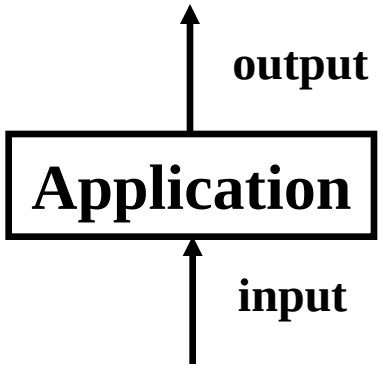
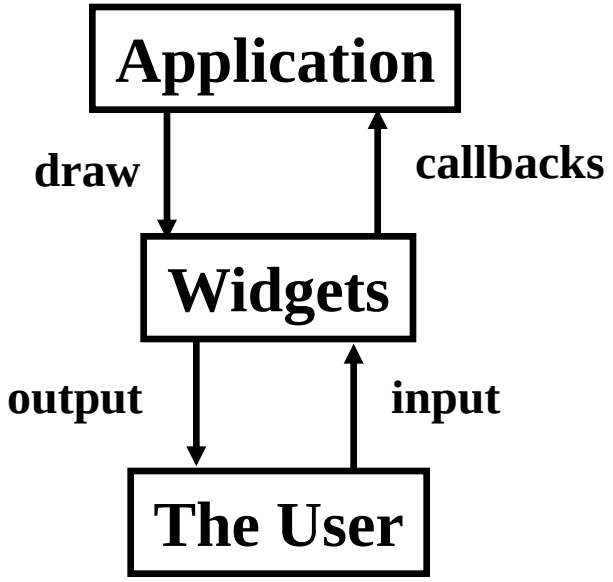
GUI-programmering med Qt

w[illegible]idgets.org

Paradigmer

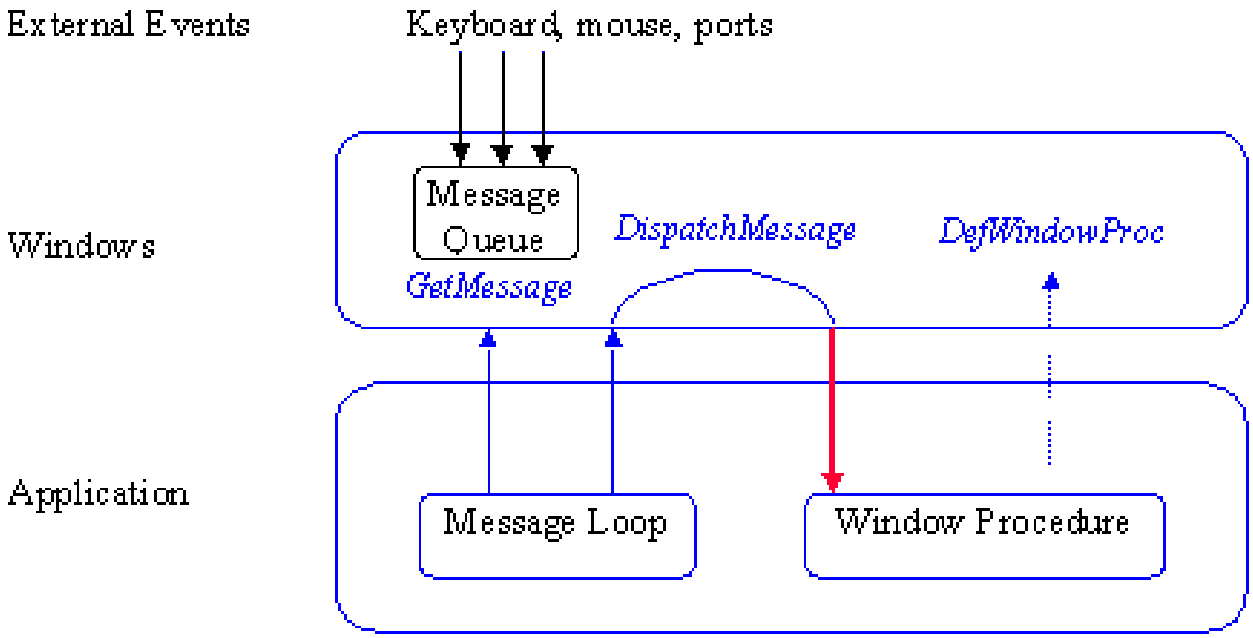




Traditionell command-line

GUI-baserad

Typisk / MS Windows Event/Message loop



Händelseloopen – pseudokod

```
int main() {
    return WinMain();
}

WinMain() {
  while (1) {  // loop forever, waiting for an event
    if (event_exists) {     //there is an event, figure out what to do
      if (event == keydown_a) display('user pressed the A key');
      else if (event == window_resize) display('window resized');
      else if (event == repaint) display('need to repaint window');
      else if (event == keydown_escape) exit_program();
    }
  }
}
```

