

GUI Tech (Qt)

Lab 4

Some good GUI frameworks for C++

- MFC
 - Microsoft Foundation Class
 - Only for Windows and Visual Studio
 - <http://msdn.microsoft.com/en-US/>
- Qt
 - A cross-platform application and UI framework
 - Windows, Linux and Mac
 - Well documented and simple to use
 - <http://qt-project.org/>

Downloads

- Qt 5.3.1 (qt-opensource-**windows-x86-msvc2013**_opengl-5.3.1.exe for VS2013 x86 or qt-opensource-windows-**x86-msvc2013_64**_opengl-5.3.1.exe for VS2013 x64)

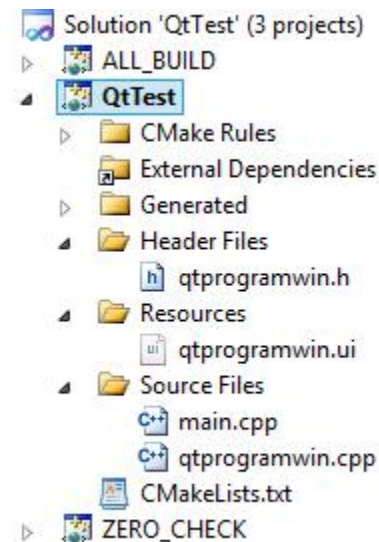
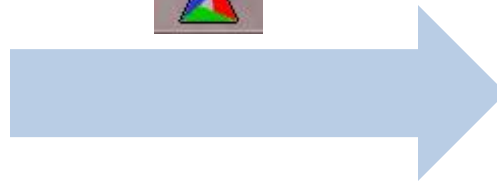
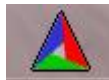
<http://download.qt.io/archive/qt/5.3/5.3.1/>

