QT

CS2141 - Software Development using C/C++

What Is Qt?

- Pronounced “cute”
- C++ Toolkit for GUI Building
- Uses native APIs to draw controls
- Runs on Linux/X11, OS X, Windows, mobile platforms
- Designed to write once, compile anywhere
- Has bindings for many languages, including Java

Where's Qt Used?

- Google Earth
- Skype
- Mathematica (on OS X and Linux)
- KDE

History of Qt

- Development began in 1991
- Developed by TrollTech
- First two versions had two flavors:
  - Qt/X11 - QPL or proprietary license
  - Qt/Windows - only available under proprietary license
- KDE 1.0 Released in 1998

More History

- Major concern that KDE depended on non-free tech
  - Harmony toolkit - compatible with Qt but free
  - GNOME
- 2000 - Qt/X11 released under GPL2
- 2001 - Qt 3 with (proprietary only) OS X support
- 2003 - Qt 3.2 - OS X version available under GPL2
- 2005 - Qt 4 - All platforms available under GPL2

Qt Today

- Current version is Qt 4.6
- Qt 4.5 available on lab machines
- Current owned/developed by Nokia
- Available with both proprietary and open licenses
  - GPL v2 & v3 (with special exemption)
  - LGPL

Using Qt

- Two different methods of interface creation
    - Raw code: Define everything in C++
    - Qt Designer: Build the interface using a GUI tool
- Methods produce equivalent programs
- Using a GUI tool may be easier for sizable programs

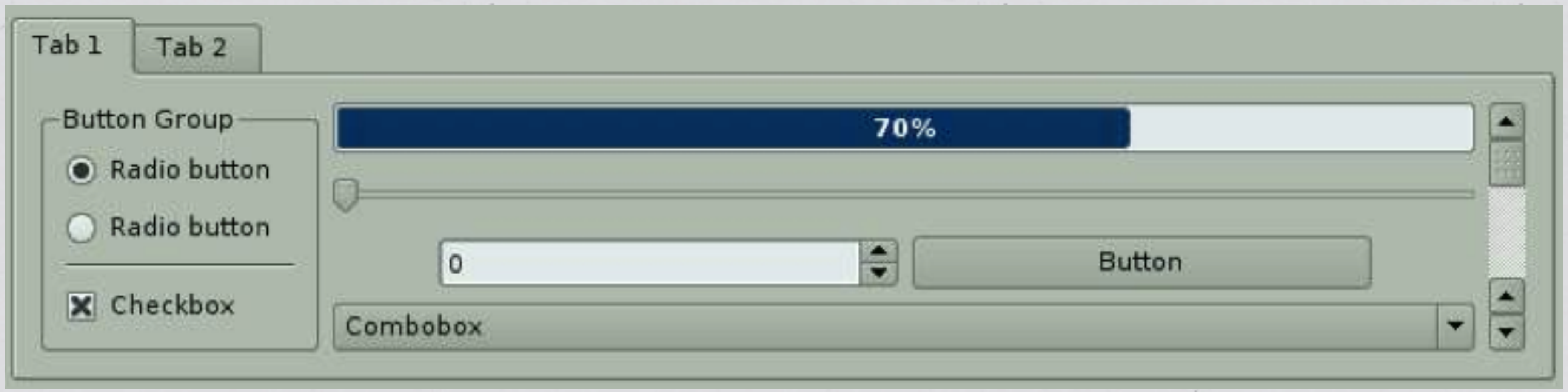
Qt Essentials

- What controls are available? - Widgets
- How are they arranged? - Layouts
- How are they interacted with? - Signals & Slots

Widgets

- Individual parts that are put together to create an interface
- All widgets inherit from QWidget
- No separation of containers and controls
- Can create new widgets by subclassing existing widgets
- A widget can have any number of child widgets
- Deleting a widget automatically deletes its children

Widgets



- Many widgets are available:

