

Einführung in die Programmierung

Wintersemester 2019/20

<https://ls11-www.cs.tu-dortmund.de/teaching/ep1920vorlesung>

Dr.-Ing. Horst Schirmeier

(mit Material von Prof. Dr. Günter Rudolph)

Arbeitsgruppe Eingebettete Systemsoftware (LS 12)
und Lehrstuhl für Algorithm Engineering (LS11)

Fakultät für Informatik

TU Dortmund

Inhalt

- Was ist eine GUI? Was ist Qt?
- Erste Schritte: „Hello World!“
- Signals & Slots: SpinBoxSlider
- Anwendung: Temperaturumrechnung
 - Lösung ohne GUI (Ein- und Ausgabe an Konsole)
 - Lösung mit GUI

GUI = **G**raphical **U**ser **I**nterface (grafische Benutzerschnittstelle)

Funktionalität wird durch Programm-Bibliothek bereitgestellt

- z.B. als Teil der MFC (*Microsoft Foundation Classes*)
- z.B. X-Window System, Version 11 (X11)

hier: Qt 5.12.2 („**Q**uasar toolkit“) → <https://www.qt.io/download>

(aktuell: Qt 5.14)

Warum?

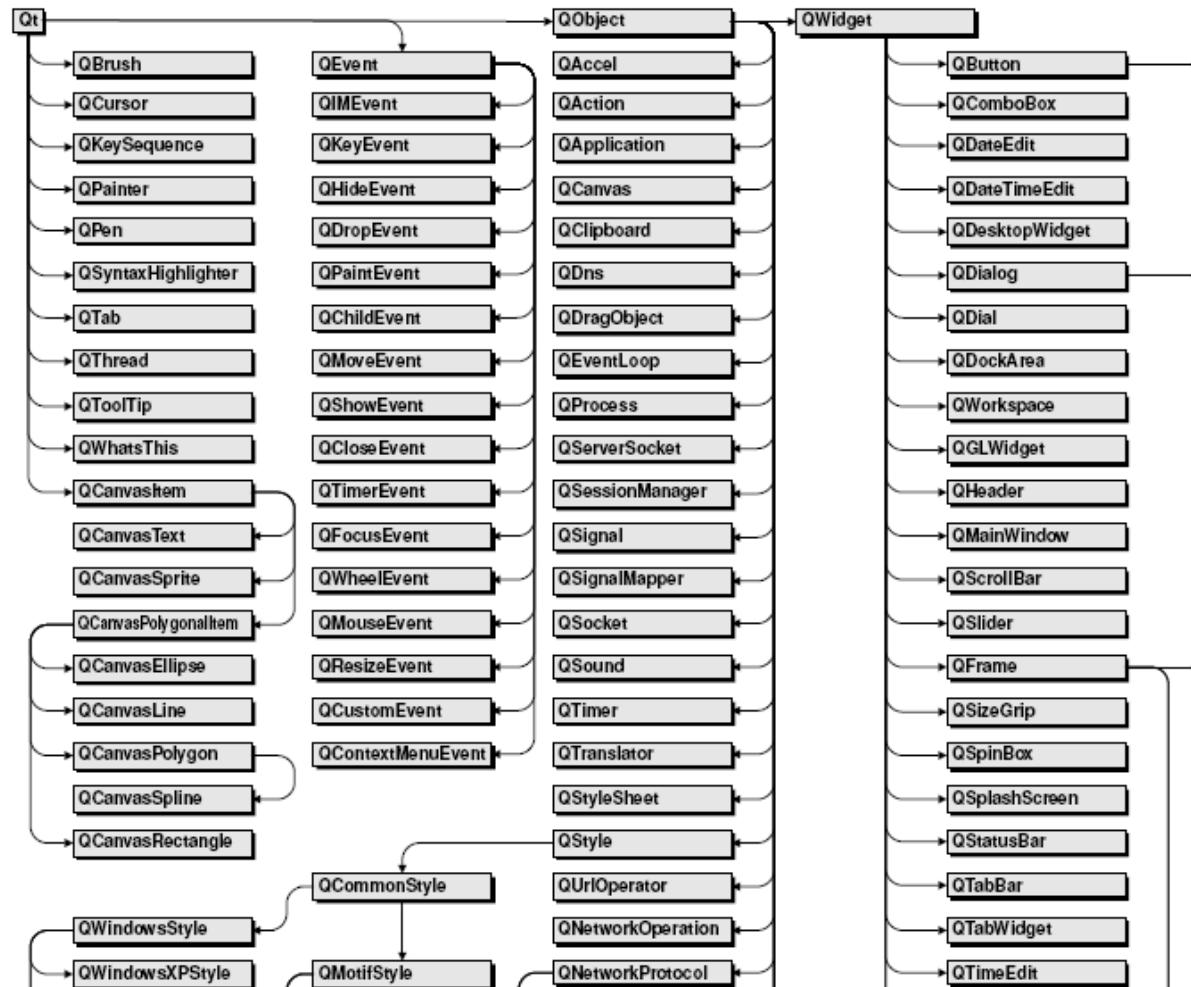
1. Plattform-unabhängig: läuft unter Linux/Unix, Windows, MacOS, Android, u.a.
2. Für nicht-kommerziellen Einsatz frei verfügbar (unter GPL), allerdings ohne Support u.a. Annehmlichkeiten