Huvudloopen i Win32 GUI

```
int WINAPI WinMain(HINSTANCE hinstance, HINSTANCE hprev, PSTR cmdline, int ishow)
{
    HWND hwnd;
    MSG msg;
    //initialization code goes here
    while(1) {
          // Get message(s) if there is one
          if(PeekMessage(&msg,hwnd,0,0,PM_REMOVE)) {
                if(msg.message == WM_QUIT)
                            break;
                TranslateMessage(&msg);
                DispatchMessage(&msg);   //this calls the CALLBACK function WinProc()
          } else {
                DrawScene(); //display the OpenGL/DirectX scene
         }
       }
}
```

NOTERA: Vanlig C-kod

Huvudloopen i Win32 GUI (2)

```
LRESULT CALLBACK WinProc(HWND hwnd, UINT message, WPARAM wparam, LPARAM
      lparam)
{
   PAINTSTRUCT ps;
    // Depending on the message -- we'll do different stuff
    switch(message) {
                case WM_PAINT:
                            Draw();
                            return 0;
                // ESC will quit program
                case WM_KEYDOWN:   //user pressed a key
                            if(GetAsyncKeyState(VK_ESCAPE))  //it was the escape key
                                        PostQuitMessage(0);         //quit program
                            return 0;
                            case WM_DESTROY:  //windows wants the program to die
                case WM_CLOSE:                        //or the user closed the window
                            PostQuitMessage(0);    //quit program
                             return 0;
    }
    return DefWindowProc(hwnd, message, wparam, lparam);
}
```

GUI koncept

- Widgets – grafiska element som erbjuder någon funktionalitet
  - button, toolbar, menu
- Window – container för widgets
- Child/parent – relationer mellan fönster
- Event / message signal – fönster ”kommunicerar” via dessa

Programmering i MS Windows

- Historia
  - WIN32 API: grundbibliotek
  - MFC: C++ -kod runt de flesta WIN32 API-funktionerna
- Många makron
- Mycket kod som ”blivit kvar”
- Icke portabelt, icke gratis
- Fungerar dock ganska bra