  - Buttons
  - Spinners
  - Comboboxes
  - Scrollbars
  - Progress bars
  - Radio buttons
  - Checkboxes
  - Tabs

Available Classes

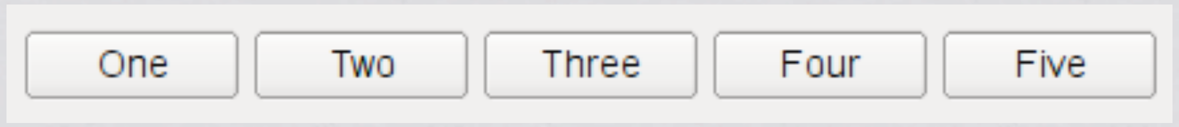
A QAbstractItemDelegate
QAbstractItemModel
QAbstractItemView
QAccessible
QAction
QApplication
B QButtonGroup
QByteArray
C QCache
QCalendarWidget
QCheckBox
QClipboard
QColor
QColorDialog
QColumnView
QComboBox
QCommandLinkButton
QCoreApplication
QCursor
D QDataStream
QDate
QDateEdit
QDateTime
QDateTimeEdit
QDebug
QDesktopWidget
QDial
QDialog
QDialogButtonBox
QDir
QDomDocument
QDomNode
QDoubleSpinBox
E QExplicitlySharedDataPointer
F QFile
QFileDialog
QFlags
QFocusFrame
QFont
QFontDialog
QFormLayout
QFrame
QFtp
G QGLWidget
QGraphicsScene
QGraphicsView
QGridLayout
QGroupBox
H QHash
QHBoxLayout
QHeaderView
QHttp
I QIcon
QImage
QInputDialog
QItemDelegate
K QKeySequence
L QLabel
QLCDNumber
QLibrary
QLibraryInfo
QLineEdit
QLinkedList
QList
QListView
QListWidget
QLocale
M QMainWindow
QMap
QMdiArea
QMdiSubWindow
QMenu
QMenuBar
QMessageBox
QModelIndex
QMultiHash
QMultiMap
QMutex
O QObject
P QPainter
QPalette
QPen
QPicture
QPixmap
QPlainTextEdit
QPluginLoader
QPointer
QPrinter
QProcess
QProgressBar
QProgressDialog
QPushButton
Q QQueue
R QRadioButton
QRegExp
QResource
QRubberBand
S QScriptable
QScriptClass
QScriptContext
QScriptContextInfo
QScriptEngine
QScriptEngineAgent
QScriptEngineDebugger
QScriptString
QScriptSyntaxCheckResult
QScriptValue
QScriptValueIterator
QScrollArea
QSet
QSettings
QSharedDataPointer
QShortcut
QSignalMapper
QSlider
QSound
QSpinBox
QSplashScreen
QSplitter
QSqlDatabase
QSqlQuery
QStack
QStackedLayout
QStackedWidget
QStatusBar
QString
QStringList
QStringListModel
QStyledItemDelegate
T QTabBar
QTableView
QTableWidget
QTabWidget
QTemporaryFile
QTextCursor
QTextDocument
QTextEdit
QThread
QThreadStorage
QTime
QTimeEdit
QTimer
QToolBar
QToolBox
QToolButton
QToolTip
QTranslator
QTreeView
QTreeWidget
U QUrl
V QValidator
QVariant
QVBoxLayout
QVector
W QWhatsThis
QWidget
QWidgetAction
X QXmlSimpleReader
QXmlStreamReader
QXmlStreamWriter

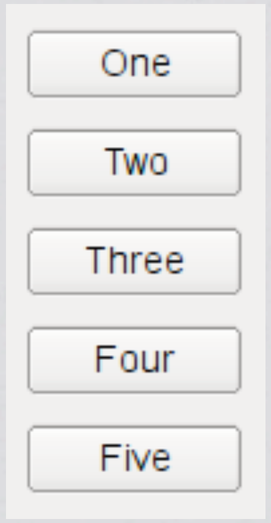
Layouts

- Many layout managers available
- QHBoxLayout - Arrange widgets horizontally
- QVBoxLayout - Arrange widgets vertically
- QGridLayout - Arrange widgets in a rectangular grid
- Layouts can be nested, and there are others