Qt (gesprochen: „Cute“)

- systemübergreifende Bibliothek
- stellt Objekte und Funktionen zur Verfügung, mit denen **unabhängig vom Betriebssystem** (Linux/Unix, Windows, MacOS, Android, u.a.) Programme erstellt werden können
- Hauptverwendungszweck:
Graphische Benutzeroberflächen (GUIs) für unterschiedliche Betriebssysteme erstellen, ohne den Code für jedes System neu zu schreiben
- Software, die auf Qt basiert: Oberfläche KDE (Linux/Mac), Google Earth, Skype, TeamViewer, Telegram Desktop, VirtualBox, Wolfram Mathematica, ...

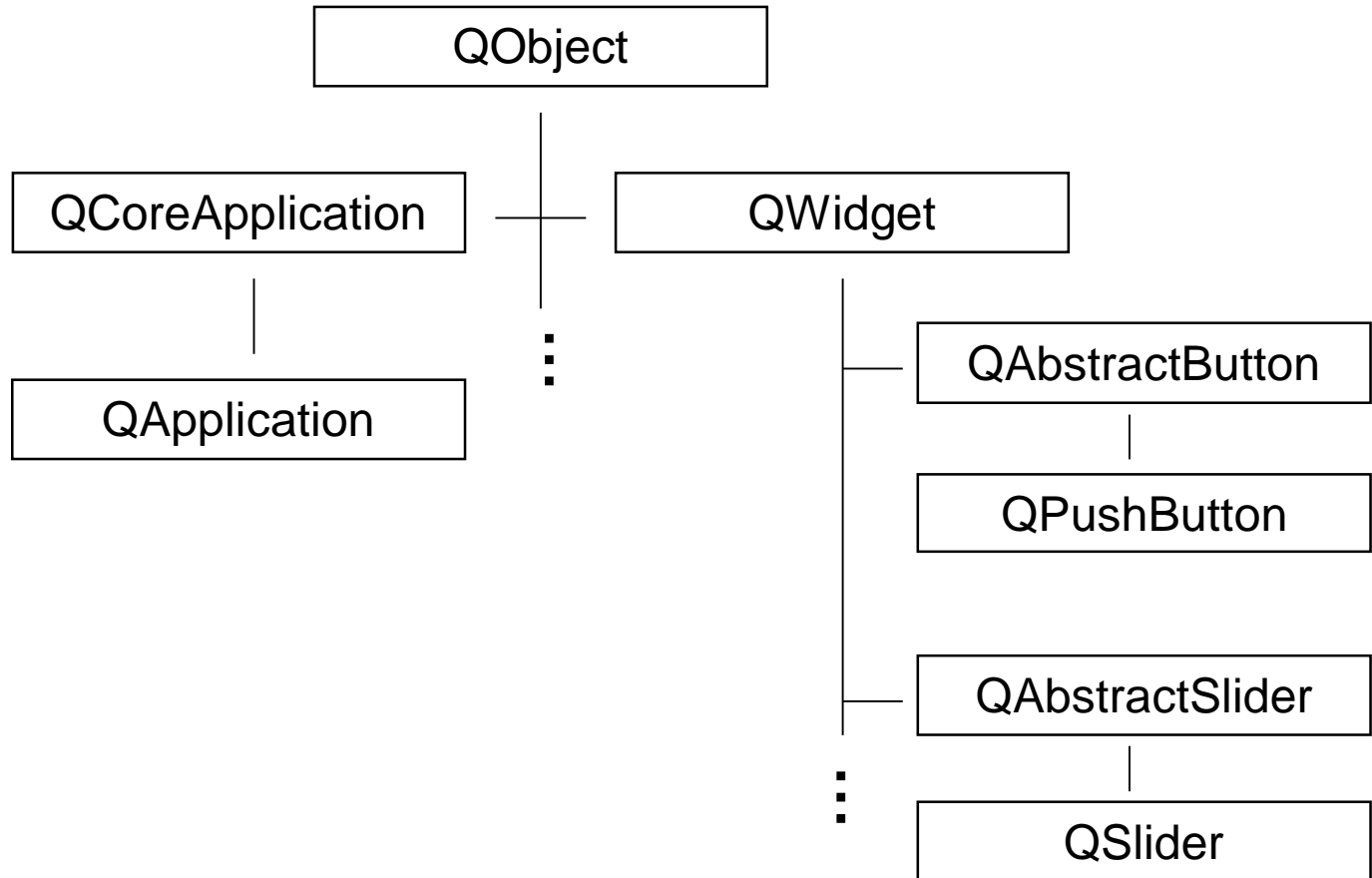
Qt-Klassen

> 1000



Liste aller Qt-C++-Klassen: <https://doc.qt.io/qt-5/classes.html>

Qt-Klassen (Ausschnitt)



Button („Schaltfläche“) mit Text „Hello World!“

```

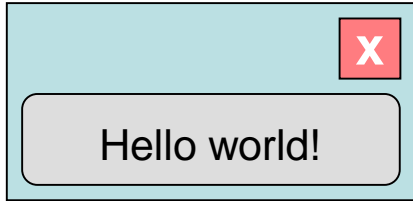
#include <QApplication>
#include <QPushButton>

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

    QPushButton hello("Hello world!", 0);

    hello.show();

    return app.exec();
}
    
```



Jedes Programm hat **genau eine** Instanz von `QApplication`

Erzeuge Button, 0=kein Elternfenster

Button darstellen!

Kontrolle an `QApplication` übergeben



Button („Schaltfläche“) mit Text „Hello World!“

- Was geschieht, wenn Button gedrückt wird? → Anscheinend nichts!
- **Tatsächlich:** Klasse `QPushButton` bemerkt die Aktion, wurde aber nicht instruiert, was sie dann machen soll!
- **Möglich:** eine Aktion in einem Objekt einer anderen Klasse auslösen

Klasse `QObject`

```
static bool connect (  
    const QObject *sender,           // Wer sendet?  
    const char *signal,             // Bei welcher Aktion?  
    const QObject *receiver,        // Wer empfängt?  
    const char *member,             // Welche Aktion ausführen?  
    Qt::ConnectionType type = Qt::AutoCompatConnection  
