프로그람개발문고

Qt실례프로그람

교육위원회 교육정보쎈터
주체99(2010)

차 례

머리말

위대한 령도자 **김정일**동지께서는 다음과 같이 지적하시였다.

《프로그람을 개발하는데서 기본은 우리 식의 프로그람을 개발하는것입니다. 우리는 우리 식의 프로그람을 개발하는 방향으로 나가야 합니다.》 (《**김정일**선집》 제15권, 196페지)

위대한 령도자 **김정일**동지의 현명한 령도에 의하여 오늘 우리 나라에서는 프로그람기술이 빠른 속도로 발전하고있다.

우리의 과학자, 연구사들은 우리 식 조작체계 《붉은별》을 개발하였으며 각종 도구들과 응용프로그람들을 개발하기 위한 연구사업을 활발히 진행하고있다.

우리는 정보공학을 전공하는 교원, 연구사들과 대학생들이 프로그람개발도구인 Qt에 의하여 프로그람을 능숙하게 작성할수 있도록 하기 위하여 《Qt일반지식》과 《Qt프로그람개발법》, 《Qt프로그람개발도구》, 《Qt실례프로그람》을 출판한다.

《Qt실례프로그람》에서는 Qt로 작성한 많은 례제프로그람들을 서술한다.

우리는 Qt의 일반원리와 프로그람작성법을 습득하여 우리 식의 조작체계에서 실행할수 있는 프로그람들을 더 많이 개발함으로써 나라의 프로그람기술을 한계단 더 발전시키고 인민경제의 정보화를 실현하는데 적극 이바지하여야 한다.

1. 상사시계

이 실례는 상사시계창문부품을 보여준다. 사용자정의창문부품(QWidget subclass)의 창조방법과 QTimer에 의하여 시계를 창조하는 방법을 보여준다.

aclock.pro

```
TEMPLATE  = app
TARGET    = aclock
CONFIG    += qt warn_on release
HEADERS      = aclock.h
SOURCES      = aclock.cpp \
          main.cpp
```

main.cpp

```
#include "aclock.h"
#include <qapplication.h>

int main( int argc, char **argv )
{
   QApplication a( argc, argv );
   AnalogClock *clock = new AnalogClock;
   if ( argc == 2 && strcmp( argv[1], "-transparent" ) == 0 )
    clock->setAutoMask( TRUE );
   clock->resize( 100, 100 );
   a.setMainWidget( clock );
   clock->setCaption("Qt Example - Analog Clock");
   clock->show();
   int result = a.exec();
   delete clock;
   return result;
}
```

aclock.cpp

```
#include "aclock.h"
#include <qtimer.h>
#include <qpainter.h>
#include <qbitmap.h>
// Constructs an analog clock widget that uses an internal QTimer.

AnalogClock::AnalogClock( QWidget *parent, const char *name )
   : QWidget( parent, name )
{
   time = QTime::currentTime();          // get current time
   internalTimer = new QTimer( this );// create internal timer
   connect( internalTimer, SIGNAL(timeout()), SLOT(timeout()) );
   internalTimer->start( 5000 );      // emit signal every 5 seconds
}

void AnalogClock::mousePressEvent( QMouseEvent *e )
{
   if(isTopLevel())
    clickPos = e->pos() + QPoint(geometry().topLeft() - frameGeometry().topLeft());
}
```

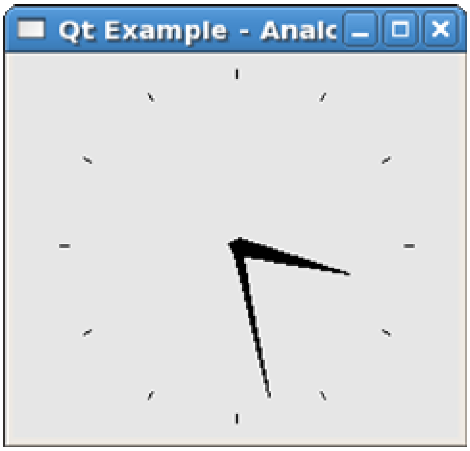
```
void AnalogClock::mouseMoveEvent( QMouseEvent *e )
{
   if(isTopLevel())
     move( e->globalPos() - clickPos );
}

// The QTimer::timeout() signal is received by this slot.
// When we set an explicit time we don't want the timeout() slot to be
// called anymore as this relies on currentTime()
void AnalogClock::setTime( const QTime & t )
{
   time = t;
   disconnect( internalTimer, SIGNAL(timeout()), this, SLOT(timeout()) );
   if (autoMask())
     updateMask();
   else
     update();
}

void AnalogClock::timeout()
{
   QTime old_time = time;
   time = QTime::currentTime();
   if ( old_time.minute() != time.minute()
     || old_time.hour() != time.hour() ) {// minute or hour has changed
     if (autoMask())
        updateMask();
     else
        update();
   }
}

void AnalogClock::paintEvent( QPaintEvent * )
{
   if ( autoMask() )
     return;
   QPainter paint( this );
   paint.setBrush( colorGroup().foreground() );
   drawClock( &paint );
}

// If the clock is transparent, we use updateMask() instead of paintEvent()
void AnalogClock::updateMask()   // paint clock mask
{
   QBitmap bm( size() );
   bm.fill( color0 );            //transparent

   QPainter paint;
   paint.begin( &bm, this );
   paint.setBrush( color1 );       // use non-transparent color
   paint.setPen( color1 );

   drawClock( &paint );
   paint.end();
   setMask( bm );
}

// The clock is painted using a 1000x1000 square coordinate system, in
// the a centered square, as big as possible.  The painter's pen and brush colors are used.
void AnalogClock::drawClock( QPainter *paint )
{
   paint->save();

   paint->setWindow( -500,-500, 1000,1000 );

   QRect v = paint->viewport();
   int d = QMIN( v.width(), v.height() );
   paint->setViewport( v.left() + (v.width()-d)/2,
             v.top() + (v.height()-d)/2, d, d );

   QPointArray pts;

   paint->save();
   paint->rotate( 30*(time.hour()%12-3) + time.minute()/2 );
   pts.setPoints( 4, -20,0,  0,-20, 300,0, 0,20 );
   paint->drawConvexPolygon( pts );
   paint->restore();

   paint->save();
   paint->rotate( (time.minute()-15)*6 );
   pts.setPoints( 4, -10,0, 0,-10, 400,0, 0,10 );
   paint->drawConvexPolygon( pts );
   paint->restore();

   for ( int i=0; i<12; i++ ) {
    paint->drawLine( 440,0, 460,0 );
    paint->rotate( 30 );
   }

   paint->restore();
}

void AnalogClock::setAutoMask(bool b)
{
   if (b)
    setBackgroundMode( PaletteForeground );
   else
    setBackgroundMode( PaletteBackground );
   QWidget::setAutoMask(b);
}
```