- CMake

<http://www.cmake.org/download/>

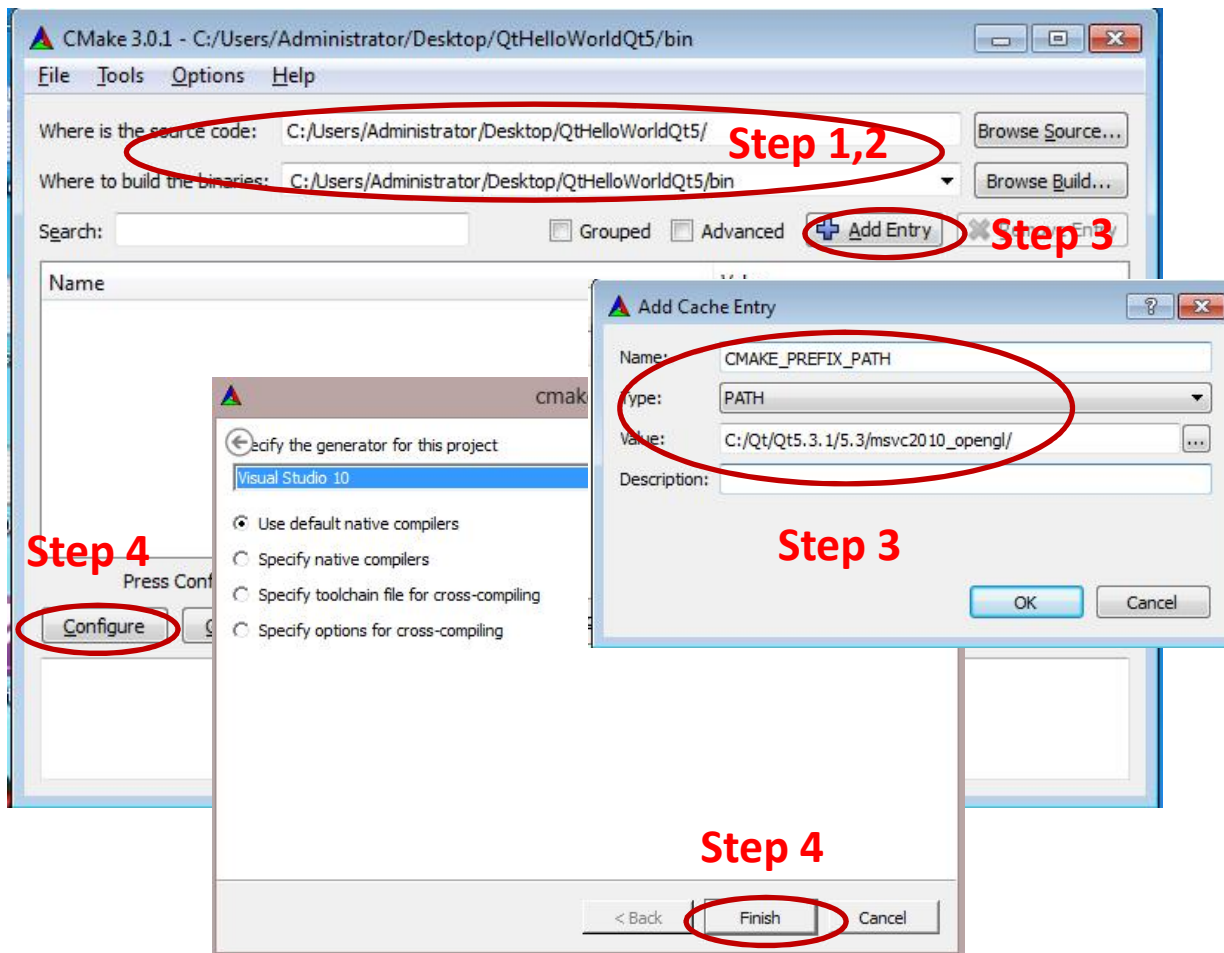
CMake

- Generate the proper project file of the source code for your IDE.
- Simply write a script and put it with your source code together.
 - See CMakeLists.txt for detail



Setup first Qt HelloWorld application

- Download Qt Hello World application from Blackboard
- Use CMake to generate the Project file



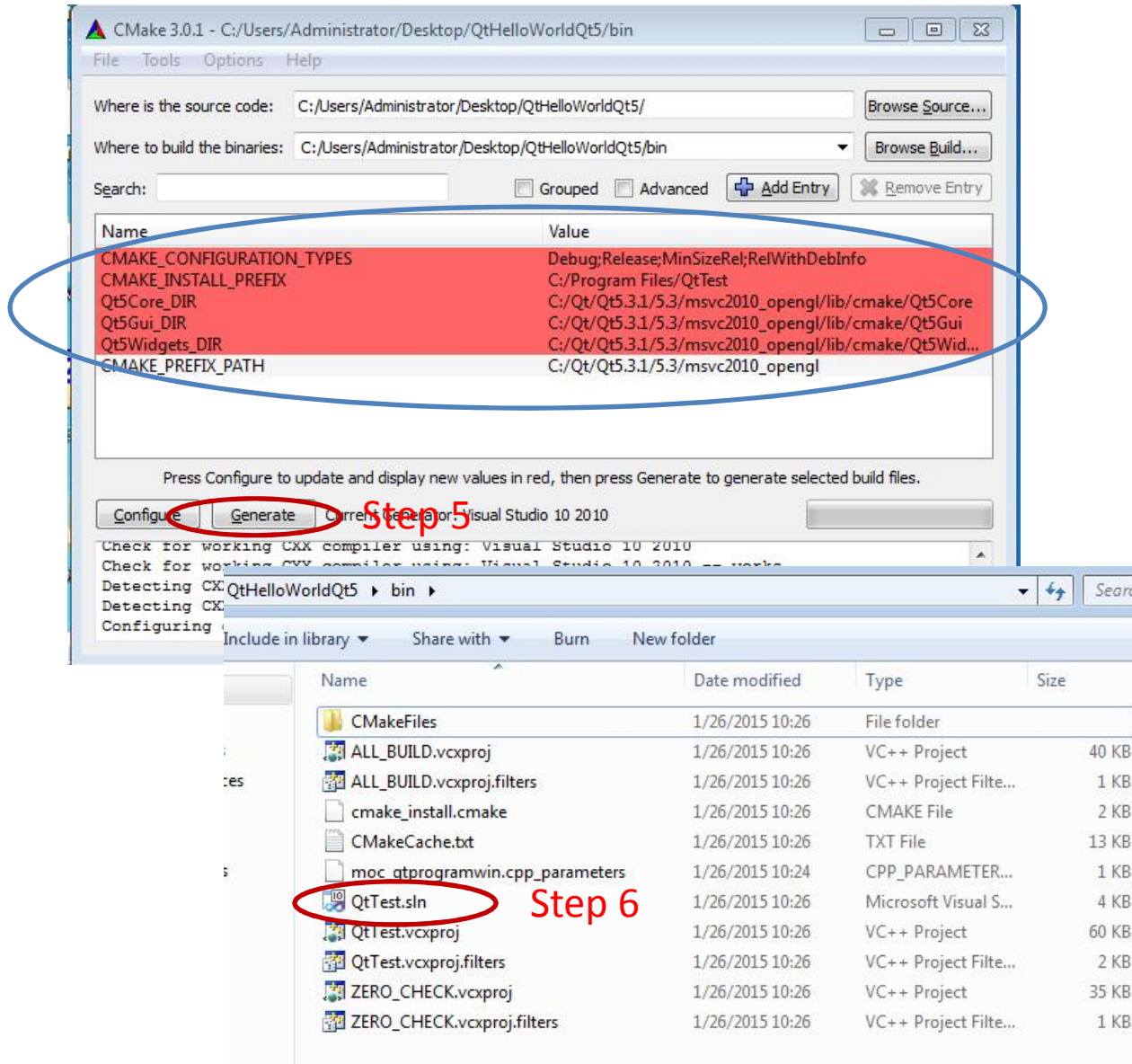
Step 1: Find the folder of source code

Step 2: Set the folder for binary files (**create a new folder inside**)

