

QT – C++ GUI Toolkit

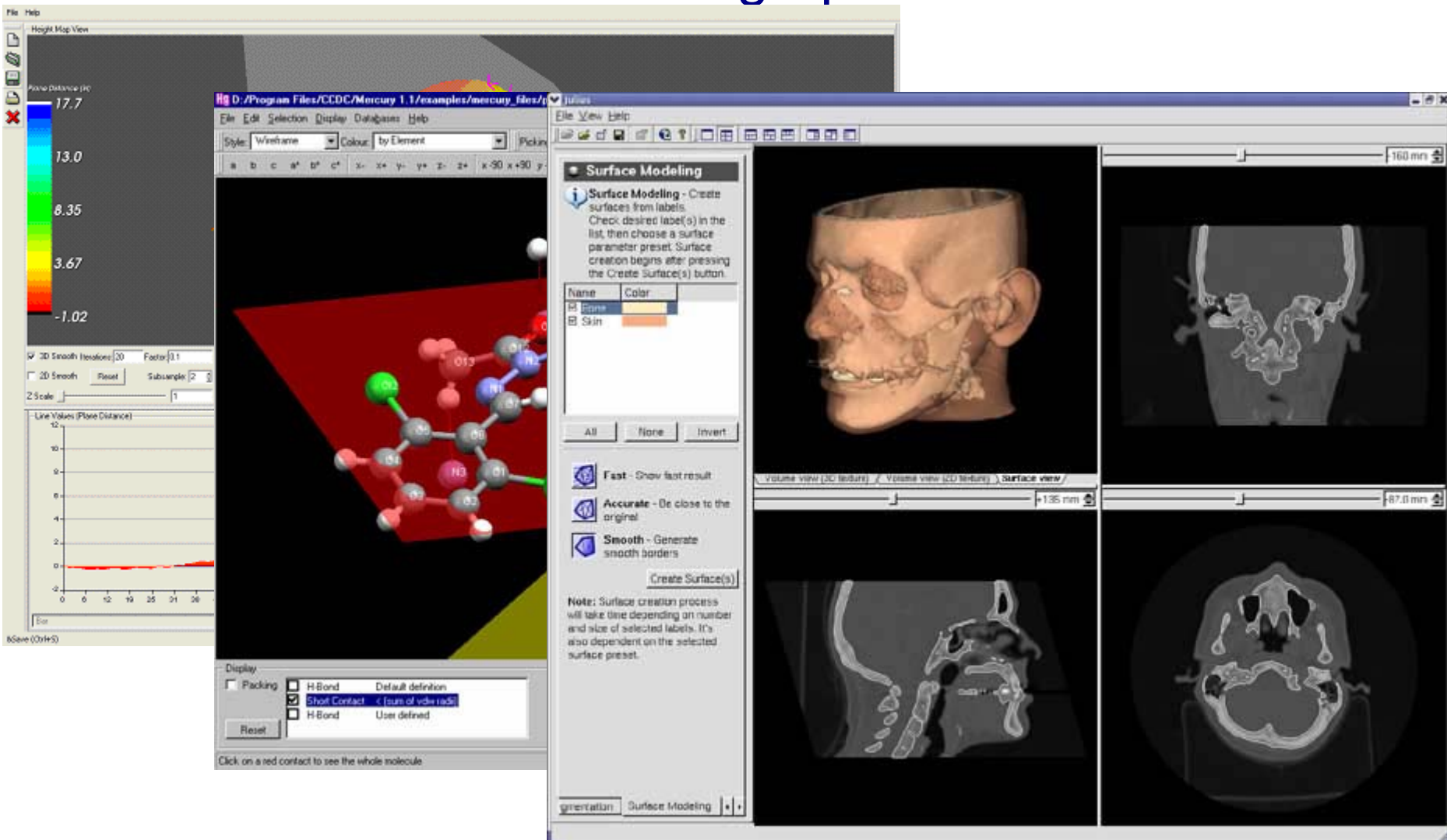
Stefanie Kettner

08 May 2007

chair for computer aided medical procedures

department of computer science | technische universität münchen

QT - Motivation für eine graphische Oberfläche



Einführung in QT

- C++ Framework
- Cross Plattform (Win, Mac, Unix/Linux,...)
- Komplette Objektorientiert
- Grosse Auswahl an *Widgets* (diverse GUI Funktionalitäten)
- Erweiterung des *C++ Object Models* mit neuen Methoden für die Inter-Objekt Kommunikation (*Signals and Slots*)