);
```


Button („Schaltfläche“) mit Text „Hello World!“, Programmende sobald gedrückt

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

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

    QPushButton hello("Hello world!", 0);
    QObject::connect(&hello, SIGNAL(clicked()),
                    &app, SLOT(quit()));
};
hello.show();

return app.exec();
}
```

Wenn **hello** angeklickt wird, dann soll in **app** die Methode **quit** ausgeführt werden.

Signals and Slots

Qt-spezifisch!

- Bereitstellung von Inter-Objekt-Kommunikation
- **Idee:** Objekte, die nichts voneinander wissen, können miteinander **verbunden** werden
- Jede von QObject abgeleitete Klasse kann **Signals** deklarieren, die von Funktionen der Klasse ausgestoßen werden.
- Jede von QObject abgeleitete Klasse kann **Slots** definieren. Slots sind Funktionen, die mit Signals assoziiert werden können.
- Technische Umsetzung: Makro Q_OBJECT in Klassendeklaration
- Signals und Slots von Objektinstanzen können **miteinander verbunden** werden:

Signal S von Objekt A verbunden mit Slot T von Objekt B →
Wenn A Signal S auslöst, so wird Slot T von B ausgeführt.

Signals and Slots

Qt-spezifisch!

- Ein **Signal** kann mit mehreren **Slots** verbunden werden.
→ Ein Ereignis löst mehrere Aktionen aus.
- Ein **Slot** kann mit mehreren **Signals** verbunden werden.
→ Verschiedene Ereignisse können gleiche Aktion auslösen.
- **Signals** können auch **Parameter** an die **Slots** übergeben.
→ Parametrisierte Aktionen.
- **Signals** können mit **Signals** verbunden werden.
→ Weitergabe / Übersetzung von Signalen.

Button als Teil eines Fensters, in dem Widgets vertikal angeordnet werden

```
#include <QApplication>
#include <QWidget>
#include <QVBoxLayout>
#include <QPushButton>