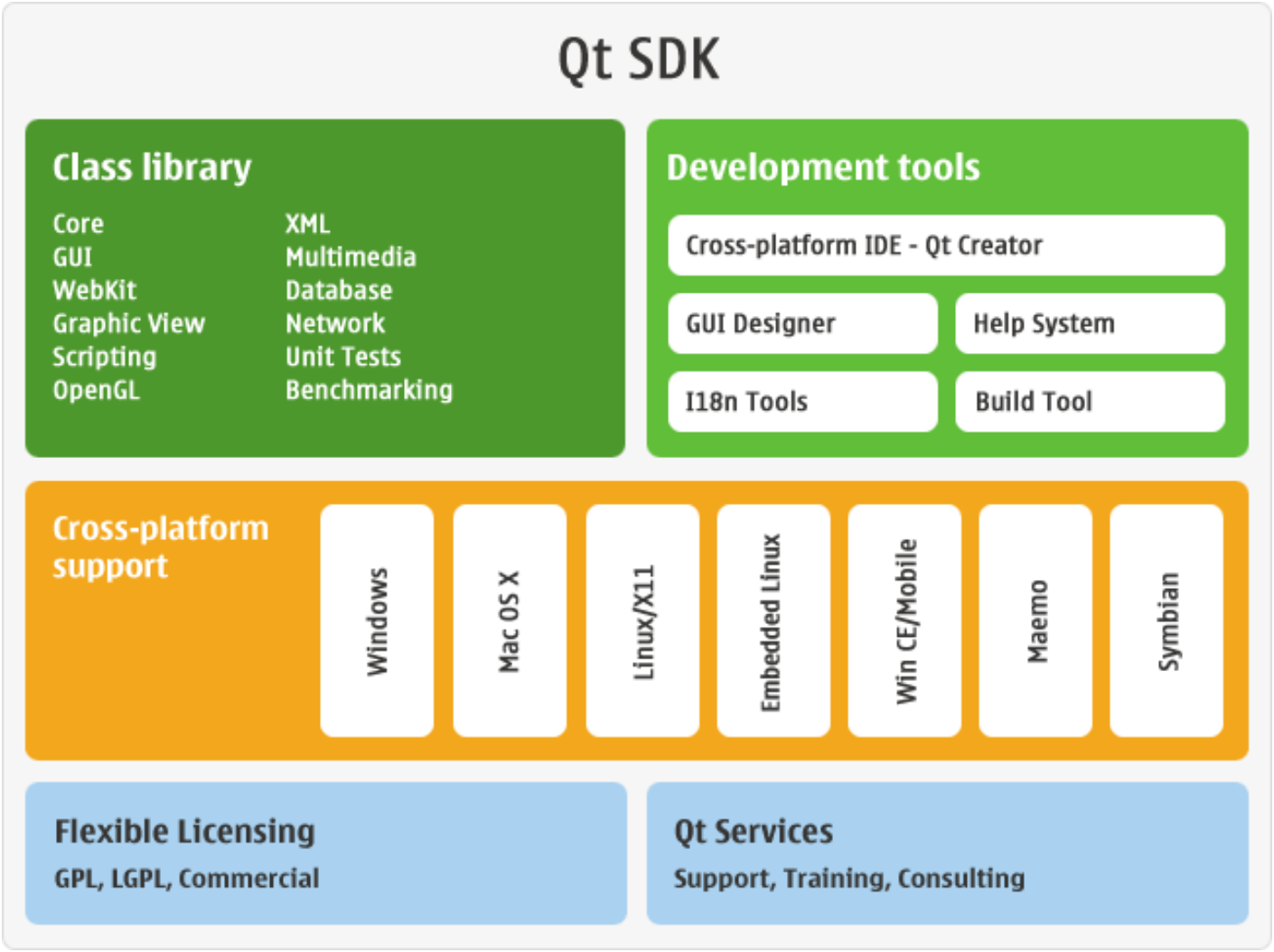
Qt

- Cross-platform C++ GUI, varför?
  - C++ är populärt bland programmerare
  - objektorienterat
  - exekvering är snabb
  - finns stort antal färdiga komponenter
  - kan även accessera lågnivå utan några problem
  - många större mjukvaruprojekt är skrivna med C++