Step 3: Add Entry for "CMAKE_PREFIX_PATH" (**directory of the folder msvc2013_opengl**)

Step 4: Choose our compiler "Visual Studio 12 2013", then finish

Setup first Qt Hello World application (cont.)

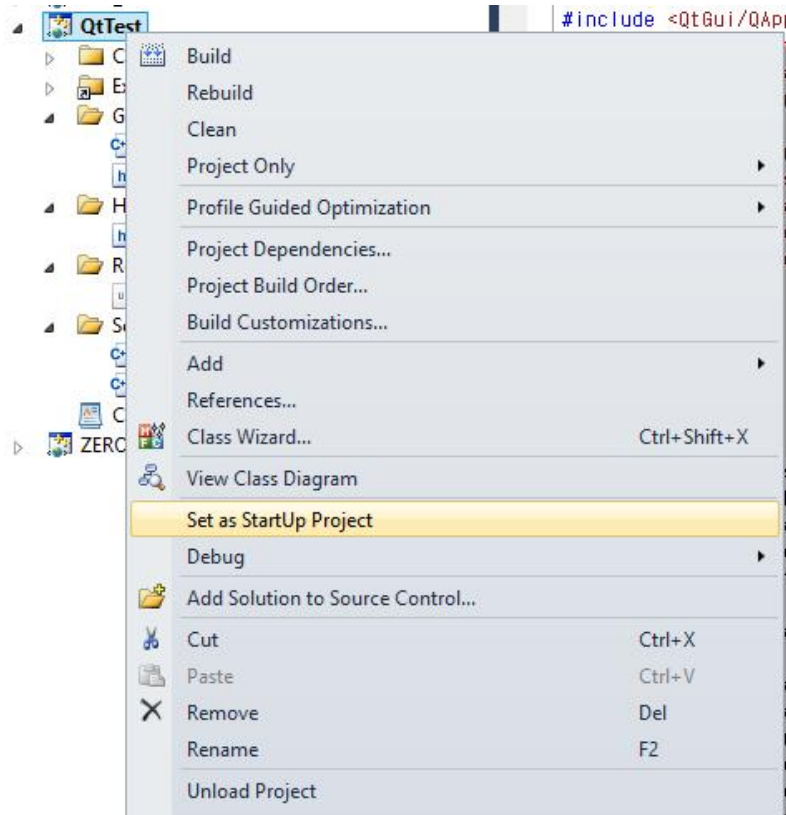


Step 5: Wait until the configuring is done, then click Configure again. (Red highlights will disappear)

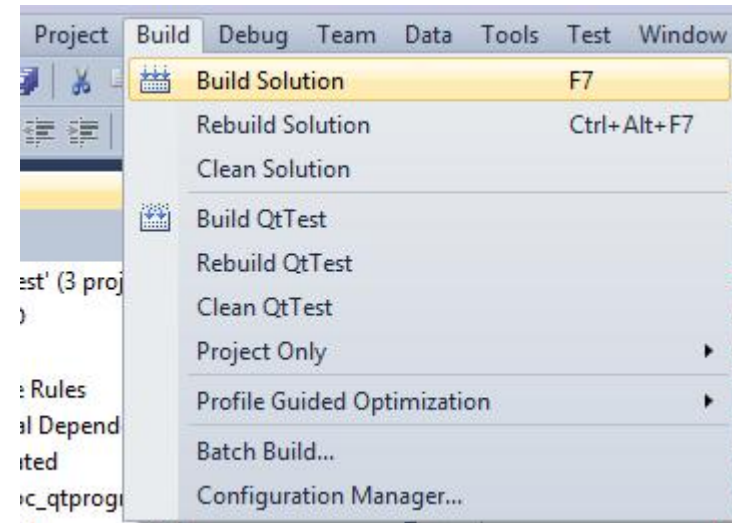
Step 6: Click Generate to generate the solution file

