Qt

CSCI 5448 Spring 2011

Mark Grebe

Introduction

- Qt is a cross-platform application framework that is widely used to develop GUI applications.
- In runs on many OS platforms, including Windows, Linux, Solaris, FreeBSD, MeeGo, and Mac OSX, with a "write-once, compile anywhere" approach.
- Qt is a C++ based framework, with enhancements from the Qt metacompiler, which is part of the toolchain provided with the framework.
- Bindings to other languages are available. PyQT's bindings to Python are one example.

History of Qt

- Qt development was started in 1991 by two developers who would become the founders of Trolltech.
- The name Qt was picked because they liked the way the letter Q appeared in the Emacs font, and the ‘t’ came from Xt, the X toolkit.
- Early versions of Qt had separate Unix and Windows flavors.
- Mac OSX support was added in 2001.
- Nokia acquired Trolltech in 2008.
- Qt on all platforms is now available in either a proprietary or an open source license (GPL, LGPL, or GPLv3 with exception, which allows apps to developed with other open source licenses.)

Well Known Qt Applications

- Many well known cross-platform applications are written using the Qt Framework. They include:
  - Google Earth
  - Autodesk
  - KDE
  - Adobe Photoshop Album
  - Skype
  - VLC Media Player
  - Virtual Box
  - Mathematica

Services Provided By Qt

- Qt service abstractions provided to the application can be grouped into the following categories:
  - QtCore – Core non-GUI classes.
  - QtGUI – Widgets and 2D Graphics.
  - QtOpenGL - 3D Graphics with OpenGL.
  - QtNetwork – TCP/UDP client/server, etc, etc.
  - QtSQL – Integration with Proprietary and Open Source Databases.
  - QtXML – XML Reader/Parser and Writers.
  - QtMultimedia - Multimedia Framework.
  - QtWebKit - WebKit Browser Integration.
  - QtScript – ECMAScript (JavaScript) based Scripting Engine.

QtCore:

- The QTCore module provides the basic non-GUI classes for application development:
  - Event Loop(s) – Event Loops for both GUI and command line applications, as well as thread event loops.
  - Data Classes – Lists, Dictionaries, Stacks, Queues, Arrays, Sets, and Strings.
  - Platform independent abstractions for:
    - Threads
    - File Handling
    - Shared Libraries
    - Shared Memory
    - Regular Expressions
    - User and Application settings - QSettings

QtCore: Signals and Slots (I)

- A core concept in the Qt Framework provided by QtCore is Signals and Slots, which is Qt's Inter-Object communication method.
- GUI Widgets (or other objects) emit Signals when events occur.
- These Signals may be connected to another object's Slots, which are methods that receive the Signals.
- Qt's Signals and Slots are a method of implementing the Observer Design Pattern.
- The signals and slots system is implemented using the C++ preprocessor (macros) and the moc (Meta-Object compiler) that is part of the Qt build system.

QtCore: Signals and Slots (II)

- Slots are defined like any other method, however there are separate sections in the class definitions for them (public slots:, protected slots:, and private slots:), i.e.:
  - `public slots:`
    `void mySlot(QString &msg)`
- The modifiers public, protected, and private only apply to using the slot as a normal method call. You may still connect a signal from another class to a private slot method.

QtCore: Signals and Slots (III)

- Signals are also defined like any other method, but again with a separate section in the class definitions for them, i.e.:
  - `signals:`
    `void mySignal(QString &sigmsg)`
- Signals are only defined in class definitions, no implementation for them is provided.
- Signals are are not invoked and executed like normal methods, instead they are emitted, i.e.:
  - `emit mySignal("Test");`

QtCore: Signals and Slots (IV)

- Multiple signals may be connected to the same slot, and multiple slots may be connected to a signal.
- As an example, to connect a Quit button's 'clicked' signal to a application's 'quit' slot, the programmer would use:
  - `connect(button, SIGNAL(clicked()), qApp, SLOT(quit()));`
- The SIGNAL and SLOT in the above statement are required macros.
- When signals and slots take parameters, only the parameter type is given in the connect statement, not the actual parameter.

QtCore: Memory Management

- Qt provides a simplified version of memory management to all objects that inherit from it's base QObject base class.
- In the inherited object's constructor, a call to QObject's constructor can be made, passing a pointer to a parent object.
- When the parent object is destroyed, all of it's children objects are destroyed as well, greatly simplifying object memory management.