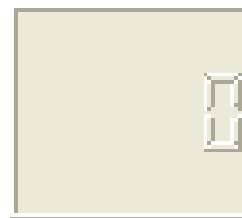
Widgets

- Widgets sind visuelle Elemente einer graphischen Benutzerschnittstelle
- Grosse Auswahl an vordefinierten Widgets für GUIs
- Einfache Vererbung von vordefinierten Widgets um individuelle Funktionalität hinzuzufügen
- Können beliebig kombiniert werden
- Einfache Knöpfe und Textfelder bis zu komplexen Dialogen
- Widget Gallery
<http://doc.trolltech.com/4.0/gallery.html>

Widget Gallery – Display Widgets



- QProgressBar

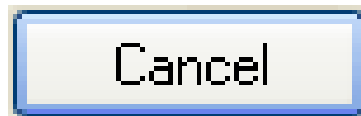


- QLCDNumber

Text Label

- QLabel

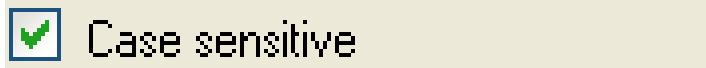
Widget Gallery – Buttons



- QPushButton



- QToolButton



- QCheckBox

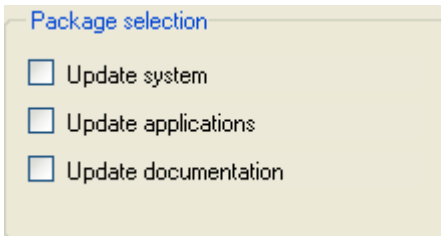


- QRadioButton

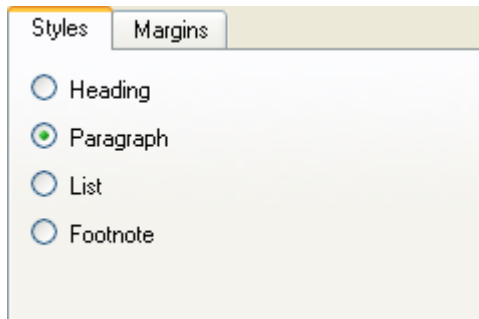
Widget Gallery – Containers



- QToolBox

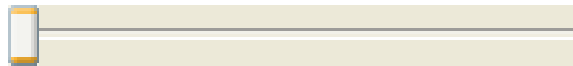


- QGroupBox



- QTabWidget

Widget Gallery – Input Widgets



The **QTextEdit** class provides a widget that is used to edit and display both plain and rich text.

QTextEdit is an advanced WYSIWYG viewer/editor that can display images, lists and tables.



