BERGISCHE UNIVERSITÄT
GESAMTHOCHSCHULE WUPPERTAL
GAUSS-STRASSE 20
42097 WUPPERTAL
(Korrespondenzanschrift)
42119 WUPPERTAL
(Lieferanschrift)
TELEX 8 592 262 bughw
TELEFAX (0202) 439-2901
TELEFON (0202) 439-1

Fachbereich 7
MATHEMATIK

Prof. Dr. Hans-Jürgen Buhl
Praktische Informatik / Numerik

e-mail: Juergen.Buhl@math.uni-wuppertal.de

Grundzüge der objektorientierten Programmierung

WS2001/2002 – optionales Übungsblatt

Abgabetermin: 18. Februar 2002

Hinweis zur GUI-Bibliothek Qt:

Eine übersetzte Version der Qt-Bibliothek steht auf den CIP-Rechnern für den GNU-Compiler `g++` zur Verfügung. Zum Linken der Bibliothek ist `-lqt` anzugeben. Eventuell muß vorher der *Meta-Objekt-Compiler* (moc) von Qt aufgerufen werden, dies ist jeweils in der Aufgabenstellung mit Aufrufsyntax angegeben. Auf den LINUX-Rechnern des Fachbereichs müssen Sie entweder mittels

```
g++ -o xxx -L$QTDIR/lib -I$QTDIR/include xxx.cpp -lqt
```

übersetzen oder können nach

```
export CPPFLAGS="-g -L$QTDIR/lib -I$QTDIR/include"
```

einfach `make xxx` nutzen.
Die Online-Dokumentation im HTML-Format zu Qt finden Sie auf den CIP-Rechnern unter
`file:/usr/local/Manuals/Libs/qt-2.2.0/index.html`,
ein einführendes Tutorial ist unter
`file:/usr/local/Manuals/Libs/qt-2.2.0/tutorial.html`
zu finden.
Weitere Informationen zur Qt-Bibliothek finden Sie im WWW auf den Seiten der Firma Trolltech `http://www.trolltech.com/`.

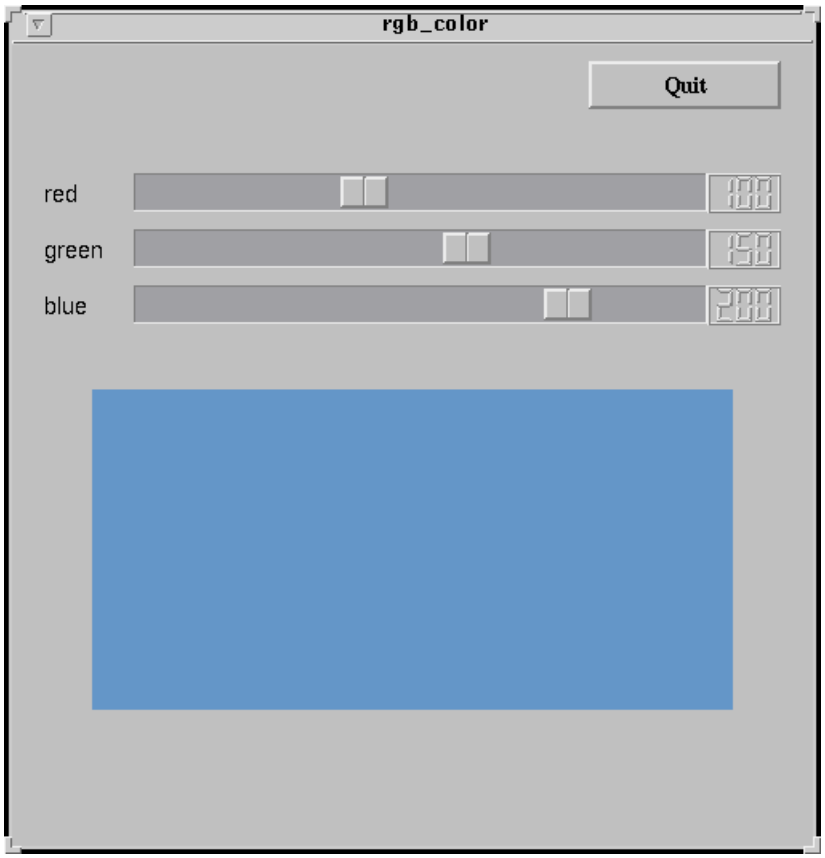
Aufgabe 1. *Farbpalette, 2 Punkte*
Mit dem Programm `rgb_color` können Sie Farben nach dem Rot-Blau-Grün-Farbschema mischen und anzeigen lassen.
Eine übersetzte ausführbare Version des Programms finden Sie unter http://www.math.uni-wuppertal.de/~buhl/oop/Farbpalette. Dort steht auch der Quellcode `rgb_color.cc` zu diesem Programm.