Step 7: Run QtTest.sln to startup

Compile



1. Set QtTest as StartUp Project

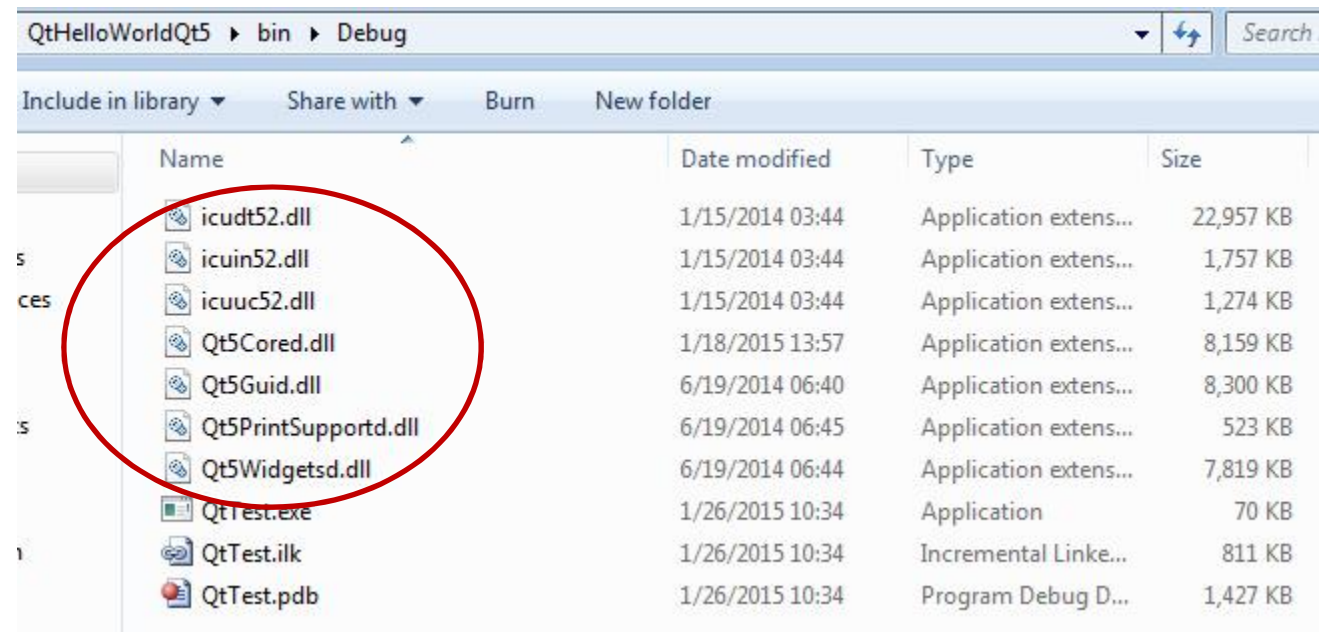


2. Build/Debug Solution

The .exe will be generated in the Debug folder if you build it as Debug version. If it is the first time to open the solution, you can directly start debugging the program.

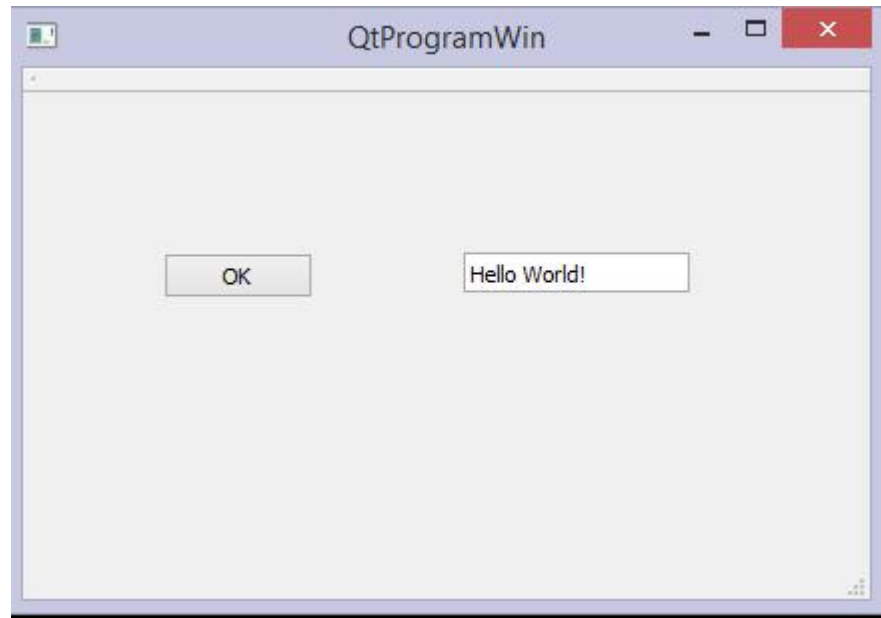
Set DLLs

1. Copy the 7 highlighted .dll files to the Debug folder
(you can find these DLLs in Qt\Qt5.3.1\5.3\msvc2013_opengl\bin
or the **directory of the folder msvc2013_opengl\bin**)
2. You will also need these 7 files for future project, so copy them
to a specific folder to avoid searching them again



Run

- Once you copied the DLLs to the Debug folder, you may press  to run the .exe file.