- QComboBox
- QLineEdit
- QSlider

- QDial

- QTextEdit

- QScrollBar

Layout Manager

- QVBoxLayout
- QHBoxLayout
- QGridLayout

`myLayout->addWidget(myWidget)`

`myLayout->addLayout(myInnerLayout)`

`myLayout->addSpacing(int space)`

or

`insert<...>(int index, <...> myWidget)`



Hello World Button – Code Snippet

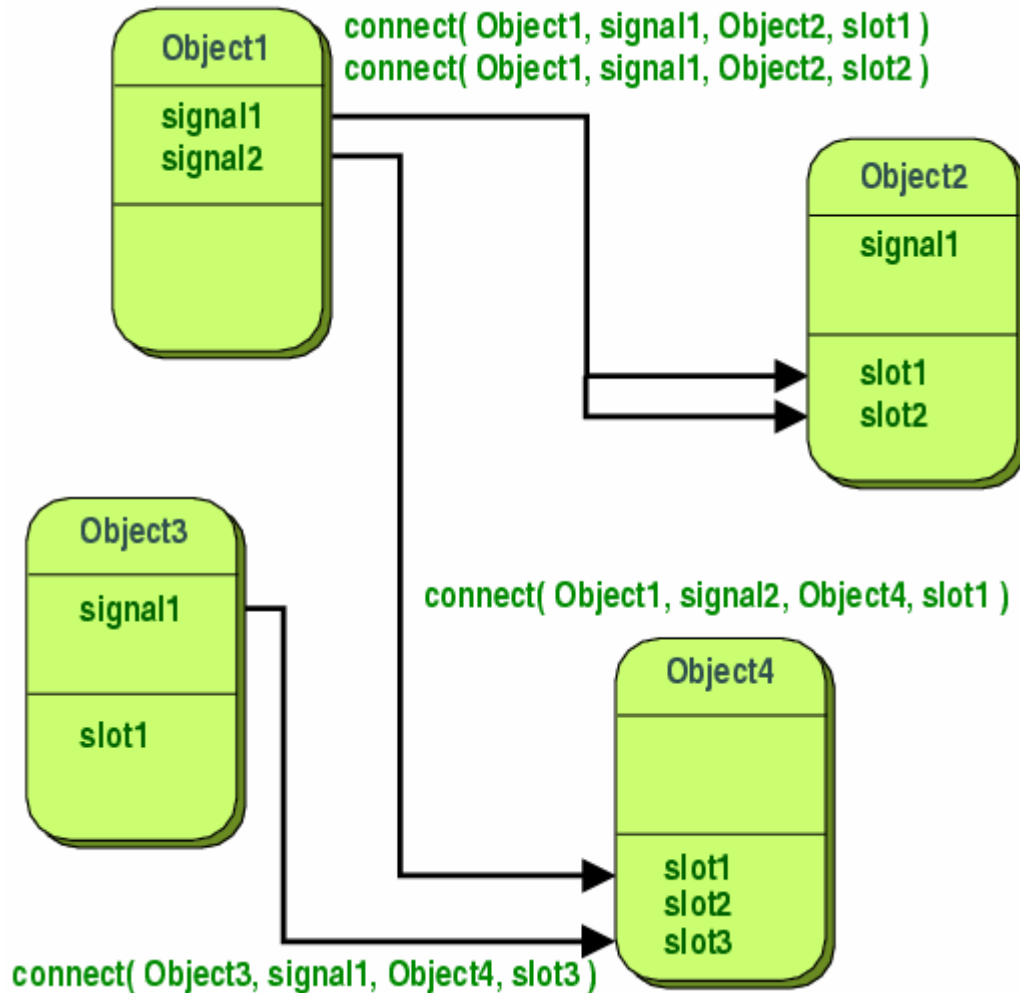
```
#include <QtGui/QApplication>
#include <QtGui/QPushButton>

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

    QPushButton hello("Hello world!");
    hello.resize(100, 30);

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

Signals and Slots



Slots and Signals

- **Signals:** Widgets erzeugen Signale wenn bestimmte Aktionen ausgeführt werden (z.B. Button -> `clicked()`)
- **Slots:** Funktionen die von *Signals* (mittels `connect()`) oder auch normal aufgerufen werden können
- **Connect:** Verbindet ein *Signal* mit einem *Slot*
- **Beispiel:**
`connect(button, SIGNAL(clicked()), QApplication, SLOT(quit()));`

Benutzerdefinierte Widgets

- Klasse muss von `QObject` oder einer Unterklasse (z.B. `QWidget`) abgeleitet sein
- Das `Q_OBJECT` macro muss in der Klassendefinition stehen
- *Slots* können in der *public slots*, *protected slots* oder *private slots* Sektion stehen
- Es muss der `moc` compiler vor dem Präprozessor verwendet werden
 - Cpp-Datei `mocen` (header-file kompilieren)
 - Entstandene Datei auch in das Projekt einfügen

MyWidget – Code Snippet

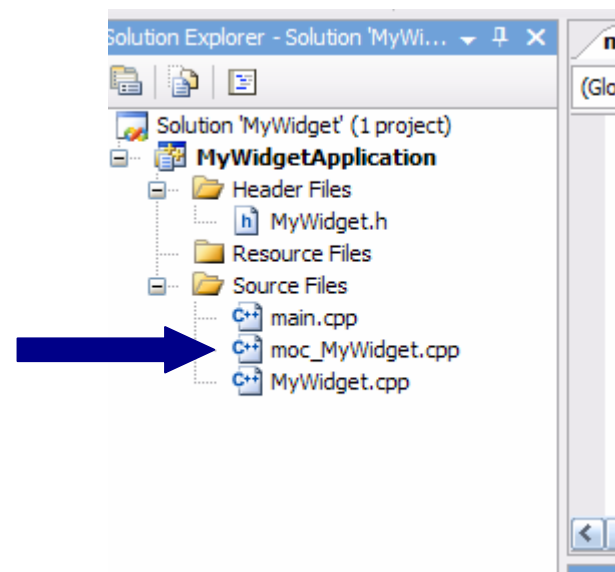
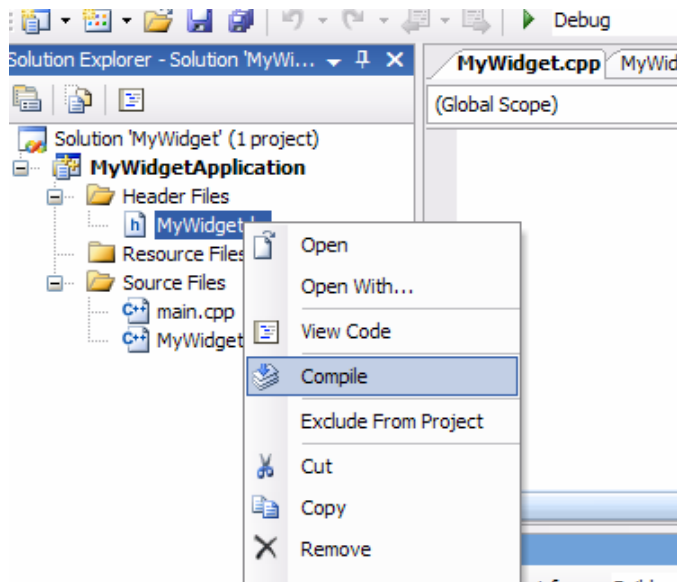
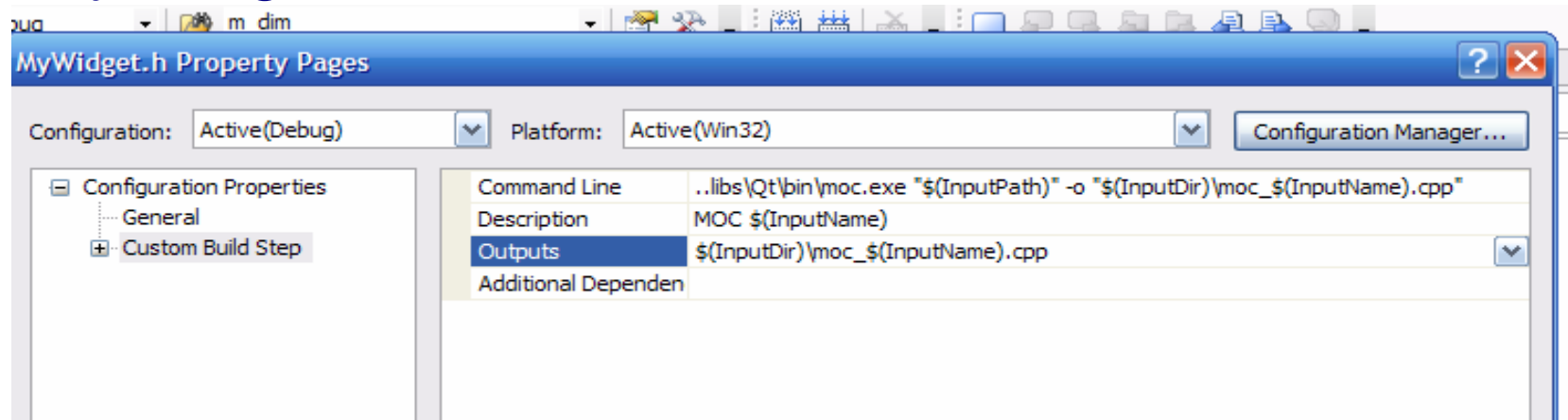
```
#include <QtGui/QWidget>