aclock.h

```
#ifndef ACLOCK_H
#define ACLOCK_H
#include <qwidget.h>
#include <qdatetime.h>
```

```
class QTimer;
class AnalogClock : public QWidget        // analog clock widget
{
   Q_OBJECT
public:
   AnalogClock( QWidget *parent=0, const char *name=0 );
   void setAutoMask(bool b);

protected:
   void updateMask();
   void paintEvent( QPaintEvent *);
   void mousePressEvent( QMouseEvent *);
   void mouseMoveEvent( QMouseEvent *);
   void drawClock( QPainter* );

private slots:
   void timeout();

public slots:
   void setTime( const QTime & t );

private:
   QPoint clickPos;
   QTime time;
   QTimer *internalTimer;
};

#endif // ACLOCK_H
```

실행



2. 작용

1) 작용을 가지는 완전한 응용프로그람창문

QAction클라스는 각이한 사용자대면부요소들로부터의 사용자입력을 추상적인 고수준작용들과 결합하는 방법을 제공한다. 이 수법은 응용프로그람들을 사용자들의 각이한 요구에 맞게 전용화하기 쉽게 한다.

이 실례프로그람은 응용프로그람실례와 같지만 QAction을 사용하여 차림표와 도구띠를 만든다.

application.pro

```
TEMPLATE  = app
TARGET      = action
CONFIG      += qt warn_on release
HEADERS        = application.h
SOURCES        = application.cpp \
            main.cpp
```