Verdeutlichen Sie sich die Funktionsweise des Programms. Schlagen Sie hierbei die verwendeten Klassen und Elementfunktionen in der *API-Reference* der Online-Reference-Documentation von Qt nach. Ergänzen Sie dann mit

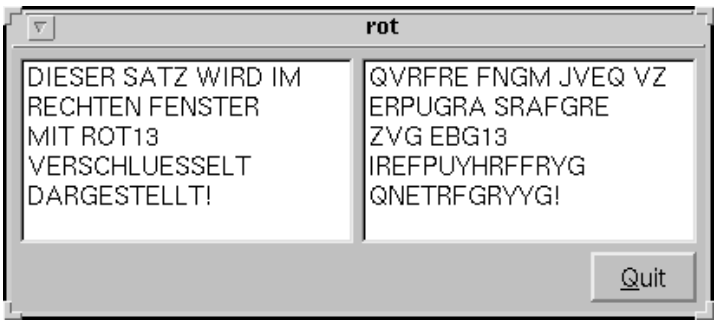
Hilfe der Online-Dokumentation alle noch fehlenden Deklarationen und Definitionen, indem Sie die entsprechenden Headerfiles von Qt mittels `#include` einbinden.
Übersetzen Sie das Programm mit

```
moc -o rgb_color.moc rgb_color.cc
g++ -o rgb_color rgb_color.cc -lqt
```

und testen Sie es anschließend.



Aufgabe 2. *GUI zur Verschlüsselung mit rot13, 6 Punkte*
Im Übungsblatt 4 haben Sie in Aufgabe 2 das einfache Kodierungsverfahren **rot13** kennengelernt. Erstellen Sie nun mit Hilfe der Qt-Bibliothek ein Programm mit einer graphischen Bedienungsoberfläche zur Kodierung bzw. Dekodierung eines mehrzeiligen Textes.
Das Programmfenster soll dabei aus einem linken und einem rechten Textfenster bestehen. Im linken Textfenster soll ein (mehrzeiliger) Text mit üblichen Editierfunktionen (wie z.B. Löschen eines Zeichens) eingegeben werden können. Der kodierte bzw. dekodierte Text soll simultan, d.h. Zeichen für Zeichen im rechten Fenster angezeigt werden. Zum Beenden des Programms soll ein *Quit-Button* vorhanden sein. Sie können sich dabei an folgendem Bildschirmphoto orientieren:



Bei Anwendung der nachfolgenden Hinweise können Sie ein Programm schreiben, das (bei sinnvoller Formatierung und Gliederung!) auf einer Seite ausgedruckt werden kann (d.h. maximal 60-70 Programmzeilen):

- Als mehrzeilige Textfenster mit Editierfunktionen können Objekte der Klasse `QMultiLineEdit` verwendet werden. Soll ein Textfenster nur zur Ausgabe von Text, d.h. nicht zur Eingabe benutzt werden, so kann dies mit der Elementfunktion `setReadOnly` eingestellt werden.

- Leiten Sie von der Qt-Basisklasse `QWidget` eine neue Klasse `Rot13` ab, die einen öffentlichen Konstruktor, einen privaten Slot `changeRight()`, die (De-)Kodierungsfunktion `rot13` als private Elementfunktion sowie zwei Zeiger `left` und `right` auf Objekte der Klasse `QMultiLineEdit` als private Datenmember besitzt.