class MyWidget : public QWidget
{
    Q_OBJECT

public:
    MyWidget(QWidget* parent = 0);
    ~MyWidget();
signals:
    positionChanged();
public slots:
    showContent();
private slots():
    ...;
}
```

MyWidget.h

MyWidget - moc



Bezug zu OpenGL - QGLWidget

- ermöglicht Integration von OpenGL Code zum Zeichnen von Objekten
- implementiert Methoden:
`initializeGL()` //initialisiert OpenGL
`paintGL()` //zeichnet OpenGL Objekte in Widget
`resizeGL()` //kann Größenänderungen handhaben

Bezug zu OpenGL - QGLWidget

- weitere Methoden zu Mousefunktionen

```
mousePressEvent (MouseEvent* event) ;
```

```
mouseMoveEvent (MouseEvent* event) ;
```

```
mouseReleaseEvent (MouseEvent* event) ;
```

- weitere Methoden zu Tastaturfunktionen

```
keyPressEvent (KeyEvent* event) ;
```

```
keyReleaseEvent (KeyEvent* event) ;
```

Tutorials und Referenzen

- Reference Manual

<http://doc.trolltech.com/4.1/index.html>

- Whitepaper

<http://www.trolltech.com/pdf/whitepapers/qt40-whitepaper-a4.pdf>