# Lecture 3.
# GUI Programming
# – Part 2: Qt4

**Guoyong Shi, PhD**
shiguoyong@ic.sjtu.edu.cn
**School of Microelectronics**
**Shanghai Jiao Tong University**
**Fall 2010**

# Outline

- Qt4 programming basics
- Beginner's tutorial

# About Qt

- Qt is a language for developing cross-platform GUI applications.

- Qt is C++.
- Qt is successful because programmers like it
- The latest release is “Qt4”
- Qt is publicly available

# About Qt

- Qt was made publicly available in 1995
- Qt was initially developed by two persons
  - Haavard Nord (Trolltech CEO) and
  - Eirik Chambe-Eng (Trolltech President),
- both graduated from Norwegian Institute of Technology in Computer Science

# A Brief History of Qt

- Haavard commissioned by a Swedish company to develop a C++ GUI framework in 1988.
- In the summer of 1990, Haavard and Eirik were working together on a C++ database application for ultrasound images.
- The system was required to run with a GUI on Unix, Macintosh, and Windows.
- One day that summer when Haavard and Eirik went outside to enjoy the sunshine on a park bench, Haavard said, "We need an object-oriented display system."
- The discussion laid the intellectual foundation for the object-oriented cross-platform GUI framework they would start to build.

# History of Qt (cont'd)

- In 1991, Haavard and  Eirik started writing the classes that eventually became Qt.
- The following year, Eirik came up with the idea for "signals and slots", a simple but powerful GUI programming paradigm that has now been embraced by several other toolkits.
- By 1993, Haavard and Eirik had developed Qt's first graphics kernel and were able to implement their own widgets.
- At the end of the year, Haavard suggested that they go into business together to build "the world's best C++ GUI framework".

-- From "A Brief Histroy of Qt"
Jasmin Blanchetter and Mark Summerfield, C++ GUI Programming with Qt 4, Prentice Hall, 2006.

# About Qt

- The letter 'Q' was chosen as the class prefix because the letter looked beautiful in Haavard's Emacs font.
- The 't' was added to stand for "toolkit", inspired by Xt, the X Toolkit.
- The company was incorporated on March 4, 1994, originally as Quasar Technologies, then as Troll Tech, and today as Trolltech.

- On June 6, 2008, Nokia acquired Trolltech at approx. $150 millions.

# Qt Licenses

- Qt was available under two licenses from day one:
  - A commercial license was required for commercial development, and
  - a free software edition was available for open source development.

- In August 1999, Qt won the LinuxWorld award for best library/tool.
- In 2000: Trolltech released Qtopia Core (then called Qt/Embedded).
- Qtopia Core won the LinuxWorld "Best Embedded Linux Solution" award in both 2001 and 2002,
- and Qtopia Phone achieved the same distinction in 2004.

# Qt3

- Qt 3.0 was released in 2001.
- Qt 3 provided 42 new classes and its code exceeded 500,000 lines.
- a completely new text viewing and editing widget, and
- a Perl-like regular expression class.

- Qt3 won the Software Development Times "Jolt Productivity Award" in 2002.

# Qt4

- Qt 4.0 was released in the summer of 2005.
- With about 500 classes and more than 9000 functions
  - include a completely new set of efficient and easy-to-use template containers,
  - advanced model/view functionality,
  - a fast and flexible 2D painting framework, and
  - powerful Unicode text viewing and editing classes.

- Qt 4 is the first Qt edition to be available for both commercial and open source development on all the platforms it supports.

# Qt is popular today

- Qt is very popular today.
- This success is a reflection both of the quality of Qt and of how enjoyable it is to use.
- In the last decade, Qt has gone from a product being used by only a few to one that is used daily by thousands of customers and tens of thousands of open source developers all around the world.

# Qt4 in CYGWIN

- Qt4 is currently included in the latest CYGWIN release.
- To run the qtdemo:
  - Start X-server
  - run in CYGWIN at any directory: $ qtdemo &

# Install Qt4-Win

- Download Qt4-Win release from http://trolltech.com/downloads
- Unpack the archive (this release has <qt_windows.h>)
- Type the following in a Windows console
  - configure
  - nmake
- The Qt4-Win release was installed on my XP machine successfully (after compiling the source code for about 4-5 hours).

# Hello Qt !

```
1 #include <QApplication>
2 #include <QLabel>
3 int main(int argc, char *argv[])
4 {
5     QApplication app(argc, argv);
6     QLabel *label = new QLabel("Hello Qt!");
7      label->show();
8      return app.exec();
9 }
```


- Save the source code to “hello.cpp” in a directory called “hello”.
- Type the following:
- cd hello
- qmake –project (generates “hello.pro”)
- qmake hello.pro (generates “Makefile”)
- make (use nmake if running on Windows)

# Qt is easy to learn

- Qt is consistent and fully object-oriented in the construction and use of widgets.
- Qt carefully chooses names for functions, parameters, enums, and so on.
- Qt signal/slot connections and layouts are easy to learn.
- Qt new widgets are easy to learn and use.