- Im Konstruktor werden zwei Textfenster erzeugt, die Datenmember `left` und `right` zeigen auf diese Textfenster. Ein Textfenster wird zur Eingabe, ein Textfenster wird nur zur Ausgabe von Text benutzt.

  Das Signal `textChanged()` des Eingabetextfensters wird mit dem Slot `changeRight()` verbunden.

  Der *Quit*-Button wird angelegt und das Signal `clicked()` wird mit dem Slot `quit()` der Anwendung `qApp` verbunden.

  Weiter wird auch die Anordnung der einzelnen graphischen Bedienelemente der Bedienungsoberfläche, d.h. der beiden Textfenster und des *Quit*-Buttons festgelegt. Hierzu kann ein Objekt der Klasse `QGridLayout` genutzt werden.

  Mit Hilfe der Elementfunktion `setFocus()` kann das Eingabetextfenster als aktives Fenster gesetzt werden.

- Die parameterlose Funktion `changeRight()` ist als Slot der Klasse `Rot13` deklariert. Beim Aufruf wird der Text im Eingabetextfenster mittels der Elementfunktion `text` abgefragt, mit Hilfe der (De-)Kodierungsfunktion `rot13` umgewandelt und dann mit der Elementfunktion `setText()` in das Ausgabetextfenster geschrieben.

- Die (De-)Kodierungsfunktion `rot13` soll einen gesamten String umwandeln können. Verwenden Sie als String-Datentyp den Qt-Datentyp `QString`.

- Im Hauptprogramm `main` soll als Anwendung ein Objekt der Klasse `QApplication` angelegt werden. Ein Objekt der neuen Klasse `Rot13`

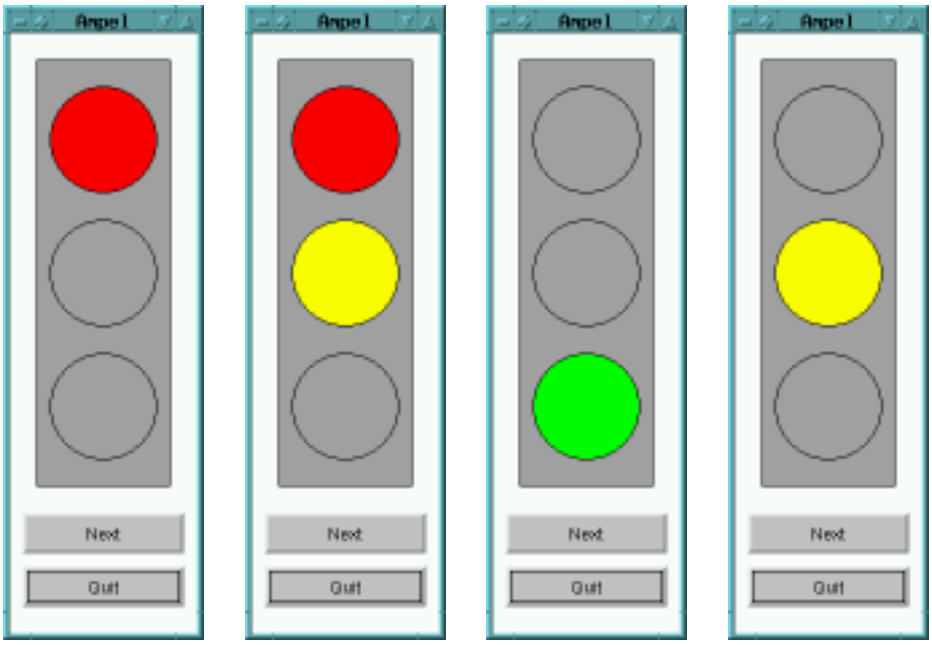
wird angelegt und mit der Elementfunktion `setMainWidget` als Hauptfenster der Anwendung festgelegt. Mit Hilfe der Elementfunktionen `resize` und `show` wird die Größe und die Sichtbarkeit festgelegt. Am Ende das Hauptprogramms wird die Anwendung gestartet.

