Presentation Overview

- Introduction to Qt/KDE
- History
- Features
- Qt Designer
- Event Handling

- Applications
- Widgets
- Layout Managers
- Event handling
- Sample Code
What is QT?

- Cross-platform graphical toolkit
- One of the most advanced GUI toolkits available
- Fast and compact
- Extensive support and documentation
What is Qt? (continued)

- Supported on the following platforms:
  - MS/Windows 95, 98, NT 4.0, ME & 2000
  - Unix/X11 (Linux, Sun, Solaris, HP-UX, Digital Unix, IBM AIX, SGI IRIX etc.)
  - Mac OS X
  - Embedded (Linux Platforms with framebuffer support)
What is Qt? (continued)

• Qt Desktop family consists of the following:
  - Qt/Windows
  - Qt/X11
  - Qt/Mac
  - Qt/Embedded
  - Qtopia
What is KDE?

- A powerful Open Source graphical desktop environment for Unix
- Is an Internet Project
KDE features

- Ease of use
- Contemporary functionality
- Outstanding graphical design
Some Facts

- Qt and KDE are written in C++
- Qt enabled KDE developers to focus on design issues for the desktop
• KDE project was founded by Matthias Ettrich in October 1996
• Original purpose: emulate CDE
• Contributors include hundreds of developers around the world
Qt Features

- Object Oriented
- Component Support
- Superior on-line documentation
- Ease of Use Portability
- Rich API
- Full Widget Set
- Extensibility
Qt Features (continued)

- High performance implementation
- GUI Emulation
- Advanced drawing operations
- 2D and 3D Graphics support
- Database support
- Internationalization
- GUI Applications
Qt Designer

- Can be used purely as a design tool or create entire applications with its built-in C++ code editor
- Drag and Drop
- Eliminates compile, link and run cycle for user interface design
- Can create both dialog style and main window style applications
- Form designs are stored in XML format (.ui) and converted into C++ header files by the User Interface Compiler
Event Handling

• Slots and Signals technology was developed independently and is not a C++ feature
• Slots and Signals are normal member functions
Understanding Slots

- Possible to connect signals
- Slot (function) execution occurs when a Signal connected to it is emitted
- Many Qt classes have predefined slots
Understanding Signals

- Internal state changes in an object send out signals
- Possible to connect multiple Slots to one Signal; Slots are executed one by one in an arbitrary order
QApplication Class

- Inherits QObject
- Manages the application event queue
- receives events from the underlying window system and sends them to the destination widgets
- Must be created before any widgets can be created
QApplication Class

- `#include <qapplication.h>
- `QApplication ( int& argc, char**argv )
- `void quit ( )
- `void lastWindowClosed ( )

**define**
**constructor**
**slots**
**signals**
#include <qapplication.h>
#include <qpushbutton.h>

int main( int argc, char **argv )
{
    QApplication app( argc, argv );
    QPushButton hello( "Hello, world!" );
    app.setMainWidget( &hello );
    connect( &hello, SIGNAL(clicked()),
             &app, SLOT(quit()) );
    hello.show();
    return app.exec();
}
Widgets

- Window
- Labels
- Scrollbars
- Text Fields
- Buttons
- Radio buttons
- Checkbox
QMainWindow

- Inherits QWidget
- `#include <qmainwindow.h>`
- `QMainWindow ( QWidget* parent = 0, const char* name = 0, Wflags = 0 )`
- `void setRightJustification (bool enable)` & `void setUsesBigPixmaps ( bool )`
- `void pixmapSizeChanged ( bool )`

Screenshot

- Define
- Constructor
- Slots
- Signals
QButton

- Inherits QWidget
- \#include <qbutton.h>
- QPushButton ( QWidget* parent = 0, const char* name = 0 )
- void animateClick ( ) & void toggle ( )
- void pressed ( ), void released ( ), void clicked ( ) & void toggled ( bool )
Screenshots

QCheckBox

- First
- Second
- Third

QLabel

This is a Label it spans multiple lines
Screenshots

QScrollView

QProgressDialog

Please wait...

34%
Layout Managers

- QLayout
- QGridLayout
- QBoxLayout
- QHBoxLayout
- QVBoxLayout
- QHBoxLayout
- QVBoxLayout
QBoxLayout

- Inherits QLayout
- #include <qlayout.h>
- QBoxLayout ( QWidget* parent, Direction d, int border=0, int autoBorder = -1, const char* name=0 )