ECE 3574: Applied Software Design

Introduction to Qt

Some Announcements/Reminders

- Project 1 Beta is due Monday at 8 AM. Be sure to **tag your intended commit** as beta (or beta1, beta2, etc) and push it before the deadline.
- Office hours tomorrow 2/17 have moved to 2-3:30.
- Office hours Monday 2/20 are canceled

The goal of today's meeting it to learn about about a popular cross-platform library called Qt.

- Windows and Event Loops
- Widgets
- Signals and Slots
- Meta-Object Compiler
- Exercise

User Interaction

In C++ (including the standard library), the built-in mechanisms for user input are

- specifying command line arguments (not interactive)
- standard input (interactive but synchronous)
- signals, e.g. Control-C (asynchronous)

C++ itself also has nothing to say about displays.

It assumes only standard output and standard error.

- The OS provides the notion of a console
- a way to enter input into standard input one line at a time,
- and a way to view standard output/error.
- multiplexes different programs input/output
- this interaction dates to the very early days of computing

This provides powerful language-style interaction but is limited the kind of user interaction that can be supported.
See: “In the Beginning was the Command Line” by Neal Stephenson.

Modern OSs often provide some abstraction of a graphical display

A library which interacts with the display hardware (vector or bitmap). It provides

- a way to draw 2D shapes and/or images on the screen
- a way to register user events related to those objects (clicks, etc)
- a way to multiplex different programs on the same display (focus)

The dominant abstraction is called WIMP

WIMP = windows, icons, menus, pointer

- the display is made up of a set of windows
- a program has access to one or more windows
- a window is a collection of widgets
- a pointing device is used to register actions on a widget (event)
- the program can change the visual appearance of the widget (draw or render)

The main concept is the event-loop.

Event Loop

1. Draw the widgets
2. Collect all events
3. Process all events
4. Goto 1

- This loop takes over the main thread of the program.
- All work (in a single threaded application) happens in the event loop.
- Called *Event Driven Programming*. Event cause code to run changing the program state and causing side effects.

The windowing system library is platform dependent

Common native windowing libraries:

- On Windows: Win32, WinForms, MFC, WPF
- On Mac: Carbon, Quartz
- On Unix: X11

Maintaining an application across all three platforms is cumbersome, but sometimes warranted.

An alternative is to use another library layer that abstracts away the platform

- GTK+
- WxWindows
- FLTK
- Qt

We will be discussing Qt, a *huge* library, focusing on the GUI part.

In Qt widgets and events are objects.

- QApplication handles the event loop
- Your user interface code is embedded in a widget (using dynamic polymorphism)
- Events are delivered to your widget if appropriate (events are filtered)
- If your widget needs to change it calls a method called `update`

Events can trigger other events. In this view a program is a collection of widgets communicating via events.
See http://doc.qt.io/qt-5/eventsandfilters.html

Exercise 10: Part 1: A Basic Qt Window

See the website.

Qt also uses another parallel form of communication among widgets.

Signals and Slots

- extends C++ syntax to add slots, special member functions
- requires a code generator (meta-object compiler or moc)
- code can emit signals, which are objects
- these signals can be connected to slots, members of other objects
- when an signal is emitted it is sent to all slots that it is connected to

Allows dynamic and one-to many communication among objects as opposed to just calling a member (one-to-one).

Exercise 10: Part 2: Signal/Slot example

See the website.

Next Actions

- Read links on Dynamic Polymorphism
- Project 1 beta is due Monday at 8 am