THE UNIVERSITY OF
ALABAMA IN HUNTSVILLE

Lecture Qt008
Main Windows

Instructor: David J. Coe

CPE 353 – Software Design and Engineering

Department of Electrical and Computer Engineering

Outline

- Review
- Motivation
- Deriving Applications from **QMainWindow**
- Hands-On Example: Virtual Slots
- Lessons Learned
- Key Points

Review

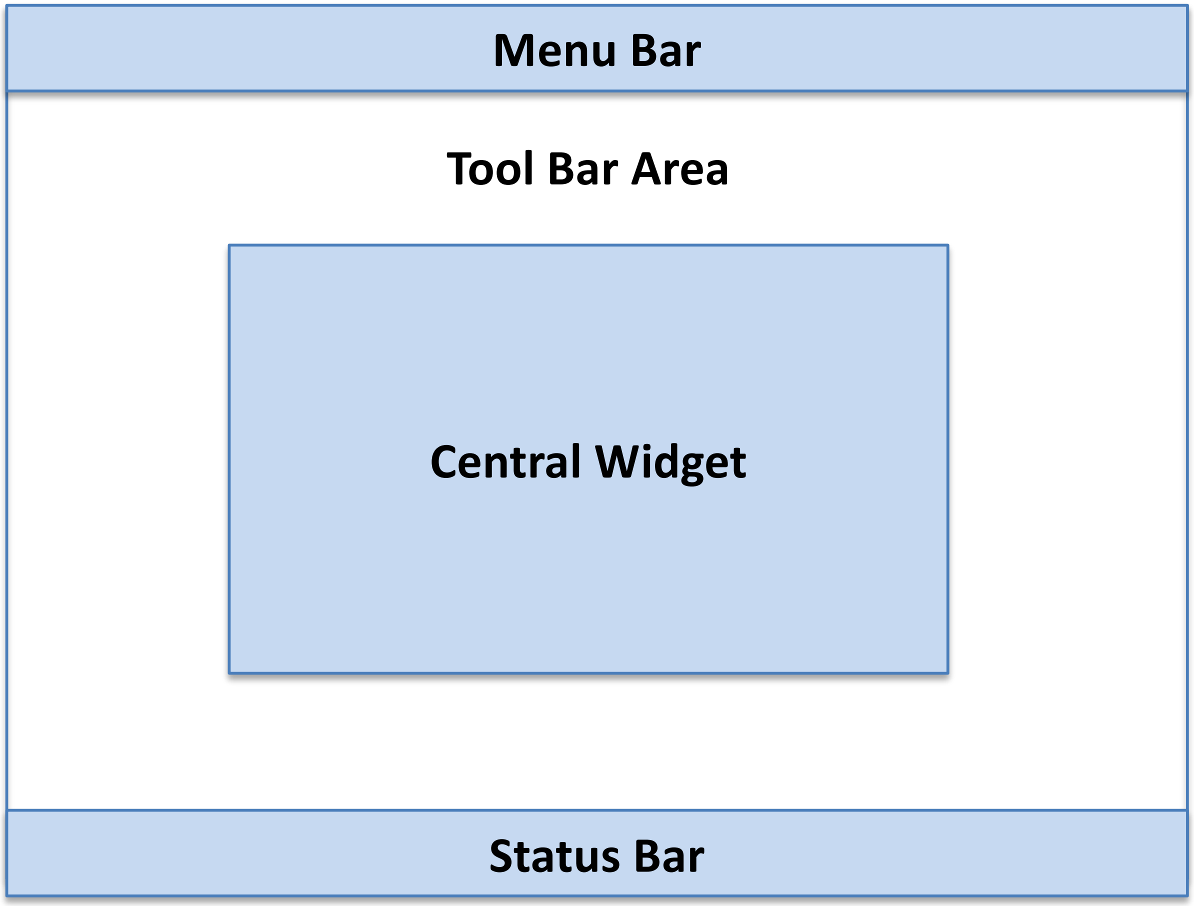
- Looked at creating dialogs in Qt
  - Creating and nesting of layouts
  - Allocating and configuring child widgets
  - Adding child widgets to layouts
  - Modifying widget properties by directly calling widget methods [such as **setText(…)** for a **QLabel**] or indirectly via the Property Editor in **QtCreator**
- Use of Qt Creator and Qt via command line

Review

- Introduced **signals** and **slots**
  - Looked at examples of establishing signal-slot connections by calling ***connect*** with the macros **SIGNAL** and **SLOT**
  - Look at the use of the Signals and Slots editor in **QtCreator** to establish signal-slot connections
  - One signal may trigger the execution of one or more slots functions
  - Multiple signals may trigger execution of the same slot function
  - Looked only at the use of pre-defined signals and slots thus far
  - Today we will see how to ***emit*** a ***custom signal*** that passes an argument value to a ***custom slot function*** that makes use of the incoming argument