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

    QVBoxLayout layout;
    window.setLayout(&layout);

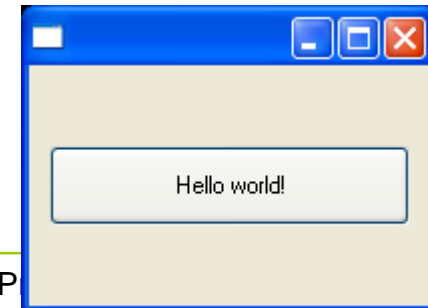
    QPushButton hello("Hello world!", &window);
    QObject::connect(&hello, SIGNAL(clicked()),
                   &app, SLOT(quit()));

    layout.addWidget(&hello);

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

QVBoxLayout:
vertikales Anordnen
von Widgets
innerhalb **window**

hello
gehört zu **window**
und wird von
layout positioniert



Button und Label als Teile eines Fensters

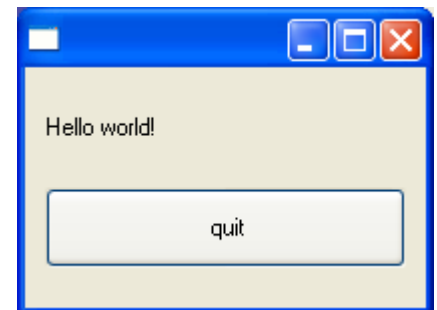
```
#include <QApplication>
#include <QWidget>
#include <QVBoxLayout>
#include <QLabel>
#include <QPushButton>
int main(int argc, char *argv[]) {
    QApplication app(argc, argv);
    QWidget window;
    QVBoxLayout layout;
    window.setLayout(&layout);

    QLabel hello("Hello world!", &window);
    layout.addWidget(&hello);
    QPushButton quit("Quit", &window);
    QObject::connect(&quit, SIGNAL(clicked()),
                   &app,      SLOT(quit()));

};
layout.addWidget(&quit);

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

QLabel zum
Beschriften des
Fensterinneren



Slider verbunden mit SpinBox

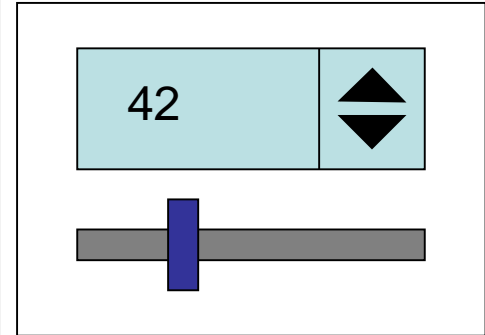
```
#include <QApplication>
#include <QWidget>
#include <QVBoxLayout>
#include <QSpinBox>
#include <QSlider>

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

    QVBoxLayout layout;
    window.setLayout(&layout);

    QSpinBox spinBox(&window);
    spinBox.setRange(0, 130);
    layout.addWidget(&spinBox);

    QSlider slider(Qt::Horizontal, &window);
    slider.setRange(0, 130);
    layout.addWidget(&slider);
}
```



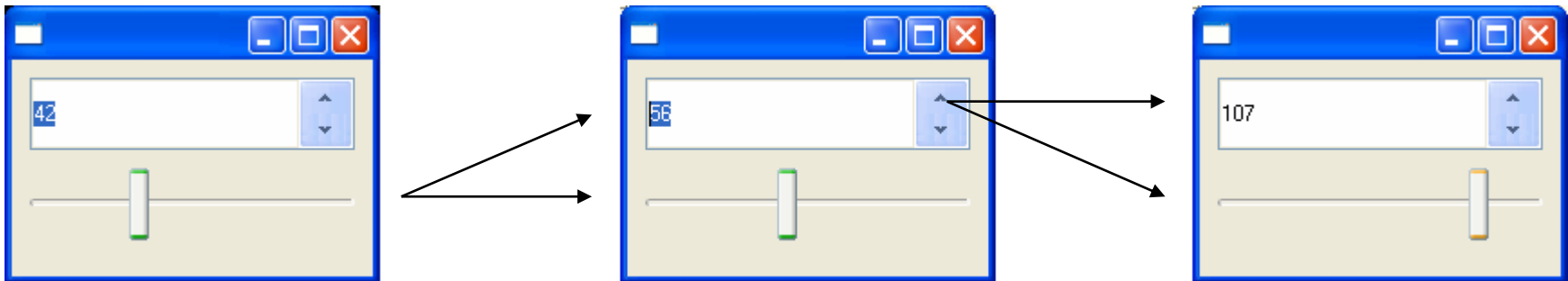
Gewünschtes Verhalten:

SpinBox wirkt auf Slider und umgekehrt.

Fortsetzung nächste Folie ...

Slider verbunden mit **SpinBox***Fortsetzung*