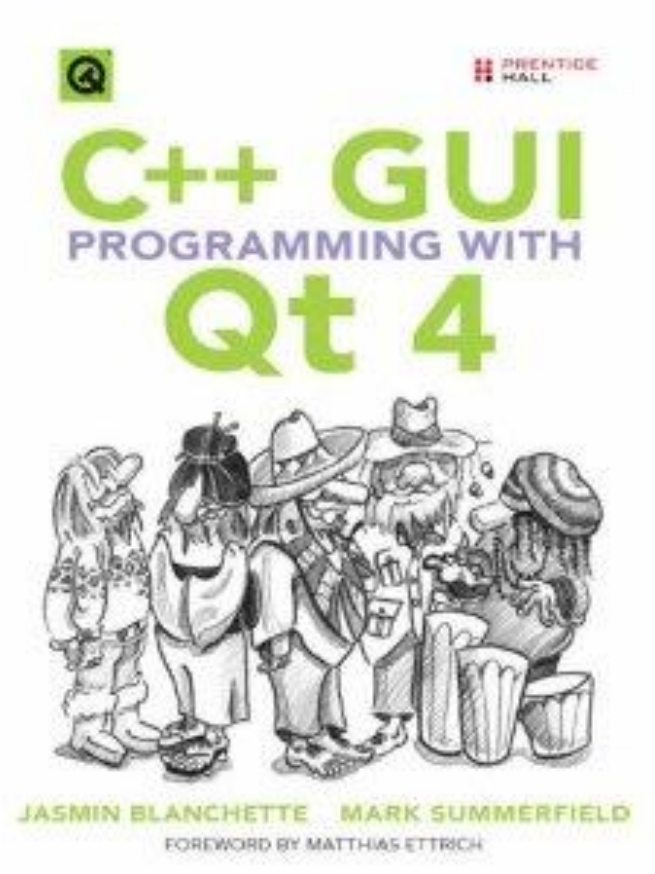
Programmering med Qt / historia

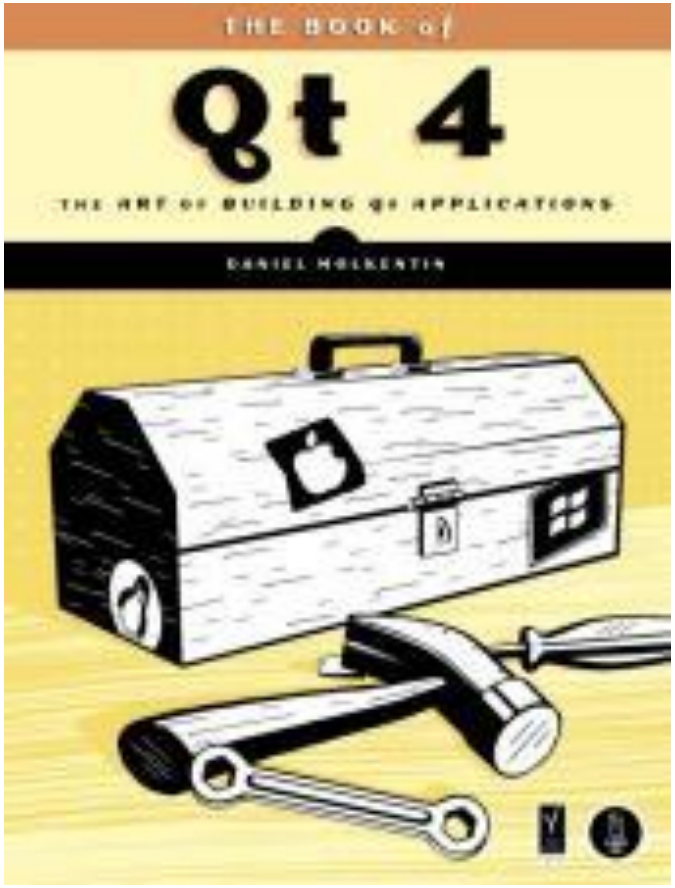
- Första version 1995 av Haavard Nord / Eirik Chambe-Eng (Trondheim, NTNU) / TrollTech
- Första kunder: Metis (.no), ESA
- Qt1.2 1997
- Qtopia Core: 2000 Qt/Embedded
- Qt 3.0 2001: Windows, Mac OS X, Unix, Linux
- Qt 4.0 2005: 500 klasser, 9000 funktioner
- Nokia köper Trolltech för 135 miljoner USD 2008
- Idag: qt.nokia.com: Desktop (Windows, Max OS X, Unix, Linux), Embedded (Linux / Maemo / Symbian)
- 2011 Nokia övergår till Windows OS; Vad händer med Qt
  - Symbian använder Qt, Nokia lovar stöd många år till
  - Qt ursprungligen för generella inbyggda system
- 2012 Digia övertar Qt affärsverksamhet från Nokia
- Qt5 lanseras December 2012

Vad är Qt ?


Qt SDK
Class library
Core
GUI
WebKit
Graphic View
Scripting
OpenGL
XML
Multimedia
Database
Network
Unit Tests
Benchmarking
Development tools
Cross-platform IDE - Qt Creator
GUI Designer
Help System
I18n Tools
Build Tool
Cross-platform support
Windows
Mac OS X
Linux/X11
Embedded Linux
Win CE/Mobile
Maemo
Symbian
Flexible Licensing
GPL, LGPL, Commercial
Qt Services
Support, Training, Consulting

