Hydroinformatik II:

Einführung in Qt mit Übungen

V4

[1]Helmholtz Centre for Environmental Research – UFZ, Leipzig
[2]Technische Universität Dresden – TUD, Dresden

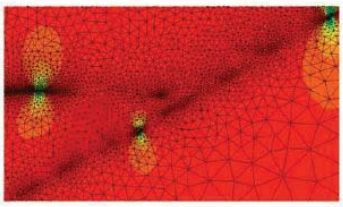
Dresden, 02. November 2012

Vorlesungsplan Hydroinformatik II WS2012/2013

#	Datum	Thema
1	12.10.2012	Einführung, Grundlagen: Kontinuumsmechanik
2	19.10.2012	Grundlagen: Hausaufgabe: Kontinuumsmechanik
3	26.10.2012	Grundlagen: Hydromechanik
4	02.11.2012	Einführung: Qt
5	09.11.2012	Grundlagen: Numerische Methoden
6	16.11.2012	Grundlagen: PDEs, Analytische Lösungen
7	30.11.2012	Numerik: Finite Differenzen Methode
8	23.11.2012	Grundlagen: Diffusionsprozesse
9	30.11.2012	Numerik: Explizite FDM
10	14.12.2012	Numerik: Implizite FDM
11	21.12.2012	Christmas Lecture
12	11.01.2013	Gerinnehydraulik: Theorie
13	18.01.2013	Gerinnehydraulik: Übung (QAD)
14	25.01.2013	Gerinnehydraulik: Programmierung (OOP)
15	01.02.2013	Gerinnehydraulik: Systemanalyse
16	08.02.2013	Ausblick "Hydrosystemanalyse" / Klausurvorbereitung
17	XX.02.2012	Klausur

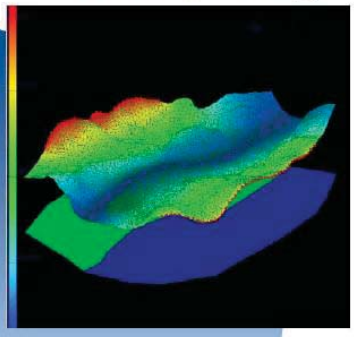
Konzept

$$\frac{d\psi}{dt} = \frac{\partial\psi}{\partial t} + \mathbf{v}^E \nabla\psi$$

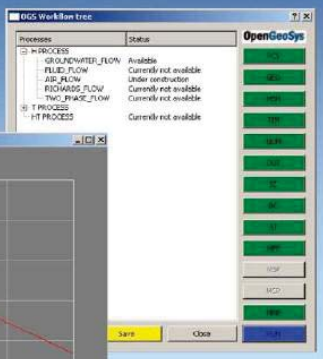


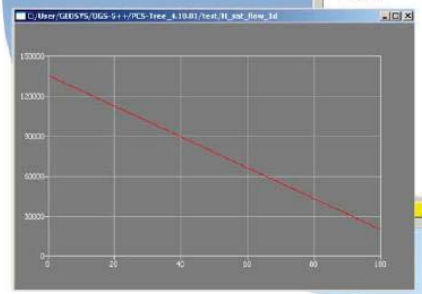
Basics
Mechanik

Anwendung



Numerische
Methoden





Programmierung
Visual C++

Prozessverständnis

Der Fahrplan für heute

- Siehe Skript Kapitel 5
- Die Qt Installation
- Die `main()` Funktion
- Das Qt Projekt
- Wir plotten mit `QPlotter` [E1]
- Was ist eine Schnittstelle ?
- wir basteln unser MatLab ... [E2]
- wir rechnen eine stationäre Diffusion (Abschn. 1.5.3)

Der Fahrplan für heute

- Siehe Skript Kapitel 5
- Die Qt Installation
- Die main() Funktion
- Das Qt Pr
- Wir plotte
- Was ist ei
- wir bastel
- wir rechne 5.3)

Kapitel 5

Qt

Die Geschichte: Bei der ersten HI-I Vorlesung gabe es ziemlich peinliche eine Panne. Die (mit viel Mühe vorbereiteten Übungen) liefen zwar mit meiner MSVC++ (professional) Version aber (nicht ohne Weiteres) mit freien MSVC++ Express Version, wo die MFC standartmäßig nicht mehr dabei ist. Dies zeigt uns aber, auch beim Programmieren niemals nur auf ein Pferd setzen. Wichtig ist, unser eigener Code muss C++ Standard sein, den können wir dann problemlos in verschiedene GUI Frameworks, wie MSVC++, Qt etc. einbauen. Warum jetzt noch Qt ? Wir werden in den Übungen sehen, dass Qt sehr dicht an C++ Standards dran ist und Qt ist a cross-platform GUI. Qt läuft auf Windows, Linux, Mac ... In der Anlage 5.1 finden sie eine Anleitung zur Installation von Qt.

Am Ende des letzten Semesters haben wir uns bereits ein bisschen mit Qt beschäftigt [9]. Nun wollen wir das Visual C++ für unsere numerischen Anwendungen benutzen.

5.1 Qt Projekt

Es ist ihnen hoffentlich gelungen, Qt für ihr Betriebssystem erfolgreich zu installieren. Eine kurze Installationsanleitung war im Skript [9] (Abschn. 12.8) zu finden. Hier noch mal die Web-Seite für den Download `http://qt.nokia.com/products`. Der große Vorteil von Qt ist die *Plattformunabhängigkeit* sowie die freie Verfügbarkeit unter der GPL (Gnu Public License).

Wir starten Qt mit einem Doppelklick auf das Desktop-Symbol (Abb. 5.1).

Wenn alles gut geht gegangen ist sollten sie im Qt User Interface (UI) landen (Abb. 5.2).

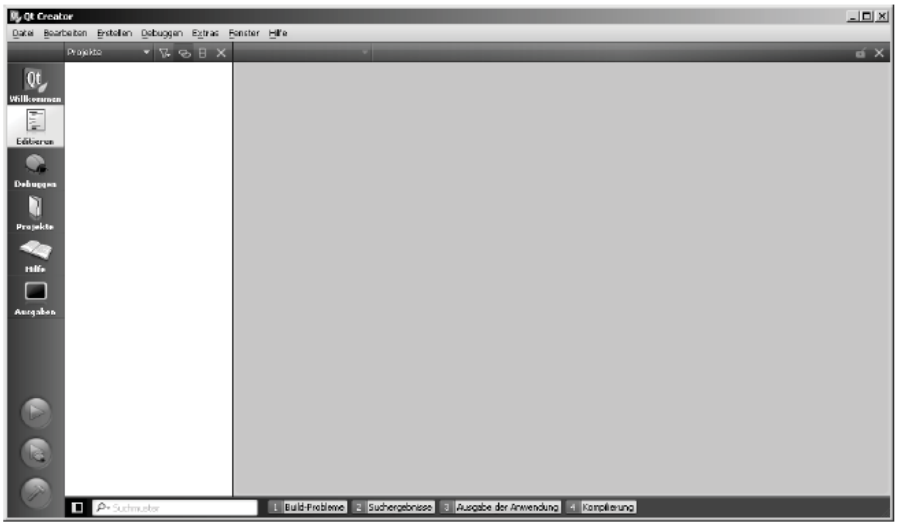


Abbildung 5.2: Qt Creator

Der Fahrplan für heute

- Siehe Skript Kapitel 5
- Die Qt Installation
- Die `main()` Funktion
- Das Qt Projekt
- Wir plotten mit `QPlotter` [E1]
- Was ist eine Schnittstelle ?
- wir basteln unser MatLab ... [E2]
- wir rechnen eine stationäre Diffusion (Abschn. 1.5.3)

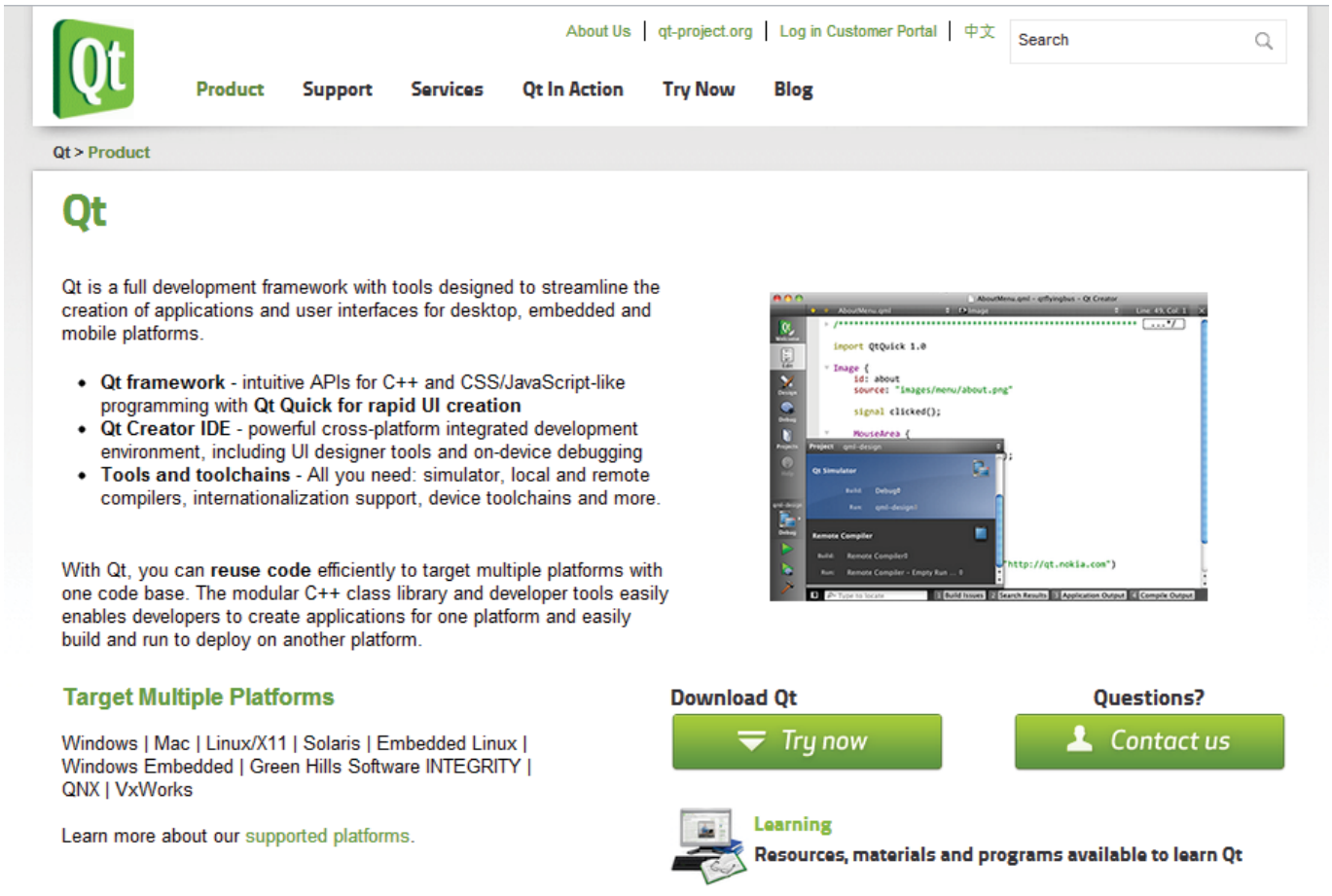
Qt Installation

Es ist ihnen hoffentlich gelungen, Qt für ihr Betriebssystem erfolgreich zu installieren. Eine kurze Installationsanleitung war im Skript Hydroinformatik-I (Abschn. 12.8) zu finden. Hier noch mal die Web-Seite für den Download `http://qt.nokia.com/products`. Der große Vorteil von Qt ist die *Plattformunabhängigkeit* sowie die freie Verfügbarkeit unter der GPL (Gnu Public License).
Wir starten Qt mit einem Doppelklick auf das Desktop-Symbol.

Abbildung: Qt Start

Wenn alles gut geht gegangen ist sollten sie im Qt User Interface (UI) landen.

http://qt.digia.com/product/



http://qt.digia.com/product/

About Us | qt-project.org | Log in Customer Portal | 中文 Search

Product Support Services Qt In Action Try Now Blog

Qt > Product

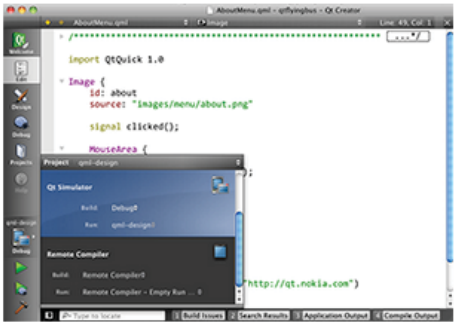
Qt

Qt: C++-Klassenbibliothek für die plattformübergreifende Programmierung grafischer Benutzeroberflächen.

Qt is a full development framework with tools designed to streamline the creation of applications and user interfaces for desktop, embedded and mobile platforms.

- **Qt framework** - intuitive APIs for C++ and CSS/JavaScript-like programming with **Qt Quick for rapid UI creation**
- **Qt Creator IDE** - powerful cross-platform integrated development environment, including UI designer tools and on-device debugging
- **Tools and toolchains** - All you need: simulator, local and remote compilers, internationalization support, device toolchains and more.

With Qt, you can **reuse code** efficiently to target multiple platforms with one code base. The modular C++ class library and developer tools easily enables developers to create applications for one platform and easily build and run to deploy on another platform.



Target Multiple Platforms

Windows | Mac | Linux/X11 | Solaris | Embedded Linux | Windows Embedded | Green Hills Software INTEGRITY | QNX | VxWorks

Learn more about our supported platforms.

Download Qt — Try now

Questions? — Contact us

Learning
Resources, materials and programs available to learn Qt

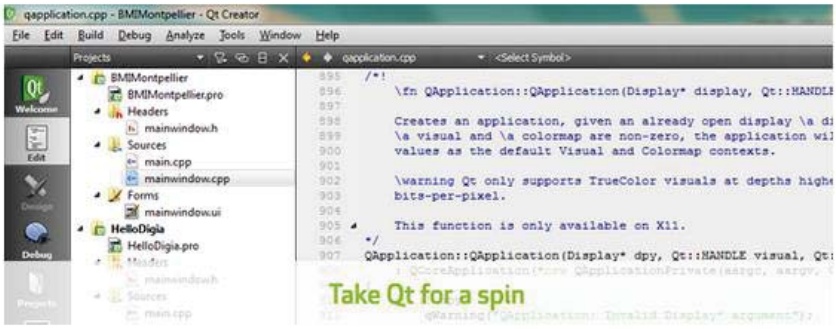
http://qt.digia.com/Try-Qt-Now/

About Us | qt-project.org | Log in Customer Portal | 中文 Search

Product Support Services Qt In Action Try Now Blog

Qt > Try Now

Try Now



Ready to purchase?

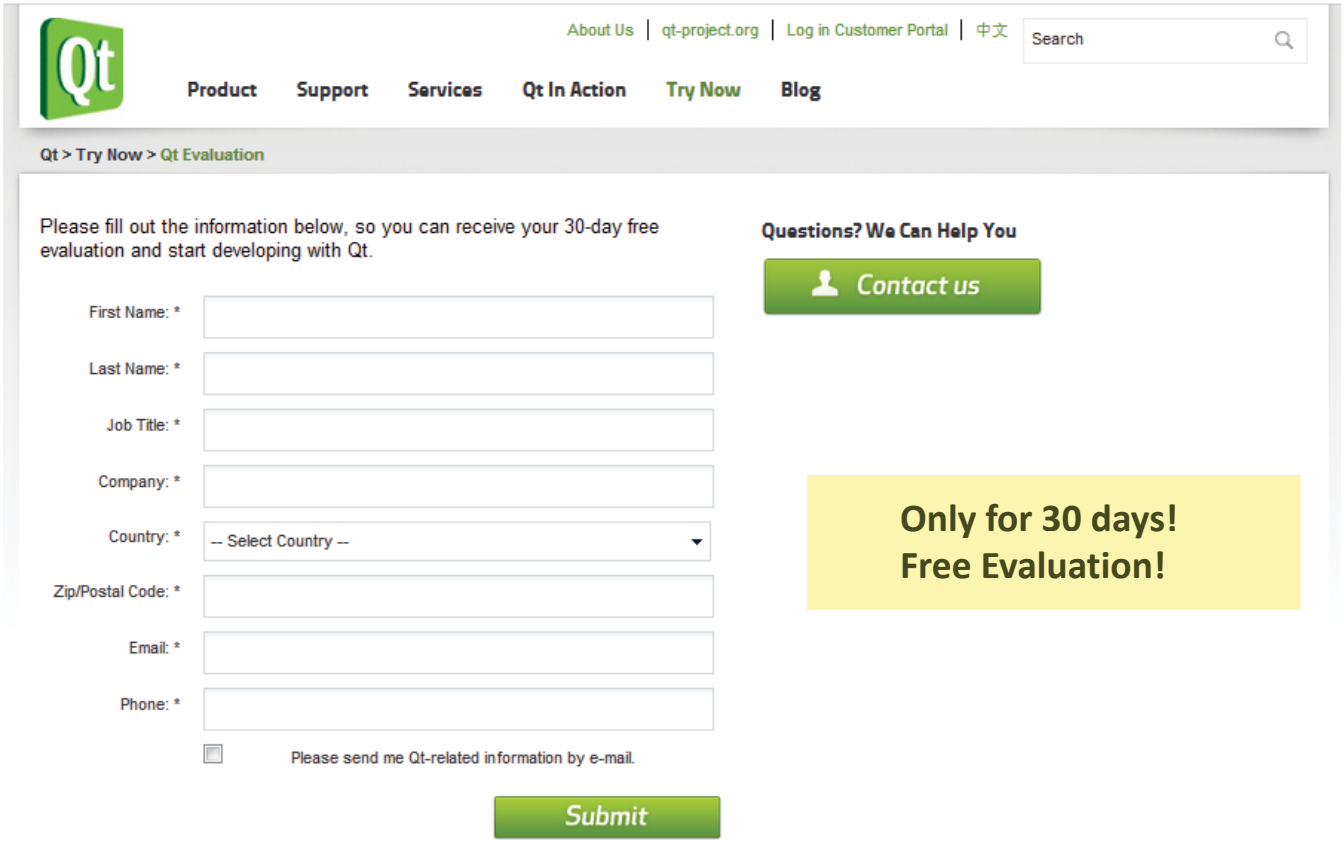
Contact us

Choose your Qt Evaluation Platform

The current evaluation version is Qt 4.8.3 released on September 13, 2012.