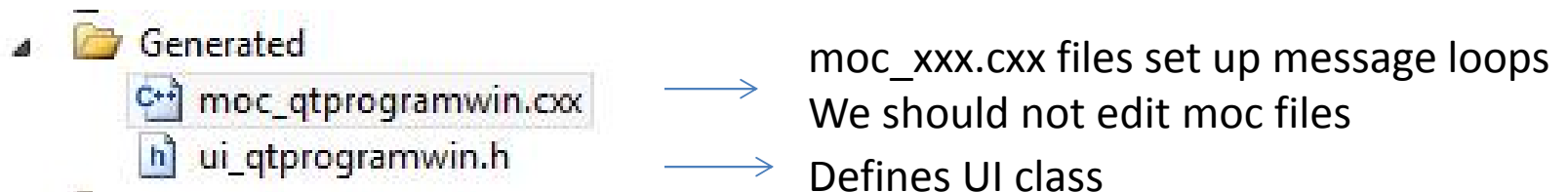
Press OK button to show "Hello World!"

A basic Qt application Structure

Files for a basic Qt application



Generated files by Qt



*.ui file will be translated to ui_xx.h file by Qt mechanism.

UI design

- Code by hand – familiar with Qt
- **Design by Qt Designer – easy for beginners**

UI design demonstration

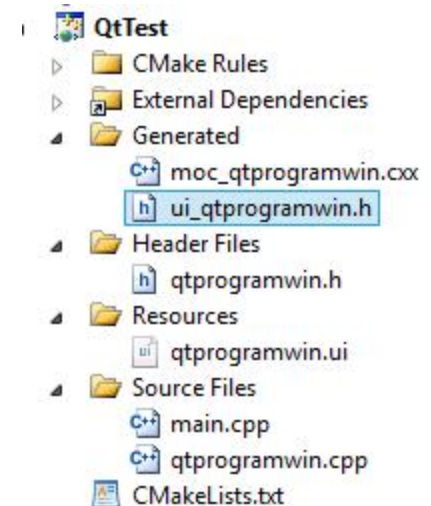
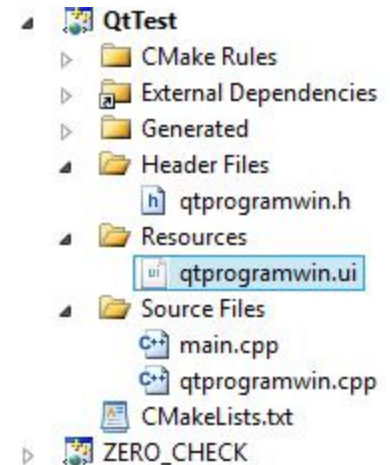
Qt Designer

- Open *.ui file with Qt Designer.
(right click *.ui file in VS and choose to open with Qt Designer)