```
QObject::connect (&spinBox, SIGNAL (valueChanged (int)) ,  
                 &slider,   SLOT (setValue (int)) );  
QObject::connect (&slider,  SIGNAL (valueChanged (int)) ,  
                 &spinBox,  SLOT (setValue (int)) );  
spinBox.setValue (42) ;  
  
window.show () ;  
return app.exec () ;  
}
```



Anwendung: Temperaturumrechnung

$$x [^{\circ}C] = \frac{9}{5} x + 32 [^{\circ}F]$$
$$y [^{\circ}F] = \frac{5}{9} (y - 32) [^{\circ}C]$$

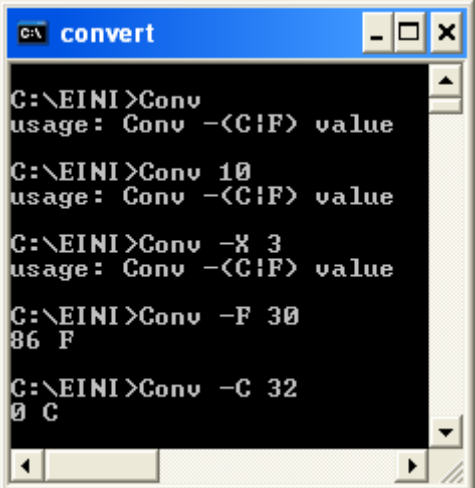
Lösung ohne GUI:

1. Einlesen einer Zahl
2. Angabe der Konvertierungsrichtung
3. Ausgabe

Lösung ohne GUI

```
#include <iostream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[]) {
    if (argc != 3 || strlen(argv[1]) != 2 || argv[1][0] != '-'
        || (argv[1][1] != 'C' && argv[1][1] != 'F')) {
        cerr << "usage: " << argv[0] << " -(C|F) value\n";
        exit(1);
    }
    double val = atof(argv[2]);
    if (argv[1][1] == 'C')
        val = 5 * (val - 32) / 9;
    else
        val = 9 * val / 5 + 32;
    cout << val << " " << argv[1][1] << endl;
}
```



```
C:\EINI>Conv
usage: Conv -(C|F) value

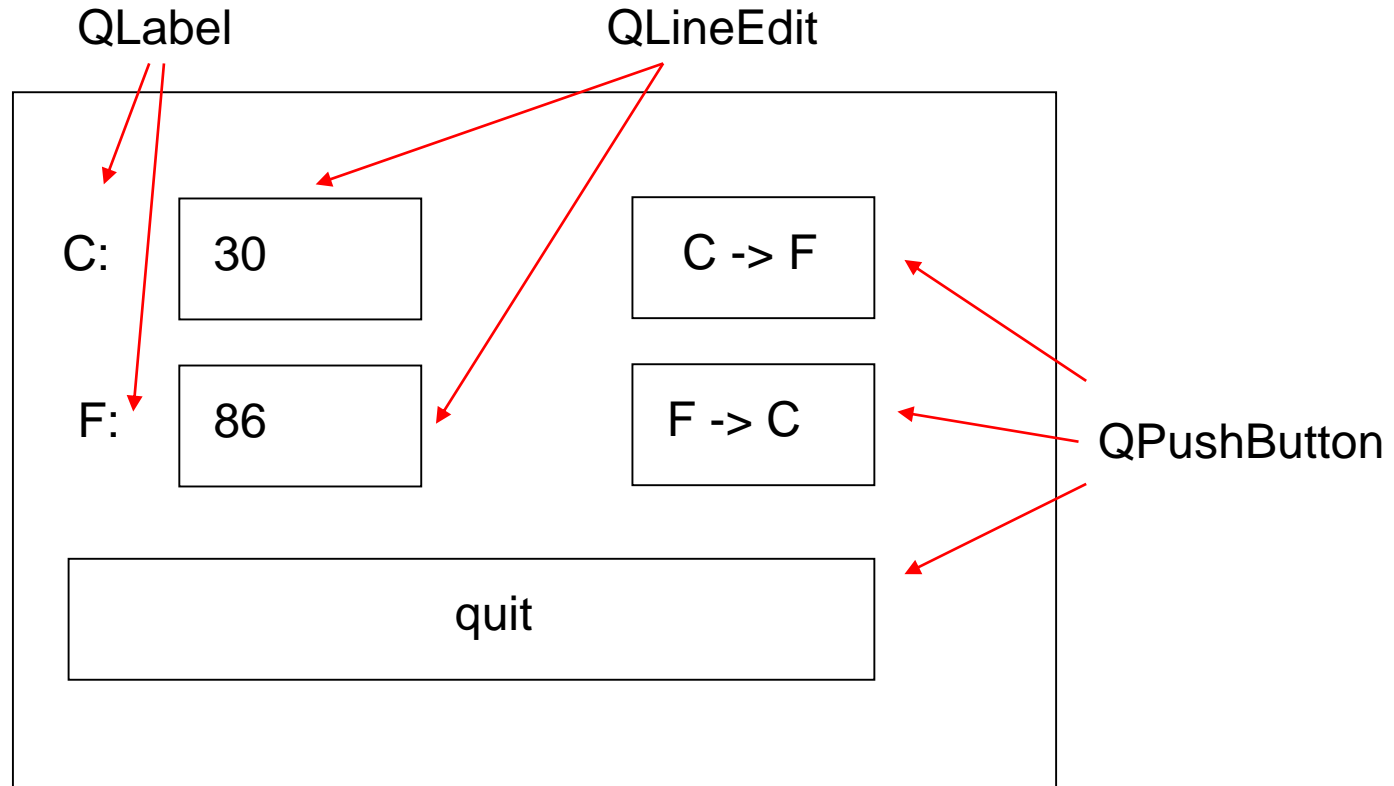
C:\EINI>Conv 10
usage: Conv -(C|F) value