Windows | Embedded Linux | Embedded Linux | Mac OS X | Windows Embedded | Windows Embedded | Linux/X11

Please note: Your download e-mail will provide a link to the appropriate SDK installer as well as the stand-alone library installers.



Qt Installation

http://qt-project.org/downloads

~~http://qt.nokia.com/downloads/~~

Download Qt, the cross-platform application framework

Jump to: Qt libraries ↓ Qt Creator ↓ Visual Studio Add-in ↓

The Qt SDK includes the tools you need to build desktop, embedded and mobile applications with Qt from a single install. This is the recommended way to get started with Qt. The latest SDK has a self updater feature that will keep you up to date as new versions are made available.

The **Qt SDK version 1.1.4** released on November 8th contains:

- Qt libraries version 4.7.4
- Qt Creator IDE version 2.3
- Qt development tools
- Symbian and MeeGo simulators
- Qt Mobility version 1.2
- Remote compilers

Windows*:
Online installer - 15 MB | Offline installer - 1.6 GB

Linux/X11 32-bit:
Online installer - 23 MB | Offline installer - 1.3 GB

Linux/X11 64-bit:
Online installer - 23 MB | Offline installer - 1.3 GB

Mac OS X 10.6 or later 64-bit:
Online installer - 12 MB | Offline installer - 1.3 GB

Qt Installation

http://qt-project.org/downloads

~~http://qt.nokia.com/downloads/~~

License

Qt is available under GPL v3, LGPL v2 and a commercial license.
Learn more about licenses here.

Developing with Qt professionally for non-mobile targets?
Qt Commercial offers varied licensing terms compared to the LGPL, plus it ensures that someone is there for you if you need help. Please contact Digia for details.

Evaluate Qt under a commercial license (qt.digia.com)

Notes

Be sure to check if Qt and Qt Creator are supported on your platform, then head over to Getting Started in the SDK documentation.

SDK Requirements
64-bit Apple Mac OS X 10.6 or later with xCode
32- or 64-bit Ubuntu Linux 8.04 or later
32- or 64-bit Microsoft Windows XP Service Pack 2, Windows Vista, or Windows 7

Qt Installation

http://qt-project.org/downloads

The Qt SDK includes the tools you need to build desktop, embedded and mobile applications with Qt from a single install. This is the recommended way to get started with Qt. The latest SDK has a self updater feature that will keep you up to date as new versions are made available.

The **Qt SDK version 1.2.1** released on April 11th contains:

- Qt libraries version 4.8.1
- Simulator for Symbian phones and the Nokia N9
- Qt Creator IDE version 2.4.1
- Qt Mobility version 1.2
- Qt development tools
- Remote compilers

INSTALLATION TEST

My download choice:
Offline installer 1.7GB
(WIN XP)
15:38 Start downloading. . . . .
. . . . . 15:47 done!

Windows:
Online installer - 15 MB | Offline installer - 1.7 GB

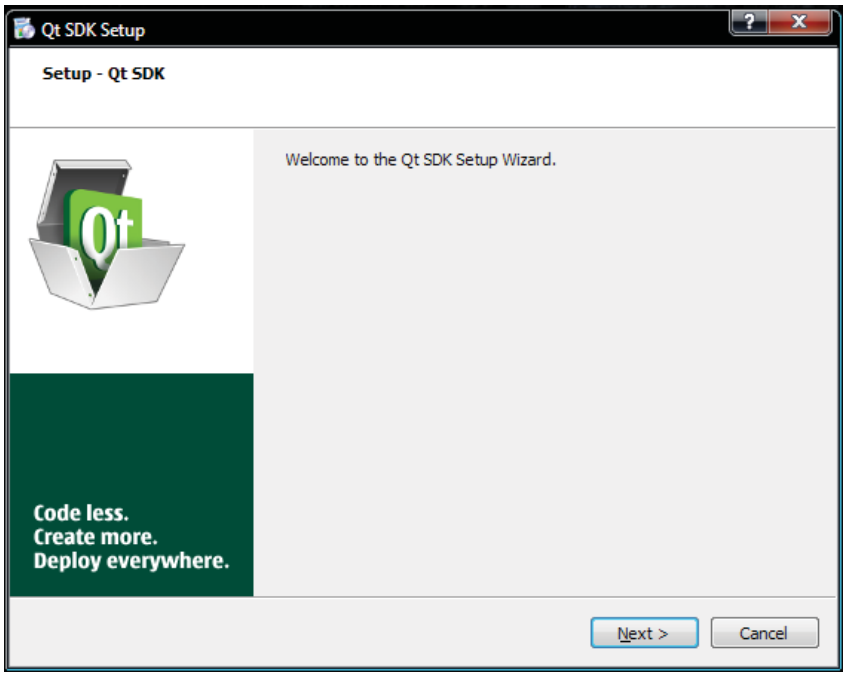
Linux/X11 32-bit:
Online installer - 26 MB | Offline installer - 764 MB

Linux/X11 64-bit:
Online installer - 26 MB | Offline installer - 767 MB

Mac OS X 10.6 or later 64-bit:
Online installer - 12 MB | Offline installer - 760 MB

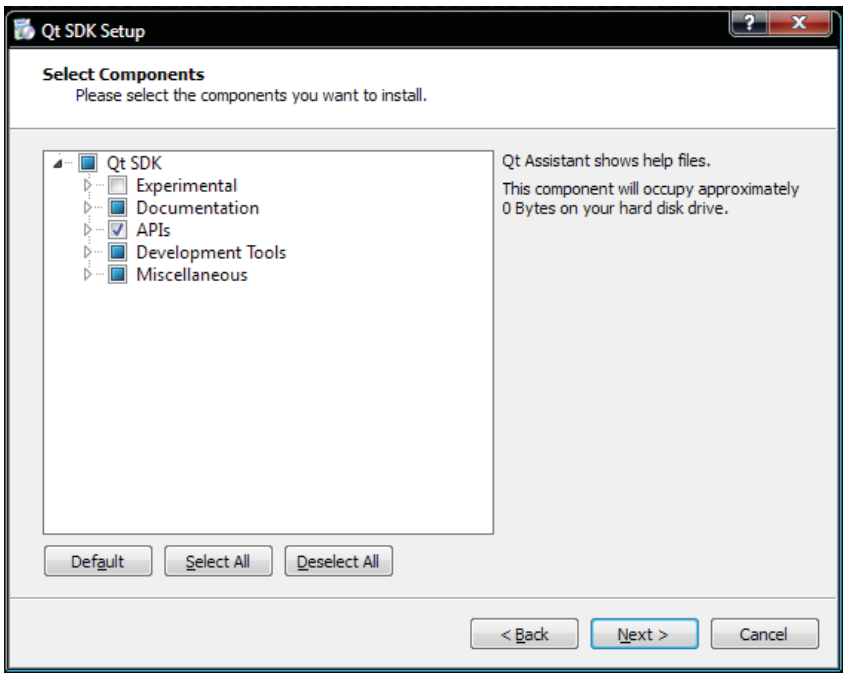
Checksums can be found on Nokia Developer