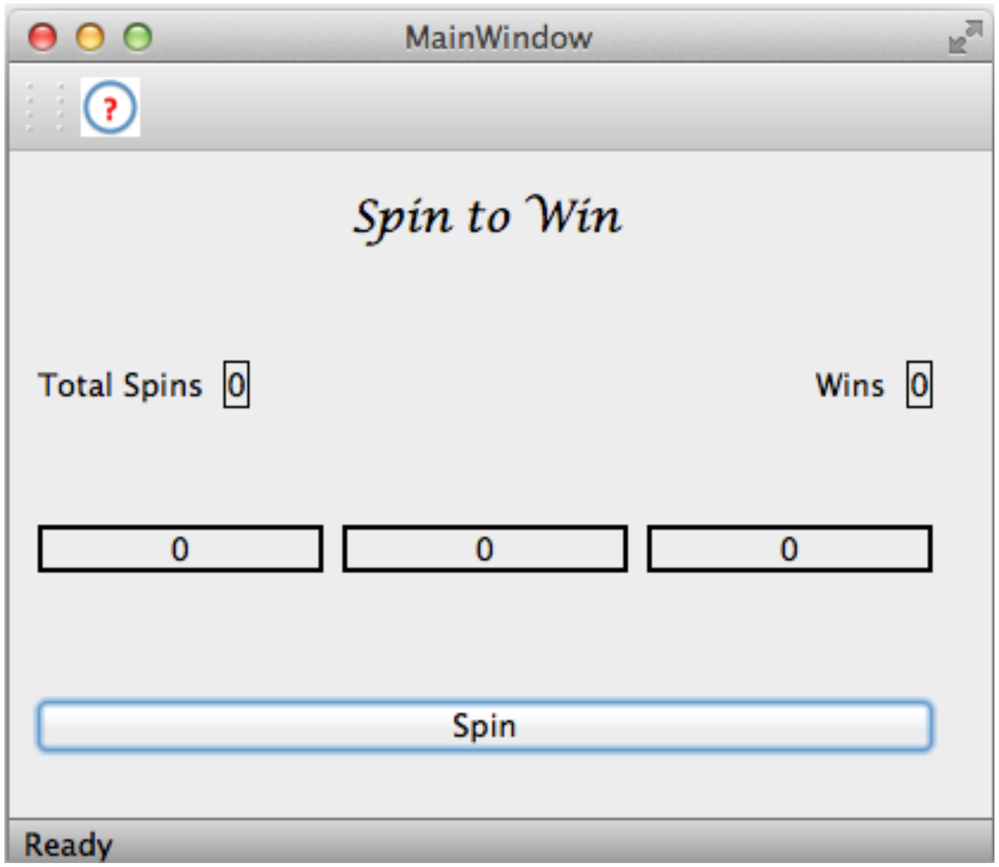
Motivation

- Most non-trivial applications inherit from **QMainWindow** and may thus include a variety of accessories including
  - Menu bar (with keyboard shortcuts)
  - Tool bar
  - Central widget (user workspace)
  - Status bar