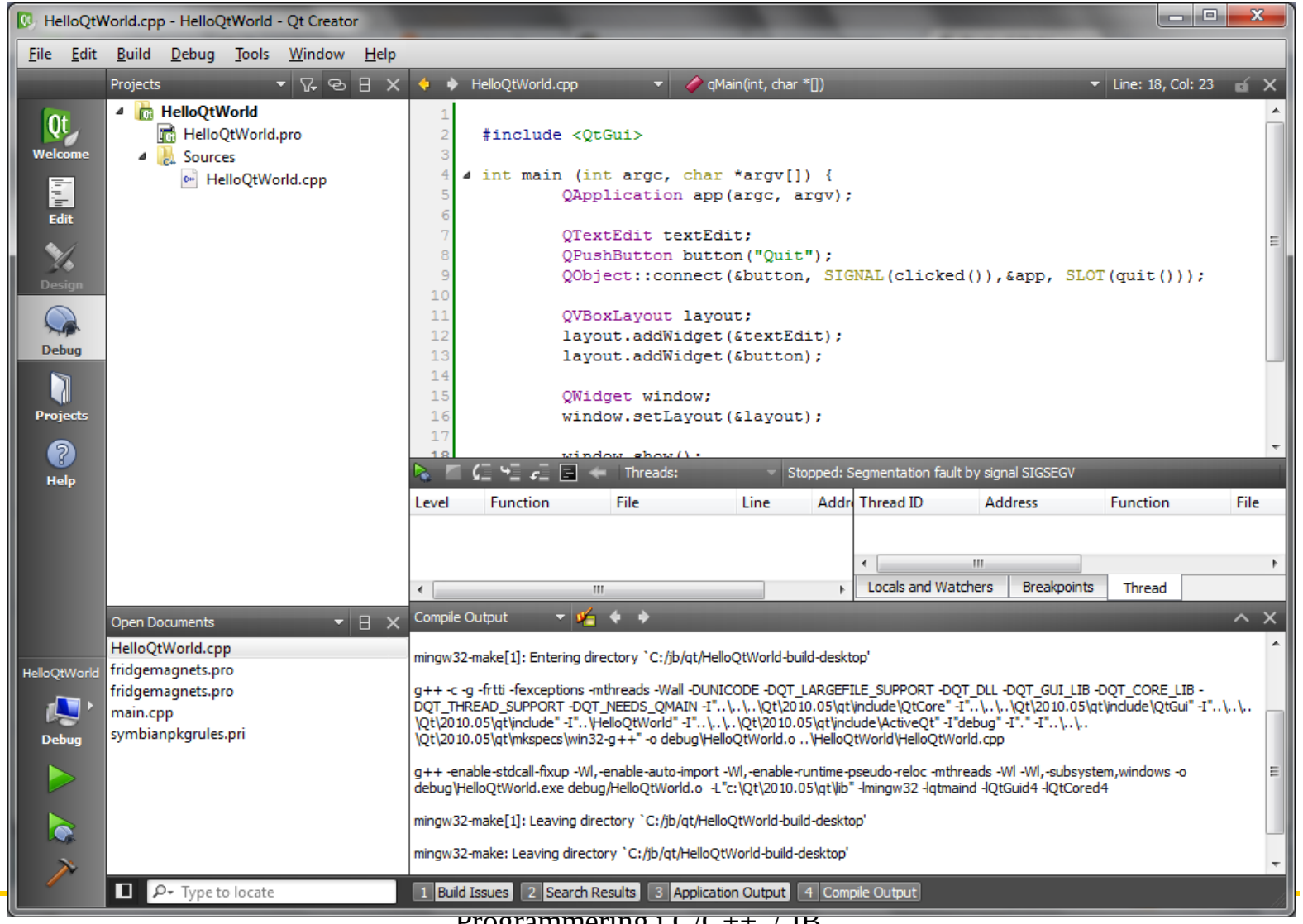
Resurser





Daniel Molkentin: The Art of Building Qt Applications (finns i ÅA:s elektroniska bibliotek)

Qt GUI

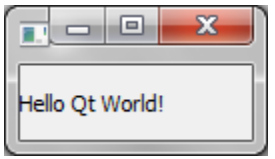

HelloQtWorld.cpp - HelloQtWorld - Qt Creator
File Edit Build Debug Tools Window Help
Projects
HelloQtWorld
HelloQtWorld.pro
Sources
HelloQtWorld.cpp
HelloQtWorld.cpp
qMain(int, char *[])
Line: 18, Col: 23
#include <QtGui>
int main (int argc, char *argv[]) {
QApplication app(argc, argv);
QTextEdit textEdit;
QPushButton button("Quit");
QObject::connect(&button, SIGNAL(clicked()),&app, SLOT(quit()));
QVBoxLayout layout;
layout.addWidget(&textEdit);
layout.addWidget(&button);
QWidget window;
window.setLayout(&layout);
Threads:
Stopped: Segmentation fault by signal SIGSEGV
Level Function File Line Addr
Thread ID Address Function File
Locals and Watchers
Breakpoints
Thread
Open Documents
HelloQtWorld.cpp
fridgemagnets.pro
fridgemagnets.pro
main.cpp
symbianpkgrules.pri
Compile Output
mingw32-make[1]: Entering directory `C:/jb/qt/HelloQtWorld-build-desktop'
g++ -c -g -frtti -fexceptions -mthreads -Wall -DUNICODE -DQT_LARGEFILE_SUPPORT -DQT_DLL -DQT_GUI_LIB -DQT_CORE_LIB -DQT_THREAD_SUPPORT -DQT_NEEDS_QMAIN -I"..\..\..\Qt\2010.05\qt\include\QtCore" -I"..\..\..\Qt\2010.05\qt\include\QtGui" -I"..\..\..\Qt\2010.05\qt\include" -I"..\HelloQtWorld" -I"..\..\..\Qt\2010.05\qt\include\ActiveQt" -I"debug" -I"." -I"..\..\..\Qt\2010.05\qt\mkspecs\win32-g++" -o debug\HelloQtWorld.o ..\HelloQtWorld\HelloQtWorld.cpp
g++ -enable-stdcall-fixup -Wl,-enable-auto-import -Wl,-enable-runtime-pseudo-reloc -mthreads -Wl -Wl,-subsystem,windows -o debug\HelloQtWorld.exe debug/HelloQtWorld.o -L"c:\Qt\2010.05\qt\lib" -lmingw32 -lqtmaind -lQtGuid4 -lQtCored4
mingw32-make[1]: Leaving directory `C:/jb/qt/HelloQtWorld-build-desktop'
mingw32-make: Leaving directory `C:/jb/qt/HelloQtWorld-build-desktop'
Type to locate
1 Build Issues
2 Search Results
3 Application Output
4 Compile Output
Welcome
Edit
Design
Debug
Projects
Help
HelloQtWorld
Debug

Qt HelloWorld

```
#include <qapplication.h>
#include <qlabel.h>

int main (int argc, char *argv[]) {
        QApplication app(argc, argv);
        QLabel *label = new QLabel(NULL);
        label->show();
        return app.exec();
}
```

Kompilering (antar filnamn HelloQtWorld.cpp

```
% qmake -project
% qmake
% make
% ./HelloQtWorld
```



