Lecture Qt005
Dialogs

Instructor: David J. Coe

CPE 353 – Software Design and Engineering

Department of Electrical and Computer Engineering

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Outline

- Dialogs
- Hands-On Exercise: Manual Coding Echo Dialog
- Hands-On Exercise: Qt Creator Echo Dialog
- Key Points

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Dialogs - 1

- Dialog boxes provide a way for the user and application to communicate
  - Data input or option selection
  - Output or status display
- Typical GUI applications consist of a main window with menus and/or toolbars, along with multiple dialog boxes

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

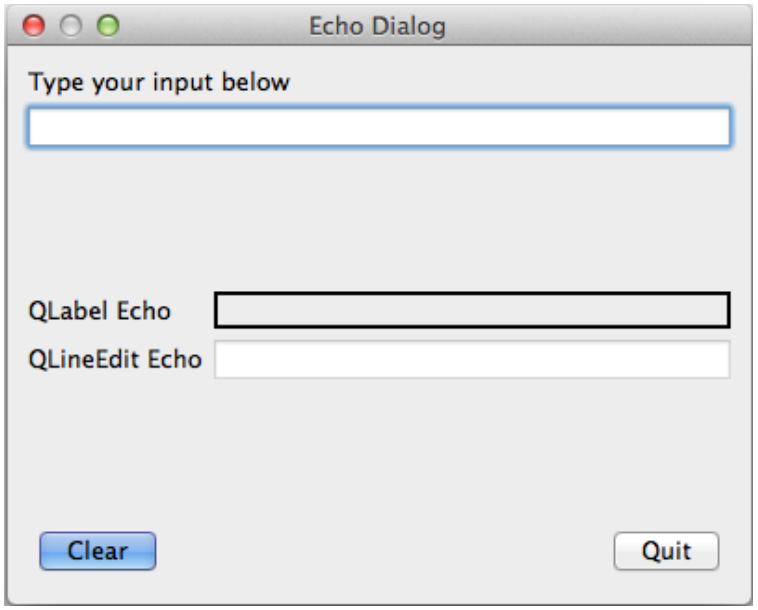
Dialogs - 2

- In practice, one would use ***Qt Designer***
  - Visual layout tool included with ***Qt Creator***
  - Quick prototyping and coding of designs
- We look first at ***manual coding*** of dialogs to give you insight on the work done for you by the ***Qt Creator***

Hands-On Example: Manual Coding of Echo Dialog

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Dialog-based Qt application
  - Custom class **EchoDialog** ***inherits basic dialog properties*** from **QDialog**
  - Add custom features to **EchoDialog**
- Features
  - User types in **QLineEdit** Object
  - Input limit is 30 characters
  - Text echoed below in **QLabel** and **QLineEdit** objects
  - Clear button erases typed and echoed text
  - Quit button closes dialog and terminates app



- Manual coding example

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Echo Dialog: main.cpp

```
// main.cpp - Echo dialog application
#include <QApplication>
#include "echodialog.h"

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

  EchoDialog  myDialog;
  myDialog.show();

  return myApp.exec();
} // End main()
```

Echo Dialog: echodialog.h

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// echodialog.h - EchoDialog specification file
#ifndef ECHODIALOG_H
#define ECHODIALOG_H

#include <QDialog>
class QLabel;                          // Forward declaration of QLabel class
class QLineEdit;                       // Forward declaration of QLineEdit class
const int MAXIMUM_LENGTH = 30;     // Maximum number of characters of input

class EchoDialog : public QDialog
{
  Q_OBJECT                             // Required Q_OBJECT macro

public:
  EchoDialog();                        // Default constructor

private:
  QLineEdit*  userInputLineEditPtr; // Pointer to user input QLineEdit
  QLineEdit*  echoLineEditPtr;     // Pointer to uneditable, echo QLineEdit
  QLabel*     echoLabelPtr;        // Pointer to echo QLabel;

  QPushButton*  quitButtonPtr;     // Pointer to quit QPushButton
  QPushButton*  clearButtonPtr;    // Pointer to clear QPushButton
};