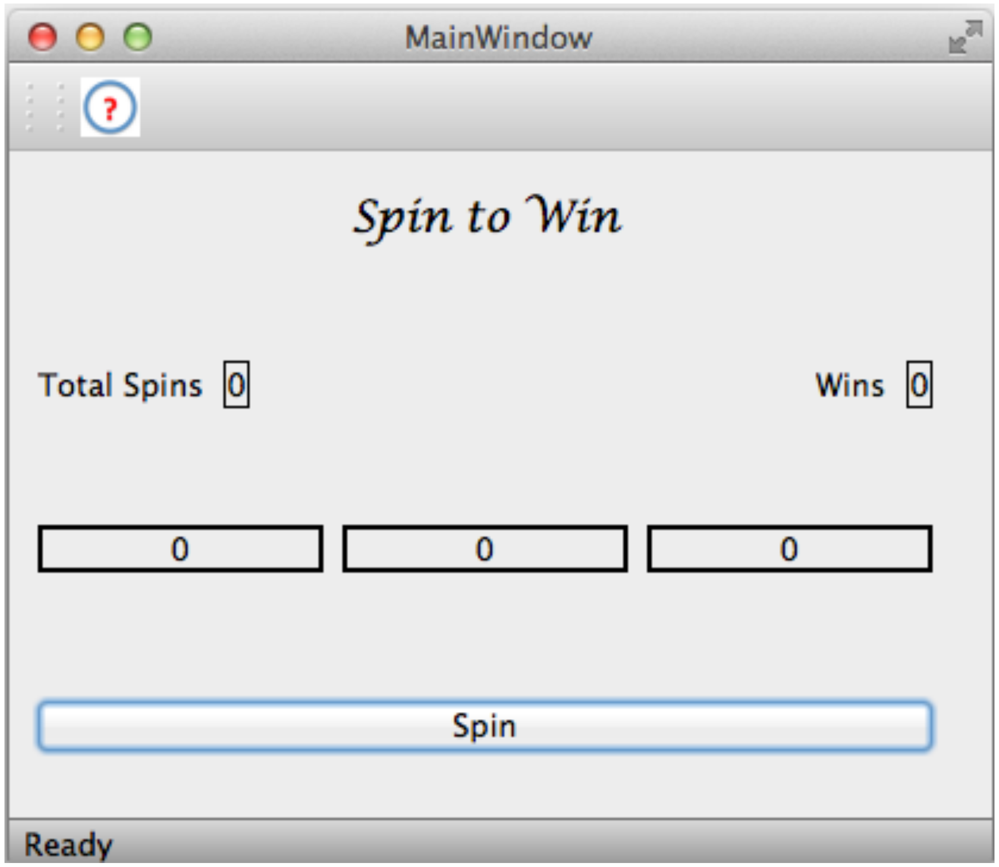
Hands-On Example: Virtual Slots

- App should inherit from **QMainWindow** and include illustrate the use of Qt to implement the following features
  - Pull-down menu
    - Help option
    - AboutQt option
    - Quit option
  - Toolbar with help button
  - Status bar with real-time status updates updates



Hands-On Example: Virtual Slots

- Inheriting from **QMainWindow** gives you
  - Menu bar
  - Tool bar
  - Status bar
  - Default central widget
- We will create a custom widget form class which contains the Spin-to-Win game display and controls
- This form will be used as the Central Widget within the application framework inherited from **QMainWindow**



Hands-On Example: Virtual Slots

```
// Customized form.h
#ifndef FORM_H
#define FORM_H
#include <QWidget>
namespace Ui
{
    class Form;
}
const int MODULUS = 4;
class Form : public QWidget
{
    Q_OBJECT
public:
    explicit Form(QWidget *parent = 0);
    ~Form();
private:
    Ui::Form *ui;
private slots:
    void processSpin();           // Custom slot function
signals:
    void updateStatus(QString);   // Custom signal with payload
};
#endif // FORM_H
```

Hands-On Example: Virtual Slots

```
// Customized form.cpp
#include "form.h"
#include "ui_form.h"
#include <stdlib.h>

Form::Form(QWidget *parent) : QWidget(parent), ui(new Ui::Form)
{
  ui->setupUi(this);

  connect(ui->spinButton, SIGNAL(clicked()),
          this, SLOT(processSpin()));
}

Form::~Form()
{
 delete ui;
}
```

Hands-On Example: Virtual Slots

```
// Customized form.cpp -- continued
void Form::processSpin()          // Custom slot function
{
  int one = qrand() % MODULUS;
  int two = qrand() % MODULUS;
  int three = qrand() % MODULUS;

  ui->dial1->setText(QString::number(one));
  ui->dial2->setText(QString::number(two));
  ui->dial3->setText(QString::number(three));

  int spins = ui->spins->text().toInt() + 1;
  ui->spins->setText(QString::number(spins));

  if ((one == two) && (two == three))
  {
    emit updateStatus(QString("Status: Winner!!"));
    int wins = ui->wins->text().toInt() + 1;
    ui->wins->setText(QString::number(wins));
  }
  else
  {
    emit updateStatus(QString("Status: Loser!! "));
  }
}
```

Hands-On Example: Virtual Slots

```
// Customized mainwindow.h -- continued

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QAction>
#include <QMenu>
#include <QToolBar>
#include <QLabel>
#include "form.h”

namespace Ui
{
  class MainWindow;
}
```

Hands-On Example: Virtual Slots

```
// Customized mainwindow.h - continued

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
private:
    Ui::MainWindow *ui;
    Form *form;                // Form that will become Central Widget
    QAction* helpAction;       // QActions may be triggered via
    QAction* aboutQtAction;    // menu options, tool bar buttons,
    QAction* quitAction;       // or keyboard shortcuts.
    QMenu* optionsMenu;
    QToolBar* toolBar;
    QLabel* statusLabel;
private slots:
    void showHelp();           // Another custom slot function
};
#endif // MAINWINDOW_H
```

Hands-On Example: Virtual Slots

```
// Customized mainwindow.cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>
#include <QMessageBox>

MainWindow::MainWindow(QWidget *parent) :
        QMainWindow(parent), ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    // Replace central widget with custom form
    form = new Form(this);
    setCentralWidget(form);
    form->setBaseSize(ui->centralWidget->frameSize());

    // Create and configure actions
    helpAction = new QAction(QIcon(":/images/help.png"), "Help", this);
    helpAction->setShortcuts(QKeySequence::AddTab);
    aboutQtAction = new QAction("About Qt", this);
    quitAction = new QAction("Quit", this);

    // Tie actions to slots
    connect(helpAction, SIGNAL(triggered()), this, SLOT(showHelp()));
    connect(aboutQtAction, SIGNAL(triggered()),
            qApp, SLOT(aboutQt()));
    connect(quitAction, SIGNAL(triggered()), this, SLOT(close()));
```

