Objektorientierte Programmierung mit C++ (WS 2010)

Dr. Andreas F. Borchert, Tobias Brosch
Institut für Angewandte Informationsverarbeitung
Universität Ulm

Blatt 10: Abgabetermin 12. Januar 2010 11 Uhr

Das erste Qt-Programm

Nach D. Molkentin. Qt 4. Einführung in die Applikationsentwicklung. Open Source Press 2006. Weiterführende Informationen zu Qt unter Anderen auf http://qt.nokia.com/. Eine Liste von Anwendungen (wie z.B. Autodesk, Google Earth, KDE, VLC media player), welche Qt verwenden, findet sich z.B. unter http://en.wikipedia.org/wiki/Qt_(framework).

Der Tradition aller Programmierbücher folgend, beginnt auch Unsere Einführung in Qt mit einem "Frohe Weihnachten und einen guten Start ins Jahr 2011!" Programm:

```
// file helloWorld.cpp
#include <QtGui>

int main(int argc, char *argv[]) {
 QApplication a(argc, argv);
 QString msg("Frohe Weihnachten und einen guten Start ins Jahr 2011!");
 QLabel label(msg);
```

```
  label.show();

  return a.exec();
}
```

Da Qt in manchen Bereichen kein reines C++ ist, sondern den C++-Standard vielmehr erweitert, gibt es den Meta-Object-Compiler moc. Dieser wird unter anderem dafür benötigt, um den "richtigen" C++-Code, der von `QObject` abgeleiteten Klassen zu generieren. Ein weiterer Fall, in welchem moc zusätzlichen Code generiert, ist das Property-System von Qt.

Dies begründet die Verwendung eines eigenen Buildsystems von Qt, namentlich `qmake`, welches sich um den Aufruf von `moc`, sowie dem Linken des Programms gegen die Qt-Libraries kümmert. Mittels `qmake` lassen sich Projektdateien erstellen und auswerten, aus denen anschließend z.B. ein Makefile generiert werden kann, mit welchem sich die Anwendung dann tatsächlich übersetzen lässt.

Mit dem Aufruf von

```
qmake -project
```

im Ordner `myFolder`, wird eine Projektdatei `myFolder.pro` erstellt (eine bestehende Datei dieses Namens wird überschrieben!), die als Basis für das eigene Projekt dienen kann.

Um Projektdateien etwas besser kennenzulernen, wollen wir jedoch zunächst selbst eine Projektdatei schreiben. Lest Euch hierzu die kleine Einführung unter http://doc.trolltech.com/4.7/qmake-tutorial.html#adding-platform-specific-source-files durch und schreibt eine kleine Projektdatei, mit dem Namen `helloWorld.pro`, welche Euch das Programm aus der Datei `helloWorld.cpp` übersetzt.

Je nach Plattform, könnt Ihr nun aus der Projektdatei ein "Makefile" generieren. Makefile in dem Sinne, dass es tatsächlich ein Makefile ist, oder aber eine Projektdatei, von einer auf der Plattform üblichen IDE, oder einem anderen Buildsystem. Hinweise dazu findet Ihr z.B. auf http://doc.qt.nokia.com/4.7/qmake-running.html. Unter Mac erzeugt man ein Makefile mit `qmake -spec macx-g++ myProject.pro`. Alternativ setzt man zuvor die Umgebunsvariable `QMAKESPEC` auf `macx-g++` mittels `export QMAKESPEC=macx-g++` und ruft anschließend `qmake myProject.pro` auf.

Nun ergänzen wir unser erstes Programm noch um einen Button, um das Programm zu beenden. In Qt kann ein Button von der Klasse `QPushButton` dargestellt werden. Dem Konstruktor von `QPushButton` kann ein String übergeben werden, der die Aufschrift des Buttons darstellt. Unsere beiden Objekte, das Label und der Button, sollen untereinander dargestellt werden. Hierzu bietet sich ein `QWidget` als Container an. Dieses nennen wir `window`. Um die beiden Objekte untereinander darzustellen, verwenden wir ein `QVBoxLayout` mit dem Namen `vBoxLayout`. Mittels der Methode `addWidget` von `QVBoxLayout`, fügen wir Unser Label bzw. Unseren Button dem Widget hinzu (so wie die meisten Methoden in Qt erwartet `addWidget` Zeiger).