QtCore: Threading

- QApplication's exec method starts the main GUI thread of the Qt app.
- Qt Objects are owned by the thread they are created on.
- The QThread class allows the Qt programmer to create other threads of execution.
- QMutex, QReadWriteLock, QSemaphore (counting) classes provide protection and synchronization between threads.
- In Qt 4.x, the ability to use Signals and Slots between threads was added.
  - When emit is called to send a signal between objects in a single thread, the slot member is called directly.
  - When emit is called to send a signal between objects in multiple threads, the slot member call is queued on the receiving thread.

QtCore: Files, Paths and Streams

- Qt provides platform independent ways of dealing with files, paths, and streams.
- The QDir class is the key to dealing with paths and drives in Qt. It's abstraction handles the translation of '/' separators into whatever is used by the underlying platform.
- Text Streams using Qt's QTextStream class handle line ending differences between platforms automatically, and also handle locale translation.
- Binary data may be serialized with the QDataStream's.

QtGUI (I):

- The QtGUI module adds GUI functionality on top of the base provided by QtCore.
- Qt uses the native graphics API's of the platforms it supports, thus providing native look and feel to cross platform applications. (On some platforms, such as MeeGo and Linux, Qt is the native API).
- Qt's QPainter, QPaintDevice, and QPaintEngine classes allow for device independent presentation of graphics, using the same code to produce screen graphics or a PDF file, for example.

QtGUI (II):

- QtGUI provides basic GUI widgets include check boxes, combo boxes, dials, labels, push buttons, radio buttons, scroll bars, sliders, spin boxes, and line and text editors
- More advanced widgets such as menus, table views, list views, date and time editors, tabbed windows, progress bars, tool tips, and tool bars are also provided.
- As mentioned earlier, Qt Widgets use the Slot and Signal mechanism for communication between widgets and application classes.

QtOpenGL:

- The QtOpenGL module provides the classes used to provide 3D graphics rendering capability to applications.
- It provides support for OpenGL on Linux, Windows, and Mac OS X platforms, and OpenGL ES on Embedded Linux.
- The module provides a widget (QGLWidget) which can be used like any other Qt widget. It opens up a OpenGL display buffer, through which the programmer may use the OpenGL API to render 3D graphics.

QtNetwork:

- The QtNetwork module in Qt provides abstractions for both TCP and UDP network applications.
- At a high level, classes for FTP (QFtp) and HTTP (QHttp) clients are provided, to simply file and web transfers.
- At a lower level, QTcpServer, QTcpSocket, and QUdpSocket classes are used to support other network protocols.
- In addition, the QSslSocket class is used to provide secure TCP connections.
- The QUrl class is provided to help the programmer construct and parse complex URL's.

QtSQL:

- The Qt module QtSQL provides the Qt programmer a method for manipulating relational databases.
- QtSQL supports many of the most popular databases including MySQL, Oracle, PostgreSQL, Sybase, DB2, SQLite, Interbase, and ODBC through drivers.
- The QSqlDatabase class provides an abstraction of a connection to a particular database (making use of the correct driver for the database).
- The QSqlQuery class is used to submit SQL statements to the database such as such as CREATE_TABLE, SELECT, INSERT, UPDATE and DELETE.
- The QSqlQueryModel, QSqlTableModel, and QSqlRelationalTableModel classes allow for easy presentation of data in Qt's GUI Widgets.

QtXML and QtXMLPatterns:

- Qt supports two ways of handing parsing and writing of XML files: DOM and SAX, and are part of the QtXML module.
- The SAX interface reads the XML files a portion at a time, parsing as it goes. This method makes it difficult to write changes.
- The DOM interface reads the entire XML file into memory, then acts on it. This allows easier modification and write operations, but requires more memory.
- The QtXMLPatterns module provides a Xquery and Xpath interface to allow for querying and searching of XML.

Other Qt Modules:

- QtMultimedia
  - Provides low level multimedia functionality, including audio input/output and video playback.
- QtWebKit
  - Provides a WebKit based Web browser engine, allowing the programmer to embed Web functionality into an application.
- QtScript and QtScriptTools
  - Provides classes which allow for the scripting of applications, based on a standard scripting language (EMCAScript, of which JavaScript is a dialect)

Qt Development Tools

- Similar to many other modern GUI Frameworks (such as Cocoa and Android, which we have examined in class), Qt provides a development toolset to be used in developing applications with the framework.
- QMake – Makefile creator, used to simply multiplatform development.
- Qt Creator – IDE (Similar to Apple's Xcode)
- Qt Designer – GUI Interface builder (Similar to Apple's Interface Builder).
- Qt Assistant – Online help builder

Example Program

- Accompanying this slide presentation is a screencast which demonstrates using the Qt Framework and tools to develop a simple shape drawing GUI application.