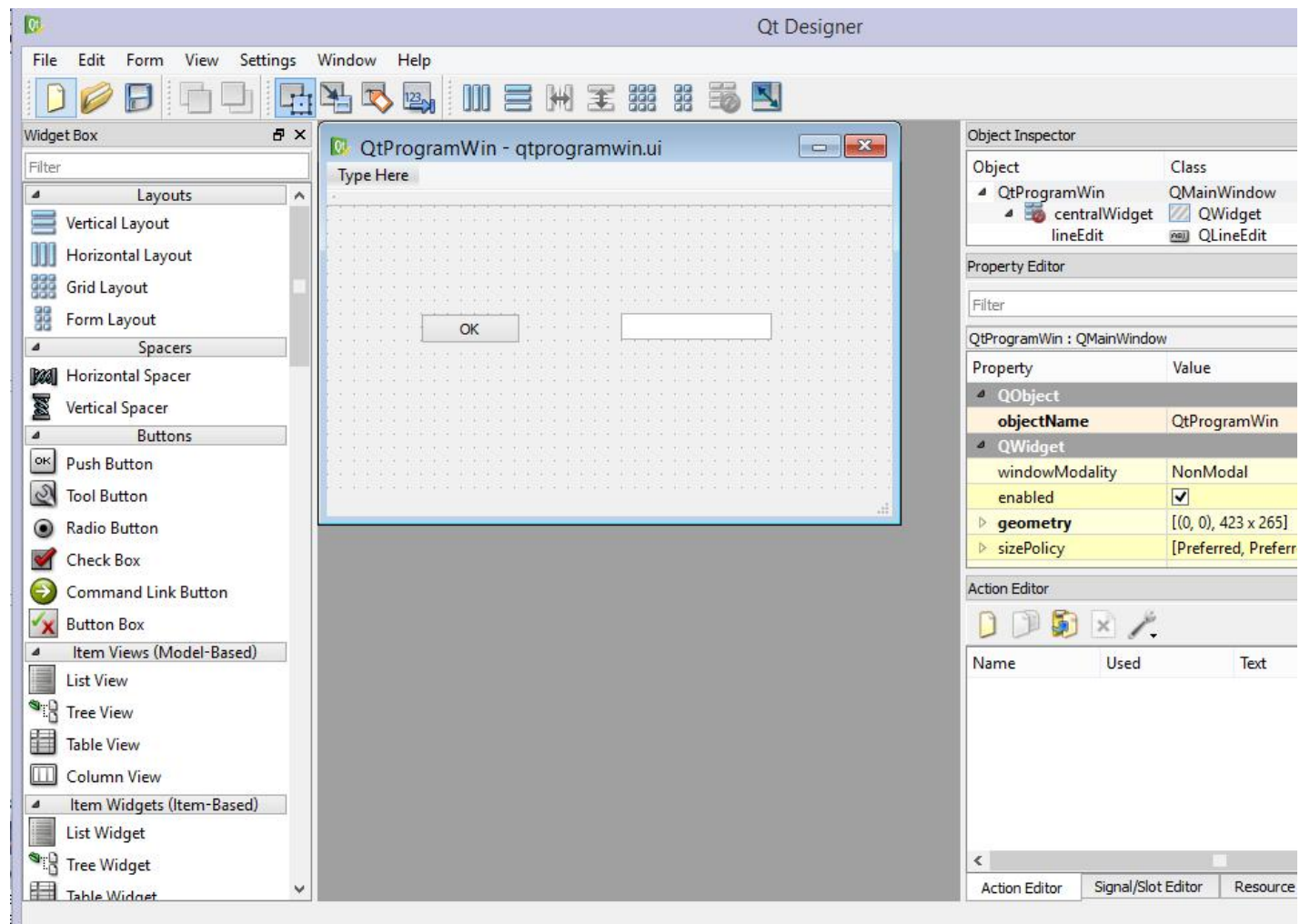
If 'qtdesigner.exe' is not in the list, then click 'Add',
select **designer.exe** from: e.g.

'C:\Qt\Qt5.3.1\5.3\msvc2013_opengl\bin\designer.exe'

- The designed GUI will be scripted in *.ui
- The *.ui file will be translated into C++ code in ui_*.h