QT / qtcreator

- **Alternativt:**
- **Fil: HelloWorld.pro**

```
****
QT += widgets
# Input SOURCES += HelloQtWorld.cpp
HEADERS += \ HelloQtWorld.h
*****
```

- **qtcreator:**
  - **öppna filen HelloWorld.pro**
  - **bygg / kör**

Qt / Koppla händelser

- Kopplar ihop en användarhändelse (klicka på en knapp) med funktionalitet

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

int main (int argc, char *argv[]) {
        QApplication app(argc, argv);
        QPushButton *button = new QPushButton("Quit");
        QObject::connect(button,
              SIGNAL(clicked()),&app, SLOT(quit()));

        button->show();
        return app.exec();
}
```

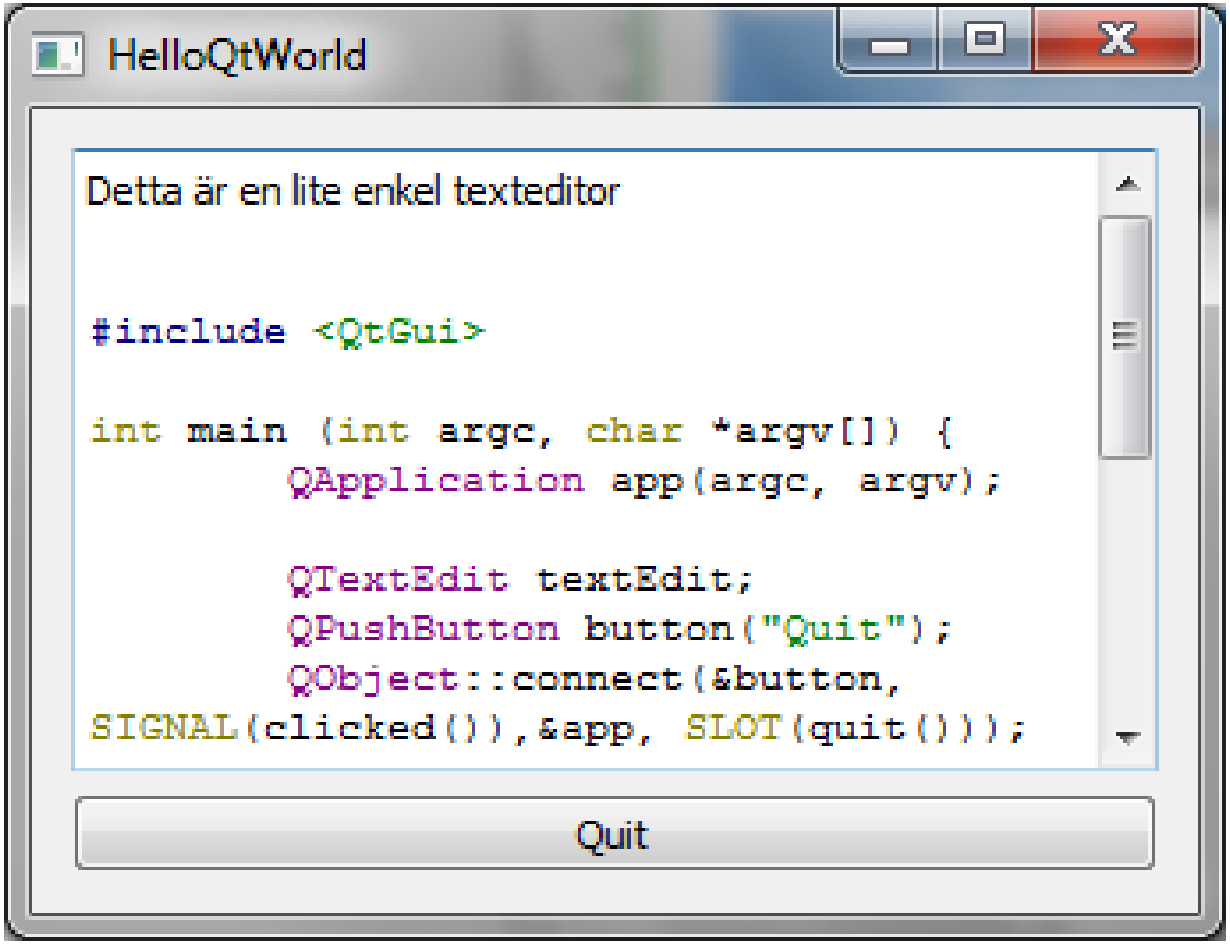
Signals och *slots*

- Via *signaler* meddelar programmoduler att något har hänt
- *Slots* är funktioner som reagerar på signaler
- En klass som förstår sig på *signal* och *slot* måste
  - ärva `QObject` (direkt, eller indirekt)
  - Aktiveras genom att i klassdeklarationen lägga till `Q_OBJECT`

Notepad

