Getting Started with C++ and Qt

Computer programming, i.e., writing computer code, can be intimidating at first. If the code is not written exactly to the liking of the machine then nothing works and it can be difficult to figure out why. In such circumstances it is easy to get discouraged and leave computer programming to "the experts." However, even the slightest success gives the new programmer a great sense of empowerment. The ability to create and distribute working apps is both fun and a valuable problem-solving skill.

About C++

Many "developer environments" are available, for free or for purchase. Most of them, but not all, run equally well on Windows, Mac OS X, and Linux. The ones that are easiest to use for beginners include Matlab, Mathematica, and Mathcad. Those environments have a host of built-in functionality, such as plotting, debugging, and advanced mathematical functions, which makes it quicker to write the code. On the other hand, those environments compile the code at run-time, i.e., the code is run by an "interpreter," and is thus slower. From the perspective of app-development, another disadvantage of those environments is that other people who want to run your code also need those environments installed.

A more powerful and computationally efficient alternative to interpreters is "compilers." Compiled programming languages include Fortran and C++. Most apps that run on modern computers are created in such languages. Here, the code is not executed until after his has been compiled into an executable file. That file, perhaps packaged with additional files, can then be distributed to others who want to run the code, without them needing to install any third-party software. Programming with compiled languages is more difficult for beginners, both in terms of understanding the language and getting the developer environment to work. The purpose of these programming documents is to help developers create apps in the C++ language, together with added features provided by the cross-platform framework Qt.

In the past, machine and assembly languages were common, but nowadays procedural and object-oriented languages have taken over. Common procedural languages are Fortran and C. In contract, C++, C#, Visual Basic, and Java allow object-oriented programming. Of these, the C++ language is selected for three reasons. First, it is a robust and modern language for "structured programming," i.e., it has reasonable constructs for for-loops, if-statements, etc. Another reason is indeed that C++ is an object-oriented language; the ability to create object-oriented code is a crucial advantage when aiming to create extensible and maintainable programs, such as Rt. The third reason is that C++ is established as a broadly used language, with many auxiliary libraries available, such as Qt.

Install the C++ Compiler

The first step towards creating programs with C++, regardless of whether the Qt framework is used, is to install a C++ compiler. Xcode is the default option on Macs, while Visual Studio and MinGW are popular options on the Windows operating system.

Installing Xcode on Mac OS X

1. Go to the App Store and download Xcode
2. Open Xcode from Applications and run it once to install it
3. In Xcode, go to Preferences and select Downloads to install the Command Line Tools; those are necessary for developing code with Qt

Installing Visual Studio on Windows

1. To develop with Qt, it is necessary to use the Professional version of Visual Studio
2. UBC students can search the internet for "msdnaa ubc" to find the version available under a site license

Installing MinGW on Windows

1. To facilitate the use of Qt, MinGW should be downloaded and installed from the Qt website, referenced below

About Qt

After the C++ compiler is installed, attention turns to Qt. In programming parlance, Qt is sometimes referred to as an "API" (application programming interface). Today it is officially referred to as a "cross-platform application and user interface framework." Also officially, it is pronounced "cute," although many continue to spell out the letters, saying, "cue-tee."

There are several reasons why Qt is used in this research group and in the development of Rt. First and foremost, Qt helps developing apps with graphical user interfaces that run across all of the popular operating systems. In other words, code written in Qt runs equally well on Windows, Mac, and Linux. In addition to this major advantage, Qt provides a host of useful tools for the C++ programmer, not limited to user interface programming, but also including object lists, and OpenGL interface, etc.

The home of Qt has moved several times since the Norwegians Haavard Nord and Eirik Chambe-Eng invented it in 1991. Three years later they created the company Quasar Technologies, which later changed name to Troll Tech subsequently Trolltech. At present, Qt is developed by Digia, who owns the Qt trademark and the Qt Project. This regime emerged after Nokia first bought Trolltech in June of 2008 and then later dropped the Symbian technologies, and effectively Qt, as mobile platform in February 2011 in favour of Microsoft. Downloads and more information about Qt are now available at http://qt-project.org. Programming in Qt is described in another document on this website, while how to get started with the installation is provided in the following.

Install Qt and Qt Creator

Qt is a framework that comes with an IDE (integrated developer environment) called Qt Creator. Although some developers prefer to stick with other IDEs, such as the ones that follow Xcode and Visual Studio, the documents on this website recommend using Qt Creator. For that reason, download and install Qt and Qt Creator, which are available in one package at the Downloads page of the Qt website. If you work on Windows, you probably want to download the 64-bit version that contains OpenGL, which is currently available towards the end of the list of versions at the aforementioned webpage.

Run a Qt Example

Once the compiler and Qt is installed it is worthwhile to run one of the examples that follow the installation. This is fun (if it works) and it serves as a milestone on which you can build if your own app does not compile properly.

Develop Good Programming Habits

The rules and syntax of programming is covered in other documents. However, before starting the programming, take a moment to commit to tidiness and clarity. Looking at computer code is like looking at someone's signature. Everyone has a different style, and some variation in tidiness is unavoidable. However, do not write code without indentation, blank lines, and comment-lines! Indentation can be automatically handled by the IDE and shows which lines are within, say, an if-block, and which are outside. Well-placed blank lines provide "air" and greatly improve the readability of the code. Comment lines are even more important. All well-written code has comment lines that explain what is taking place, to help the developer and later developers understand the code. Comment lines can be inserted anywhere with two preceding forward-slashes, and these lines are simply ignored by the compiler:

```
// This is a comment line in C++
```

Sometimes it is useful to "comment out" many lines. To accomplish this, the text is encapsulated by /* and */ in the following manner:

```
/* These symbols encapsulate several
comment lines in C++ */
```