QT Installation: 29.10.2012

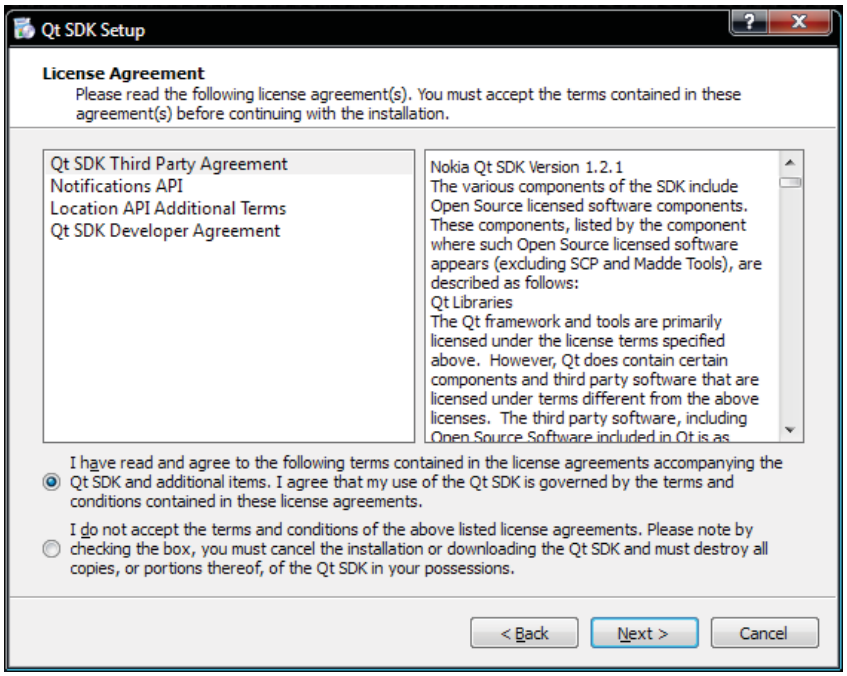


. . . .00:00

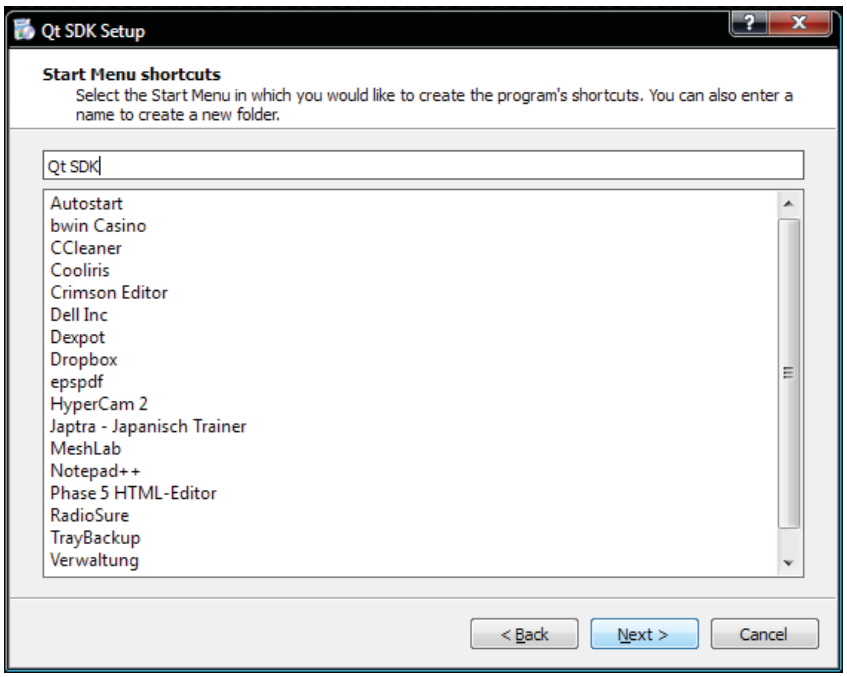
QT Installation: 29.10.2012



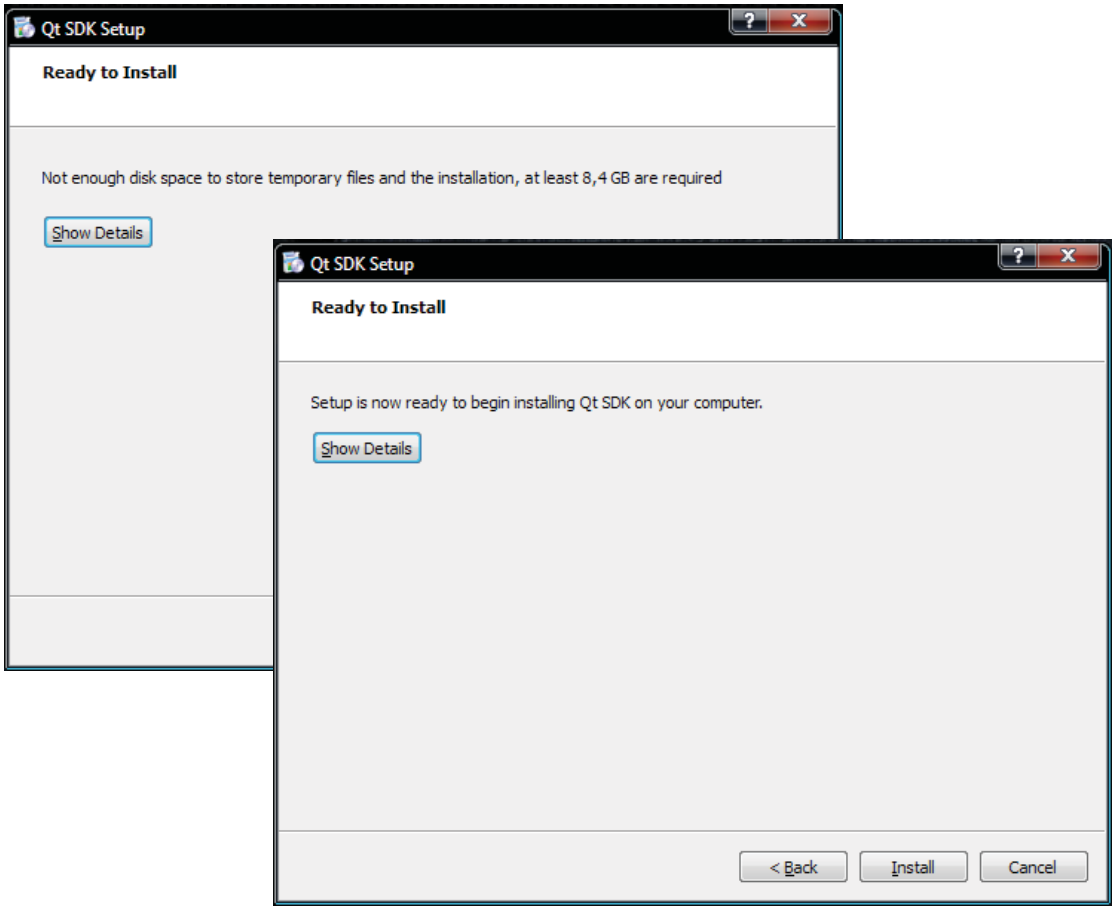
QT Installation: 29.10.2012



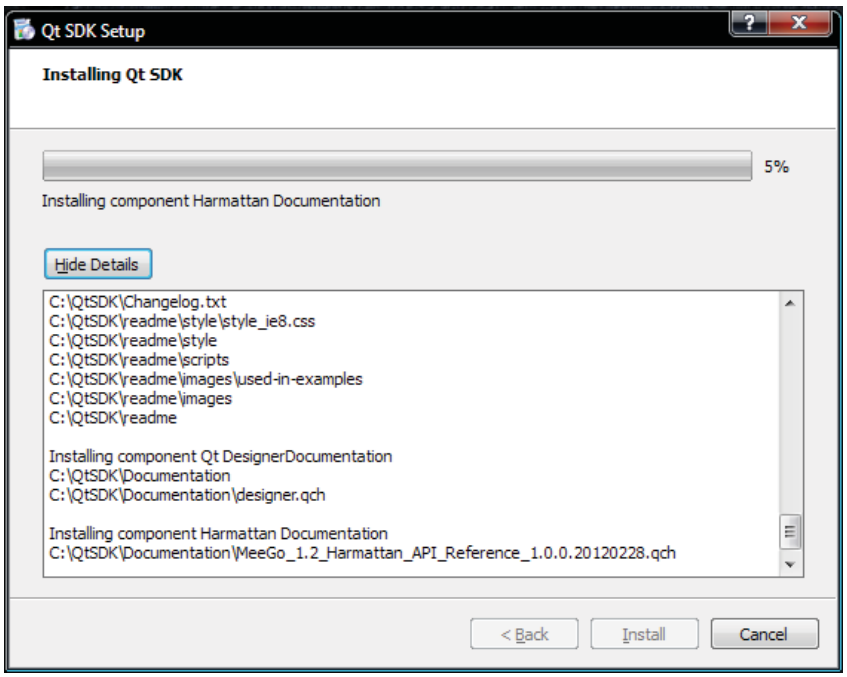
QT Installation: 29.10.2012



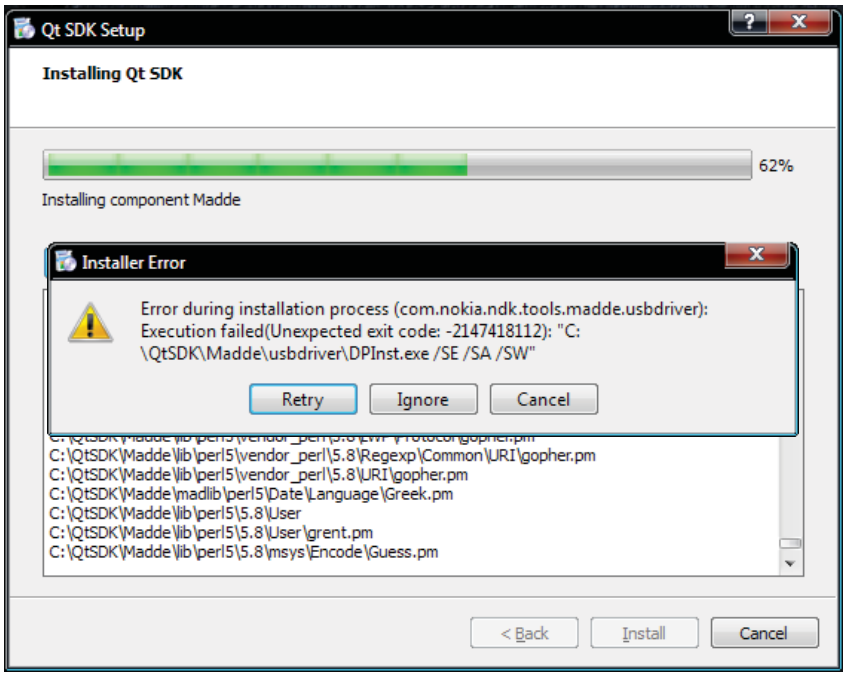
QT Installation: 29.10.2012



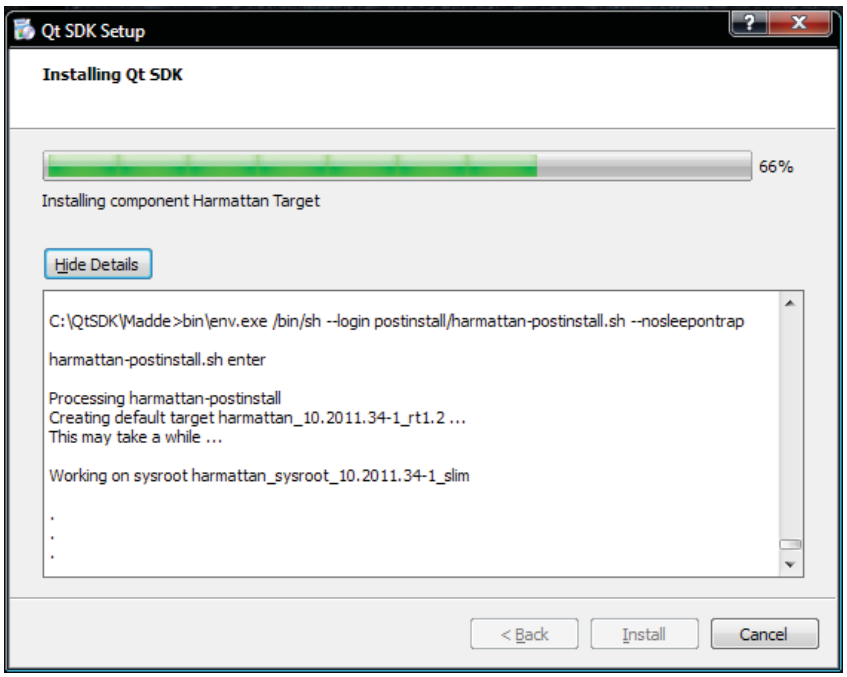
QT Installation: 29.10.2012



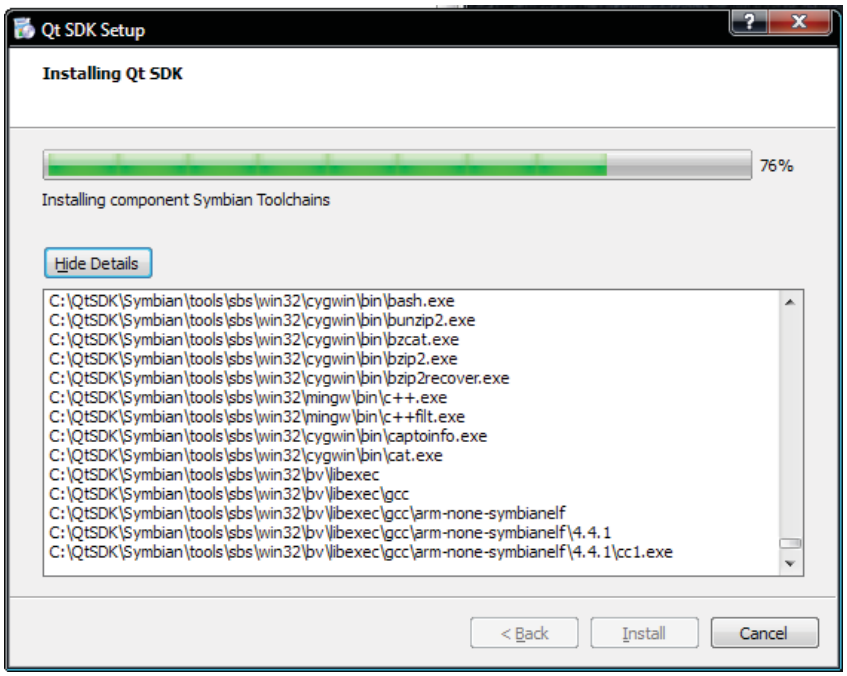
. . . .00:00


Qt SDK Setup
Installing Qt SDK
62%
Installing component Madde
Installer Error
Error during installation process (com.nokia.ndk.tools.madde.usbdriver):
Execution failed(Unexpected exit code: -2147418112): "C:
\QtSDK\Madde\usbdriver\DPInst.exe /SE /SA /SW"
Retry
Ignore
Cancel
C:\QtSDK\Madde\lib\perl5\vendor_perl\5.8\Regexp\Common\URI\gopher.pm
C:\QtSDK\Madde\lib\perl5\vendor_perl\5.8\URI\gopher.pm
C:\QtSDK\Madde\madlib\perl5\Date\Language\Greek.pm
C:\QtSDK\Madde\lib\perl5\5.8\User
C:\QtSDK\Madde\lib\perl5\5.8\User\grent.pm
C:\QtSDK\Madde\lib\perl5\5.8\msys\Encode\Guess.pm
< Back
Install
Cancel

. . . .00:04

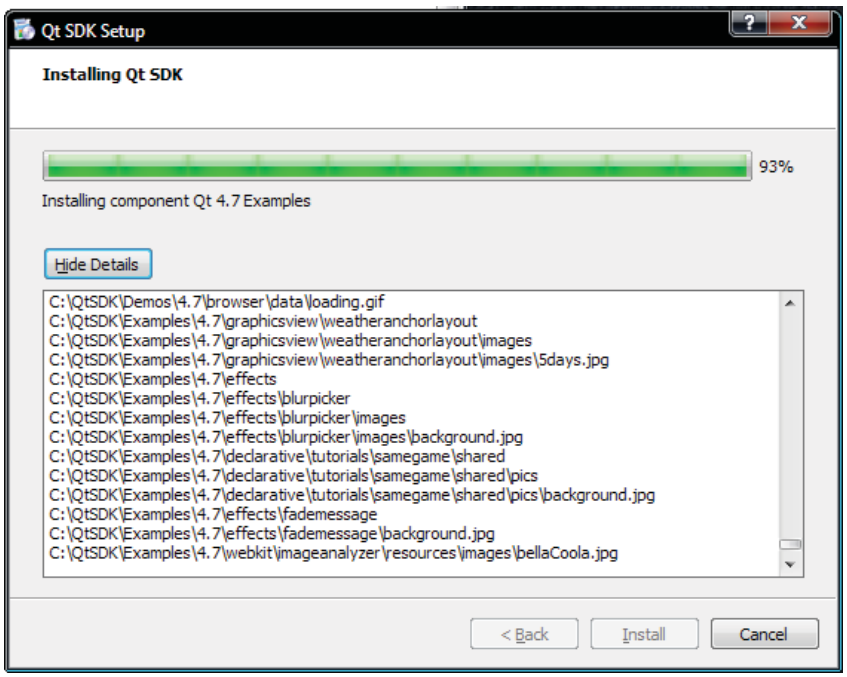

Qt SDK Setup
Installing Qt SDK
66%
Installing component Harmattan Target
Hide Details
C:\QtSDK\Madde>bin\env.exe /bin/sh --login postinstall/harmattan-postinstall.sh --nosleepontrap
harmattan-postinstall.sh enter
Processing harmattan-postinstall
Creating default target harmattan_10.2011.34-1_rt1.2 ...
This may take a while ...
Working on sysroot harmattan_sysroot_10.2011.34-1_slim
.
.
.
< Back
Install
Cancel

. . . .00:04

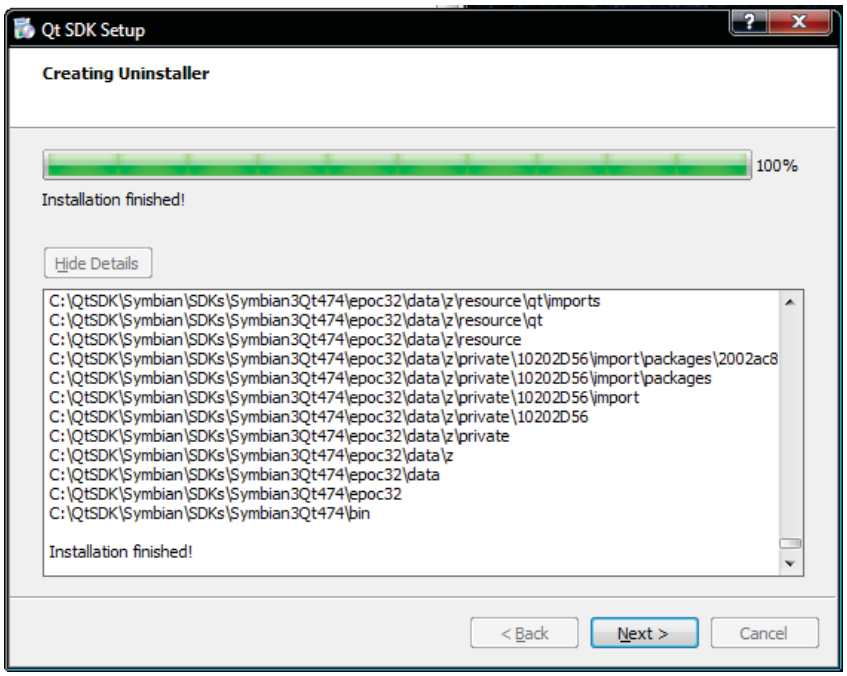


. . . .00:20

. . . .18:20. . . takes time and memory: developer environments for smartphones and Internet tablets . . .



. . . .00:32



. . . .00:34

In meinem Fall: Zeit für Download, Festplatte putzen und Installation: ca 1 Std. . .

Qt Installation

Es ist ihnen hoffentlich gelungen, Qt für ihr Betriebssystem erfolgreich zu installieren. Eine kurze Installationsanleitung war im Skript Hydroinformatik-I (Abschn. 12.8) zu finden. Hier noch mal die Web-Seite für den Download `http://qt.nokia.com/products`. Der große Vorteil von Qt ist die *Plattformunabhängigkeit* sowie die freie Verfügbarkeit unter der GPL (Gnu Public License).
Wir starten Qt mit einem Doppelklick auf das Desktop-Symbol.

Abbildung: Qt Start

Wenn alles gut geht gegangen ist sollten sie im Qt User Interface (UI) landen.

Der Fahrplan für heute

- Siehe Skript Kapitel 5
- Die Qt Installation
- Die `main()` Funktion
- Das Qt Projekt
- Wir plotten mit `QPlotter` [E1]
- Was ist eine Schnittstelle ?
- wir basteln unser MatLab ... [E2]
- wir rechnen eine stationäre Diffusion (Abschn. 1.5.3)

Der Fahrplan für heute

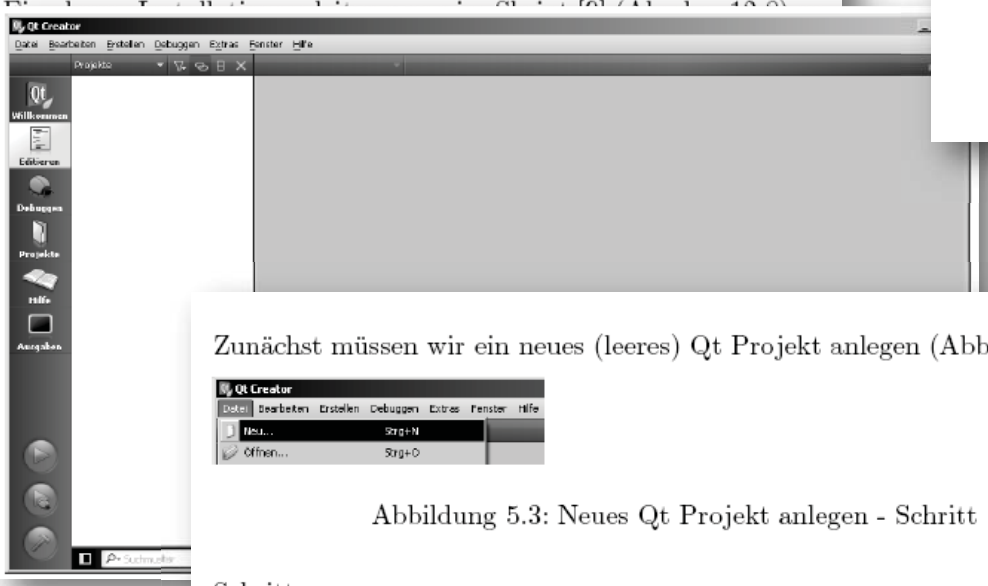
- Siehe Skript Kapitel 5
- Die Qt Installation
- Die `main()` Funktion
- Das Qt Projekt
- Wir plotten mit `QPlotter` [E1]
- Was ist eine Schnittstelle ?
- wir basteln unser MatLab ... [E2]
- wir rechnen eine stationäre Diffusion (Abschn. 1.5.3)

5.1 Qt Projekt

Es ist ihnen hoffentlich gelungen, Qt für ihr Betriebssystem erfolgreich zu installieren. Hi
finden.
http://q
unabhäng
se).

Wir starte

Wenn alle
(Abb. 5.2



Zunächst müssen wir ein neues (leeres) Qt Projekt anlegen (Abb. 5.3-5.6).