```
#include <QtWidgets>

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

        QTextEdit textEdit;
        QPushButton button("Quit");
        QObject::connect(&button,
              SIGNAL(clicked()),&app, SLOT(quit()));

        QVBoxLayout layout;
        layout.addWidget(&textEdit);
        layout.addWidget(&button);

        QWidget window;
        window.setLayout(&layout);

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

Notepad - resultat



Notepad 2

- Ärver Qwidget

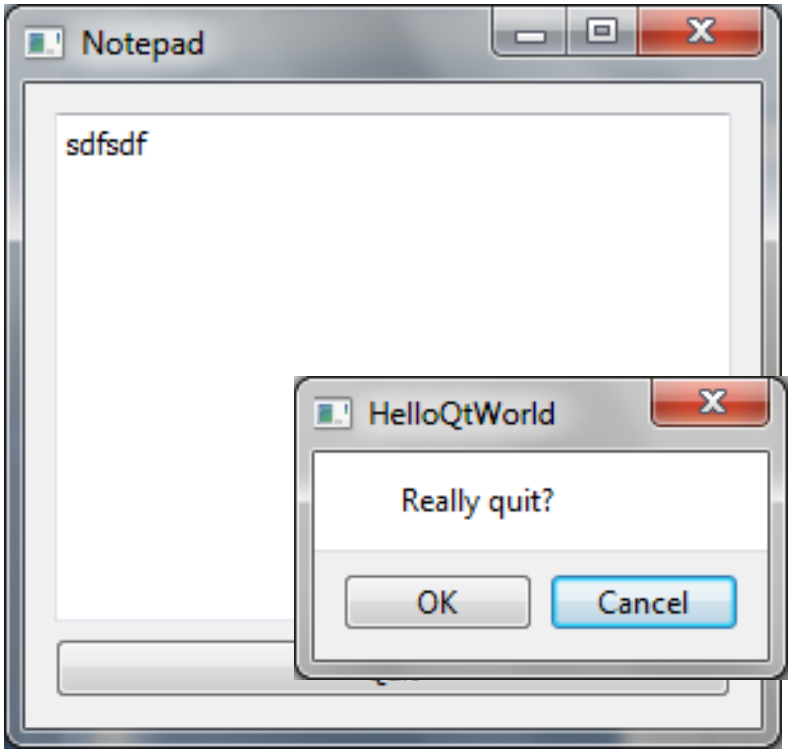
```
//HelloQtWorld.h
#ifndef HELLOQTWORLD_H
#define HELLOQTWORLD_H
#include <QtWidgets>

class myWidget: public QWidget {
    Q_OBJECT
public:
    myWidget();

private slots:
    void quit();

private:
    QTextEdit *textEdit;
    QPushButton *button;
};

#endif // HELLOQTWORLD_H
```

Notepad 2

```
//HelloQtWorld.cpp
#include <QtWidgets>
#include "HelloQtWorld.h"

myWidget::myWidget() {
     textEdit = new QTextEdit;
     button = new
   QPushButton("Quit");

     connect(button,
   SIGNAL(clicked()),this,
   SLOT(quit()));

     QVBoxLayout *layout = new
   QVBoxLayout;
     layout->addWidget(textEdit);
     layout->addWidget(button);

     setLayout(layout);
     setWindowTitle("Notepad");
}
return app.exec();
}
```

```
void myWidget::quit() {
    QMessageBox msgBox;
    msgBox.setText("Really quit?");

   msgBox.setStandardButtons(QMessageBox
   ::Ok | QMessageBox::Cancel);

   msgBox.setDefaultButton(QMessageBox::
   Cancel);
    if (QMessageBox::Ok ==
   msgBox.exec()) {
        QWidget::close();
    }
}

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

        myWidget window;
        window.show();

        return app.exec();
}
```

Notepad 2



Menyer....

- Bara ett text-editor-fönster är något oanvändbart, borde ha fil-access, sköts genom att:
  - 1. Addera menyer
  - 2. Koppla menyerna till funktionalitet