Q{H|V}BoxLayout

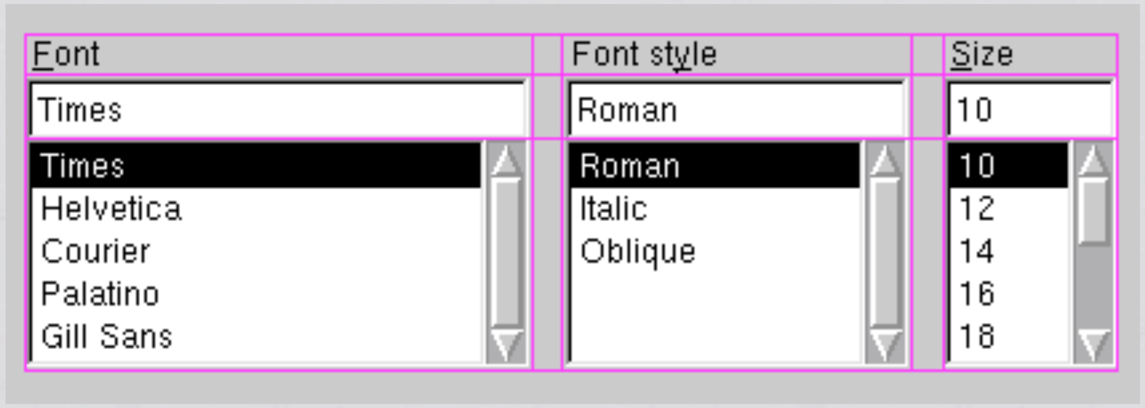
- Splits given space into boxes
- Each widget fills one box
- Can insert empty boxes or padding