- Ergänzen Sie die Klasse `rot13` um das Makro `Q_OBJECT` und binden Sie die vom *Meta-Objekt-Compiler* (moc) zu erzeugende Datei mit `#include "rot13.moc"` ein. Übersetzen und Linken des unter dem Namen `rot13.cc` abgespeicherten Quelltextes ist dann mit den beiden Aufrufen

```
moc -o rot13.moc rot13.cc
g++ -o rot13 rot13.cc -lqt
```

  möglich.

Aufgabe 3. *Ampelsimulation, 6 Punkte*
Erstellen Sie unter Verwendung der Qt-Graphikbibliothek ein Programm, das auf dem Bildschirm ein Fenster anzeigt, in dem eine Verkehrsampel simuliert wird. Die in Deutschland üblichen vier Ampelphasen (rot, rot-gelb, grün, gelb) sollen dabei korrekt angezeigt werden. Das Weiterschalten zur nächsten Ampelphase soll durch das Anklicken eines *Next*-Buttons mit der Maustaste ausgelöst werden. Nach dem Start des Programms soll zunächst die Ampelphase *rot* angezeigt werden. Das Fenster und damit das Programm soll mit Hilfe eines *Quit*-Buttons beendet werden können.



Empfehlungen zur Programmerstellung:

- Vollziehen Sie das unter

  http://www.math.uni-wuppertal.de/~buhl/oop/Warnlicht

  bereitgestellte Beispielprogramm `warnlicht.C` (gelb blinkendes Warnlicht) nach. Übersetzen Sie dieses Programm mit

  ```
  moc -o warnlicht.moc warnlicht.C
  g++ -o warnlicht warnlicht.C -lqt
  ```

  und testen Sie die Funktionsweise dieses Programms.

- Nehmen Sie das Programm `warnlicht.C` als Vorlage für Ihr Programm zur Simulation einer Verkehrsampel.

  Ändern Sie hierbei die Elementfunktion `nextphase()` der Klasse `myWidget` so ab, dass vier Ampelphasen berücksichtigt werden.

  Ergänzen Sie die eigentliche Zeichenfunktion `paintEvent` um das Zeichnen einer unteren grünen Lampe (Farbe: Je nach Ampelphase `green` oder `gray`) und einer oberen roten Lampe (Farbe: Je nach Ampelphase `red` oder `gray`). Die mittlere gelbe Lampe soll je nach Ampelphase die Farbe `yellow` oder `gray` anzeigen.

  Passen Sie die Größe des äußeren Rechtecks geeignet an (Elementfunktion `drawRect` des Objekts `paint`).

  Ändern Sie die Größe des Hauptfensters im Hauptprogramm `main` ab (Elementfunktion `resize` des Objekts `wid`).

- Übersetzen und testen Sie das abgeänderte Programm.

Aufgabe 4. *Das Spiel Tetrix, 6 Punkte*
Unter http://www.math.uni-wuppertal.de/~buhl/oop/Tetrix finden Sie die Headerdateien

`gtetrix.h` `qdragapp.h` `qtetrix.h` `qtetrixb.h` `tpiece.h`

die Programmdateien

`gtetrix.cpp` `qdragapp.cpp` `qtetrix.cpp` `qtetrixb.cpp` `tpiece.cpp` `tetrix.cpp`

und ein `Makefile` für eine Implementierung des Spiels Tetris unter Verwendung der C++-Klassenbibliothek Qt.

- Machen Sie sich mit dem Quellcode vertraut und übersetzen Sie das Programm mit Hilfe des Kommandos `make`.

- Ersetzen Sie die englischsprachigen Ausgabetexte (wie z.B. Next, Level, Score, Lines Removed, New Game usw.) im Programm durch entsprechende deutsche Texte.

- Ändern Sie die Farbe der Spielelemente ab. Sie müssen hierzu in der Programmdatei `qtetrixb.cpp` neue Farbwerte im RGB-Schema den Elementen des Feldes `colors` zuweisen. Um leuchtendere Farben zu bekommen können Sie z.B. die Farbtripel $(255, 0, 0)$, $(0, 255, 0)$, $(0, 0, 255)$, $(255, 255, 0)$, $(255, 0, 255)$, $(0, 255, 255)$ und $(255, 255, 255)$ ausprobieren.