C:\EINI>Conv -X 3
usage: Conv -(C|F) value

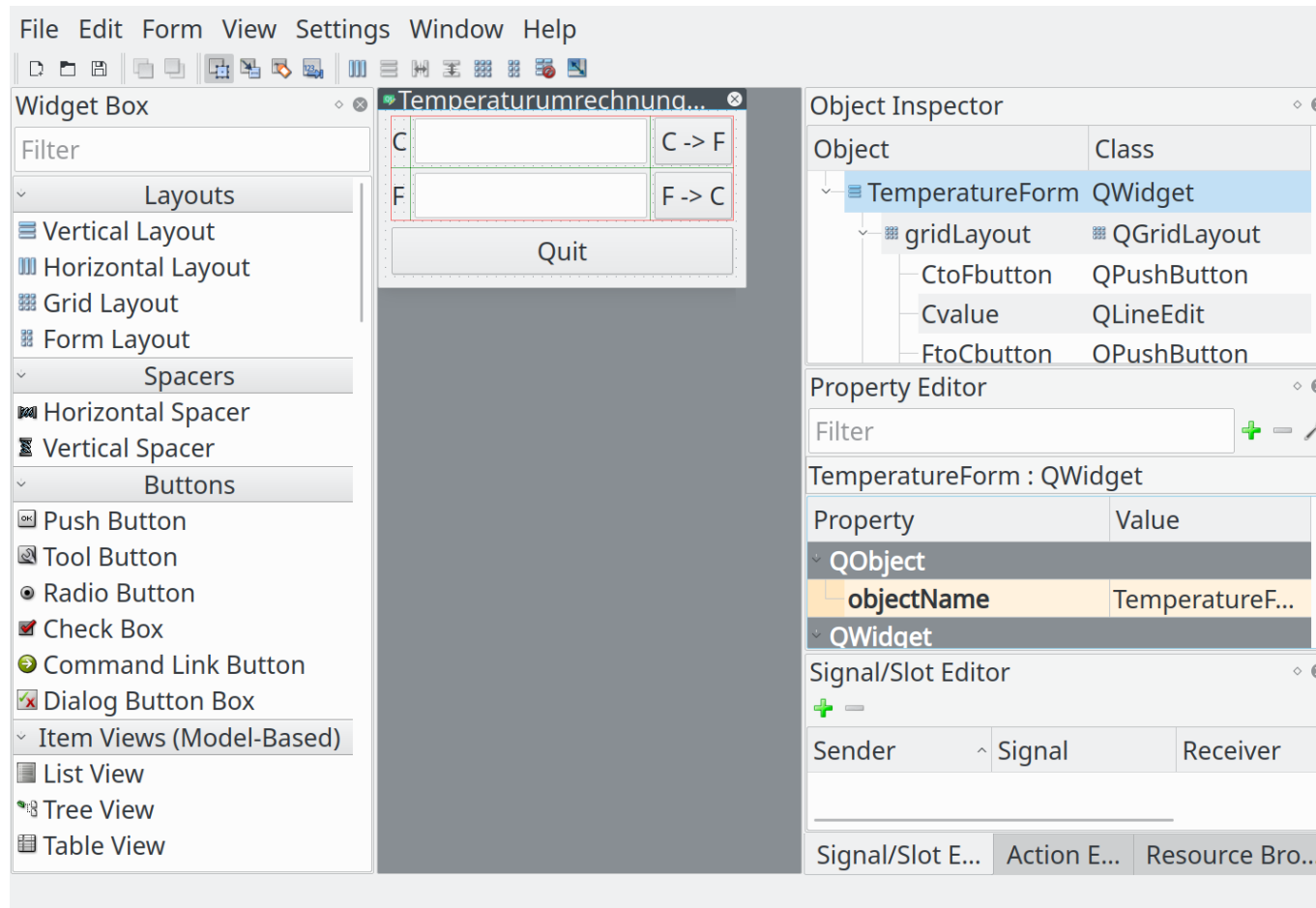
C:\EINI>Conv -F 30
86 F

C:\EINI>Conv -C 32
0 C
```

Lösung mit GUI

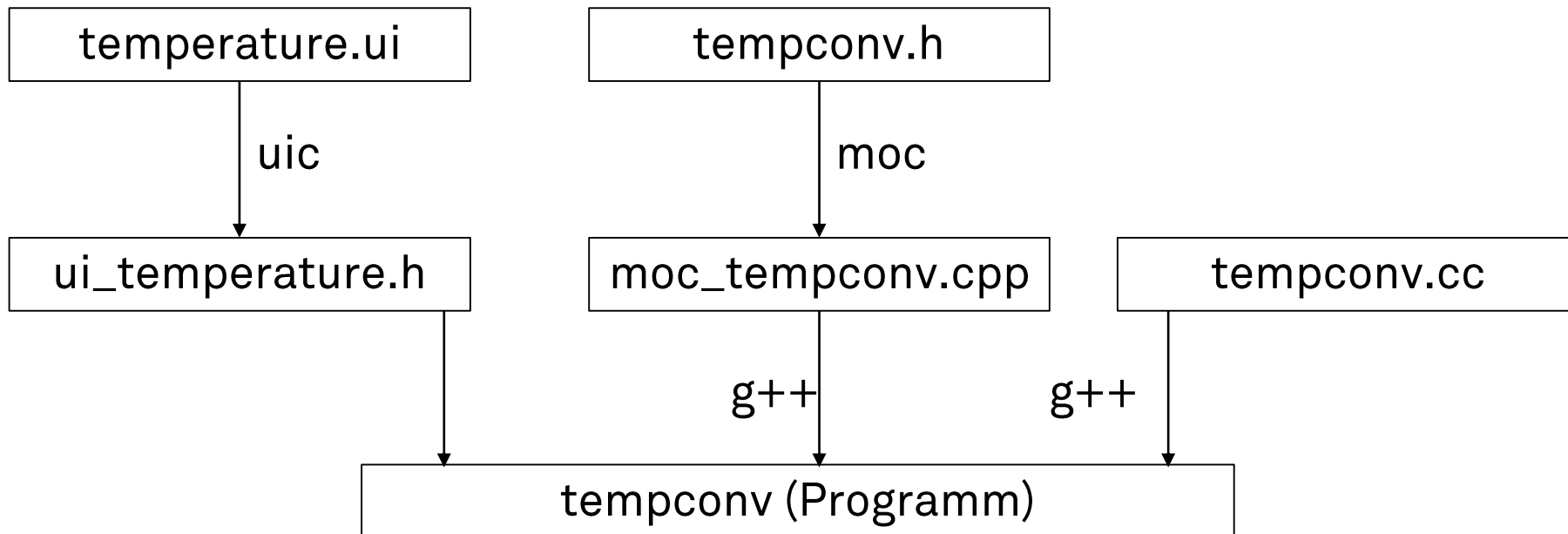


UI-Erstellung mit Qt Designer



temperature.ui

```
# tempconv.pro (qmake project file)
TEMPLATE += app
QT += widgets
FORMS += temperature.ui
HEADERS += tempconv.h
SOURCES += tempconv.cc
```



```
#include <QLineEdit>
#include "ui_temperature.h"

class TempConv : public QWidget {
    Q_OBJECT
public:
    explicit TempConv(QWidget *parent
                      = nullptr);

private slots:
    void on_CtoFbutton_clicked();
    void on_FtoCbutton_clicked();
    void on_quitButton_clicked();

private:
    Ui::TemperatureForm ui;
    QLineEdit *cvalue, *fvalue;
};
```

Was ist das?

Spracherweiterung?

Erst Aufruf von **moc**
(*meta object compiler*),
der generiert zusätzlichen
C++ Code, dann Aufruf des
C++ Compilers!