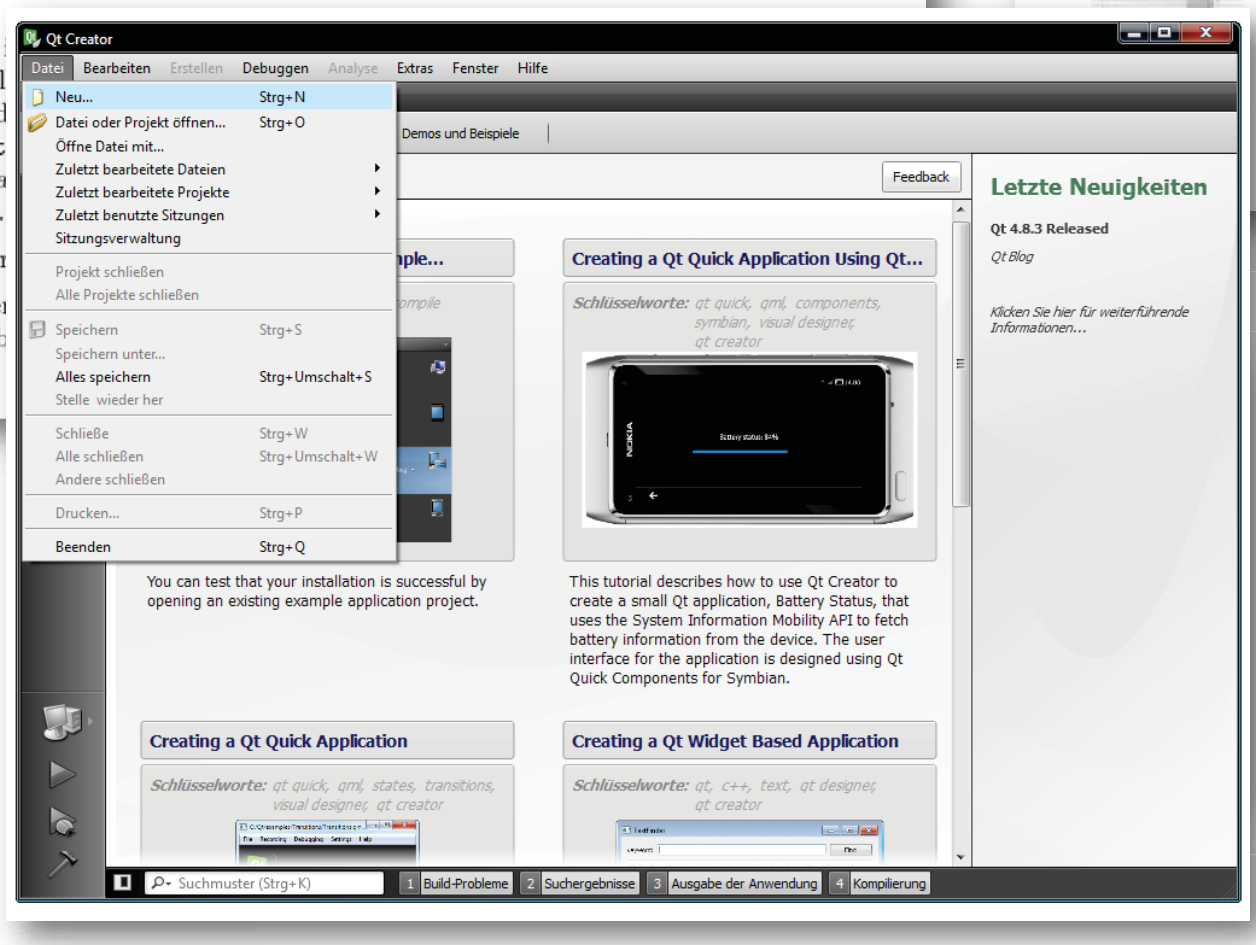
Abbildung 5.3: Neues Qt Projekt anlegen - Schritt 1

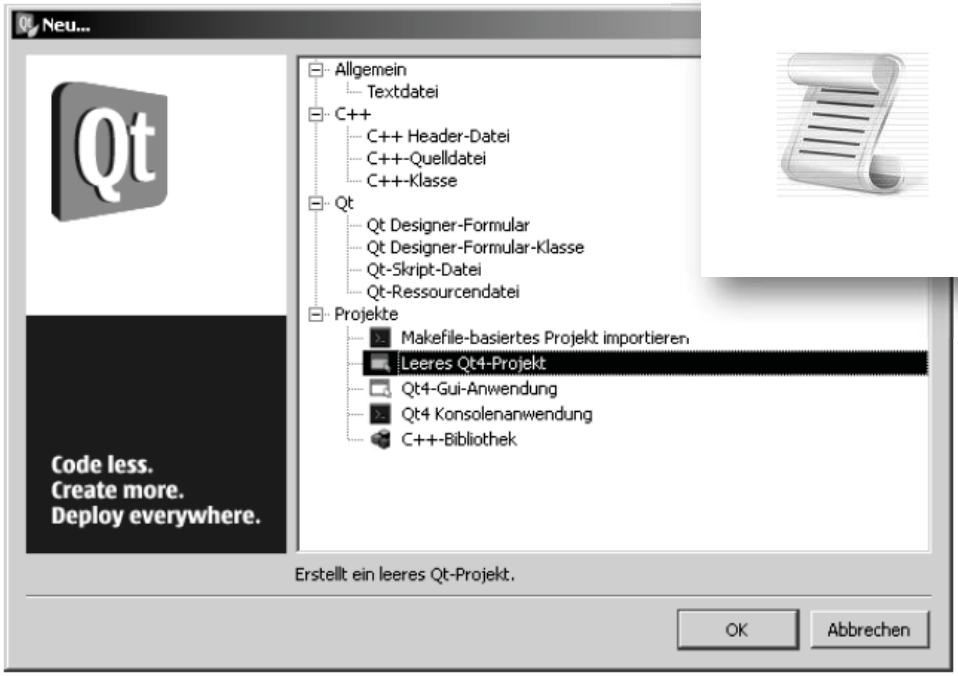
Schritte:

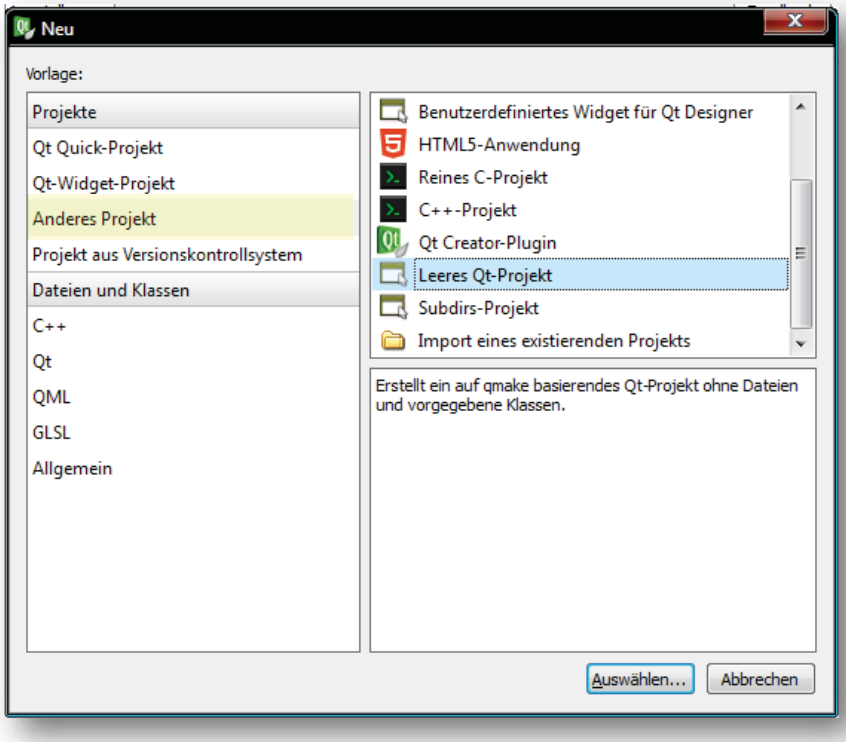
- pro Datei

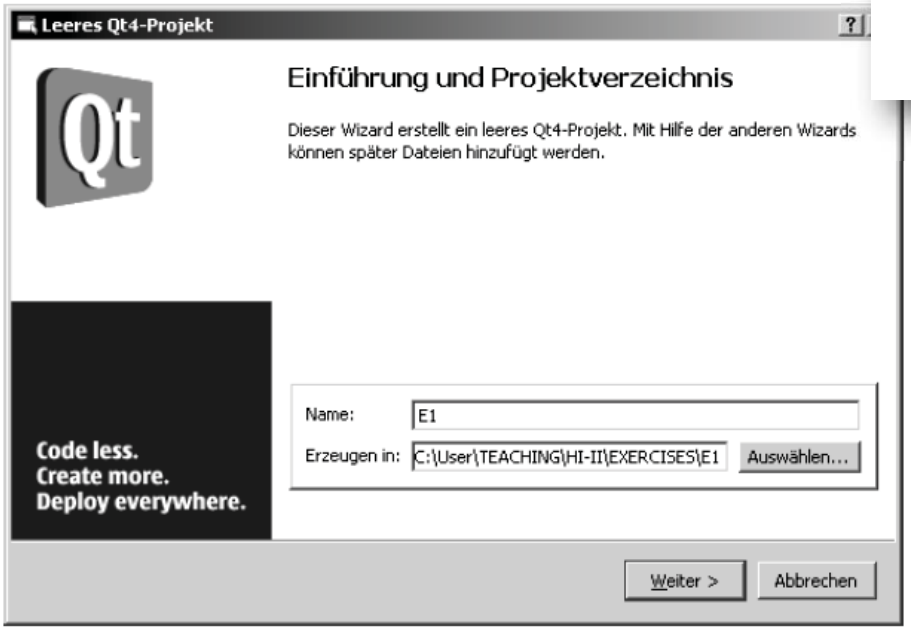
5.1 Qt Projekt

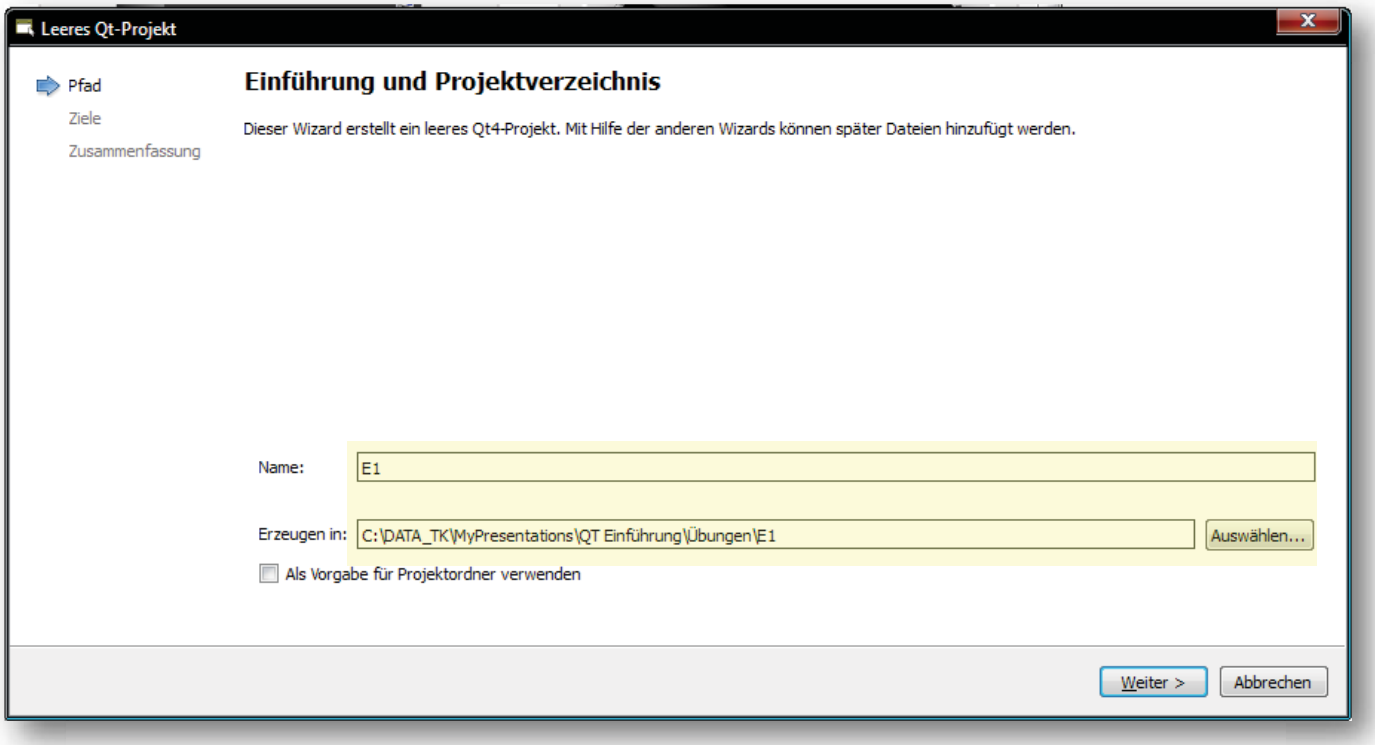
Es
stal
find
htt
una
se).
Wir
Wer
(Ab

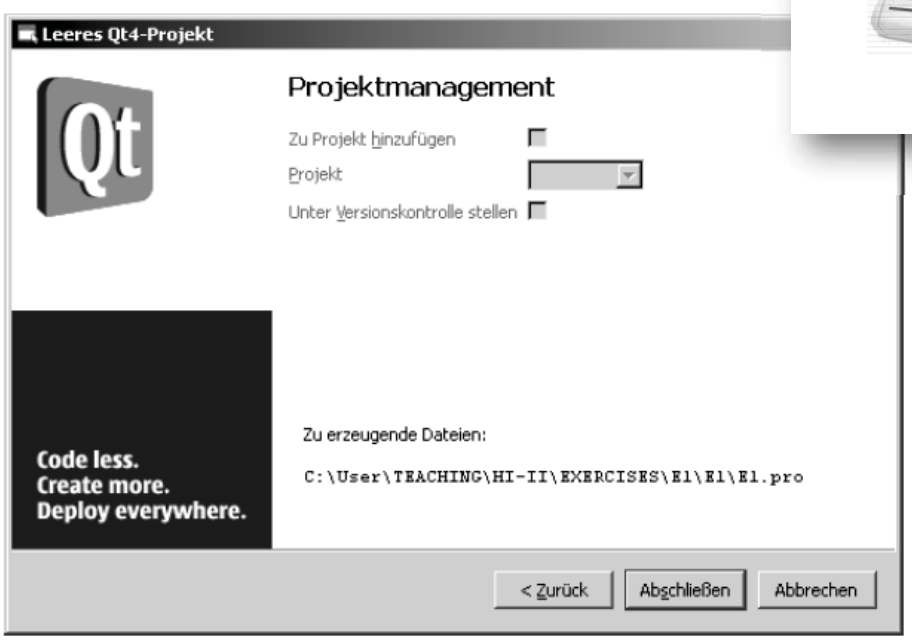


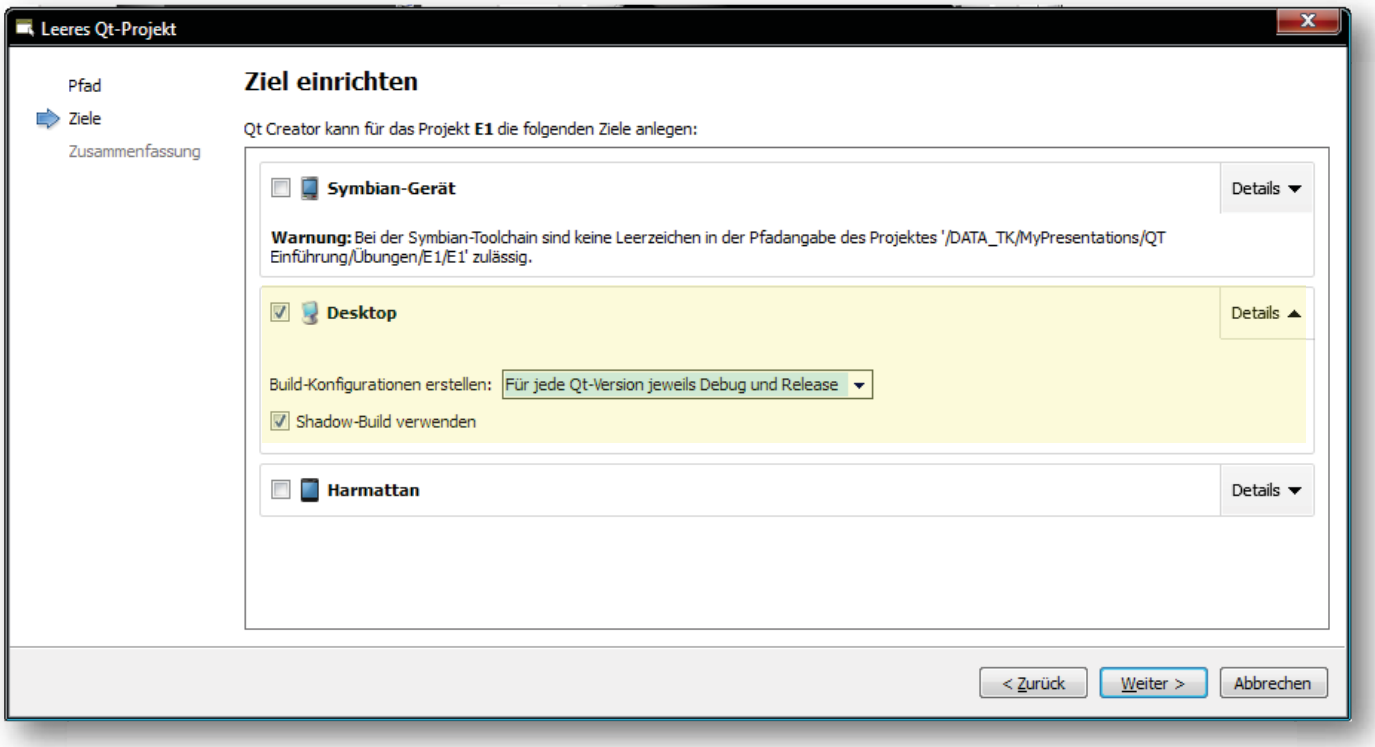


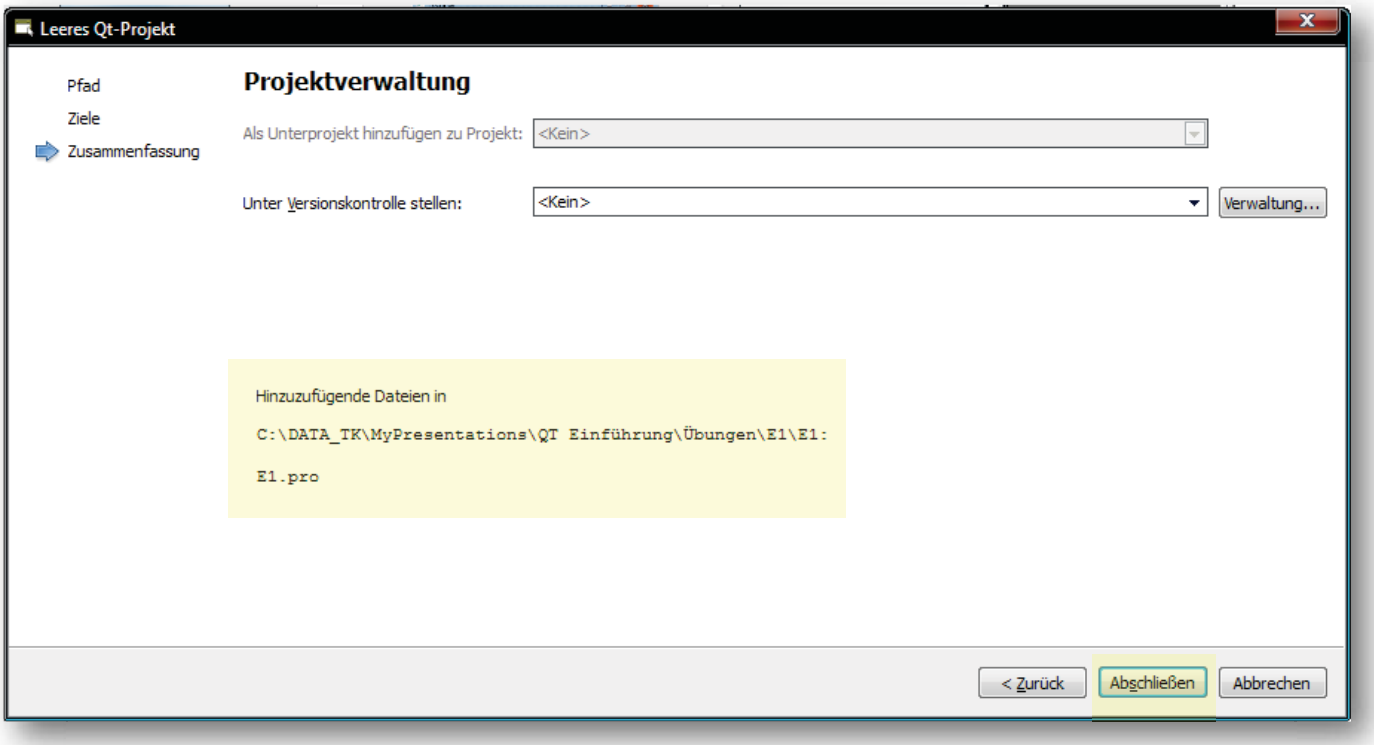

Neu
Vorlage:
Projekte
Qt Quick-Projekt
Qt-Widget-Projekt
Anderes Projekt
Projekt aus Versionskontrollsystem
Dateien und Klassen
C++
Qt
QML
GLSL
Allgemein
Benutzerdefiniertes Widget für Qt Designer
HTML5-Anwendung
Reines C-Projekt
C++-Projekt
Qt Creator-Plugin
Leeres Qt-Projekt
Subdirs-Projekt
Import eines existierenden Projekts
Erstellt ein auf qmake basierendes Qt-Projekt ohne Dateien und vorgegebene Klassen.
Auswählen...
Abbrechen

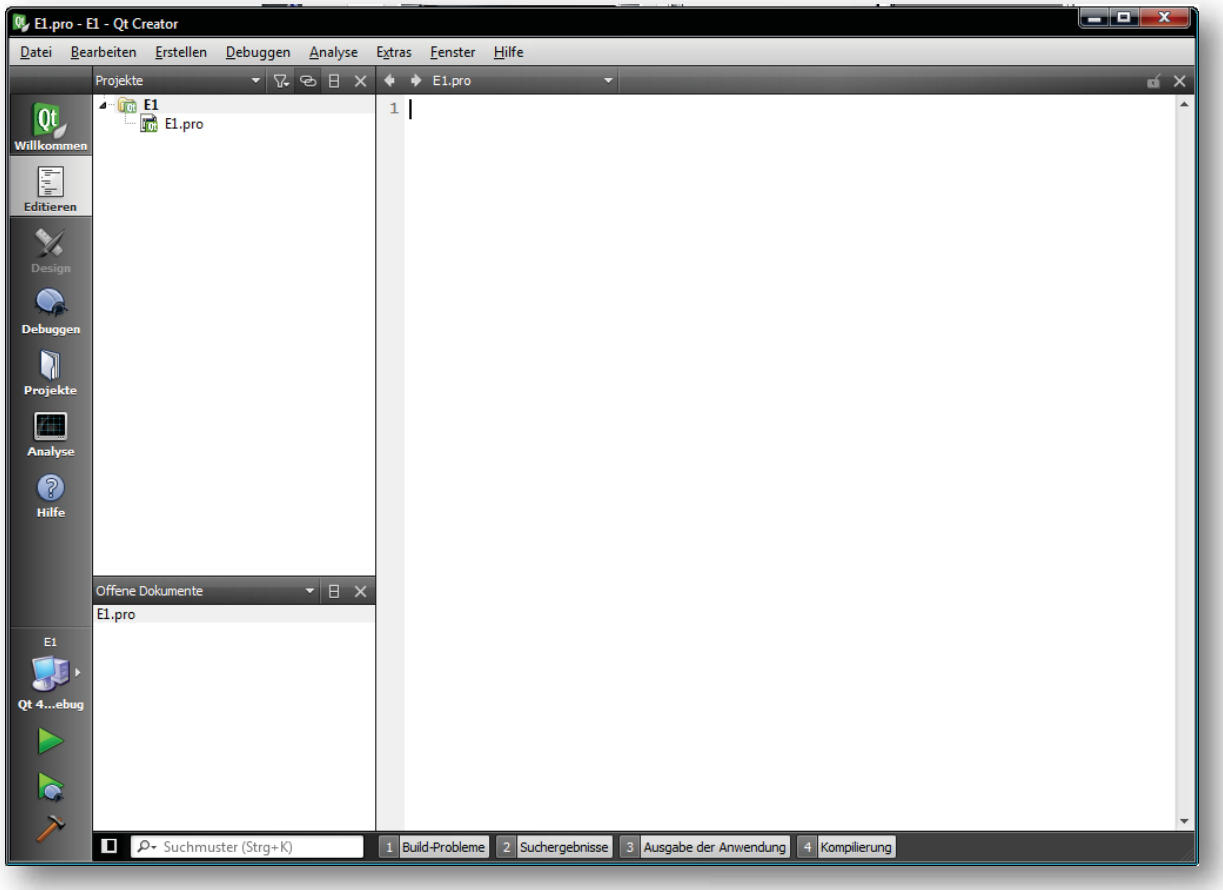

Leeres Qt4-Projekt
Einführung und Projektverzeichnis
Dieser Wizard erstellt ein leeres Qt4-Projekt. Mit Hilfe der anderen Wizards können später Dateien hinzufügt werden.
Code less.
Create more.
Deploy everywhere.
Name:
E1
Erzeugen in:
C:\User\TEACHING\HI-II\EXERCISES\E1
Auswählen...
Weiter >
Abbrechen


Leeres Qt-Projekt
Pfad
Ziele
Zusammenfassung
Einführung und Projektverzeichnis
Dieser Wizard erstellt ein leeres Qt4-Projekt. Mit Hilfe der anderen Wizards können später Dateien hinzufügt werden.
Name:
E1
Erzeugen in:
C:\DATA_TK\MyPresentations\QT Einführung\Übungen\E1
Auswählen...
Als Vorgabe für Projektordner verwenden
Weiter >
Abbrechen

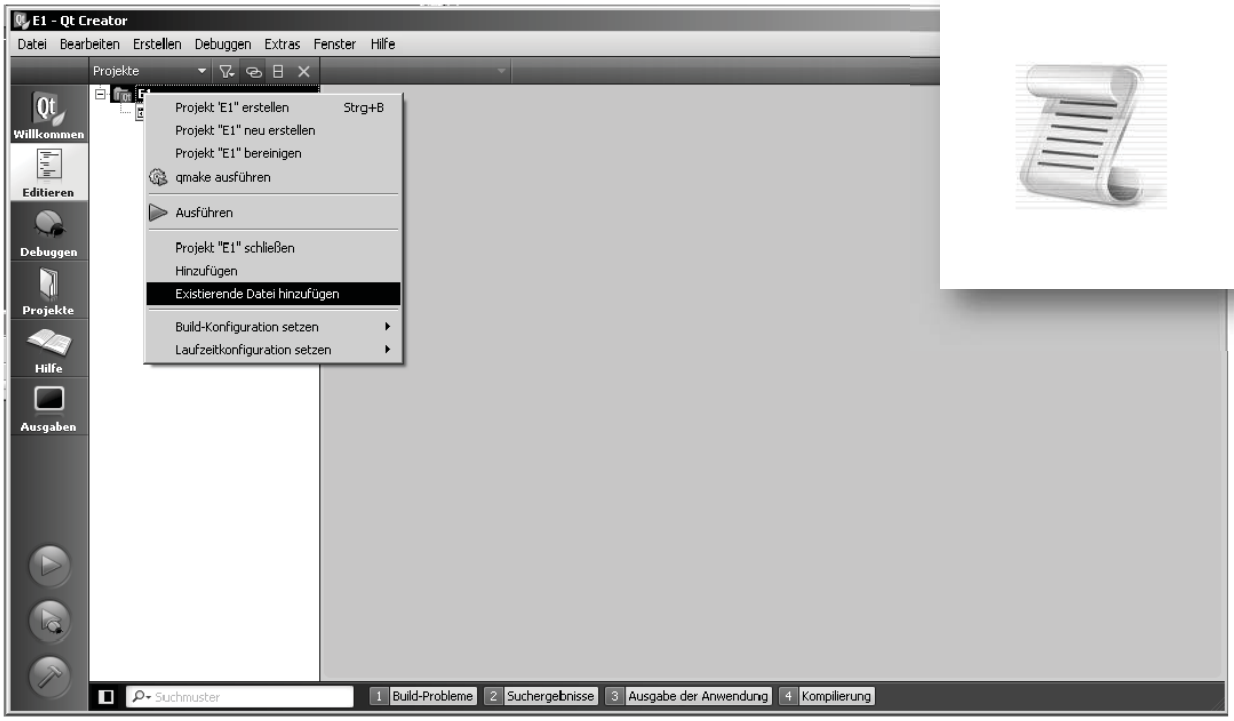

Leeres Qt4-Projekt
Projektmanagement
Zu Projekt hinzufügen
Projekt
Unter Versionskontrolle stellen
Code less.
Create more.
Deploy everywhere.
Zu erzeugende Dateien:
C:\User\TEACHING\HI-II\EXERCISES\E1\E1\E1.pro
< Zurück
Abschließen
Abbrechen


Leeres Qt-Projekt
Pfad
Ziele
Zusammenfassung
Ziel einrichten
Qt Creator kann für das Projekt E1 die folgenden Ziele anlegen:
Symbian-Gerät
Details
Warnung: Bei der Symbian-Toolchain sind keine Leerzeichen in der Pfadangabe des Projektes '/DATA_TK/MyPresentations/QT Einführung/Übungen/E1/E1' zulässig.
Desktop
Details
Build-Konfigurationen erstellen:
Für jede Qt-Version jeweils Debug und Release
Shadow-Build verwenden
Harmattan
Details
< Zurück
Weiter >
Abbrechen

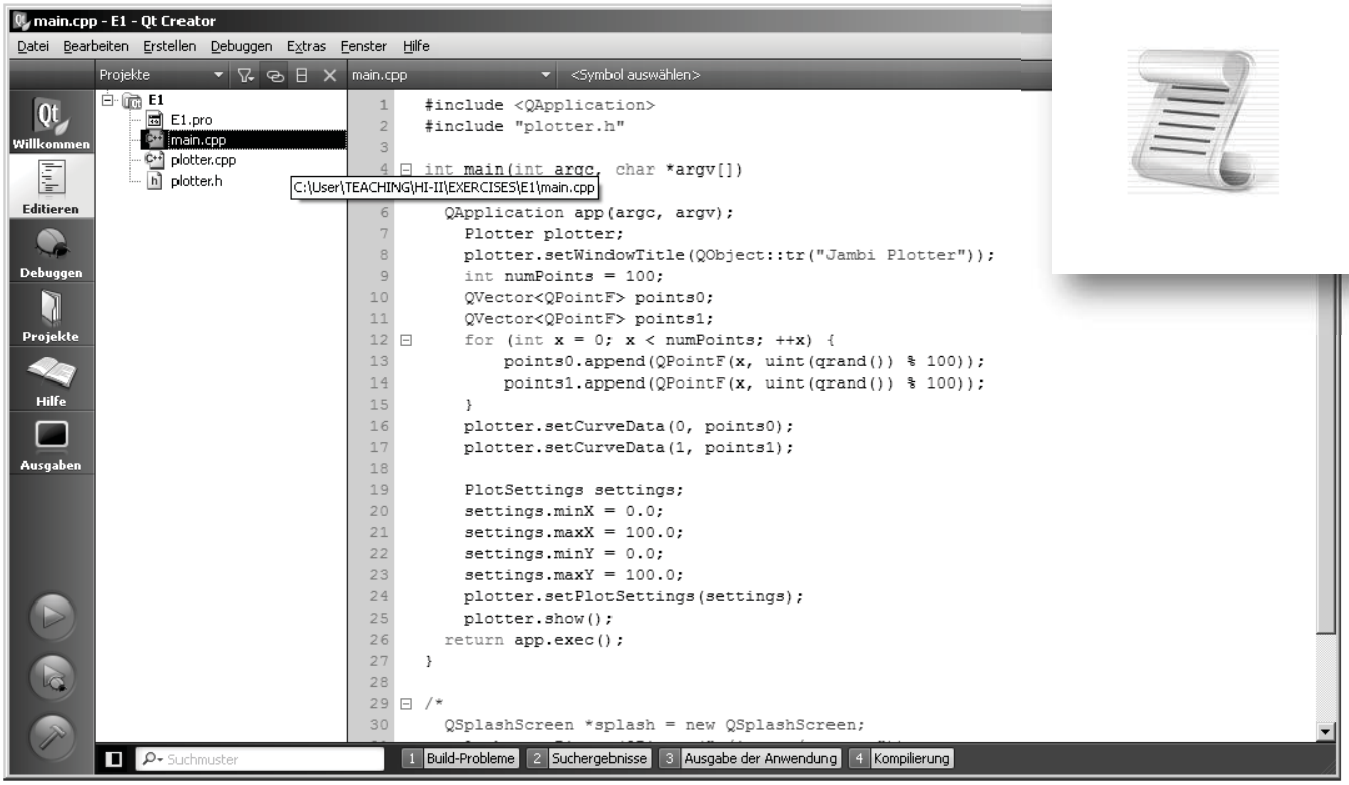

Leeres Qt-Projekt
Pfad
Ziele
Zusammenfassung
Projektverwaltung
Als Unterprojekt hinzufügen zu Projekt:
<Kein>
Unter Versionskontrolle stellen:
<Kein>
Verwaltung...
Hinzuzufügende Dateien in
C:\DATA_TK\MyPresentations\QT Einführung\Übungen\E1\E1:
E1.pro
< Zurück
Abschließen
Abbrechen