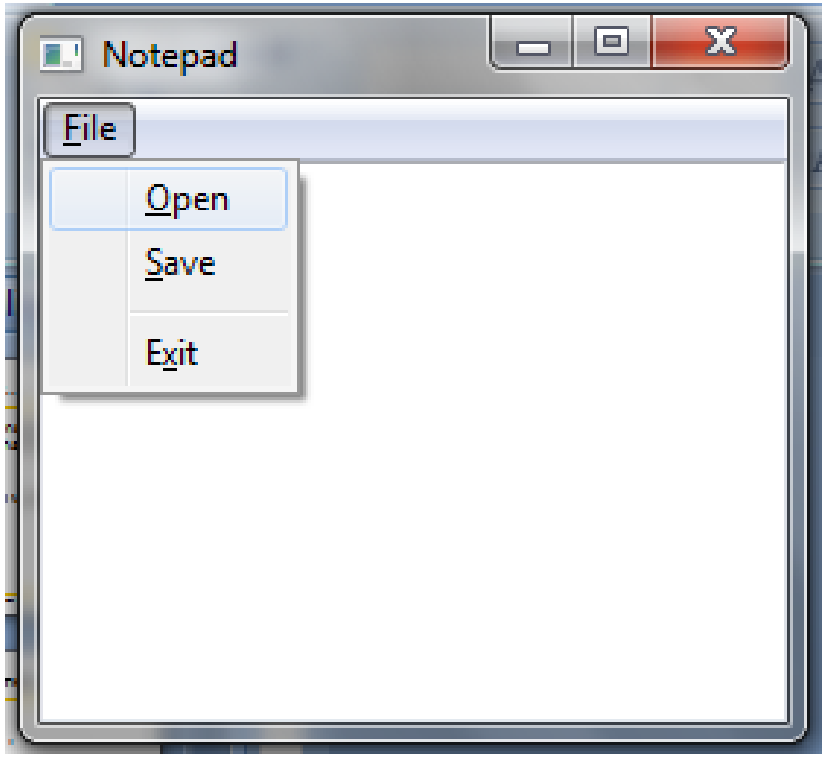
Adderar meny

```
//HelloQtWorld.h
#ifndef HELLOQTWORLD_H
#define HELLOQTWORLD_H
#include <QtWidgets>

class myWidget: public QMainWindow {
    Q_OBJECT
public:
    myWidget();

private slots:
    void open();
    void save();
    void quit();

private:
    QTextEdit *textEdit;
    QAction *openA;
    QAction *saveA;
    QAction *exitA;

    QMenu *fileMenu;
};

#endif // HELLOQTWORLD_H
```

Adderar meny

```
myWidget::myWidget() {
    openA = new QAction("&Open", this);
    saveA = new QAction("&Save", this);
    exitA = new QAction("E&xit", this);

    fileMenu = menuBar()->addMenu("&File");
    fileMenu->addAction(openA);
    fileMenu->addAction(exitA);
    fileMenu->addSeparator();
    fileMenu->addAction(saveA);

    textEdit = new QTextEdit;
    setCentralWidget(textEdit);

    setWindowTitle("Notepad");
}
```

Adderar meny



Koppla aktioner till menyn

- Kopplar ihop trigger-signalerna från menyn med mottaggarfunktionerna (slot) i klassen

```
myWidget::myWidget() {
    openA = new QAction("&Open", this);
    saveA = new QAction("&Save", this);
    exitA = new QAction("E&xit", this);
    connect(openA, SIGNAL(triggered()),this, SLOT(open()));
    connect(saveA,SIGNAL(triggered()),this, SLOT(save()));
    connect(exitA, SIGNAL(triggered()),this, SLOT(quit()));
    fileMenu = menuBar()->addMenu("&File");
    ...
    ...
    ...
}
```

Koppla aktioner till menyn

- Sedan måste det finnas någon implementation av meny-funktionaliteten

```
// wxhello.cpp
....
void myWidget::open() {
    statusBar()->showMessage("Open triggered");
}
void myWidget::save() {
    statusBar()->showMessage("Save triggered");
}
....
```

File/open funktionalitet

- 1. Låt användaren öppna en fil
  - Det finns färdiga komponenter för att välja filer: QFileDialog
- 2. Öppna filen, och läs in innehållet till text-editorn
  - Igen färdiga komponenter, såsom QFile

QFileDialog

- I koden:

```
void myWidget::open() {
    QString fileName = QFileDialog::getOpenFileName(this, "Open File",
   "", "Text Files (*.txt);;C++ Files (*.cpp *.h)");
    if (fileName != "") {
        QFile file(fileName);
        if (!file.open(QIODevice::ReadOnly)) {
            QMessageBox::critical(this, "Error", "Could not open
   file");
            return;
        }
    QString contents = file.readAll().constData();
    textEdit->setPlainText(contents);
    file.close();
    }
}
```

QFileDialog (2)

```
void myWidget::save() {
    QString fileName = QFileDialog::getSaveFileName(this,
   "Save File", "", "Text Files (*.txt);;C++ Files (*.cpp
   *.h)");
    if (fileName != "") {
        QFile file(fileName);
        if (!file.open(QIODevice::ReadWrite)) {
            QMessageBox::critical(this, "Error", "Could not
   open file");
            return;
        }
    QTextStream st(&file);
    st<<textEdit->toPlainText();
    file.close();
    }
}
```

QMessageBox

- Denna funktion visar ett enkelt meddelande på skärmen
  - Använder här för ”About-stil”-boxen från QMessageBox-klassen

```
void myWidget::about() {
    QMessageBox::about ( this, "About", "ÅA/C++/Qt
   Demonstration JB/2013");
}
```