Installing Qt

Author: Charles Parker and Zia Khan
Email: charles.w.parker95@gmail.com, zia@cs.umd.edu

For Windows

1. Check for any Windows updates using the Windows Update Center and check for any graphics drivers updates if you have a dedicated graphics card
2. Go to https://www.it.umd.edu/slic/products/microsoft/dreamspark.html and download and install Visual Studio 2013
3. Go to http://qt-project.org/downloads, click show downloads and select Qt 5.3 or newer for Windows 64-bit (VS 2013, OpenGL, 571 MB)
4. Run the installer.

For Linux

1. Check for any system updates
2. Install and/or update gcc using the Ubuntu equivalent of `sudo apt-get install gcc` and `sudo apt-get update gcc`
3. Go to http://qt-project.org/downloads and click the online installer for Linux
4. Run the installer

For OSX

1. Make sure Xcode is installed.
2. Run Xcode and accept the licensing agreement if you haven't already.
3. Run "xcode-select --install" to install the command line tools.
4. Go to http://qt-project.org/downloads, download and run the installer for MacOSX.

Using the Command Line

For Windows

1. Go to Control Panel | System and Security | System. Click on Advanced Systems settings. Click Environment Variables and find PATH under System variables. Double-click it and add ";C:\Qt\Qt5.x\5.x\msvc2013_64_opengl\bin" replacing the x's with your version of Qt.
2. Open the Visual Studio Command Prompt. If you do not have a shortcut to it, hit the windows key (in between the fn and alt keys on most keyboards) and type "command prompt" and click on the one that resembles "Open VS2013 x64 Native Tools Command Prompt"
3. You can now navigate to the project folders and run `qmake` and `nmake` (Visual Studio's version of make). Reminder: `dir` is the equivalent of `ls` in windows.

For Linux/OSX

1. Look for where the Qt installer put the qmake Add this directory to your path (e.g /Users/zia/Qt /5.x/clang_64 for MacOSX or /home/zia/Qt /5.x/gcc_64/bin for Linux). The x.x's are just subversion numbers of Qt. You want to be using 5.3 or newer.
2. To change your PATH variable, edit the .profile or .bash_profile file in your home directory. Add this line at the end of the file to add another directory to PATH: `export PATH="$PATH:your_directory_containing_qmake"`

3. To compile your application, in the directory containing the .pro file. Run `qmake` and then `make`. NOTE: If you have trouble running qmake or building using Qt under Linux. Try the following https://qt-project.org/wiki/Install_Qt_5_on_Ubuntu, installing additional OpenGL libraries as well as the g++ compiler.

Using Qt-Creator

When you first open Qt-Creator, it will prompt you to configure a compiler and the Qt version. It should automatically detect the VS compiler in Windows and GCC in Linux along with the correct version of Qt.

A few notes on Qt-Creator

- It resembles Eclipse, but is much simpler than it
- The execute, debug, and build icons are located in the bottom left hand corner
- By default, the compiled files and executable are placed in the debug folder. You can change it to the release folder by clicking on the computer icon located above the execute button.
- You can create a new project under File | New File or Project. The easiest option is to choose Other Project | Empty Qt Project to avoid any unnecessary files or inclusions.
- To add source and header files, right click on the project in the left pane and choose Add New. Here, you can easily add header and source files simultaneously by choosing C++ class as the type of file. For shaders, choose GLSL | Vertex Shader (or Fragment Shader). For shaders, make sure the file extension is .vsh for vertex shaders or .fsh for fragment shaders so that the IDE highlights variables, functions, etc. correctly for those files.

Running Applications with Command-line Options

For those of you that are having some trouble with the command-line (in particular, the Windows users), there is an alternative way you can run your code from within Qt Creator with command-line arguments. Here are the instructions:

1) With the project open in Qt Creator, click on the "Projects" tab on the left-hand side

2) Near the top of the window that comes up, there will be two buttons that say "Build" and "Run". Click on the "Run" button.

3) You will see a text field called "Arguments". This is where you can type in the arguments to be passed into the program when you run it. For example, I could put

cmsc427.hdr output.hdr crop 100 100 1000 500

into the text field and then click the run button.

Addressing Could not resolve SDK path for 'macosxXX.X'

If you recently upgraded to the latest version of Mac OS, then the following message may apply to you. If you are trying to compile your code and are getting the error message

Could not resolve SDK path for 'macosx10.8'

Then try this fix.
Edit the file at <Path_to_Qt_Install>/5.x/clang_64/mkspecs/qdevice.pri
Comment out the following line by putting a hashtag (also called pound) in front of it like so

#!host_build:QMAKE_MAC_SDK = macosx10.x

Setting up Break Points and Line-by-Line Debugging

Change the release into debug in the second line of the the .pro file:

```
CONFIG += qt warn_on debug embed_manifest_exe
```

And also delete the two lines:

```
win32 {
   QMAKE_CFLAGS += "/D_HAS_ITERATOR_DEBUGGING=0 /D_SECURE_SCL=0"
   QMAKE_CXXFLAGS += "/D_HAS_ITERATOR_DEBUGGING=0
/D_SECURE_SCL=0"
}
```

Finally you should remove the Shadow Build option in the project configuration.
Then you could easily insert breakpoints and debug your program!!