QGridLayout

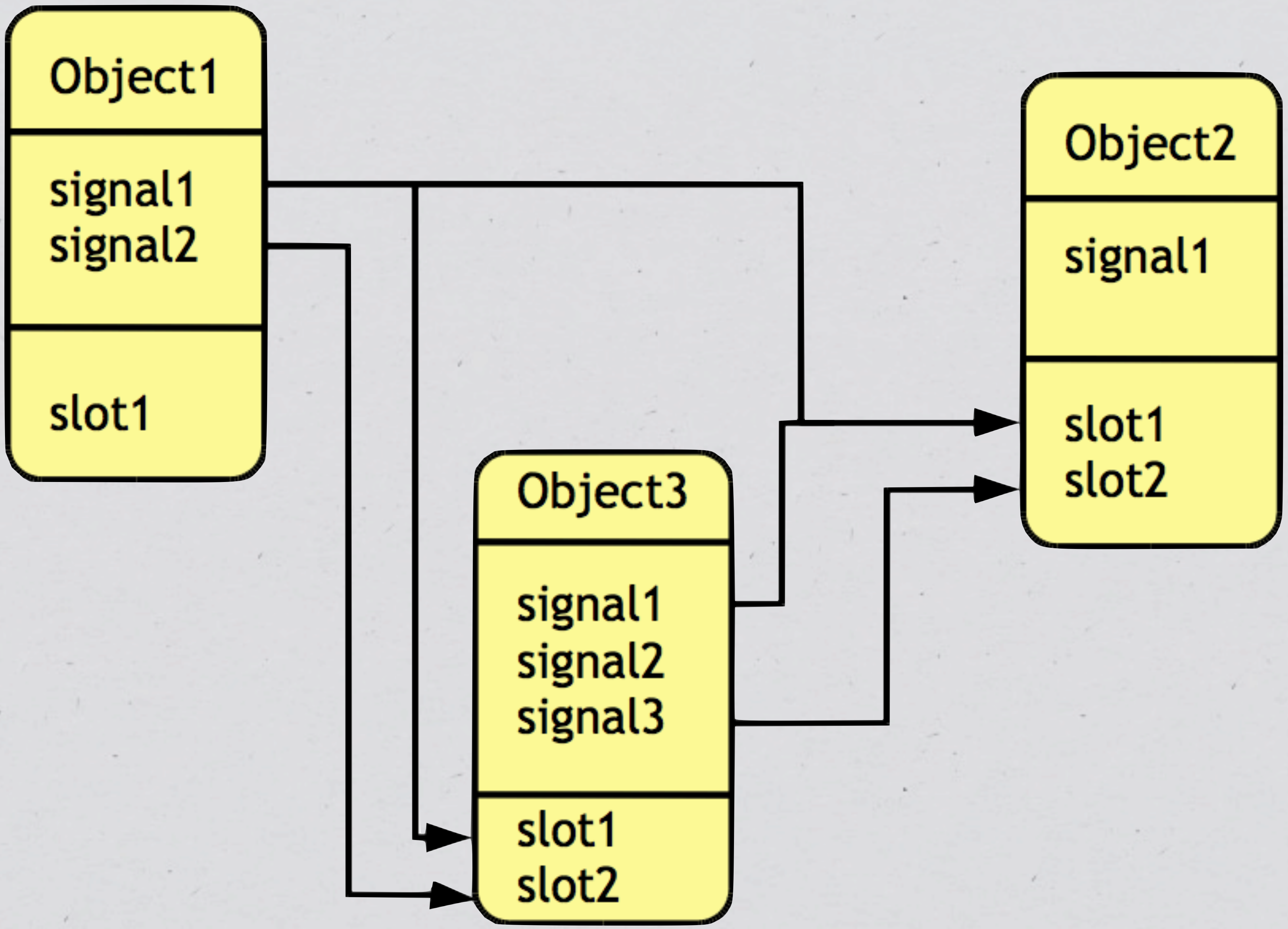
- Similar to grid layout from Java
- Can place widgets directly in any cell
- Widgets can span cells
- Rows / columns need not be same size



Signals & Slots

- Form the basis of communication between QWidgets
- Operates like message passing
- Not a one-to-one relationship
  - Signals can go to multiple slots
  - Slots can receive multiple signals
- Implemented as functions behind the scenes

Signals & Slots



Signals & Slots

- Use QObject::connect() to link signals and slots
- Many signals (eg clicked()) and slots (eg quit()) are provided
- User-defined signals/slots are supported
  - Must include the Q_OBJECT macro in private section of the class definition
  - Class definition includes new sections: signals, slots
  - Signals not implemented, just defined

Compilation

- Qt implements many features not directly supported by C++
  - signal and slot keywords
- Qt-specific tools must be used during compilation
- Requires a specialized Makefile
- Good news: tools are provided to automate creating Makefile

qmake

- Analyze files to create a project file
  - Assumes all C++ files in current directory are part of project
- Use project file to create Makefile
- Can also create Visual Studio or XCode compatible projects
- Note: On a Mac, you need to include -spec macx-g++ to get a Makefile

Quick Qt Compilation

- Create the project (only necessary if things change)
  - `qmake -project`
- Create a Makefile
  - `qmake`
- Compile as usual
  - `make`

Hello, World!

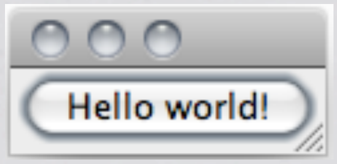
```
#include <QApplication>
#include <QPushButton>

int main(int argc, char *argv[])
{
   QApplication app(argc, argv);

   QPushButton hello("Hello world!");

   QObject::connect(
      &hello, SIGNAL(clicked()),
      &app, SLOT(quit()));

   hello.show();
   return app.exec();
}
```


Hello world!