the Export Template Wizard choose the Project template radio button and click Next. Name the template `OpenGLProjectTemplate` and check the box "Automatically import the template into Visual Studio" and uncheck the box "Display an explorer window on the output files folder". Click Finish. The template `OpenGLProjectTemplate` has been created and you can, in fact, now delete the project `OpenGLProjectTemplate` from the folder you created it in.

12. To use the template, open Visual Studio 2015 from the Start Menu to bring up the MSVS Start Page. Click New Project and select Visual C++. Scroll down the middle pane to find and select `OpenGLProjectTemplate`. Name the project and an appropriate folder for its location and click OK. The new project comes up. To check if the template is ok, repeat steps (3), (4) and (9) above with this project.

It should now be clear how to run the book programs even if you weren't able to place `OpenGLwrappers` in `C:`; particularly, bring up the project, open the Property Pages and edit the paths to the Additional Include Directories (refer step (6) above) and the Additional Libary Directories (refer step (7)) according to where they are in your system.

OpenGL support: Check what level of OpenGL your graphics card supports by running an OpenGL extensions viewer (you can download one from `http://www.realtech-vr.com/glview/download.php`). If it doesn't support OpenGL 4.3 then a book program may compile but not run as the system is unable to provide the OpenGL 4.3 rendering context asked by the command `glutInitContextVersion(4, 3)` in the main routine. What you might do in this case is replace `glutInitContextVersion(4, 3)` in main with `glutInitContextVersion(3, 3)`, or even `glutInitContextVersion(2, 1)`, instead. Of course, then, programs using later-generation calls will not run, but you should be fine early on in the book.

MSVS 2017: Though our development was all on MSVS 2015/Win10, subsequently we installed MSVS 2017 and successfully opened and ran every program using the latter IDE. However, we did see that developing from scratch in 2017 after creating a project template in that IDE required changes to the existing code for each program, which we did not think worth the effort to make, given that everything was already working smoothly in 2015.

Problems? What I described above is exactly how I set up my own environment. Moreover, this guide has been used the past few months by several students, none of whom seemed to have encountered a problem serious enough that they came back to me with a bug report. So, I am fairly confident that this guide is accurate modulo "Microsoft quirks" (I love the MSVS IDE but Microsoft has a way of changing little things from one version to another, which can trip up even the most experienced). So, keep in mind that almost certainly, even if you are on MSVS Community 2015/Win10, your environment and mine are not exactly same and there might be steps you have to do a bit differently.

Now, I will be the first to admit that, though I claim to be a fairly competent programmer, I am a total klutz when it comes to systems matters. So, if you have problems along the lines of "Error xyz123 ...." or "File xyz123.dll not found.", then apologies in advance that I am not going to be of much help. What I have found with such issues, though, is that often a round of internet searching comes up with a solution; if not, there are tech forums teeming with gurus who can fix anything. Beyond this, if you see bugs or have ideas for improvement I will be grateful for an email to `sg@sumantaguha.com`.

Happy Coding!