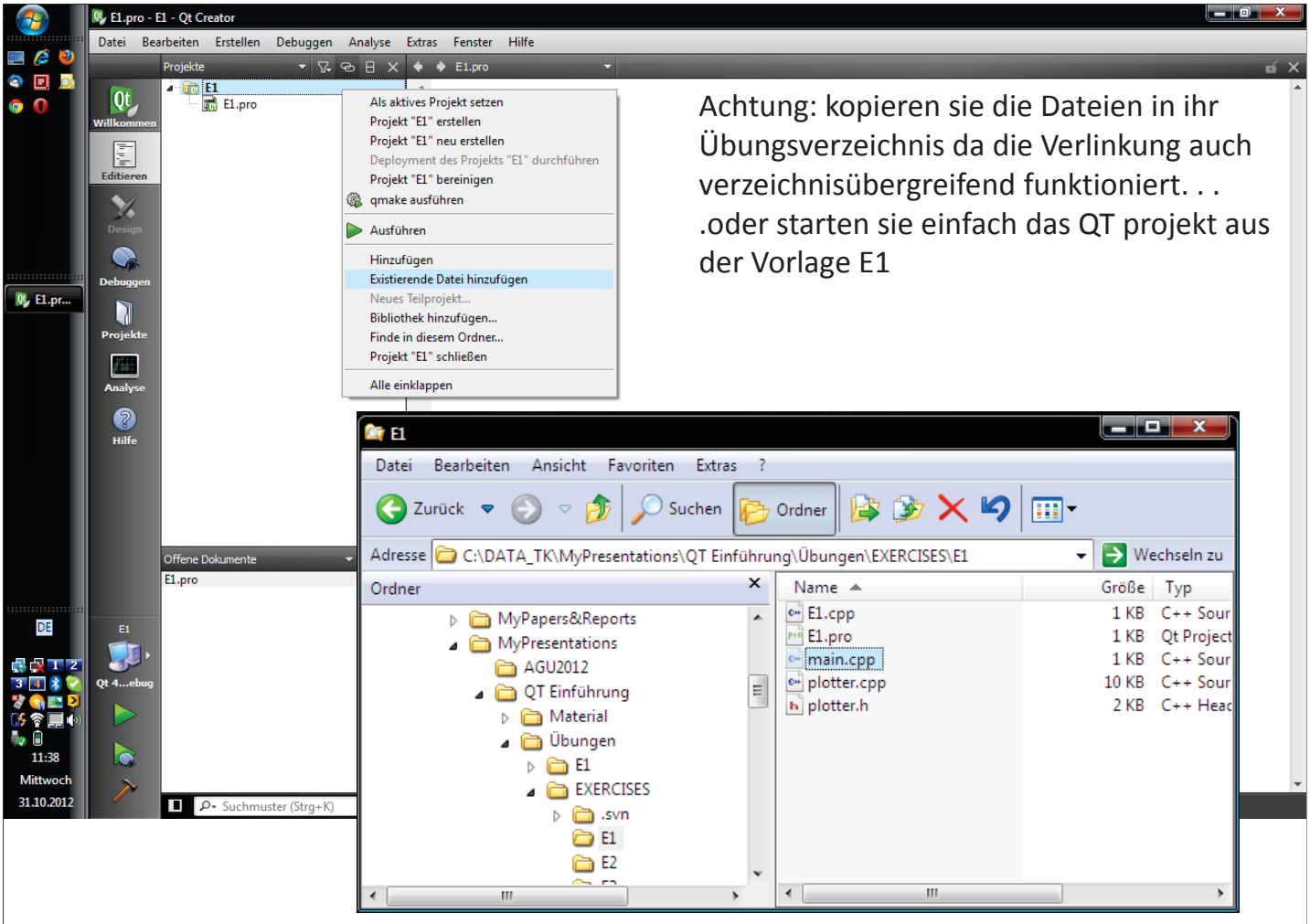


Nachdem ein leeres Qt Projekt angelegt ist, fügen wir zunächst eine existierende Quell-Datei mit der `main` Funktion hinzu (Abb. 5.7).

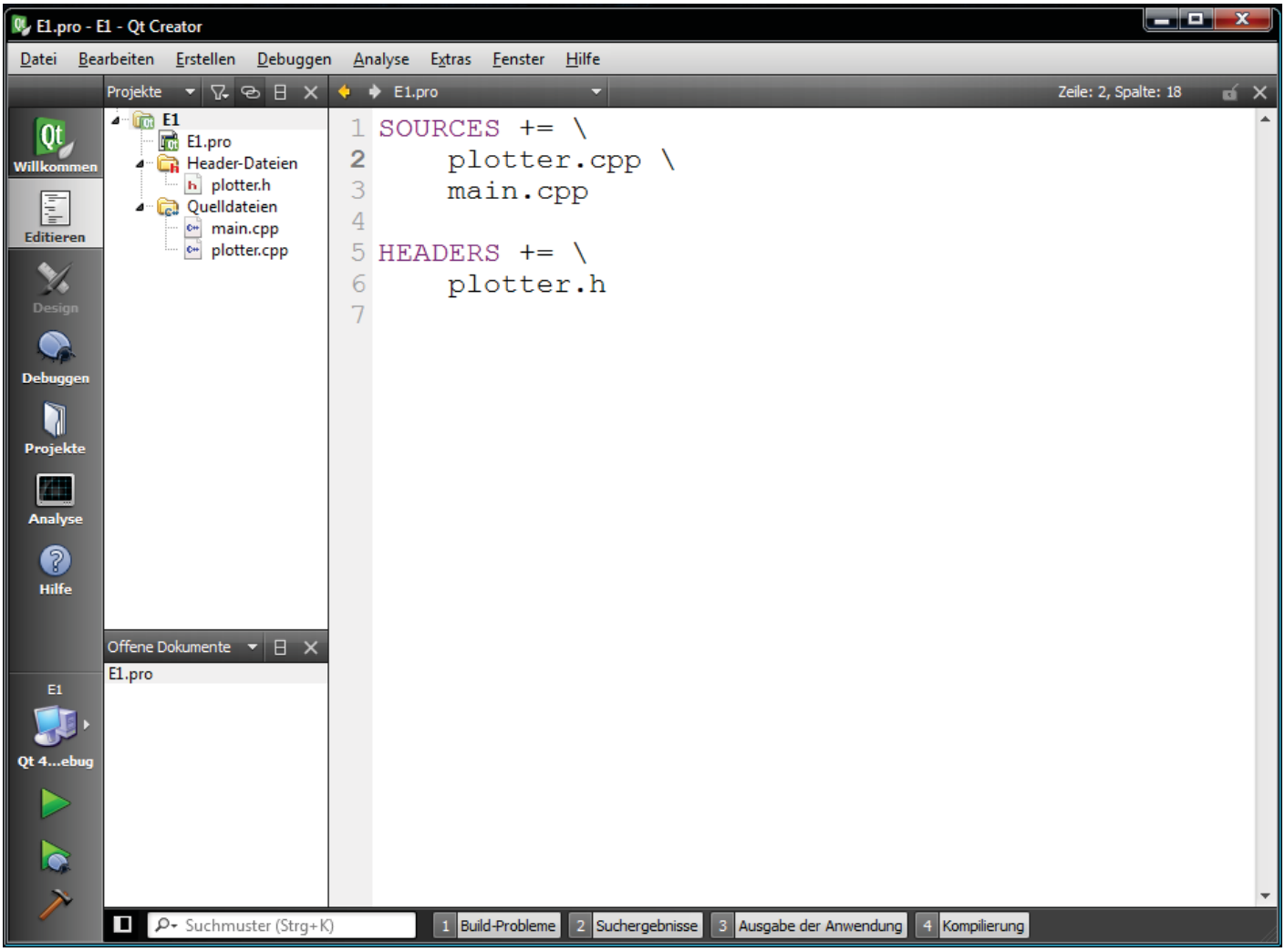


Es können weitere Quell-Dateien hinzugefügt werden, wie z.B. der Qt Plotter (Abb. 5.8).



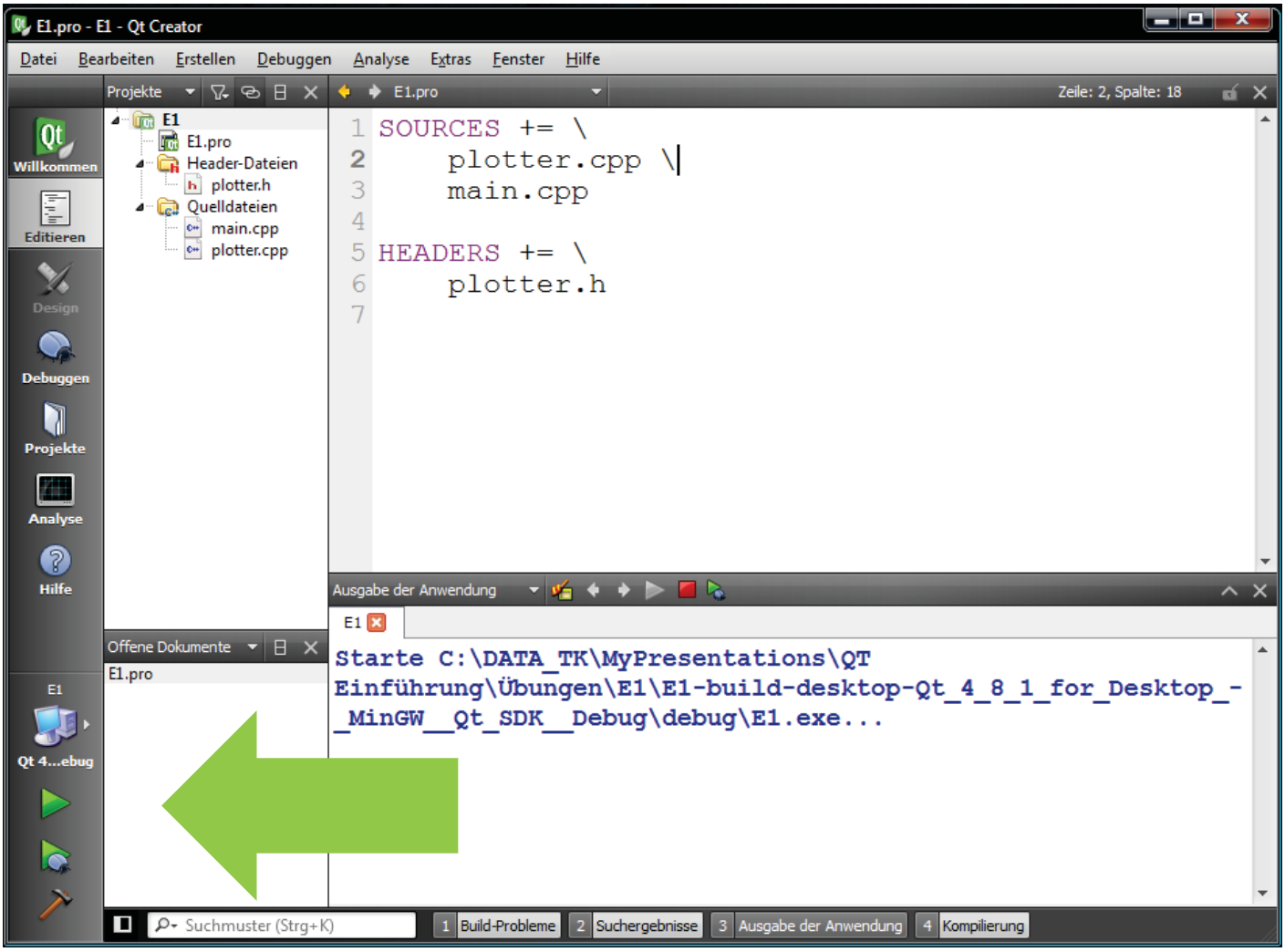


Achtung: kopieren sie die Dateien in ihr Übungsverzeichnis da die Verlinkung auch verzeichnisübergreifend funktioniert. . . .oder starten sie einfach das QT projekt aus der Vorlage E1



Unser Projekt ist schon kompilierfähig. Mit einem Druck auf das "grüne Knöpfchen" (Abb. 5.9) wird kompliert und das Programm ausgeführt.

Abbildung 5.9: Kompilation und Programmstart



Im Ergebnis haben wir unseren ersten Qt Plot (Abb. 5.10), unheimlich - nicht wahr.

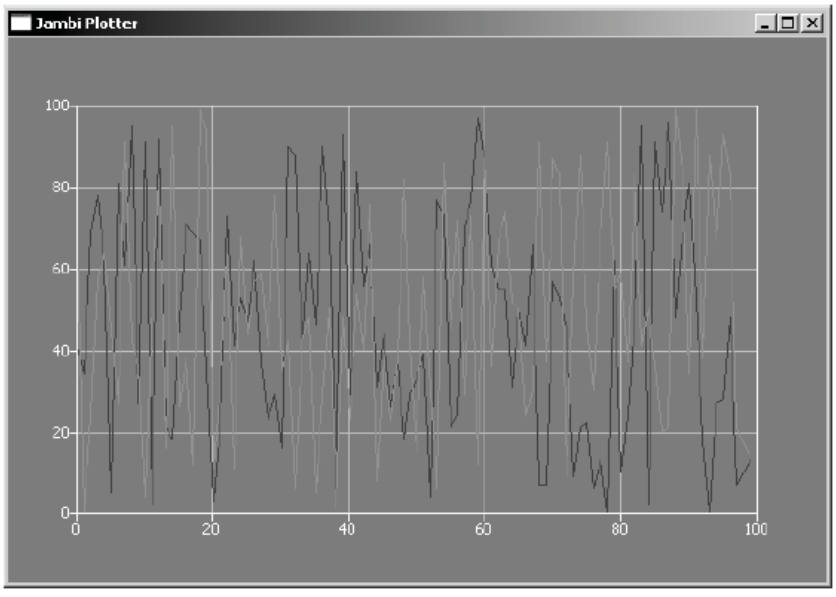
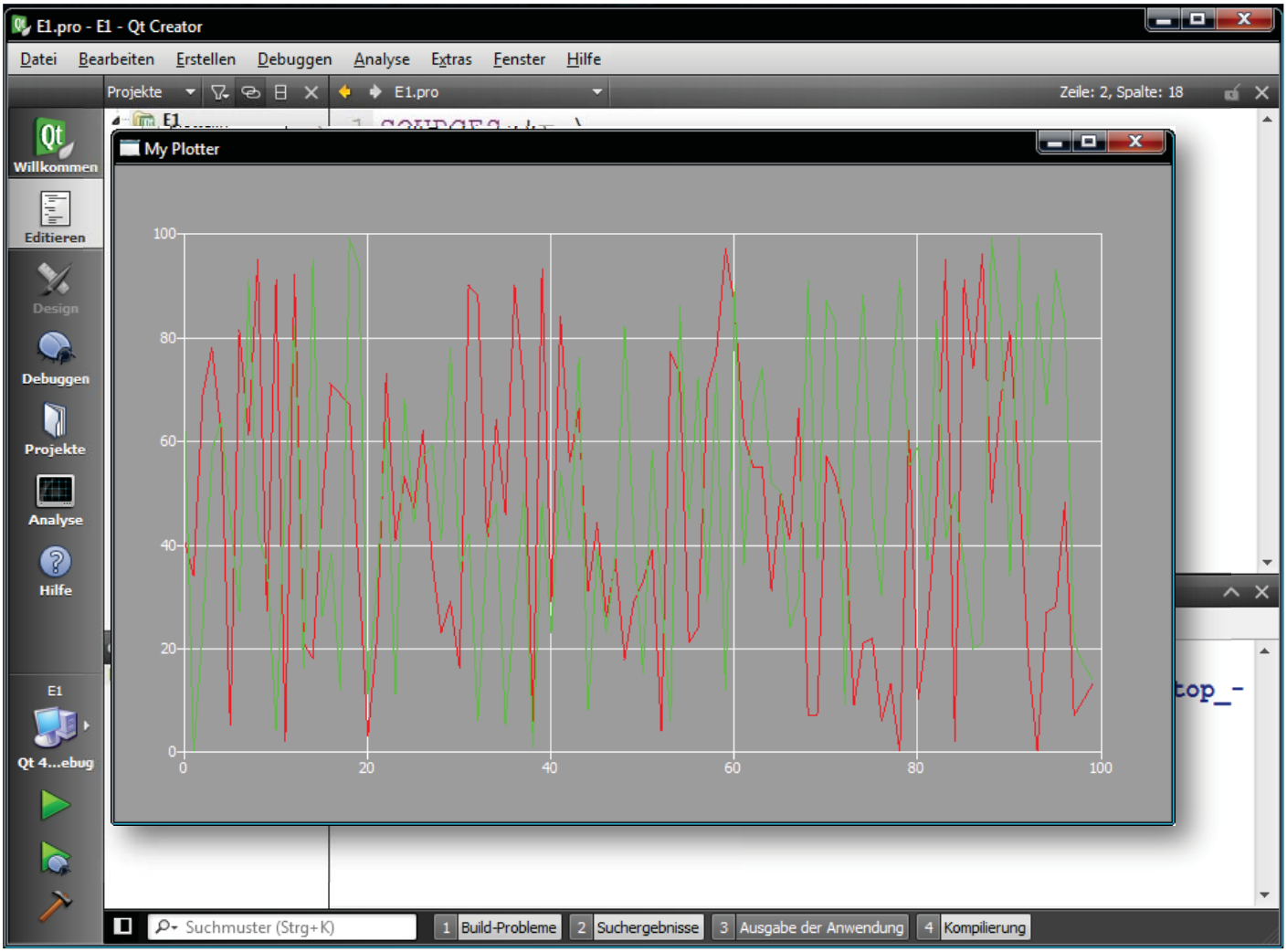


Abbildung 5.10: Qt Plotter



Der Fahrplan für heute

- Siehe Skript Kapitel 5
- Die Qt Installation
- Die `main()` Funktion
- Das Qt Projekt
- Wir plotten mit `QPlotter` [E1]
- Was ist eine Schnittstelle ?
- wir basteln unser MatLab ... [E2]
- wir rechnen eine stationäre Diffusion (Abschn. 1.5.3)

Jetzt schauen wir uns die Sache mal etwas genauer an.

