Programmer's Guide for

QT Gui + openFrameworks (OF)

in C++

(Visual Studio 2008 & 2010)

Diogo Cabral

João Valente

CITI and DI, FCT/UNL, Portugal

Wednesday, October 26, 2011

Requirements

QT GUI VS2008: http://qt.nokia.com/downloads/windows-cpp-vs2008

QT Creator: http://qt.nokia.com/downloads/qt-creator-binary-for-windows

QT VS Add-in: http://qt.nokia.com/downloads/visual-studio-add-in

QTwinmigrate: ftp://ftp.qt.nokia.com/qt/solutions/lgpl/qtwinmigrate-2.8_1-opensource.zip

openFrameworks (OF): http://www.openframeworks.cc/download (v0.061 for VS2008, v0.062 for VS2010)

NOTE: This integration was tested on Windows XP, Windows 7, Visual Studio 2008, Visual Studio 2010, QtWinmigrate 2.8.1, Qt Gui 4.7.0, Qt Gui 4.7.1, openFrameworks v0.061 and openFrameworks v0.062.

VisualStudio 2010 NOTE: The $(QTDIR) should be replaced by the absolute path "C:\yourQtpath". We used the QT GUI VS2008, this caused an error in console but the application still worked. Perhaps, building QT GUI for VS2010 the error will disappear.

QT Gui and openFrameworks (OF) working together

Diogo Cabral

1. Download and Install **QT_GUI_VS2008**. Add the **"C:\yourQtpath\bin"** to your environment variable **PATH**.

2. Download and Install **QT Creator** and **QT VS Add-in**.

3. Download and install **openFrameworks**.

4. Download **QTwinmigrate**.

   Run **configure.bat** in the windows command prompt and use QT Creator to build the buildlib project (buildlib\buildlib.pro), with the aim to create the lib file (QtSolutions_MFCMigrationFramework-2.8d.lib) in the \libs directory (if the file is not created, try to build and run the qtwinmigrate project in the Qt Creator).

   OR

   You can also use the Qt command prompt (**Start -> Qt by Nokia -> Qt Command Prompt**) for this step: run the **configure.bat** file, cd buildlib, run nmake.

5. Open any VS Solution from the OF Examples and create a **New Project**. Choose **Qt4Projects -> Qt Library** and add to the solution with the **static library checked** (.lib).

6. In Solution Properties, go to Project Dependencies and choose the OF Project from the Project list. In the "**Depends On**" box, check your Qt Library Project. For VS2010 you need also to check Project Properties -> Common Properties -> Framework and References.

7. Configuration of QT Library Project Properties:
   - Add to C/C++ -> General -> Additional Directories:
     $(QTDIR)\include, (your qtwinmigratedirectory)\src
   - Add to C/C++ -> Preprocessor -> Preprocessor Definitions:
     QT_QTWINMIGRATE_IMPORT, QT_DLL, QT_GUI_LIB, QT_CORE_LIB

8. Configuration of OF Project:
   - Add to C/C++ -> General -> Additional Directories:
     $(QTDIR)\include, (your qtwinmigratedirectory)\src
   - Add to C/C++ -> Preprocessor -> Preprocessor Definitions:
     QT_QTWINMIGRATE_IMPORT, QT_DLL, QT_GUI_LIB, QT_CORE_LIB, QT_THREAD_SUPPORT
   - Add to Linker -> Additional Library Directories :
     (your qtwinmigratedirectory)\lib, $(QTDIR)\lib

- Add to Linker -> Input : **qtmaind.lib, QtCored4.lib, QtGuid4.lib, QtSolutions_MFCMigrationFramework-2.8d.lib**

9. Add this code (highlighted in red) to the **main.cpp** of the OF Project:

```
#include <QtGui\QApplication>

int main(int argc, char *argv[]){//you have to add
                                 //int argc, char *argv[]

    QApplication app(argc, argv); //Qt app
    //Q_INIT_RESOURCE(yourResources); //Load Qt resource (icons)
    //named "yourResources" in the lib. Only needed if you have
    //image icons

    ofAppGlutWindow window;
    ofSetupOpenGL(&window, 1024,768, OF_WINDOW); // <--- setup the
                                                 //GL context

    // this kicks off the running of my app
    // can be OF_WINDOW or OF_FULLSCREEN
    // pass in width and height too:
    ofRunApp( new testApp());

    return app.exec(); //Qt app
}
```

10. Add a Qt Widget Form to the **Form Files** folder in the Qt Library Project**: Right click in Forms folder -> Add-> New Item -> Qt Forms -> Qt Widget Form**. This will add a .ui file to the Form Files and can be edited in the Qt Designer. In the Qt Designer change the name of the Form: **click in the form, using the right button of the mouse and choose Change objectName or use the Object Inspector or Property Editor and Save it**. In this form you can add any Qt widget. This will also generate a ui_youruifile.h where is the code generated from the created form. (see Generated Files folder).