application.cpp

```
#include "application.h"
#include <qimage.h>
#include <qpixmap.h>
#include <qtoolbar.h>
#include <qtoolbutton.h>
#include <qpopupmenu.h>
#include <qmenubar.h>
#include <qtextedit.h>
#include <qfile.h>
#include <qfiledialog.h>
#include <qstatusbar.h>
#include <qmessagebox.h>
#include <qprinter.h>
#include <qapplication.h>
#include <qaccel.h>
#include <qtextstream.h>
#include <qpainter.h>
#include <qpaintdevicemetrics.h>
#include <qwhatsthis.h>
#include <qaction.h>
#include <qsimplerichtext.h>

#include "filesave.xpm"
#include "fileopen.xpm"
#include "fileprint.xpm"

ApplicationWindow::ApplicationWindow()
    : QMainWindow( 0, "example application main window", WDestructiveClose )
{
    printer = new QPrinter( QPrinter::HighResolution );

    QAction * fileNewAction;
    QAction * fileOpenAction;
    QAction * fileSaveAction, * fileSaveAsAction, * filePrintAction;
    QAction * fileCloseAction, * fileQuitAction;

    fileNewAction = new QAction( "&New", CTRL+Key_N, this, "new" );
    connect( fileNewAction, SIGNAL( activated() ) , this,  SLOT( newDoc() ) );

    fileOpenAction = new QAction( QPixmap( fileopen ), "&Open...",  CTRL+Key_O, this, "open" );
    connect( fileOpenAction, SIGNAL( activated() ) , this, SLOT( choose() ) );

    const char * fileOpenText = "<p><img source=\"fileopen\"> "
```

```
            "Click this button to open a <em>new file</em>. <br>"
            "You can also select the <b>Open</b> command "
            "from the <b>File</b> menu.</p>";
QMimeSourceFactory::defaultFactory()->setPixmap( "fileopen",
                fileOpenAction->iconSet().pixmap() );
fileOpenAction->setWhatsThis( fileOpenText );

fileSaveAction = new QAction( QPixmap( filesave ),
                    "&Save", CTRL+Key_S, this, "save" );
connect( fileSaveAction, SIGNAL( activated() ) , this, SLOT( save() ) );

const char * fileSaveText = "<p>Click this button to save the file you "
            "are editing. You will be prompted for a file name.\n"
            "You can also select the <b>Save</b> command "
            "from the <b>File</b> menu.</p>";
fileSaveAction->setWhatsThis( fileSaveText );

fileSaveAsAction = new QAction( "Save File As", "Save &As...", 0,  this,  "save as" );
connect( fileSaveAsAction, SIGNAL( activated() ) , this,  SLOT( saveAs() ) );
fileSaveAsAction->setWhatsThis( fileSaveText );

filePrintAction = new QAction( "Print File", QPixmap( fileprint ),
                    "&Print...", CTRL+Key_P, this, "print" );
connect( filePrintAction, SIGNAL( activated() ) , this,  SLOT( print() ) );

const char * filePrintText = "Click this button to print the file you "
            "are editing.\n You can also select the Print "
            "command from the File menu.";
filePrintAction->setWhatsThis( filePrintText );

fileCloseAction = new QAction( "Close", "&Close", CTRL+Key_W, this, "close" );
connect( fileCloseAction, SIGNAL( activated() ) , this, SLOT( close() ) );

fileQuitAction = new QAction( "Quit", "&Quit", CTRL+Key_Q, this, "quit" );
connect( fileQuitAction, SIGNAL( activated() ) , qApp,  SLOT( closeAllWindows() ) );

// populate a tool bar with some actions

QToolBar * fileTools = new QToolBar( this, "file operations" );
fileTools->setLabel( "File Operations" );
fileOpenAction->addTo( fileTools );
fileSaveAction->addTo( fileTools );
filePrintAction->addTo( fileTools );
(void)QWhatsThis::whatsThisButton( fileTools );

// populate a menu with all actions

QPopupMenu * file = new QPopupMenu( this );
menuBar()->insertItem( "&File", file );
fileNewAction->addTo( file );
fileOpenAction->addTo( file );
fileSaveAction->addTo( file );
fileSaveAsAction->addTo( file );
file->insertSeparator();
```

```
    filePrintAction->addTo( file );
    file->insertSeparator();
    fileCloseAction->addTo( file );
    fileQuitAction->addTo( file );

    menuBar()->insertSeparator();

    // add a help menu

    QPopupMenu * help = new QPopupMenu( this );
    menuBar()->insertItem( "&Help", help );
    help->insertItem( "&About", this, SLOT(about()), Key_F1 );
    help->insertItem( "About &Qt", this, SLOT(aboutQt()) );
    help->insertSeparator();
    help->insertItem( "What's &This", this, SLOT(whatsThis()), SHIFT+Key_F1 );

    // create and define the central widget

    e = new QTextEdit( this, "editor" );
    e->setFocus();
    setCentralWidget( e );
    statusBar()->message( "Ready", 2000 );

    resize( 450, 600 );
}

ApplicationWindow::~ApplicationWindow()
{
    delete printer;
}

void ApplicationWindow::newDoc()
{
    ApplicationWindow *ed = new ApplicationWindow;
    ed->show();
}

void ApplicationWindow::choose()
{
    QString fn = QFileDialog::getOpenFileName( QString::null, QString::null,  this);
    if ( !fn.isEmpty() )
      load( fn );
    else
      statusBar()->message( "Loading aborted", 2000 );
}

void ApplicationWindow::load( const QString &fileName )
{
    QFile f( fileName );
    if ( !f.open( IO_ReadOnly ) )
      return;

    QTextStream ts( &f );
    e->setText( ts.read() );
    e->setModified( FALSE );
    setCaption( fileName );
    statusBar()->message( "Loaded document " + fileName, 2000 );
}

void ApplicationWindow::save()
{
    if ( filename.isEmpty() ) {
     saveAs();
     return;
    }

    QString text = e->text();
    QFile f( filename );
    if ( !f.open( IO_WriteOnly ) ) {
     statusBar()->message( QString("Could not write to %1").arg(filename),  2000 );
     return;
    }

    QTextStream t( &f );
    t << text;
    f.close();

    e->setModified( FALSE );
    setCaption( filename );
    statusBar()->message( QString( "File %1 saved" ).arg( filename ), 2000 );
}

void ApplicationWindow::saveAs()
{
    QString fn = QFileDialog::getSaveFileName( QString::null, QString::null,  this );
    if ( !fn.isEmpty() ) {
     filename = fn;
     save();
    } else {
     statusBar()->message( "Saving aborted", 2000 );
    }
}

void ApplicationWindow::print()
{
    printer->setFullPage( TRUE );
    if ( printer->setup(this) ) {          // printer dialog
     statusBar()->message( "Printing..." );
     QPainter p;
     if( !p.begin( printer ) ) {               // paint on printer
         statusBar()->message( "Printing aborted", 2000 );
         return;
     }

     QPaintDeviceMetrics metrics( p.device() );
     int dpiy = metrics.logicalDpiY();
     int margin = (int) ( (2/2.54)*dpiy ); // 2 cm margins
     QRect view( margin, margin, metrics.width() - 2*margin, metrics.height() - 2*margin );
    QSimpleRichText richText( QStyleSheet::convertFromPlainText(e->text()),
                  QFont(),
                  e->context(),
                  e->styleSheet(),
                  e->mimeSourceFactory(),
                  view.height() );
    richText.setWidth( &p, view.width() );
    int page = 1;
    do {
       richText.draw( &p, margin, margin, view, colorGroup() );
       view.moveBy( 0, view.height() );
       p.translate( 0 , -view.height() );
       p.drawText( view.right() - p.fontMetrics().width( QString::number( page ) ),
             view.bottom() + p.fontMetrics().ascent() + 5, QString::number( page ) );
       if ( view.top() - margin >= richText.height() )
         break;
       printer->newPage();
       page++;
    } while (TRUE);

    statusBar()->message( "Printing completed", 2000 );
   } else {
    statusBar()->message( "Printing aborted", 2000 );
   }
}

void ApplicationWindow::closeEvent( QCloseEvent* ce )
{
   if ( !e->isModified() ) {
    ce->accept();
    return;
   }

   switch( QMessageBox::information( this, "Qt Application Example",
                   "The document has been changed since "
                   "the last save.",
                   "Save Now", "Cancel", "Leave Anyway",
                   0, 1 ) ) {
   case 0:
    save();
    ce->accept();
    break;
   case 1:
   default: // just for sanity
    ce->ignore();
    break;
   case 2:
    ce->accept();
    break;
   }
}

void ApplicationWindow::about()
{
    QMessageBox::about( this, "Qt Application Example",
                "This example demonstrates simple use of "
                "QMainWindow,\nQMenuBar and QToolBar.");
}

void ApplicationWindow::aboutQt()
{
    QMessageBox::aboutQt( this, "Qt Application Example" );
}
```

application.h

```
#ifndef APPLICATION_H
#define APPLICATION_H

#include <qmainwindow.h>

class QTextEdit;

class ApplicationWindow: public QMainWindow
{
    Q_OBJECT

public:
    ApplicationWindow();
    ~ApplicationWindow();

protected:
    void closeEvent( QCloseEvent* );

private slots:
    void newDoc();
    void choose();
    void load( const QString &fileName );
    void save();
    void saveAs();
    void print();

    void about();
    void aboutQt();

private:
    QPrinter *printer;
    QTextEdit *e;
    QString filename;
};

#endif
```