#endif
```

Echo Dialog: echodialog.cpp

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// echodialog.cpp -- EchoDialog implementation file
#include "echodialog.h"
#include <QLabel>
#include <QLineEdit>
#include <QPushButton>
#include <QVBoxLayout>
#include <QHBoxLayout>
#include <QGridLayout>

EchoDialog::EchoDialog()
{
  // Allocate layouts
  QVBoxLayout*  mainLayoutPtr = new QVBoxLayout(this);
  QVBoxLayout*  inputLayoutPtr = new QVBoxLayout;
  QGridLayout*  echoLayoutPtr = new QGridLayout;
  QHBoxLayout*  buttonLayoutPtr = new QHBoxLayout;

  setWindowTitle("Double Echo Dialog”);     // Set dialog title

  // Build main dialog layout
  mainLayoutPtr->addLayout(inputLayoutPtr);     // Place input layout at top
  mainLayoutPtr->addStretch();                  // Add stretch
  mainLayoutPtr->addLayout(echoLayoutPtr);      // Place echo layout in middle
  mainLayoutPtr->addStretch();                  // Add stretch
  mainLayoutPtr->addLayout(buttonLayoutPtr);    // Place button layout at bottom
```

Echo Dialog: echodialog.cpp

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// echodialog.cpp -- EchoDialog implementation file – continued

  // Build input layout
  QLabel*  inputLabelPtr = new QLabel("Type your input below”);   // Allocate and

  userInputLineEditPtr = new QLineEdit;                  // Allocate input line edit
  userInputLineEditPtr->setMaxLength(MAXIMUM_LENGTH);    // Limit total number of characters
  userInputLineEditPtr->setFrame(true);                  // Add frame to define input line edit

  inputLayoutPtr->addWidget(inputLabelPtr);              // Add widgets to input layout
  inputLayoutPtr->addWidget(userInputLineEditPtr);


  // Build echo layout
  QLabel*  firstLabelPtr = new QLabel("QLabel Echo”);    // Initialize first label
  QLabel*  secondLabelPtr = new QLabel("QLineEdit Echo”); // Initialize second label

  echoLabelPtr = new QLabel;                                 // Allocate echo label
  echoLabelPtr->setFrameShape(QFrame::Box);                  // Add frame to define echo label
  echoLineEditPtr = new QLineEdit;                           // Allocate echo line edit
  echoLineEditPtr->setReadOnly(true);                        // Make read only
  echoLineEditPtr->setMaxLength(MAXIMUM_LENGTH);          // Limit total number of characters
  echoLineEditPtr->setFrame(true);                           // Draw frame around line edit

  echoLayoutPtr->addWidget(firstLabelPtr, 0, 0);             // Place widgets into grid
  echoLayoutPtr->addWidget(echoLabelPtr, 0, 1);
  echoLayoutPtr->addWidget(secondLabelPtr, 1, 0);
  echoLayoutPtr->addWidget(echoLineEditPtr, 1, 1);
```

Echo Dialog: echodialog.cpp

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// echodialog.cpp -- EchoDialog implementation file – continued

  // Build close button layout
  clearButtonPtr = new QPushButton("Clear”);  // Allocate & initialize clear button widget
  quitButtonPtr = new QPushButton("Quit”);     // Allocate & initialize quit button widget

  quitButtonPtr->setDefault(true);                // Make it default button of dialog
  buttonLayoutPtr->addWidget(clearButtonPtr);     // Add clear button to button layout
  buttonLayoutPtr->addStretch();                  // Add stretch
  buttonLayoutPtr->addWidget(quitButtonPtr);      // Add quit button to button layout


  // Make Qt4-style signal/slot connections
  connect(quitButtonPtr, SIGNAL(clicked()), this, SLOT(accept()));

  connect(clearButtonPtr, SIGNAL(clicked()), userInputLineEditPtr, SLOT(clear()));

  connect(userInputLineEditPtr, SIGNAL(textChanged(QString)),
          echoLabelPtr, SLOT(setText(QString)));

  connect(userInputLineEditPtr, SIGNAL(textChanged(QString)),
          echoLineEditPtr, SLOT(setText(QString)));

} // End EchoDialog::EchoDialog()
```

Manual Coding: Lessons Learned

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Simple dialog box with one text input, two outputs, and two buttons required a substantial amount of code to implement
  - Lots of details to remember such as available methods associated with each widget, required arguments for those methods, etc.
  - Manual coding of GUI elements time consuming and error prone
- Most GUI applications are developed using tools that automate GUI generation code
- Unless otherwise specified in the assignment, we will be using ***Qt Creator*** in this course