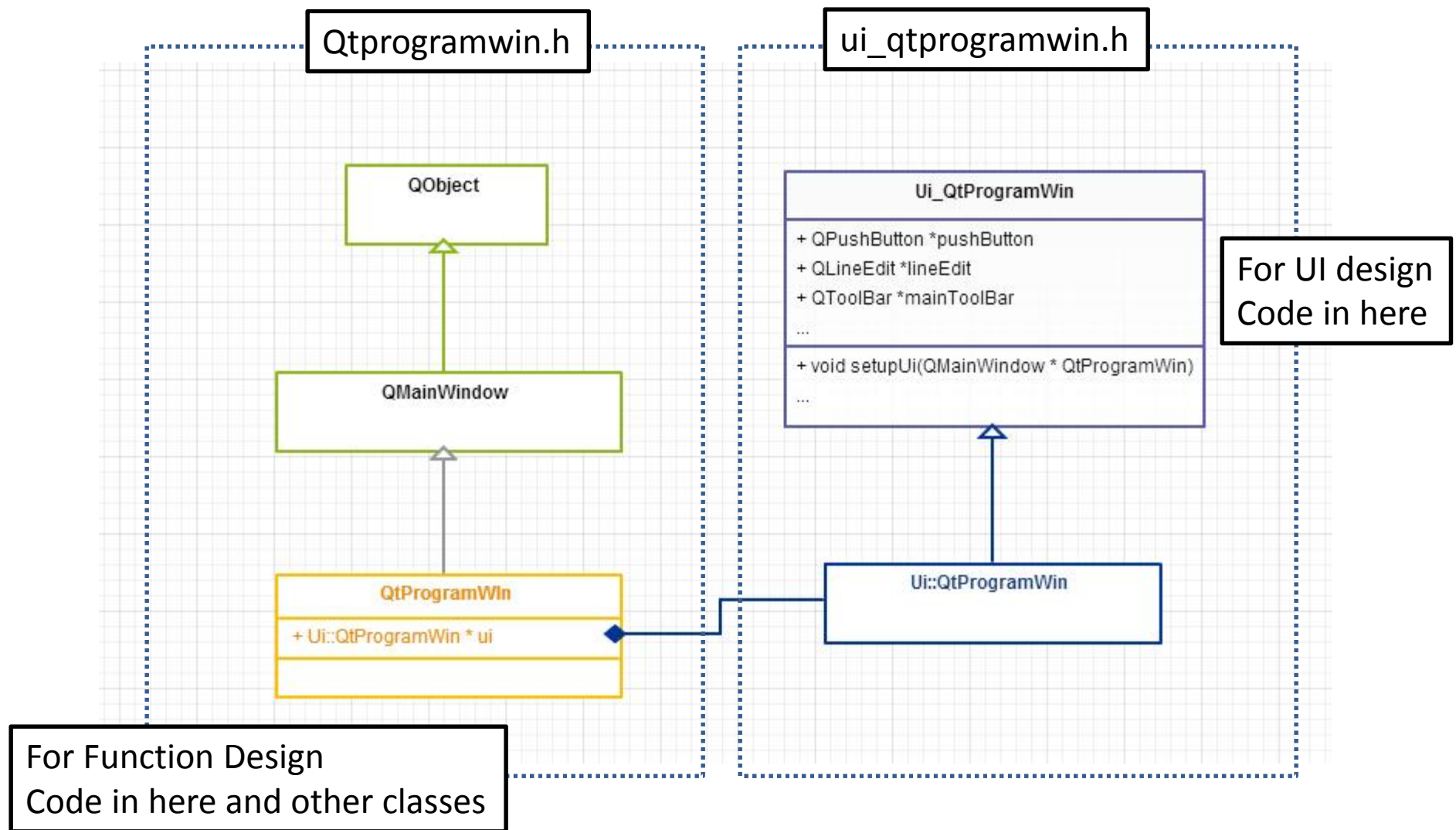
Qt Designer



Qt Designer (cont.)

- After saving the designed GUI, *.ui file will be modified, however, ui_*.h will be translated when you build the solution.
- So if you want to code the newly added button or other objects, you have to build the solution once to generate the new ui_*.h file. And then you will find the code of the new GUI in ui_*.h file.
- If you are designing the GUI without QT Designer(code the GUI directly), there won't be above problems.

UML Diagram of Hello World Qt application



Explain QtHelloWorld

In QtHelloWorld

- The core content of GUI programming is event handling, which is designing a response to a certain event (e.g. mouse clicking).
- In the HelloWorld, ***connect function*** connects the button clicking event to the method setLineEditText(), so that response method will be called when the button is clicked.