Übung: E5.1

```
#include <QApplication>
#include "plotter.h"

int main(int argc, char *argv[])
{
  QApplication app(arg
  // Data
  int numPoints = 100;
  QVector<QPointF> poi
  QVector<QPointF> poi
  for (int x = 0; x <
    points0.append(QPo
    points1.append(QPo
  }
  // Plotter
  Plotter plotter;
  plotter.setWindowTitle(QObject::tr("Jambi Plotter"));
  plotter.setCurveData(0, points0);
  plotter.setCurveData(1, points1);
  PlotSettings settings;
  settings.minX = 0.0;
  settings.maxX = 100.0;
  settings.minY = 0.0;
  settings.maxY = 100.0;
  plotter.setPlotSettings(settings);
  plotter.show();
  //
  return app.exec();
}
```


Der Fahrplan für heute

- Siehe Skript Kapitel 5
- Die Qt Installation
- Die `main()` Funktion
- Das Qt Projekt
- Wir plotten mit `QPlotter` [E1]
- Was ist eine Schnittstelle ?
- wir basteln unser MatLab ... [E2]
- wir rechnen eine stationäre Diffusion (Abschn. 1.5.3)

Die main() Funktion

```
int main(int argc, char *argv[])
{
  ...
  return 0;
}
```

```
#include <QApplication>

int main(int argc, char *argv[])
{
  // Start application
  QApplication app(argc,argv);
  ...
  // End application
  app.exec();
  return 0;
}
```


Die main() Funktion

```
int main(int argc, char *argv[])
{
  ...
  return 0;
}
```

```
#include <stdio.h>                    C++
// main function for Hello
int main()
{
     printf("Hello!");
     return 0;
}
```

main() needs to return an int

In C++, this function takes an int and a char * array (an array of character strings) as parameters (Comand Line Arguments).

argc (*argument count*)
argv (*argument vector*)

Die main() Funktion

```
int main(int argc, char *argv[])
{
  ...
  return 0;
}
```

The QApplication constructors requires argc and argv because Qt supports a few command-line arguments of its own. (*The first argument is e.g. the name of the executable*)

```
#include <QApplication>

int main(int argc, char *argv[])
{
  // Start application
  QApplication app(argc,argv);
  ...
  // End application
  app.exec();
  return 0;
}
```


Die main() Funktion

QApplication class:

- **For every Qt class, there is a header file with the same name as the class that contains the class's definition.**

```
#include <QApplication>

int main(int argc, char *argv[])
{
  // Start application
  QApplication app(argc,argv);
  ...
  // End application
  app.exec();
  return 0;
}
```

QApplication class manages the GUI application's control flow and main settings

- QApplication contains the main event loop, where all events from the window system and other sources are processed and dispatched.
- It also handles the application's initialization and finalization.
- For any GUI application using Qt, there is precisely **one** QApplication object, no matter whether the application has 0, 1, 2 or more windows at any given time.
- For non-GUI Qt applications, use QCoreApplication instead, as it does not depend on theQtGui library.

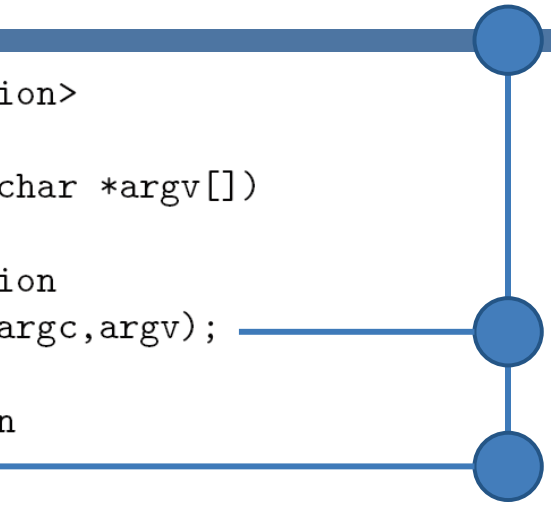
QApplication class main areas of responsibility are:

- **Initialization of the application and desktop settings** such as palette(), font() and doubleClickInterval().
- **Tracking** of these properties
- **Event handling**, meaning that it receives events from the underlying window system and dispatches them to the relevant widgets.
- **Parsing CCLA** common command line arguments
- **Application's look** (QStyle objects e.g. see setColorSpec() for details).
- **Window management** (e.g. position, size, etc)
- **Session Management** (e.g. terminate gracefully when the user logs out)

Die main() Funktion

QApplication class:

- **The QApplication object is accessible through the instance() function that returns a pointer**



```
#include <QApplication>

int main(int argc, char *argv[])
{
  // Start application
  QApplication app(argc,argv);
  ...
  // End application
  app.exec();
  return 0;
}
```


OOP: Class

In C++:

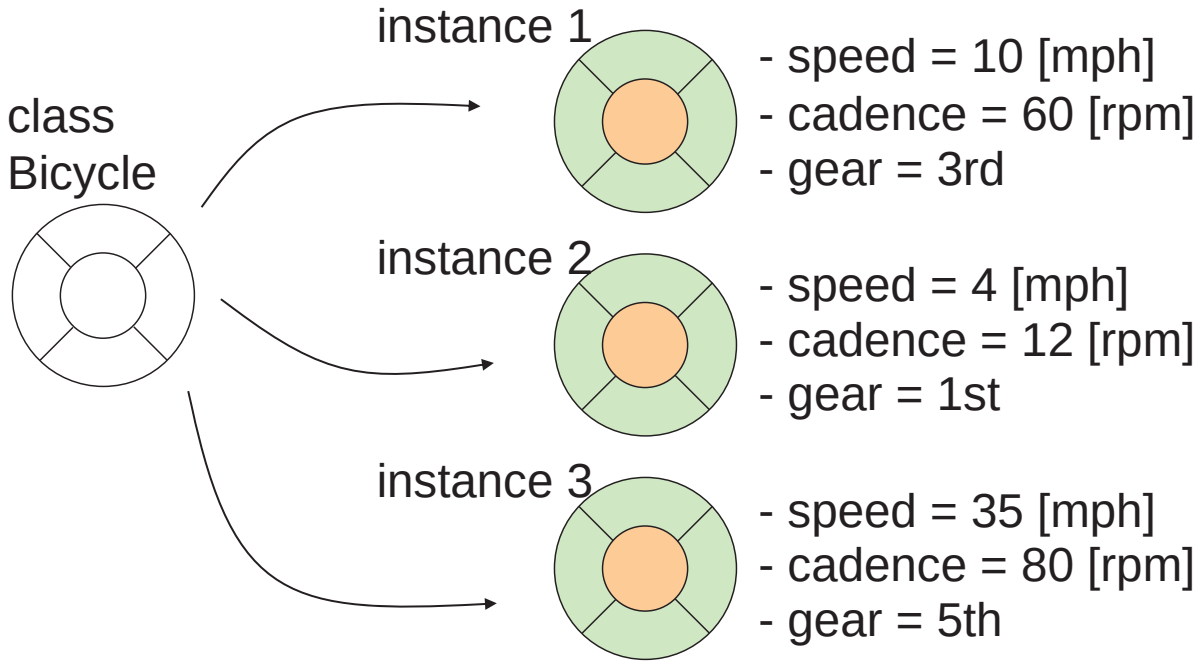

```
class Bicycle {
private:
    float speed;
    float cadence;
    int gear;
public:
    void change_gears(int gear);
    void break();
    void chage_cadence(float cadence);
};
```

- A class is a template definition of the methods and variables.
- An object is a specific instance of a class; it contains real values instead of variables. The object or class instance is what you run in the computer.

OOP: Class

INTRO CONCEPTS OOP GEO MSH PCS

Different instances can have different values of member variables



Der Fahrplan für heute

- Siehe Skript Kapitel 5
- Die Qt Installation
- Die `main()` Funktion
- Das Qt Projekt
- Wir plotten mit `QPlotter` [E1]
- Was ist eine Schnittstelle ?
- wir basteln unser MatLab ... [E2]
- wir rechnen eine stationäre Diffusion (Abschn. 1.5.3)

Der Fahrplan für heute

- Siehe Skript Kapitel 5
- Die Qt Installation
- Die `main()` Funktion
- Das Qt Projekt
- Wir plotten mit `QPlotter` [E1]
- Was ist eine Schnittstelle ?
- wir basteln unser MatLab ... [E2]
- wir rechnen eine stationäre Diffusion (Abschn. 1.5.3)

Locker plotten mit `QPlotter` (5.1)[E1]

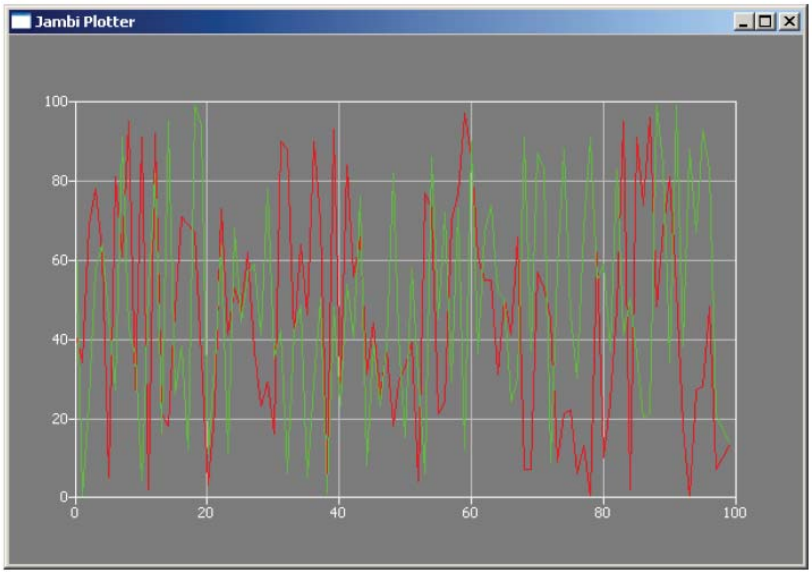


Abbildung: Qt Plotter

Jetzt schauen wir uns die Sache mal etwas genauer an.

Übung: E5.1

```
#include <QApplication>
#include "plotter.h"

int main(int argc, char *argv[])
{
  QApplication app(arg
  // Data
  int numPoints = 100;
  QVector<QPointF> poi
  QVector<QPointF> poi
  for (int x = 0; x <
    points0.append(QPo
    points1.append(QPo
  }
  // Plotter
  Plotter plotter;
  plotter.setWindowTitle(QObject::tr("Jambi Plotter"));
  plotter.setCurveData(0, points0);
  plotter.setCurveData(1, points1);
  PlotSettings settings;
  settings.minX = 0.0;
  settings.maxX = 100.0;
  settings.minY = 0.0;
  settings.maxY = 100.0;
  plotter.setPlotSettings(settings);
  plotter.show();
  //
  return app.exec();
}
```

plotter.cpp - E1 - Qt Creator

Datei Bearbeiten Erstellen Debuggen Analyse Extras Fenster Hilfe

Projekte | plotter.cpp | Plotter::setPlotSettings(const PlotSettings &) | Zeile: 37, Spalte: 1

Willkommen
Editieren
Design
Debuggen
Projekte
Analyse
Hilfe
E1
Qt 4...ebug

- E1
  - E1.pro
  - Header-Dateien
    - plotter.h
  - Quelldateien
    - main.cpp
    - plotter.cpp

Offene Dokumente
- E1.pro
- main.cpp*
- plotter.cpp
- plotter.h

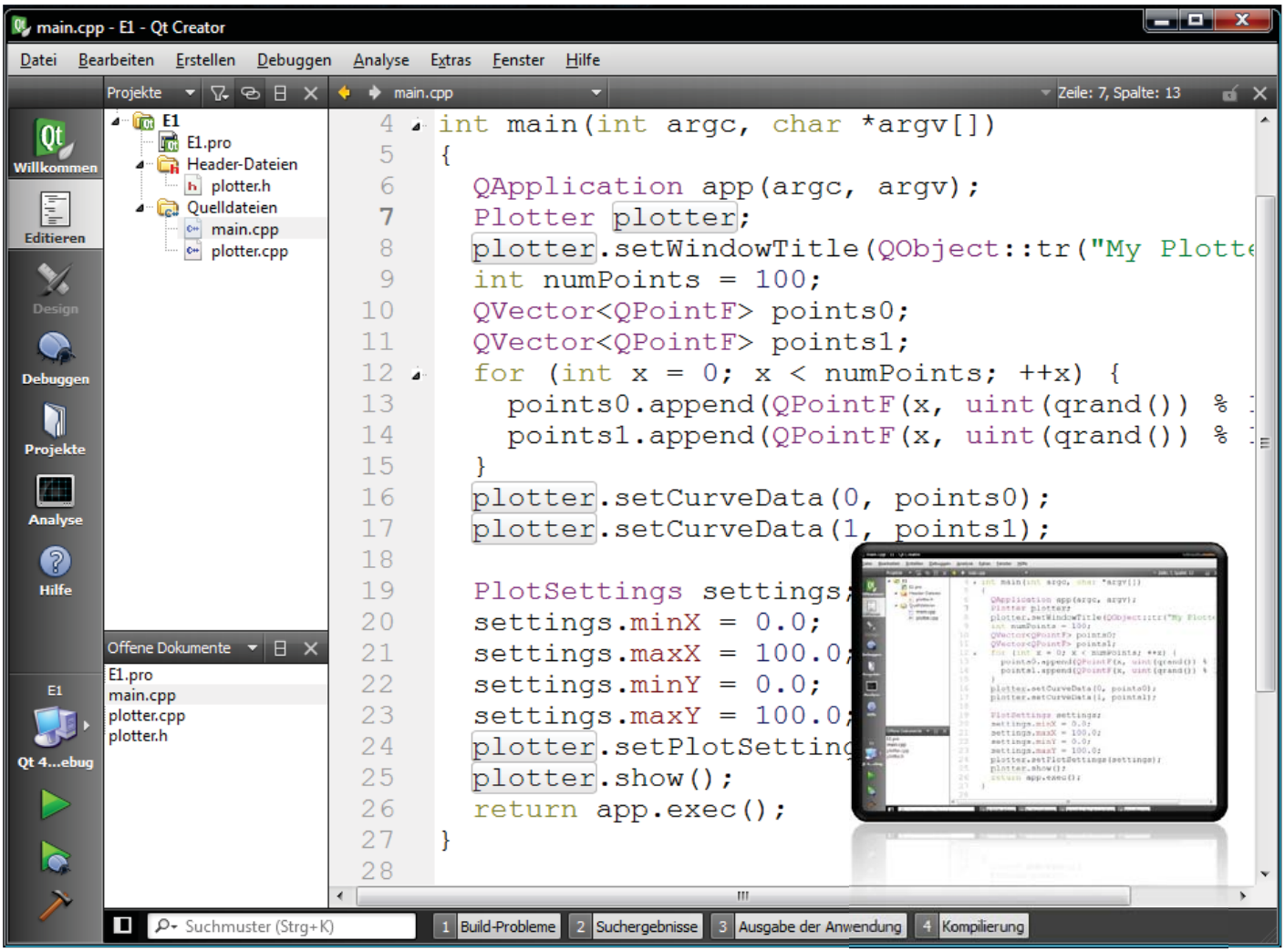
```
1  #include <QtGui>
2  #include <cmath>
3
4  #include "plotter.h"
5
6  Plotter::Plotter(QWidget *parent)
7      : QWidget(parent)
8  {
   ...nding, QSi...
   ...this);
   ...images/zoo...
   ...icked()),
20     zoomOutB...
21     zoomOutB...                    .../zo...
22     zoomOutButton->adjustSize();
23     connect(zoomOutButton, SIGNAL(clicked()),
24
25     setPlotSettings(PlotSettings());
```

Einbinden des Qt –XY-Plotters

- copy and paste plotter.h /plotter.cpp
- Zum Projekt hinzufügen
- #include “plotter.h“

The QWidget class is the base class of all user interface objects.