Das Layout von Unserem `QWidget`, setzen wir mit der Methode `setLayout`. Um Unserem Exit-Button die gewünschte Funktionalität zu verleihen, nutzen wir den Signal-Slot-Mechanismus von Qt. Objekte können Signale aussenden, welche von Slots empfangen werden. Eine solche Verbindung kommt mittels der Methode

```
bool QObject::connect (const QObject* sender, const char* signal, \\
        const QObject* receiver, const char* method)
```

zustande, welche uns (zur Runtime) zurückliefert, ob die Verbindung hergestellt werden konnte. In unserem Fall, ist der Sender der Button, und der Receiver unsere `QApplication`. Die Zeichenketten

sollten immer von den Makros `SIGNAL` und `SLOT` generiert werden. Dies sieht dann wie folgt aus:

```
QObject::connect(exitButton, SIGNAL(clicked()), &a, SLOT(quit()));
```

Vergesst final nicht, mittels der Methode `show` von `QWidget` unser `window` sowie alle darin enthaltenen Objekte anzuzeigen.

Speicherverwaltung in Objekthierarchien (wichtig!)

Qt kümmert sich teilweise um die delete-Aufrufe Eurer Objekte. Von `QObject` abgeleitete Klassen (in unseren GUIs werden dies die Meisten sein), können Kind-Objekte besitzen. Wird ein `QObject` gelöscht, so wird auf allen Kind-Objekten delete aufgerufen. Das bedeutet, dass sich der Programmierer um das löschen aller Vaterobjekte kümmern muss, und das alle Kind-Objekte, mit `new` auf dem Heap angelegt werden.

In unserem Beispiel heißt das z.B. ,dass wir unsere QAppllication und unser QWidget normal auf dem Stack generieren können (dann werden diese am Ende unserer Methode automatisch zerstört) und alle Anderen Objekte mit `new` erzeugt werden.

Hinweise zur Nutzung auf Unseren Rechnern

Qt bietet eine GUI-Bibliothek (darüber hinaus auch einige sehr interessante Bibliotheken, welche nichts mit GUIs zu tun haben), die auf vielen Platformen (viele Linux-Systeme, teilweise auch auf Mobile Devices, Mac OSX sowie Windows) nahezu identische Ergebnisse liefert. Leider wird derzeit Solaris in der aktuellen Version nicht unterstützt. Daher kann lediglich auf folgenden Rechnern in unserem Netz, welche im Pool-Raum E44 in der Helmholtzstraße 18 (direkt neben der Bibliothek) stehen, mit Qt gearbeitet werden:

- aprikose
- feige
- hohentwiel
- pflaume
- zugspitze
- birne
- feldberg
- mandarine
- quitte
- bodensee
- fortuna
- pfirsich
- watzmann

Falls Ihr über ssh auf einem dieser Rechner arbeiten wollt, zunächst mittels

```
ssh –XY username@thales.mathematik.uni–ulm.de
```

auf die thales (oder theseus) verbinden und anschließend mit

```
ssh einerDerRechnernamenOben
```

auf einen der Rechner verbinden (in unserem Netz wird ssh automatisch mit den Optionen `-XY` aufgerufen).

Beachtet bitte, dass dieses Vorgehen eine schnelle Internetverbindung voraussetzt. Ein Arbeiten im Uni-Netz (hier kann die Verbindung zu thales/theseus natürlich entfallen) oder lokal (am eigenen Rechner oder an der Universität), ist daher vorzuziehen.

Auf der Webseite `http://www.mathematik.uni-ulm.de/cgi-bin/cgi-wo.pl` unter He18/E44, könnt Ihr nachsehen, welche der oben genannten Rechner gerade frei sind (Diese sind grün gekennzeichnet. Ihr könnt auch auf einem bereits benutzten Rechner arbeiten, jedoch wird ein Rechner dadurch natürlich nicht schneller).

Einreichen der Lösung mit:

```
submit cpp 10 helloWorld.cpp helloWorld.pro
```

Viel Spaß, ein frohes Fest und einen guten Start in das neue Jahr!