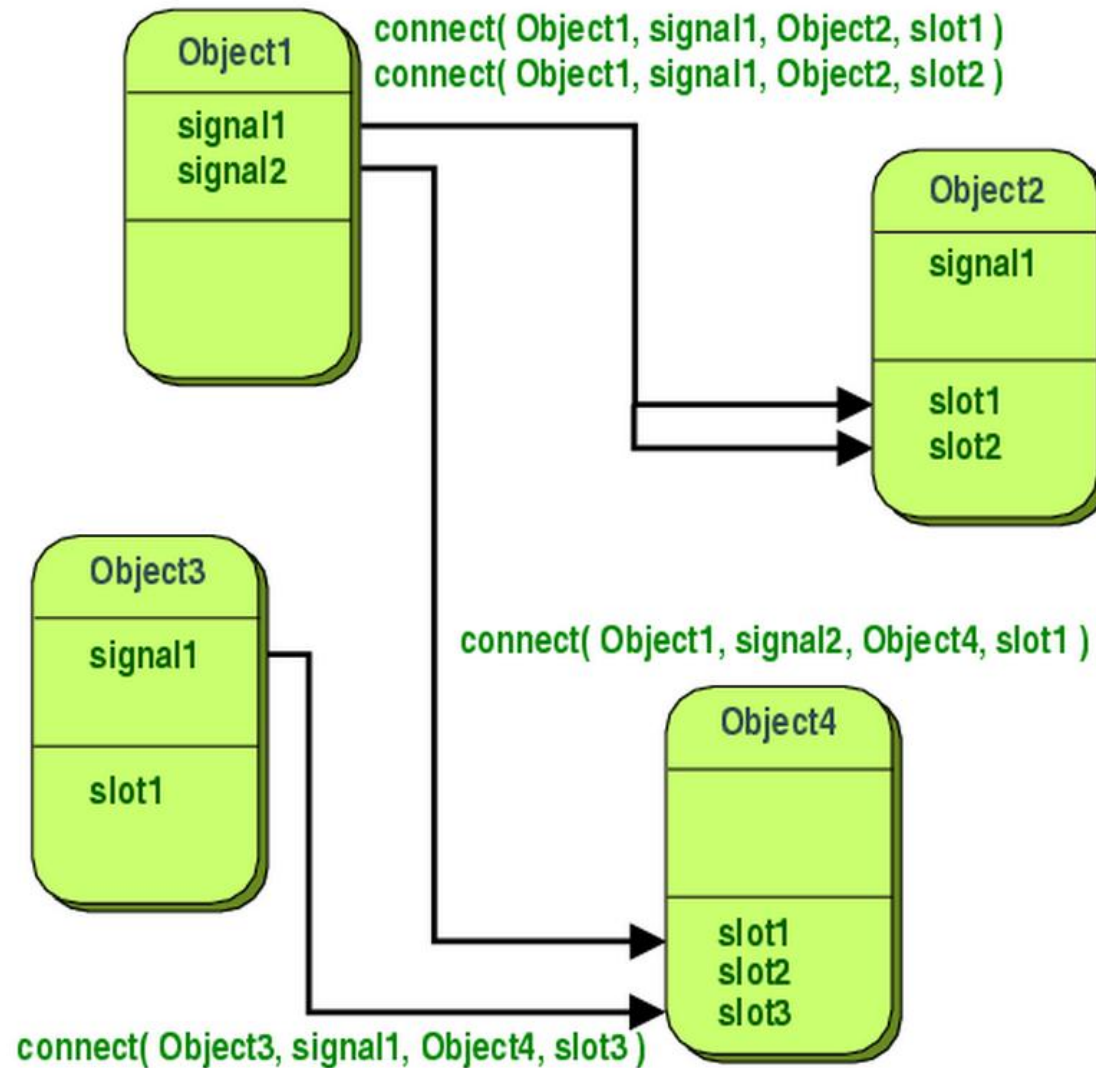
```
QtProgramWin::QtProgramWin(QWidget *parent) :  
    QMainWindow(parent),  
    ui(new Ui::QtProgramWin)  
{  
    ui->setupUi(this);  
    connect(ui->pushButton, SIGNAL(clicked()), this, SLOT(setLineEditText()));  
}
```

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

```
void QtProgramWin::setLineEditText()  
{  
    ui->lineEdit->setText(QString("Hello World!"));  
}
```

In Qt, the events are called ***Signals*** and the response methods are called ***Slot Functions***

Event connection in Qt



QObject-based class declaration

Almost every class in Qt library is derived from base class QObject.
Also you can write your own Qt-style class

```
class Counter
{
public:
    Counter() { m_value = 0; }

    int value() const { return m_value; }
    void setValue(int value);

private:
    int m_value;
};
```

A minimal C++ class declaration

A minimal QObject-based class declaration

```
#include <QObject>

class Counter : public QObject
{
    Q_OBJECT

public:
    Counter() { m_value = 0; }

    int value() const { return m_value; }

    public slots:
        void setValue(int value);

    signals:
        void valueChanged(int newValue);

private:
    int m_value;
};
```

Example of connection

Besides the predefined signals like clicked(), you can define your own signals

```
#include <QObject>

class Counter : public QObject
{
    Q_OBJECT

public:
    Counter() { m_value = 0; }

    int value() const { return m_value; }

public slots:
    void setValue(int value);


signals:
    void valueChanged(int newValue);

private:
    int m_value;
};
```

Connect QObject a's signal to QObject b's slot

```
Counter a, b;
QObject::connect(&a, SIGNAL(valueChanged(int)),
                &b, SLOT(setValue(int)));

a.setValue(12);    // a.value() == 12, b.value() == 12
b.setValue(48);    // a.value() == 12, b.value() == 48
```



Emit a signal

- Sometimes, slots can be implemented by another way:

```
void Counter::setValue(int value)
{
    if (value != m_value) {
        m_value = value;
        emit valueChanged(value);
    }
}
```

emit: Qt keyword for sending signals

When the method setValue() is called, it will send out user defined signal valueChanged().

Exercise

- In HelloWorld app, add a horizontal Sliderbar and connect it to QLineEdit object
- Set the range of the sliderbar from 0 to 999
- When you slide the bar, QLineEdit should show the current value of sliderbar
- Use Google and Qt documents for help

<http://doc.qt.io/qt-5/qslider.html>

<http://doc.qt.io/qt-5/qtwidgets-widgets-sliders-example.html>