Hands-On Example: Qt Creator Echo Dialog

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Use Qt Creator to recreate the functionality of the Echo Dialog class
- Basic steps
  - In a Linux terminal window, enter **qtcreator**
  - From the ***File*** menu, select ***New File or Project***
  - Be sure that ***Applications*** and ***Qt GUI Application*** are selected then hit the ***Choose*** button
  - Select a name and location for the project then **Continue**
  - Hit **Continue** again
  - Use the **Base Class** spin box to select **QDialog**
    - Default name of **Dialog** will be used to distinguish code from prior example
  - Hit **Continue** followed by **Done**
  - An executable project skeleton should now be available for you

Qt Creator Echo Dialog: main.cpp

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
#include "dialog.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Dialog w;
    w.show();
    return a.exec();
}
```

- Standard Qt main()
- Using default name **Dialog** instead of **EchoDialog**

- **All of this code autogenerated**

Qt Creator Echo Dialog: dialog.h

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
#ifndef DIALOG_H
#define DIALOG_H

#include <QDialog>

namespace Ui
{
  class Dialog;
}

class Dialog : public QDialog
{
    Q_OBJECT

public:
    explicit Dialog(QWidget *parent = 0);
    ~Dialog();

private:
    Ui::Dialog *ui;
};

#endif // DIALOG_H
```

- **All of this code autogenerated**

Qt Creator Echo Dialog: dialog.cpp

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
#include "dialog.h"
#include "ui_dialog.h"

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


    /* Qt-4 style connect statements using SIGNAL-SLOT macros */
    connect(ui->quitButton,  SIGNAL(clicked()), this, SLOT(accept()));

    connect(ui->clearButton, SIGNAL(clicked()),
            ui->inputEdit,   SLOT(clear()));

    connect(ui->inputEdit,   SIGNAL(textChanged(QString)),
            ui->echoEdit,    SLOT(setText(QString)));

    connect(ui->inputEdit,   SIGNAL(textChanged(QString)),
            ui->echoLabel,   SLOT(setText(QString)));
}

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

Constructor Initializer

Signals & Slots
(only manually written code)

Qt Creator Echo Dialog: alternate version - dialog.cpp

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Constructor Initializer

```
#include "dialog.h"
#include "ui_dialog.h"

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


    /* Qt-5 style connect statements using function pointers */
    connect(ui->quitButton,  &QPushButton::clicked, this, &Dialog::accept);

    connect(ui->clearButton, &QPushButton::clicked,
            ui->inputEdit,   &QLineEdit::clear);

    connect(ui->inputEdit,   &QLineEdit::textChanged,
            ui->echoEdit,    &QLineEdit::setText);

    connect(ui->inputEdit,   &QLineEdit::textChanged,
            ui->echoLabel,   &QLabel::setText);
}

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

Signals & Slots
(only manually written code)

Key Observation #1: Qt Creator Echo Dialog

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- How is the **dialog.ui** form integrated with the rest of the code?
  - The form **dialog.ui** is an XML format description of the contents of the form
  - The XML is converted into C++ class **Ui_Dialog** and stored in the **ui_dialog.h** file
  - The member variable **ui** represents the form contents as an object of the type **Ui_Dialog**

Key Observation #2: Qt Creator Echo Dialog

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Where are the widgets allocated and formatted?
  - In the automatically generated file **ui_dialog.h** you will find the public method **setupUI(…)** for the class **Ui_Dialog**
  - The **setupUI(…)** function contains the code which dynamically allocates all widgets that appear on the form
  - **setupUI(…)** must execute ***BEFORE*** you try to interact with the widgets otherwise the pointer variables will be uninitialized
  - The call to **setupUI(…)** is typically the first statement in the constructor to make sure the program does not crash

Key Observation #3:
Qt Creator, Generated Files, and Portability

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Qt Creator attempts to keep source files that the developer modifies directly separated from the automatically generated code
  - Build-related files
    - someproject.pro, someproject.pro.user, Makefile
  - Developer files
    - main.cpp, dialog.h, dialog.cpp, dialog.ui
  - Automatically generated C++ files
    - moc_dialog.cpp, ui_dialog.h
  - Object and executable files generated
    - main.o, dialog.o, moc_dialog.o, someproject
- Attempting to move the **.pro.user** file or **Makefile** from platform to platform will likely cause build problems because they contain platform-specific path information
  - Move only the .pro file and the developer files

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Key Points

- Dialogs are critical components of many GUI applications
- Dialogs may be manually coded or developed using the Qt Designer component of Qt Creator
- The signal-slot mechanism ties user interaction with the dialog to the code that must respond to the interaction or other event
- Qt4 and Qt5 style signal-slot connections may be added directly into the source code
- The Designer Signal-Slot Editor may also be used to create signal-slot connections