2) 절환작용을 수행하는 간단한 실례

이 실례프로그람은 절환작용으로서 QAction의 사용을 구체화하여 보여준다.

main.cpp

```
#include <qapplication.h>
```

```
#include "application.h"

int main( int argc, char ** argv ) {
   QApplication a( argc, argv );
   ApplicationWindow * mw = new ApplicationWindow();
   mw->setCaption( "Document 1" );
   mw->show();
   a.connect( &a, SIGNAL(lastWindowClosed()), &a, SLOT(quit()) );
   return a.exec();
}
```

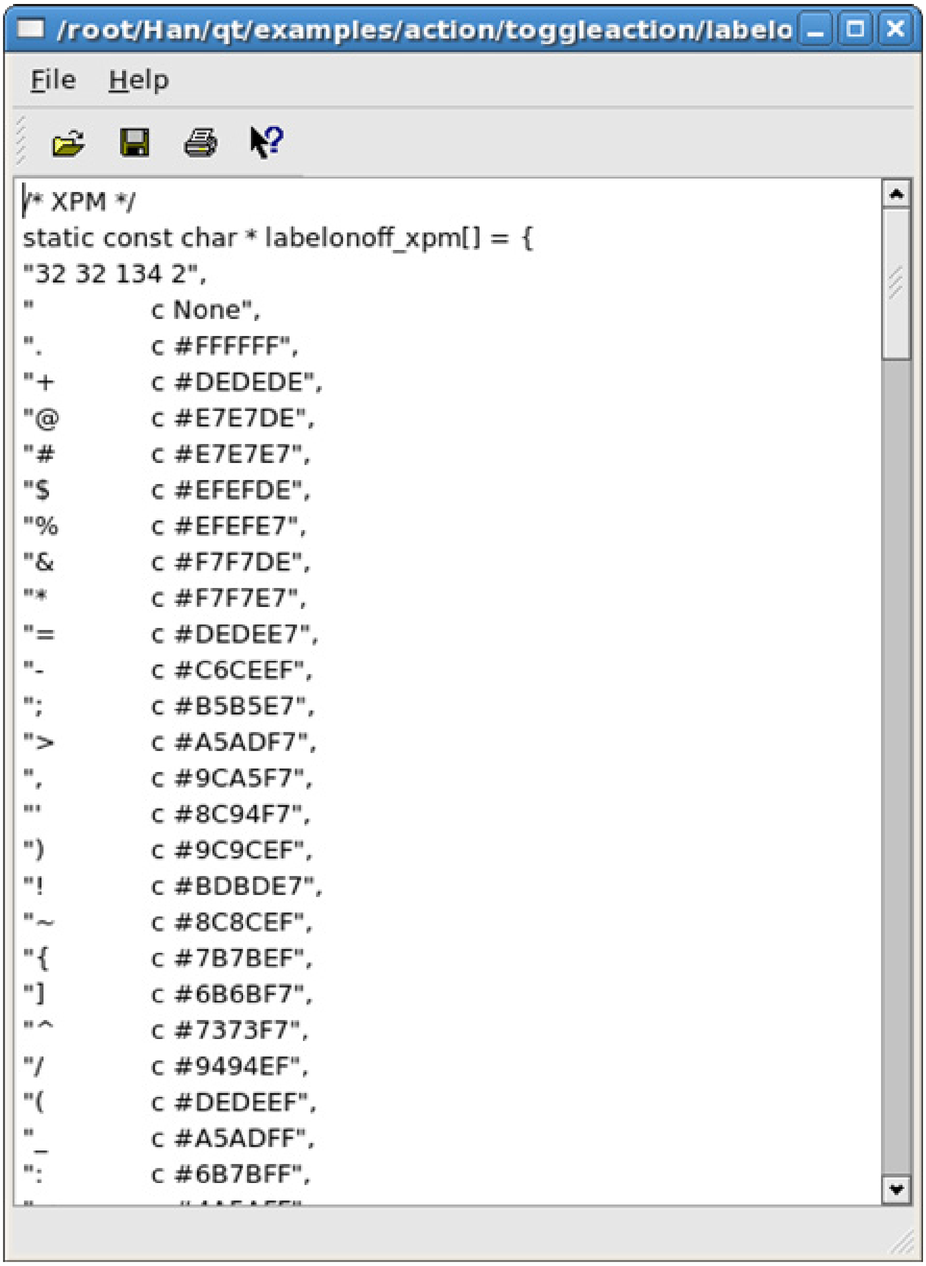
toggleaction/toggleaction.pro

```
TEMPLATE  = app
TARGET      = toggleaction
CONFIG      += qt warn_on release
HEADERS        =
SOURCES        = toggleaction.cpp
```

toggleaction/toggleaction.cpp

```
#include <qapplication.h>
#include <qmainwindow.h>
#include <qtoolbar.h>
#include <qaction.h>
#include "labelonoff.xpm"

int main( int argc, char **argv )
{
   QApplication app( argc, argv );
   QMainWindow * window = new QMainWindow;
   window->setCaption("Qt Example - Toggleaction");
   QToolBar * toolbar = new QToolBar( window );

   QAction * labelonoffaction = new QAction( window, "labelonoff" );
   labelonoffaction->setToggleAction( TRUE );

   labelonoffaction->setText( "labels on/off" );
   labelonoffaction->setAccel( Qt::ALT+Qt::Key_L );
   labelonoffaction->setIconSet( (QPixmap) labelonoff_xpm );

   QObject::connect( labelonoffaction, SIGNAL( toggled( bool ) ),
                 window, SLOT( setUsesTextLabel( bool ) ) );

   labelonoffaction->addTo( toolbar );

   app.setMainWidget( window );
   window->show();
   return app.exec();
}
```