11. Add this code (highlighted in red) in your **header** file (.h) of the Qt Library Project:

```
#include <qwinwidget.h>
#include "GeneratedFiles\ui_yourgeneratedfile.h"


class yourQtClass {
     …
     public:
       void addDock(HWND wnd);

     private:
       QWinWidget * w;
     …
}
```

12. Add this code (highlighted in red) in your **source** file (.cpp) of the Qt Library Project:

```
#include "windows.h"
#include "ui_youruifile.h"

void yourQtClass::addDock(HWND wnd){

    w = new QWinWidget(wnd); //creates an QWinWidget object,
    //which is associated with the HWND wnd (child of).

    QWidget *widget = new QWidget(w);
    //creates an QWidget object, which is associated with the
    //QWinWidget w (child of).

    Ui::yourFormName ui; // this line loads a form created
    //using Qt Designer, see Qt/C++ manuals for this issue[1]

    ui.setupUi(widget); // this line associates ui form to the
    //object widget, see Qt/C++ manuals for this issue[1]

    w->setGeometry(0,0,100,100); // set position (0,0) and
    //dimensions (100,100) for the QWinWidget object

    widget->show();// shows Qwidget object

    w->show();//shows QWinWidget object

}
```

[1] C++ GUI Programming with Qt 4 (First Edition) (ISBN 0131872494) by Jasmin Blanchette & Mark Summerfield (see page 26) or http://doc.qt.nokia.com/4.4/designer-using-a-component.html (last visit: Tuesday, November 23, 2010)

NOTE: QWinWidget objects have priority over regular QWidgets. Properties, like position and dimensions, should be defined on QWinWidget and not on QWidgets.

13. Add your Qt object in the **testApp.h** file (OF Project):

```
#include "..\..\yourQtLibrary\youQtClass.h"

yourQtClass Qtstuff
```

14. Add this code (highlighted in red) to the **testApp.cpp** file (OF Project):

```
#include "windows.h"

void testApp::setup(){
      …
      Qtstuff = yourQtClass(); // constructs your Qt object

      HWND wnd = WindowFromDC(wglGetCurrentDC());
      // gets the HWND of the OF main window

      Qtstuff.addDock(wnd); //class addDock defined above
      …
}
```

15. Compile, Link and Run the Solution.

NOTE: When using this method you are NOT able to USE the Qt Main Window. If you want to know why, just try it and see the result for yourself. ;-)

Qt Signals and Slots in openFrameworks (OF)

João Valente

If you want to connect actions performed in widgets to the OF content, you must use **QObject::connect()** function. To do this, you should derive the class that calls QObject::connect() from Q_Object class[2] and declare the SLOT methods.

For example in **testApp.h**:

```
class testApp : public QObject, public ofBaseApp{

     Q_OBJECT

     private:
                bool drawing;
     public slots:
                void buttonClicked();
…
}
```

In **testApp.cpp**, you should implement the corresponding SLOT method:

```
void testApp::setup(){
  Qtstuff = yourQtClass(); // constructs your Qt object
  HWND wnd = WindowFromDC(wglGetCurrentDC()); // gets the HWND of
  //the OF main window
  Qtstuff.addDock(wnd); //class addDock defined above
  drawing = false;

  QObject::connect(Qtstuff.yourFormName.pushButton,
  SIGNAL(clicked()), this, SLOT(buttonClicked()));
}


void testApp::buttonClicked(){
  drawing = !drawing;
}

void testApp::draw(){
  if(drawing){
    ofSetColor(0,0,0);
    ofDrawBitmapString("hello world", 600,600);
   }
}
```

[2] C++ GUI Programming with Qt 4 (First Edition) (ISBN 0131872494) by Jasmin Blanchette & Mark Summerfield (see page 21)

After that, you have to moc the class (in this case testApp.h) [3]. You can moc the class by running on the windows command prompt the following command line, moc –o yourclass.moc.cc youclass.h, or by setting the following configuration in Custom Build Tool on Properties Menu –> Custom Build Step -> General of the yourclass.h file (just right click on the yourclass.h file to access this menu):

Note1: In VS2010, go first to Properties Menu –> Configuration Properties –> General –> Item Type and choose Custom Build Tool an click on Apply to access the Custom Build Tool.

Note2: You can also do the pairing signals/slots using the Qt Designer, but only between widgets and not between widgets and OF.

In the Solution Explorer, right Click on ***MyHeaderFile.h*** and select **Properties -> Custom Build Step -> General**

Set **Command Line** to	**$(QTDIR)\bin\moc "$(ProjectDir)\src\MyHeaderFile.moc.cc" -o "$(ProjectDir)\src\MyHeaderFile.h"**
Set **Description** to	**Moc'ing *MyHeaderFile.h* ...**
Set **Outputs** to	MyHeaderFile.moc.cc
Set **Additional Dependences**	**$(QTDIR)\bin\moc.exe**

Replace ***MyHeaderFile*** with the actual name of the header file to be Moc'd.

Then you have to add the generated file, **MyHeaderFile.moc.cc**, to your OF Project.

When finished, you'll accomplish something like that:

http://img.di.fct.unl.pt/~diogocabral/qt+ofresult.swf

ACKNOWLEDGMENTS: We would like to thank our supervisor, Prof. Nuno Correia, for his advice and support. We also would like to thank João Silva for improving the method by adding step 6. The work, presented in this guide, was partially funded by the UTAustin-Portugal Program, Digital Media FCT Grant - SFRH / BD / 42662 / 2007 and funded by the FCT/MCTES project "TKB - A Transmedia Knowledge Base for contemporary dance" (PTDC/EAT/AVP/098220/2008).

[3] http://ldmartin68.com/QTSetup4VSNET.html