```
#include <QWidget>
#include <QMessageBox>
#include <iostream>
#include "tempconv.h"

TempConv::TempConv(QWidget *parent) : QWidget(parent)
{
    ui.setupUi(this);
    cvalue = findChild<QLineEdit *>("Cvalue");
    fvalue = findChild<QLineEdit *>("Fvalue");
}
```

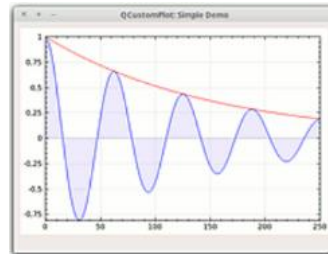
```
void TempConv::on_CtoFbutton_clicked() {
    QString s = cvalue->text();
    bool ok;
    double val = s.toDouble(&ok);
    if (ok) {
        val = 9.0 / 5.0 * val + 32;
        fvalue->setText(QString("%1").arg(val, 0, 'f', 1));
    } else {
        QMessageBox::information(this, "invalid input",
            "please enter numbers");
    }
}

void TempConv::on_FtoCbutton_clicked() {
    /* analog mit fvalue / cvalue und
       der umgekehrten Berechnung */
}
```

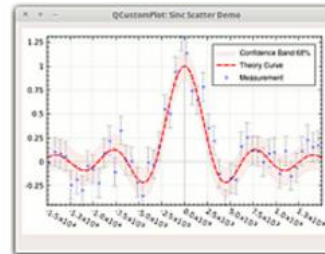
```
void TempConv::on_quitButton_clicked()  
{  
    QApplication::quit();  
}  
  
int main(int argc, char **argv)  
{  
    QApplication app(argc, argv);  
    TempConv tv;  
    tv.show();  
    return app.exec();  
}
```


Qt-Widgets für wissenschaftliche Anwendungen: QCustomPlot

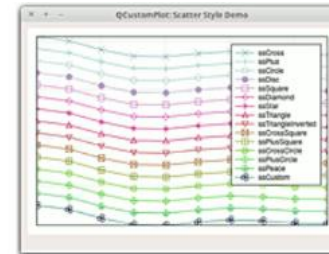
- qcustomplot.com



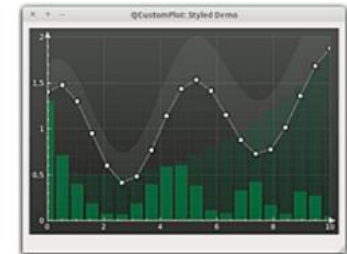
A simple decaying sine function with fill and its exponential envelope in red



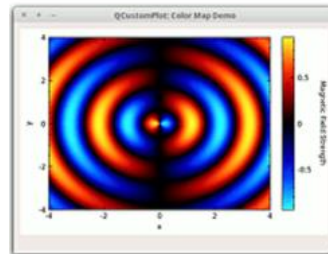
sinc function with data points, corresponding error bars and a 2-sigma confidence band



A demonstration of several scatter point styles



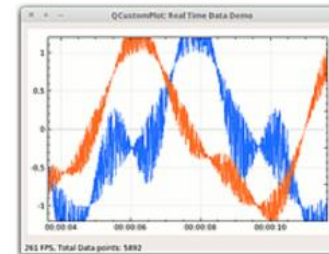
Demonstrating QCustomPlot's versatility in styling the plot



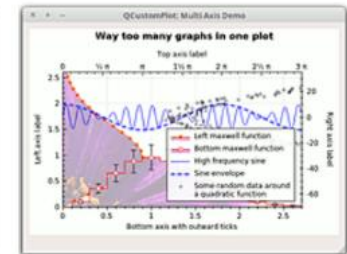
A 2D color map with color scale. Color scales can be dragged and zoomed just like axes



Pixmap scatter points and a multi-lined axis label, as well as a plot title at the top



Real time generated data and time bottom axis



Multiple plot styles with different key/value axes and pi tick labeling at top axis