실행



```
/* XPM */
static const char * labelonoff_xpm[] = {
"32 32 134 2",
"           c None",
".          c #FFFFFF",
"+          c #DEDEDE",
"@          c #E7E7DE",
"#          c #E7E7E7",
"$          c #EFEFDE",
"%          c #EFEFE7",
"&          c #F7F7DE",
"*          c #F7F7E7",
"=          c #DEDEE7",
"-          c #C6CEEF",
";          c #B5B5E7",
">          c #A5ADF7",
",          c #9CA5F7",
"'          c #8C94F7",
")          c #9C9CEF",
"!          c #BDBDE7",
"~          c #8C8CEF",
"{          c #7B7BEF",
"]          c #6B6BF7",
"^          c #7373F7",
"/          c #9494EF",
"(          c #DEDEEF",
"_          c #A5ADFF",
":          c #6B7BFF",
```

3. 주소록

이 실례프로그람은 매우 간단한 주소록을 사용하는 실례이다.

addressbook.pro

```
TEMPLATE  = app
TARGET    = addressbook
CONFIG    += qt warn_on release
HEADERS       = centralwidget.h \
        mainwindow.h
SOURCES       = centralwidget.cpp \
        main.cpp \
        mainwindow.cpp
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "centralwidget.h"

#include <qtoolbar.h>
#include <qtoolbutton.h>
#include <qpopupmenu.h>
#include <qmenubar.h>
#include <qstatusbar.h>
#include <qapplication.h>
#include <qfiledialog.h>

ABMainWindow::ABMainWindow()
    : QMainWindow( 0, "example addressbook application" ),
      filename( QString::null )
{
    setupMenuBar();
    setupFileTools();
    setupStatusBar();
    setupCentralWidget();
}

ABMainWindow::~ABMainWindow()
{
}

void ABMainWindow::setupMenuBar()
{
    QPopupMenu *file = new QPopupMenu( this );
    menuBar()->insertItem( "&File", file );

    file->insertItem( "New", this, SLOT( fileNew() ), CTRL + Key_N );
    file->insertItem( QPixmap( "fileopen.xpm" ), "Open", this, SLOT( fileOpen() ), CTRL + Key_O );
    file->insertSeparator();
    file->insertItem( QPixmap( "filesave.xpm" ), "Save", this, SLOT( fileSave() ), CTRL + Key_S );
    file->insertItem( "Save As...", this, SLOT( fileSaveAs() ) );
    file->insertSeparator();
    file->insertItem( QPixmap( "fileprint.xpm" ), "Print...", this, SLOT( filePrint() ), CTRL + Key_P );
    file->insertSeparator();
    file->insertItem( "Close", this, SLOT( closeWindow() ), CTRL + Key_W );
    file->insertItem( "Quit", qApp, SLOT( quit() ), CTRL + Key_Q );
}

void ABMainWindow::setupFileTools()
{
    //fileTools = new QToolBar( this, "file operations" );
}

void ABMainWindow::setupStatusBar()
{
    //statusBar()->message( "Ready", 2000 );
}

void ABMainWindow::setupCentralWidget()
{
   view = new ABCentralWidget( this );
   setCentralWidget( view );
}

void ABMainWindow::closeWindow()
{
   close();
}

void ABMainWindow::fileNew()
{
}

void ABMainWindow::fileOpen()
{
   QString fn = QFileDialog::getOpenFileName( QString::null, QString::null, this );
   if ( !fn.isEmpty() ) {
      filename = fn;
      view->load( filename );
   }
}

void ABMainWindow::fileSave()
{
   if ( filename.isEmpty() ) {
      fileSaveAs();
      return;
   }

   view->save( filename );
}

void ABMainWindow::fileSaveAs()
{
   QString fn = QFileDialog::getSaveFileName( QString::null, QString::null, this );
   if ( !fn.isEmpty() ) {
      filename = fn;
      fileSave();
   }
}

void ABMainWindow::filePrint()
{
}
```

mainwindow.h

```
#ifndef AB_MAINWINDOW_H
#define AB_MAINWINDOW_H

#include <qmainwindow.h>
```

```
#include <qstring.h>

class QToolBar;
class QPopupMenu;
class ABCentralWidget;

class ABMainWindow: public QMainWindow
{
   Q_OBJECT

public:
   ABMainWindow();
   ~ABMainWindow();

protected slots:
   void fileNew();
   void fileOpen();
   void fileSave();
   void fileSaveAs();
   void filePrint();
   void closeWindow();

protected:
   void setupMenuBar();
   void setupFileTools();
   void setupStatusBar();
   void setupCentralWidget();

   QToolBar *fileTools;
   QString filename;
   ABCentralWidget *view;

};

#endif
```