Suchmuster (Strg+K) | 1 Build-Probleme | 2 Suchergebnisse | 3 Ausgabe der Anwendung | 4 Kompilierung



Der Fahrplan für heute

- Siehe Skript Kapitel 5
- Die Qt Installation
- Die `main()` Funktion
- Das Qt Projekt
- Wir plotten mit `QPlotter` [E1]
- Was ist eine Schnittstelle ?
- wir basteln unser MatLab ... [E2]
- wir rechnen eine stationäre Diffusion (Abschn. 1.5.3)

Der Fahrplan für heute

- Siehe Skript Kapitel 5
- Die Qt Installation
- Die `main()` Funktion
- Das Qt Projekt
- Wir plotten mit `QPlotter` [E1]
- Was ist eine Schnittstelle ?
- wir basteln unser MatLab ... [E2]
- wir rechnen eine stationäre Diffusion (Abschn. 1.5.3)

Was ist eine Schnittstelle ?

```
#include <QApplication>
#include "plotter.h"

int main(int argc, char *argv[])
{
  QApplication app(argc, argv);
  // Data
  int numPoints = 100;
  QVector<QPointF> points0;
  QVector<QPointF> points1;
  for (int x = 0; x < numPoints; ++x) {
    points0.append(QPointF(x, uint(qrand()) % 100));
    points1.append(QPointF(x, uint(qrand()) % 100));
  }
  // Plotter
  Plotter plotter;
  plotter.setWindowTitle(QObject::tr("Jambi Plotter"));
  plotter.setCurveData(0, points0);
  plotter.setCurveData(1, points1);
  PlotSettings settings;
  settings.minX = 0.0;
  settings.maxX = 100.0;
  settings.minY = 0.0;
  settings.maxY = 100.0;
  plotter.setPlotSettings(settings);
  plotter.show();
  //
  return app.exec();
}
```

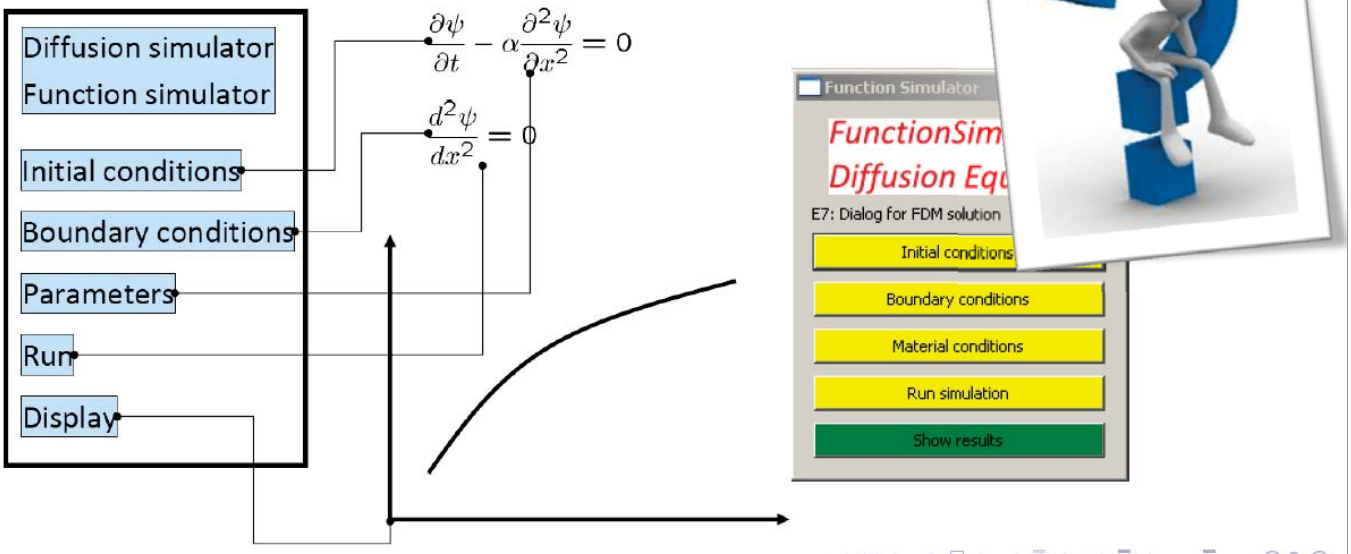
Wikipedia: Die Schnittstelle oder das Interface, englisch für Grenzfläche) ist der Teil eines Systems, welcher der Kommunikation dient.

Die Dateien plotter.h und plotter-cpp enthalten den Qt-Code für die Benutzer-schnittstelle (ui steht für engl. user interface).

An application programming interface (API) is a specification intended to be used as an interface by software components to communicate with each other.

Wir basteln unser MatLab ...

Wir haben gesehen, wie unkompliziert ist, einfache Plots mit Qt zu erzeugen ("Matlab" mit Qt). Dabei haben wir den Qt x-y Plotter in unsere Anwendung eingebunden. Nun wollen wir aber nicht irgendwelche Bildchens produzieren, sondern unsere Anwendungsbeispiele bearbeiten.



Der Fahrplan für heute

- Siehe Skript Kapitel 5
- Die Qt Installation
- Die `main()` Funktion
- Das Qt Projekt
- Wir plotten mit `QPlotter` [E1]
- Was ist eine Schnittstelle ?
- wir basteln unser MatLab ... [E2]
- wir rechnen eine stationäre Diffusion (Abschn. 1.5.3)

?s???iffusion
Fick'sche Gesetze

Erstes Fick'sches Gesetz

Nach dem Ersten Fick'schen Gesetz ist die Teilchenstromdichte (Fluss) J (mol m^{-2} s^{-1}) proportional zum Konzentrationsgradienten entgegen der Diffusionsrichtung $\partial c/\partial x$ (mol·m^{-4}). Die Proportionalitätskonstante ist der Diffusionskoeffizient D (m^2 s^{-1}).

$$J = -D\frac{\partial c}{\partial x}$$

Die Teilchenstromdichte macht eine quantitative Aussage über die (im statistischen Mittel) gerichtete Bewegung von Teilchen, d. h. wie viele Teilchen einer Stoffmenge sich pro Zeiteinheit durch eine Flächeneinheit, die senkrecht zur Diffusionsrichtung liegt, netto bewegen. Die angegebene Gleichung gilt auch für den allgemeinen Fall, dass der Diffusionskoeffizient nicht konstant ist, sondern von der Konzentration abhängt (das ist aber streng genommen nicht mehr die Aussage des Ersten Fick'schen Gesetzes).

Mit Hilfe der Kontinuitätsgleichung (Massenerhaltung)

$$\frac{\partial c}{\partial t} = -\frac{\partial J}{\partial x}$$

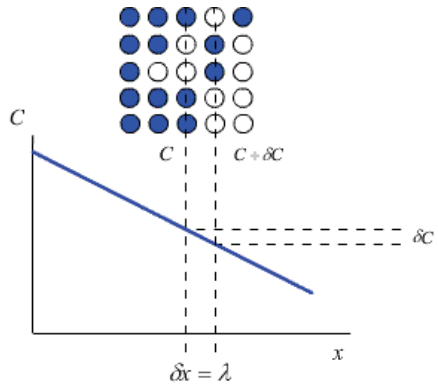
ergibt sich aus dem Ersten Fick'schen Gesetz die Diffusionsgleichung

$$\frac{\partial c}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial c}{\partial x}\right),$$

für konstante Diffusionskoeffizienten ergibt sich hieraus

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}$$

Sie stellt eine Beziehung zwischen zeitlichen und örtlichen Konzentrationsunterschieden dar und eignet sich somit zur Darstellung instationärer Diffusion, im Gegensatz zum 1. Fick'schen Gesetz, das einen zeitlich konstanten Diffusionsfluss beschreibt.



3.2 Diffusion Equation

Typical diffusion problems are mass diffusion, heat conduction, and viscous fluid flow. The diffusion equation is a parabolic PDE.

$$\frac{\partial u}{\partial t} - \alpha\frac{\partial^2 u}{\partial x^2} = 0 \tag{3.13}$$

The unknown field variable u may be interpreted as velocity, vorticity, temperature or concentration depending on whether the diffusion of momentum, vorticity, heat or mass is being considered. Boundary as well as initial conditions must be specified to obtain an unique solution of such a PDE.

In this chapter we introduce several (explicit and implicit) finite difference schemes for the numerical solution of the diffusion equation (Fig. 3.2) Attention will be paid to stability and accuracy of the several algorithms. Moreover, the implementation of initial as well as boundary conditions will also be considered.

Analysis of approximation schemes consists of three steps:

- Develop the **algebraic scheme**,
- Check **consistency** of the algebraic approximate equation,
- Investigate **stability** behavior of the scheme.

1.5.3 Elliptic Equations

Elliptic PDEs are related to equilibrium and steady-state problems, e.g. steady state temperature distribution in a rod of solid material, equilibrium stress of a solid under a given load or irrotational flow of an incompressible fluid as well as potential flow.

Example: Steady state heat conduction (1-D)

A very simple example of an equilibrium problem is steady state heat conduction ($\psi = T$) in an insulated rod whose ends are kept at different constant temperatures. This problem is governed by the following equation

$$\frac{d^2\psi}{dx^2} = 0 \tag{1.61}$$

The solution will be a linear temperature distribution (Fig. 1.6).

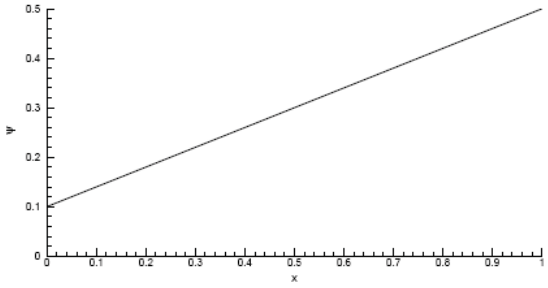


Abbildung 1.6: Solution of an elliptic equation

Steady Diffusion

Die (quellenfreie) stationäre Diffusionsgleichung hat im eindimensionalen Fall

$$\frac{d^2y}{dx^2} = 0 \tag{1}$$

Steady Diffusion

Die (quellenfreie) stationäre Diffusionsgleichung hat im eindimensionalen Fall

$$\frac{d^2y}{dx^2} = 0 \tag{1}$$

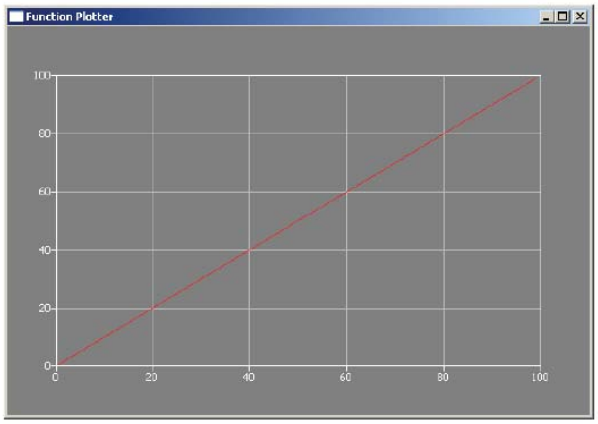
ein ziemlich triviale Lösung, eine lineare Funktion.

$$y = ax + b \tag{2}$$

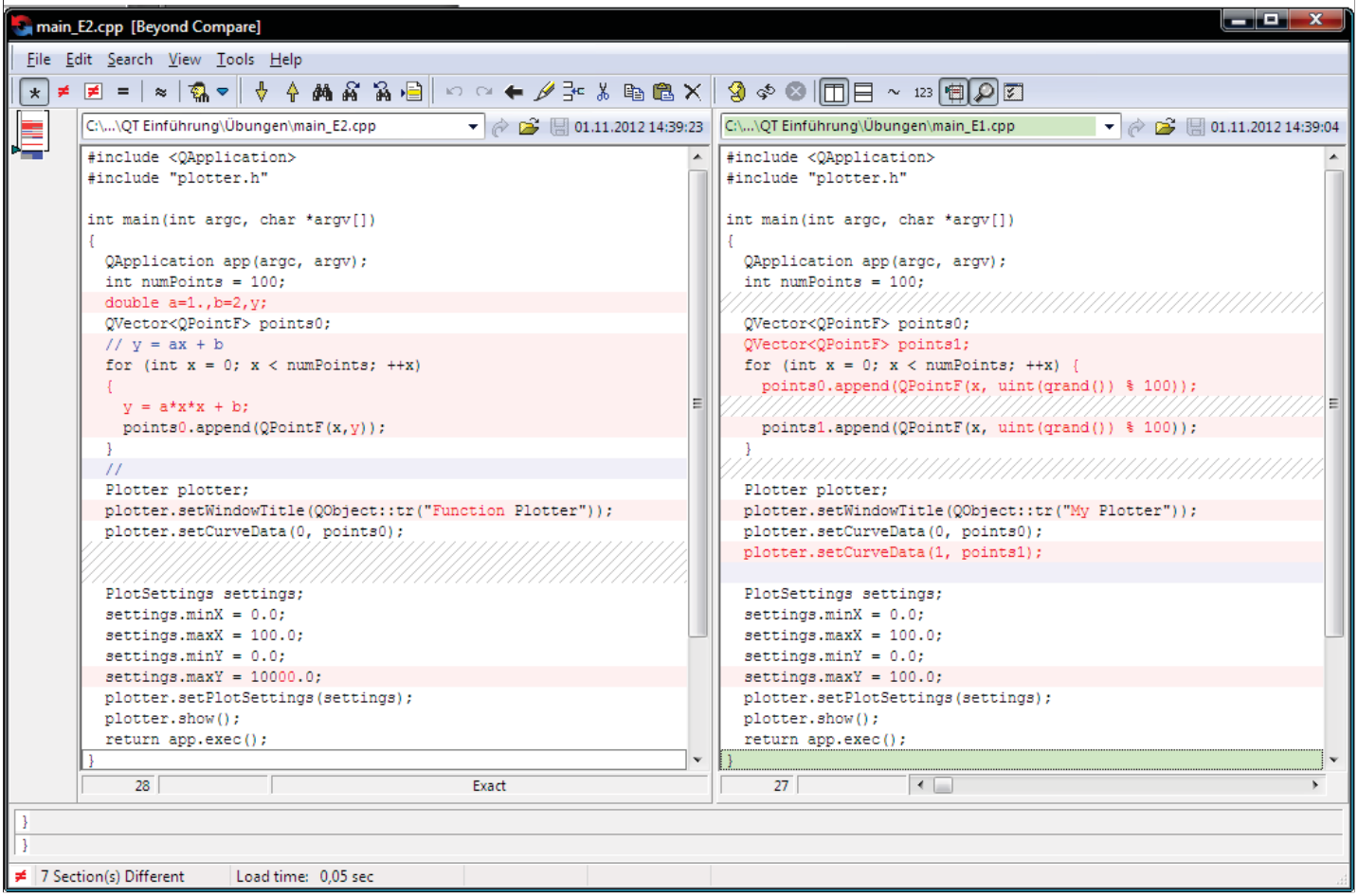
Wir wollen nun diese Funktion in unseren "function viewer" implementieren.

Steady Diffusion

```
double a=1.,b=0.,y;
// y = ax + b
for (int x = 0; x < numPoints; ++x)
{
  y = a*x + b;
  points0.append(QPointF(x,y));
}
```



Vergleich



5.2 Function Viewer

Wir haben gesehen, wie unkompliziert ist, einfache Plots mit Qt zu erzeugen (*"Matlab" mit Qt*). Dabei haben wir den Qt x-y Plotter in unsere Anwendung *eingebunden*. Nun wollen wir aber nicht irgendwelche Bildchens produzieren, *sondern unsere Anwendungsbeispiele bearbeiten*.

5.2.1 Steady diffusion

Die (quellenfreie) stationäre Diffusionsgleichung hat im eindimensionalen Fall

$$\frac{d^2y}{dx^2} = 0 \tag{5.1}$$

ein ziemlich triviale Lösung, eine lineare Funktion.

$$y = ax + b \tag{5.2}$$

Wir wollen nun diese Funktion in unseren "function viewer" implementieren.

Übung: E5.2.1

```
double a=1.,b=0.,y;
// y = ax + b
for (int x = 0; x < numPoints; ++x)
{
  y = a*x + b;
  points0.append(QPointF(x,y));
}
```

Steady Diffusion

Das war einfach. Nun müssen wir uns Gedanken über die physikalische Bedeutung der Koeffizienten a und b machen. Dies ist ihre 3. HW.

HW3: Berechnen sie die Koeffizienten a und b für Wärmediffusion, wenn die linke Temperatur $y = 283$ K und die rechte Temperatur $y = 300$ K sind.