TNM046 Computer Graphics
Programming project setup

Stefan Gustavson

April 13, 2016

---

1 Development environment

C and C++ programming can be done with nothing but a text editor and a compiler, but it is more convenient to use an *integrated development environment* (IDE) to make it easier to write the code, to handle multiple files, to browse and search in the code, to compile and to debug. There are many IDEs to choose from, and you may use any tool of your choice. The one we recommend for use in the course labs at ITN is *Code::Blocks*. It is a full-featured, free and open cross-platform development environment with versions for Windows, MacOS X and Linux which are easy to set up and use. The Windows version comes with the likewise free Gnu GCC compiler bundled. The code for this lab series can be compiled on Windows, MacOS X and Linux without changes. The additional libraries and extra header files you need are provided in the lab material. We changed the names of the GLFW library for Linux and MacOS X to avoid a name clash with the Windows version, so the package you download from the course page should work straight off for all platforms.

Code::Blocks is already installed for you in the course lab (K4507), and if you have downloaded the zip archive with lab files from the course page, you have all the files you need, including a Code::Blocks project file with the extension `.cbp`. However, please read the following sections anyway to learn exactly what we did to prepare the lab enviroment for you. **You should not be dependent on other people's templates to write C++ programs**, and we will not have this much preparations done for you in upcoming courses.

2 Installing Code::Blocks

Code::Blocks is available for free download from the project's own web site:

`http://http://www.codeblocks.org/`

For Windows, download the version that includes a compiler (GCC) even if you already have a compiler installed. For other platforms, you need to install GCC by other means if you don't already have it. How to do that depends on your individual setup, so instructions are not included here. There are documented, supported and reasonably convenient ways to install the GCC compiler both under Linux and MacOS X. The current Windows download includes only a 32-bit compiler, but 32-bit executables are compatible with 64-bit Windows as well. (The converse is not true: 64-bit binaries can be run only in 64-bit versions of Windows.)

Code::Blocks version 16.01 was released in January 2016. Unfortunately, the version of Code::Blocks that is installed in ITN:s course labs is the previous version (version 13 - versions 14 and 15 were never released). The project files are compatible between the two versions, and the code works equally well in either version, but the user interface and the organization of menus have changed slightly in version 16. The instructions below still refer to version 13. We apologize for any inconvenience.

3 Downloading glext.h

In Windows, you probably need an up-to-date version of the file `glext.h`, because the version that is included with most compiler downloads is badly outdated. This large file describes all extensions that have been made to OpenGL since version 1.0. The file is included in the lab material, but it changes rather frequently, sometimes several times per year. The original source for this file is at `www.opengl.org`:

`http://www.opengl.org/registry/api/glext.h`

4 Installing GLFW

You only need three files to use GLFW: the header files `glfw3.h` and `glfw3native.h`, and the library file `libglfw3.a`. All three are provided in the lab material for your convenience, in the subfolder named `GLFW`. The library files for MacOS X and Linux are included as well, with the altered names `libglfw3_macosx.a` and `libglfw3_linux.a`.

You can compile and install GLFW yourself if you want to. Instructions on how to do that is on the GLFW web site, `http://www.glfw.org`.

From `http://www.glfw.org`, you can also download a precompiled package for Windows which contains the library file `libglfw3.a` for the compiler MinGW. We recommend that you download the 32-bit version, which is the version we provide in the lab material, and which is compatible with all versions of Windows. If you are using a 64-bit version of Windows (most people are), you can instead download the 64-bit package, but then you also need to make sure you have a 64-bit compiler and set up your project to use it. The current download package for Code::Blocks does not include a 64-bit version of the GCC compiler, but you can find that separately from other sources.

5 Setting up a C++ project

Before you start coding, you need to create a new project and set it up for your particular purpose of programming in C++ under Windows using OpenGL and GLFW. Please follow the instructions below *exactly*. All the steps are required.

- Start Code::Blocks.
- In the menu, select *File→New→Project...* click *Empty project*, click *Next*, name the project "GLprimer", specify the folder where you want your project files, click *Next* and *Finish*. The project file is created and saved.
- Place all the `.cpp`, `.hpp` and (if present) `.glsl` files from the lab material and the folders `GL`, `GLFW`, `meshes` and `textures` (along with their contents) in the newly created project folder.
- Right click on the project name in the left hand panel. Select *Add Files...*, select all files with extensions `.hpp`, `.cpp` in the folder, and click *Open*. If you have any `.glsl` files in the project, add them as well. (You will write these files yourself in the first lab session. They are not included in the archive you download from the course page.)
- **In Windows**, select *Project→Build Options...*, select the tab *Linker Settings*. Under *Link libraries*, click *Add*, type `glfw3` and click *OK*. Click *Add* again, type `opengl32` and click *OK*.

  Under *Other linker options*, type `-mwindows -mconsole`. This will make both a graphics window and a text console appear when you run the program. The console is useful for displaying debugging output.

  In the tab *Search Directories→Compiler*, click `Add...` and type a dot (`.`), and in the tab *Search Directories→Linker*, click `Add...` and type `.\GLFW` (dot, backslash, capital "GLFW"). This will make the compiler look for header files and library files also in the local project directory in addition to the compiler's own directories.
- **In MacOS X**, libraries are handled differently than on other platforms, so if you are using Code::Blocks under MacOS X you need to do the following change to the settings compared to what was presented above:

  Instead of writing `opengl32` in the section *Linker Settings→Link libraries*, you should type the following text into the field *Linker Settings→Other linker options*:

  ```
  -framework opengl
  -framework cocoa
  -framework IOKit
  -framework CoreVideo
  ```

  These extra libraries are part of MacOS X and should be on your system if you have installed the GCC compiler and XCode, but they need to be linked to like this for the GLFW library to work.

  In the the tab *Search Directories→Compiler*, click `Add...` and type a dot (`.`), and in the field under the tab *Search Directories→Linker*, click `Add...` and type `./GLFW` (dot, slash, capital "GLFW"). This will make the compiler look for header

files and library files also in the local project directory in addition to the compiler's own directories.

- **In Linux**, you need a slightly larger set of external libraries than on Windows. In the section *Linker Settings→Link libraries*, you should add the following libraries:

  `glfw3 GL m X11 Xxf86vm Xrandr Xi`

  These libraries are GLFW itself and the libraries used indirectly by GLFW, and they are all required for a GLFW program under Linux to work. If you don't already have these installed on your system, please refer to the documentation for your Linux distribution for instructions on how to install them.

  In the text field in the tab *Search Directories→Compiler*, add a line with a single dot (`.`), and in the field under the tab *Search Directories→Linker*, add a line with the text `./GLFW` (dot, slash, capital "GLFW"). This will make the compiler look for header files and library files also in the local project directory in addition to the compiler's own directories.

- Select *File→Save Project* to save the project again with the updated settings.

5.1 Test run

To test your project settings, press the "Build" button. If you misspelled a library or forgot some step in the setup, you will see errors. When you manage to build without errors, press "Run". A window should appear and remain open, waiting for you to close it. There's nothing in it yet, it's just a gray window, but you are going to add content to it during the lab sessions.