centralwidget.cpp

```
#include "centralwidget.h"

#include <qtabwidget.h>
#include <qlistview.h>
#include <qlayout.h>
#include <qwidget.h>
#include <qlabel.h>
#include <qpushbutton.h>
#include <qlineedit.h>
#include <qlabel.h>
#include <qcheckbox.h>
#include <qfile.h>
#include <qtextstream.h>

ABCentralWidget::ABCentralWidget( QWidget *parent, const char *name )
   : QWidget( parent, name )
{
   mainGrid = new QGridLayout( this, 2, 1, 5, 5 );

   setupTabWidget();
   setupListView();

   mainGrid->setRowStretch( 0, 0 );
   mainGrid->setRowStretch( 1, 1 );
}

void ABCentralWidget::save( const QString &filename )
{
   if ( !listView->firstChild() )
       return;

   QFile f( filename );
   if ( !f.open( IO_WriteOnly ) )
       return;

   QTextStream t( &f );
   t.setEncoding(QTextStream::UnicodeUTF8);

   QListViewItemIterator it( listView );

   for ( ; it.current(); ++it )
       for ( unsigned int i = 0; i < 4; i++ )
           t << it.current()->text( i ) << "\n";

   f.close();
}

void ABCentralWidget::load( const QString &filename )
{
   listView->clear();

   QFile f( filename );
   if ( !f.open( IO_ReadOnly ) )
       return;

   QTextStream t( &f );
   t.setEncoding(QTextStream::UnicodeUTF8);

   while ( !t.atEnd() ) {
       QListViewItem *item = new QListViewItem( listView );
       for ( unsigned int i = 0; i < 4; i++ )
           item->setText( i, t.readLine() );
   }

   f.close();
}

void ABCentralWidget::setupTabWidget()
{
   tabWidget = new QTabWidget( this );
```

```
QWidget *input = new QWidget( tabWidget );
QGridLayout *grid1 = new QGridLayout( input, 2, 5, 5, 5 );

QLabel *liFirstName = new QLabel( "First &Name", input );
liFirstName->resize( liFirstName->sizeHint() );
grid1->addWidget( liFirstName, 0, 0 );

QLabel *liLastName = new QLabel( "&Last Name", input );
liLastName->resize( liLastName->sizeHint() );
grid1->addWidget( liLastName, 0, 1 );

QLabel *liAddress = new QLabel( "Add&ress", input );
liAddress->resize( liAddress->sizeHint() );
grid1->addWidget( liAddress, 0, 2 );

QLabel *liEMail = new QLabel( "&E-Mail", input );
liEMail->resize( liEMail->sizeHint() );
grid1->addWidget( liEMail, 0, 3 );

add = new QPushButton( "A&dd", input );
add->resize( add->sizeHint() );
grid1->addWidget( add, 0, 4 );
connect( add, SIGNAL( clicked() ), this, SLOT( addEntry() ) );

iFirstName = new QLineEdit( input );
iFirstName->resize( iFirstName->sizeHint() );
grid1->addWidget( iFirstName, 1, 0 );
liFirstName->setBuddy( iFirstName );

iLastName = new QLineEdit( input );
iLastName->resize( iLastName->sizeHint() );
grid1->addWidget( iLastName, 1, 1 );
liLastName->setBuddy( iLastName );

iAddress = new QLineEdit( input );
iAddress->resize( iAddress->sizeHint() );
grid1->addWidget( iAddress, 1, 2 );
liAddress->setBuddy( iAddress );

iEMail = new QLineEdit( input );
iEMail->resize( iEMail->sizeHint() );
grid1->addWidget( iEMail, 1, 3 );
liEMail->setBuddy( iEMail );

change = new QPushButton( "&Change", input );
change->resize( change->sizeHint() );
grid1->addWidget( change, 1, 4 );
connect( change, SIGNAL( clicked() ), this, SLOT( changeEntry() ) );

tabWidget->addTab( input, "&Add/Change Entry" );

// -------------------------------------

QWidget *search = new QWidget( this );
```

```
    QGridLayout *grid2 = new QGridLayout( search, 2, 5, 5, 5 );

    cFirstName = new QCheckBox( "First &Name", search );
    cFirstName->resize( cFirstName->sizeHint() );
    grid2->addWidget( cFirstName, 0, 0 );
    connect( cFirstName, SIGNAL( clicked() ), this, SLOT( toggleFirstName() ) );

    cLastName = new QCheckBox( "&Last Name", search );
    cLastName->resize( cLastName->sizeHint() );
    grid2->addWidget( cLastName, 0, 1 );
    connect( cLastName, SIGNAL( clicked() ), this, SLOT( toggleLastName() ) );

    cAddress = new QCheckBox( "Add&ress", search );
    cAddress->resize( cAddress->sizeHint() );
    grid2->addWidget( cAddress, 0, 2 );
    connect( cAddress, SIGNAL( clicked() ), this, SLOT( toggleAddress() ) );

    cEMail = new QCheckBox( "&E-Mail", search );
    cEMail->resize( cEMail->sizeHint() );
    grid2->addWidget( cEMail, 0, 3 );
    connect( cEMail, SIGNAL( clicked() ), this, SLOT( toggleEMail() ) );

    sFirstName = new QLineEdit( search );
    sFirstName->resize( sFirstName->sizeHint() );
    grid2->addWidget( sFirstName, 1, 0 );

    sLastName = new QLineEdit( search );
    sLastName->resize( sLastName->sizeHint() );
    grid2->addWidget( sLastName, 1, 1 );

    sAddress = new QLineEdit( search );
    sAddress->resize( sAddress->sizeHint() );
    grid2->addWidget( sAddress, 1, 2 );

    sEMail = new QLineEdit( search );
    sEMail->resize( sEMail->sizeHint() );
    grid2->addWidget( sEMail, 1, 3 );

    find = new QPushButton( "F&ind", search );
    find->resize( find->sizeHint() );
    grid2->addWidget( find, 1, 4 );
    connect( find, SIGNAL( clicked() ), this, SLOT( findEntries() ) );

    cFirstName->setChecked( TRUE );
    sFirstName->setEnabled( TRUE );
    sLastName->setEnabled( FALSE );
    sAddress->setEnabled( FALSE );
    sEMail->setEnabled( FALSE );

    tabWidget->addTab( search, "&Search" );

    mainGrid->addWidget( tabWidget, 0, 0 );
}
```

```
void ABCentralWidget::setupListView()
{
   listView = new QListView( this );
   listView->addColumn( "First Name" );
   listView->addColumn( "Last Name" );
   listView->addColumn( "Address" );
   listView->addColumn( "E-Mail" );

   listView->setSelectionMode( QListView::Single );

   connect( listView, SIGNAL( clicked( QListViewItem* ) ), this,
SLOT( itemSelected( QListViewItem* ) ) );

   mainGrid->addWidget( listView, 1, 0 );
   listView->setAllColumnsShowFocus( TRUE );
}

void ABCentralWidget::addEntry()
{
   if ( !iFirstName->text().isEmpty() || !iLastName->text().isEmpty() ||
        !iAddress->text().isEmpty() || !iEMail->text().isEmpty() ) {
       QListViewItem *item = new QListViewItem( listView );
       item->setText( 0, iFirstName->text() );
       item->setText( 1, iLastName->text() );
       item->setText( 2, iAddress->text() );
       item->setText( 3, iEMail->text() );
   }

   iFirstName->setText( "" );
   iLastName->setText( "" );
   iAddress->setText( "" );
   iEMail->setText( "" );
}

void ABCentralWidget::changeEntry()
{
   QListViewItem *item = listView->currentItem();

   if ( item &&
        ( !iFirstName->text().isEmpty() || !iLastName->text().isEmpty() ||
          !iAddress->text().isEmpty() || !iEMail->text().isEmpty() ) ) {
       item->setText( 0, iFirstName->text() );
       item->setText( 1, iLastName->text() );
       item->setText( 2, iAddress->text() );
       item->setText( 3, iEMail->text() );
   }
}

void ABCentralWidget::selectionChanged()
{
   iFirstName->setText( "" );
   iLastName->setText( "" );
   iAddress->setText( "" );
   iEMail->setText( "" );
```

```
}

void ABCentralWidget::itemSelected( QListViewItem *item )
{
    if ( !item )
      return;
    item->setSelected( TRUE );
    item->repaint();

    iFirstName->setText( item->text( 0 ) );
    iLastName->setText( item->text( 1 ) );
    iAddress->setText( item->text( 2 ) );
    iEMail->setText( item->text( 3 ) );
}

void ABCentralWidget::toggleFirstName()
{
    sFirstName->setText( "" );

    if ( cFirstName->isChecked() ) {
        sFirstName->setEnabled( TRUE );
        sFirstName->setFocus();
    }
    else
        sFirstName->setEnabled( FALSE );
}

void ABCentralWidget::toggleLastName()
{
    sLastName->setText( "" );

    if ( cLastName->isChecked() ) {
        sLastName->setEnabled( TRUE );
        sLastName->setFocus();
    }
    else
        sLastName->setEnabled( FALSE );
}

void ABCentralWidget::toggleAddress()
{
    sAddress->setText( "" );

    if ( cAddress->isChecked() ) {
        sAddress->setEnabled( TRUE );
        sAddress->setFocus();
    }
    else
        sAddress->setEnabled( FALSE );
}

void ABCentralWidget::toggleEMail()
{
    sEMail->setText( "" );
    if ( cEMail->isChecked() ) {
        sEMail->setEnabled( TRUE );
        sEMail->setFocus();
    }
    else
        sEMail->setEnabled( FALSE );
}

void ABCentralWidget::findEntries()
{
    if ( !cFirstName->isChecked() &&
         !cLastName->isChecked() &&
         !cAddress->isChecked() &&
         !cEMail->isChecked() ) {
        listView->clearSelection();
        return;
    }

    QListViewItemIterator it( listView );

    for ( ; it.current(); ++it ) {
        bool select = TRUE;

        if ( cFirstName->isChecked() ) {
            if ( select && it.current()->text( 0 ).contains( sFirstName->text() ) )
                select = TRUE;
            else
                select = FALSE;
        }
        if ( cLastName->isChecked() ) {
            if ( select && it.current()->text( 1 ).contains( sLastName->text() ) )
                select = TRUE;
            else
                select = FALSE;
        }
        if ( cAddress->isChecked() ) {
            if ( select && it.current()->text( 2 ).contains( sAddress->text() ) )
                select = TRUE;
            else
                select = FALSE;
        }
        if ( cEMail->isChecked() ) {
            if ( select && it.current()->text( 3 ).contains( sEMail->text() ) )
                select = TRUE;
            else
                select = FALSE;
        }

        if ( select )
            it.current()->setSelected( TRUE );
        else
            it.current()->setSelected( FALSE );
        it.current()->repaint();
```

```
   }
}
```