# Command Line Compile

- qmake –project (generates “hello.pro”, platform-independent)
- qmake hello.pro (or simply “qmake” to generate “Makefile”)
- make

- You may type all in one line:

- qmake –project && qmake && make (on CYGWIN/LINUX)
- qmake –project && qmake && nmake (on Windows)
- .\debug\hello.exe (to run)

# In case ...

- In case you have “qt3” installed as well, use the following “qmake”:
- /lib/qt4/bin/qmake
- You can create a Visual Studio project file from hello.pro by typing:
- qmake –tp vc hello.pro

# Hello Qt !

```
1 #include <QApplication>
2 #include <QLabel>
3 int main(int argc, char *argv[])
4 {
5     QApplication app(argc, argv);
6     QLabel *label = new QLabel("Hello Qt!");
7      label->show();
8      return app.exec();
9 }
```


Replace

```
QLabel *label = new QLabel("<h2><i>Hello</i> " "<font color=red>Qt!</font></h2>");
```

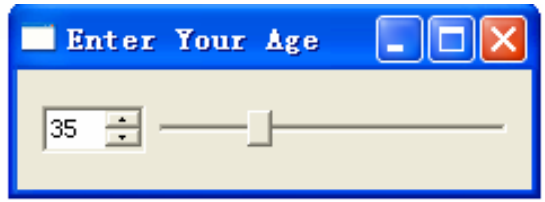
- You may use Marking Language (like in HTML) to set the label fonts.

# *Qt Designer*

- Qt Designer is Qt's visual design tool (just like Microsoft Visual Studio.)
- Using *Qt Designer* is a lot faster than hand-coding
- The Qt Designer comes with the installation of Qt4.

# Example of Signal & Slot

```
10      QSpinBox *spinBox = new QSpinBox;
11      QSlider *slider = new QSlider(Qt::Horizontal);
12      spinBox->setRange(0, 130);
13      slider->setRange(0, 130);
14      QObject::connect(spinBox, SIGNAL(valueChanged(int)),
15                       slider, SLOT(setValue(int)));
16      QObject::connect(slider, SIGNAL(valueChanged(int)),
17                       spinBox, SLOT(setValue(int)));
18      spinBox->setValue(35);
```




- **The two QObject::connect() calls shown in lines 14 to 17 ensure that the spin box and the slider are synchronized (i.e., always showing the same value).**
- **Whenever the value of one widget changes, its valueChanged(int) signal is emitted, and the setValue(int) slot of the other widget is called with the new value.**

# *Signals and Slots*

- You should learn the signals and slots mechanism for Qt programming.
- Signals and slots *bind objects together*.
- Slots are like ordinary C++ member functions.
  - They can be virtual; they can be overloaded; they can be public; protected, or private, they can be directly invoked like any other C++ member functions; and their parameters can be of any types.
- "By *connecting* a signal to a slot" it means that whenever the signal is emitted, the slot is called automatically.

# connect( )

- The connect() statement looks like this:
  - connect(sender, SIGNAL(signal), receiver, SLOT(slot));

- sender and receiver are pointers to QObjects,
- signal and slot are function signatures without parameter names.
- The SIGNAL() and SLOT() macros essentially convert their argument to a string.

# Connecting signals and slots

- One signal can be connected to many slots.
- Many signals can be connected to the same slot.
- A signal can be connected to another signal.
- Connections can be removed.

- (See Qt4 documentation for more details.)

# Qt 2D Graphics

- Qt's 2D graphics engine is based on the QPainter class.
- QPainter can draw geometric shapes (points, lines, rectangles, ellipses, arcs, chords, pie segments, polygons, and Bezier curves), as well as pixmaps, images, and text.

# Qt Modules

- Qt consists of several modules, each lives in its own library.
- The most important modules are
  - QtCore
  - QtGui
  - Qtnetwork
  - QtOpenGL
  - QtScript
  - QtSql
  - QtSvg
  - QtXml

# QtOpenGL Module

- An alternative to QPainter is to use OpenGL commands.
- OpenGL is a standard library for drawing 3D graphics.
- QtOpenGL module makes it easy to integrate OpenGL code into Qt applications.

# Chinese Language

- Qt4 can display Chinese Language.

```
#include <QTextCodec.h>
...
int main (...)
{
        QApplication app(argc, argv);
        QTextCodec::setCodecForTr(QTextCodec::codecForName("GB18030"));

        QPushButton *button = new QPushButton(QWidget::tr("退出"));
}
```