Hands-On Example: Virtual Slots

```
// Customized mainwindow.cpp -- continued

    // Add file menu to menubar and populate with actions
    optionsMenu = menuBar()->addMenu(”&Options");  // & makes shortcut
    optionsMenu->addAction(helpAction);
    optionsMenu->addAction(aboutQtAction);
    optionsMenu->addSeparator();
    optionsMenu->addAction(quitAction);

    // Add toolbar and populate with help action
    toolBar = addToolBar("Options");
    toolBar->addAction(helpAction);
    toolBar->setIconSize(QSize(25,25));

    // Add label to status bar
    statusLabel = new QLabel(" Ready ");
    statusBar()->addWidget(statusLabel);

    // Allow custom signal to update status display
    connect(form, SIGNAL(updateStatus(QString)),
            statusLabel, SLOT(setText(QString)));
}
```

Hands-On Example: Virtual Slots

```
// Customized mainwindow.cpp – continued

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::showHelp()
{
    int status = QMessageBox::information(this, "Help",
         "Press spin button to cycle wheels.\n”
         “Exact match of all three wheels wins.");

    qDebug() << "Help Action = " << status;
}
```

Hands-On Example: Virtual Slots

```
#-------- Contents of .pro file below ---------

QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = QtMainWindowApp
TEMPLATE = app

SOURCES += main.cpp\
           mainwindow.cpp \
           form.cpp

HEADERS  += mainwindow.h \
            form.h
FORMS    += mainwindow.ui \
            form.ui

OTHER_FILES += \
    images/help.png

RESOURCES += \
    icons.qrc
```

- Icon image stored in an ***images*** subdirectory
- Image file added as "Other Files" to project
  - File now include in revision control
- Resource file **icons.qrc** created that results in compile-time pre-digesting of image into a C++ char array
  - Speeds execution later on

Hands-On Example: Virtual Slots

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Contents of icons.qrc resource file

```
<RCC>
    <qresource prefix="/">
        <file>images/help.png</file>
    </qresource>
</RCC>
```

Partial contents of qrc_icons.cpp from build folder (icon image digested into C++)

```
#include <QtCore/qglobal.h>

static const unsigned char qt_resource_data[] = {
  // /Users/blah/QtMainWindowApp/images/help.png
  0x0,0x0,0x2f,0xfa,
  0x89,
  0x50,0x4e,0x47,0xd,0xa,0x1a,0xa,0x0,0x0,0x0,0xd,0x49,0x48,0x44,0x52,0x0,
  0x0,0x0,0x5b,0x0,0x0,0x0,0x5b,0x8,0x2,0x0,0x0,0x0,0x93,0x54,0x6e,0xce,
  0x0,0x0,0x18,0x21,0x69,0x43,0x43,0x50,0x49,0x43,0x43,0x20,0x50,0x72,0x6f,0x66,
  0x69,0x6c,0x65,0x0,0x0,0x58,0x9,0xad,0x59,0x67,0x58,0x14,0x4b,0xb3,0xee,0x99,
…
```

Lessons Learned: QMainWindow

- Inheritance from **QMainWindow** gives your applications a number of features commonly found in GUI applications including
  - Menus
  - Toolbars
  - Status bar

Lessons Learned: QAction

- A **QAction** may be defined so that a particular slot function executes when it is triggered
- An action may be triggered by a variety of means including menu option, toolbar button, or keyboard shortcut
- A **QAction** may have an associated icon that will be displayed on the tool bar or pull down menu
- Define the action once and decide which triggering options you wish to have

Lessons Learned: Signals/Slots

- If pre-defined signals or slots are inadequate, you can define custom signals and slots as needed
  - The **emit** statement can be used to send signals
  - Signals can convey information (payload) via the argument list
  - Slots may make use of the incoming argument values
- The **connect** statement may be used to link pre-defined or custom signals to pre-defined or custom slots
- The **disconnect** statement may similarly be used to break the linkage when it is no longer required

Lessons Learned: QDebug Module

- The **qDebug()** method may be used to write debugging information to the console
  - One advantage of **qDebug()** over a traditional output statement is that some data type information may also displayed with the value
  - A constant can be defined during the compilation process to suppress all **qDebug()** output

Key Points

- The use of inheritance an essential technique of Qt software development
  - Extend and customize class functionality
  - Reuse code developed by others