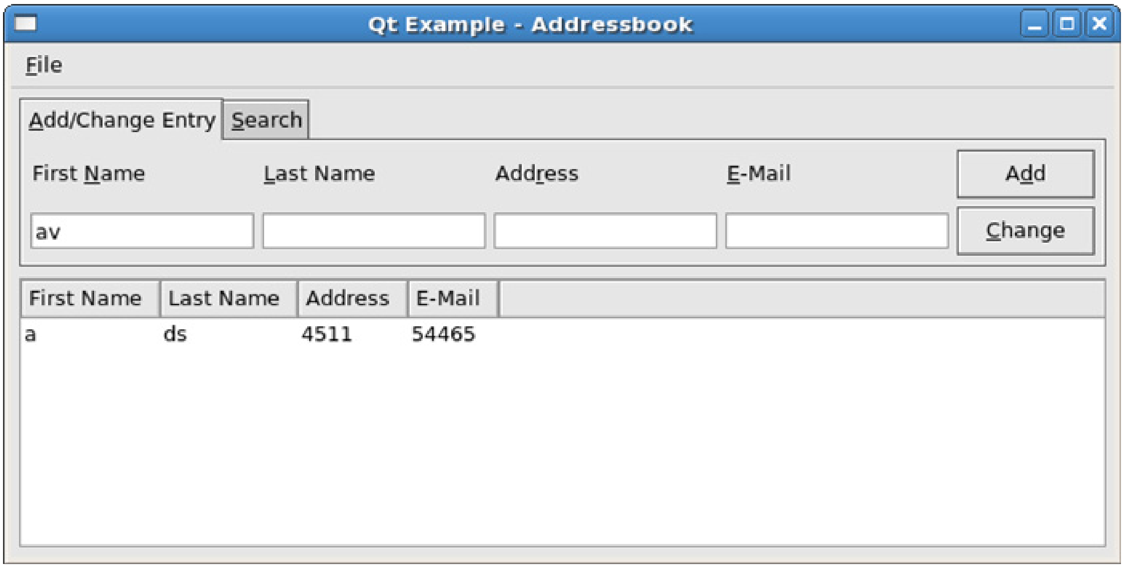
centralwidget.h

```
#ifndef AB_CENTRALWIDGET_H
#define AB_CENTRALWIDGET_H

#include <qwidget.h>
#include <qstring.h>

class QTabWidget;
class QListView;
class QGridLayout;
class QLineEdit;
class QPushButton;
class QListViewItem;
class QCheckBox;

class ABCentralWidget : public QWidget
{
   Q_OBJECT

public:
   ABCentralWidget( QWidget *parent, const char *name = 0 );

   void save( const QString &filename );
   void load( const QString &filename );

protected slots:
   void addEntry();
   void changeEntry();
   void itemSelected( QListViewItem* );
   void selectionChanged();
   void toggleFirstName();
   void toggleLastName();
   void toggleAddress();
   void toggleEMail();
   void findEntries();

protected:
   void setupTabWidget();
   void setupListView();

   QGridLayout *mainGrid;
   QTabWidget *tabWidget;
   QListView *listView;
   QPushButton *add, *change, *find;
   QLineEdit *iFirstName, *iLastName, *iAddress, *iEMail,
      *sFirstName, *sLastName, *sAddress, *sEMail;
   QCheckBox *cFirstName, *cLastName, *cAddress, *cEMail;

};

#endif
```

main.cpp

```
#include <qapplication.h>
#include "mainwindow.h"

int main( int argc, char ** argv )
{
   QApplication a( argc, argv );

   ABMainWindow *mw = new ABMainWindow();
   mw->setCaption( "Qt Example - Addressbook" );
   a.setMainWidget( mw );
   mw->show();

   a.connect( &a, SIGNAL( lastWindowClosed() ), &a, SLOT( quit() ) );
   int result = a.exec();
   delete mw;
   return result;
}
```

실행



First Name	Last Name	Address	E-Mail
av			

First Name	Last Name	Address	E-Mail
a	ds	4511	54465

4. 완전한 응용프로그람창문

이 실례프로그람은 완전한 현대응용프로그람처럼 보인다. 차림표띠, 도구띠, 상태띠를 가지고 단순본문편집기처럼 작업한다.

application.pro

```
TEMPLATE  = app
TARGET      = application
CONFIG     += qt warn_on release
HEADERS        = application.h
SOURCES        = application.cpp \
          main.cpp
```

main.cpp

```
#include <qapplication.h>
#include "application.h"

int main( int argc, char ** argv ) {
   QApplication a( argc, argv );
   ApplicationWindow *mw = new ApplicationWindow();
   mw->setCaption( "Qt Example - Application" );
   mw->show();
   a.connect( &a, SIGNAL(lastWindowClosed()), &a, SLOT(quit()) );
   return a.exec();
}
```

application.cpp

```
#include "application.h"
#include <qimage.h>
#include <qpixmap.h>
#include <qtoolbar.h>
#include <qtoolbutton.h>
#include <qpopupmenu.h>
#include <qmenubar.h>
#include <qtextedit.h>
#include <qfile.h>
#include <qfiledialog.h>
#include <qstatusbar.h>
#include <qmessagebox.h>
#include <qprinter.h>
#include <qapplication.h>
#include <qaccel.h>
#include <qtextstream.h>
#include <qpainter.h>
#include <qpaintdevicemetrics.h>
#include <qwhatsthis.h>
#include <qsimplerichtext.h>

#include "filesave.xpm"
#include "fileopen.xpm"
#include "fileprint.xpm"

ApplicationWindow::ApplicationWindow()
   : QMainWindow( 0, "example application main window", WDestructiveClose | WGroupLeader )
{
   printer = new QPrinter( QPrinter::HighResolution );
   QPixmap openIcon, saveIcon, printIcon;

   QToolBar * fileTools = new QToolBar( this, "file operations" );
   fileTools->setLabel( "File Operations" );

   openIcon = QPixmap( fileopen );
   QToolButton * fileOpen
     = new QToolButton( openIcon, "Open File", QString::null,
                 this, SLOT(choose()), fileTools, "open file" );

   saveIcon = QPixmap( filesave );
```

```
QToolButton * fileSave
  = new QToolButton( saveIcon, "Save File", QString::null,
            this, SLOT(save()), fileTools, "save file" );

printIcon = QPixmap( fileprint );
QToolButton * filePrint
  = new QToolButton( printIcon, "Print File", QString::null,
            this, SLOT(print()), fileTools, "print file" );

(void)QWhatsThis::whatsThisButton( fileTools );

const char * fileOpenText = "<p><img source=\"fileopen\"> "
        "Click this button to open a <em>new file</em>.<br>"
          "You can also select the <b>Open</b> command "
          "from the <b>File</b> menu.</p>";

QWhatsThis::add( fileOpen, fileOpenText );

QMimeSourceFactory::defaultFactory()->setPixmap( "fileopen", openIcon );

const char * fileSaveText = "<p>Click this button to save the file you "
          "are editing. You will be prompted for a file name.\n"
          "You can also select the <b>Save</b> command "
          "from the <b>File</b> menu.</p>";

QWhatsThis::add( fileSave, fileSaveText );

const char * filePrintText = "Click this button to print the file you "
          "are editing.\n"
      "You can also select the Print command "
      "from the File menu.";

QWhatsThis::add( filePrint, filePrintText );

QPopupMenu * file = new QPopupMenu( this );
menuBar()->insertItem( "&File", file );

file->insertItem( "&New", this, SLOT(newDoc()), CTRL+Key_N );

int id;
id = file->insertItem( openIcon, "&Open...",
            this, SLOT(choose()), CTRL+Key_O );
file->setWhatsThis( id, fileOpenText );

id = file->insertItem( saveIcon, "&Save",
            this, SLOT(save()), CTRL+Key_S );
file->setWhatsThis( id, fileSaveText );

id = file->insertItem( "Save &As...", this, SLOT(saveAs()) );
file->setWhatsThis( id, fileSaveText );

file->insertSeparator();

id = file->insertItem( printIcon, "&Print...",
             this, SLOT(print()), CTRL+Key_P );
  file->setWhatsThis( id, filePrintText );

  file->insertSeparator();

  file->insertItem( "&Close", this, SLOT(close()), CTRL+Key_W );

  file->insertItem( "&Quit", qApp, SLOT( closeAllWindows() ), CTRL+Key_Q );

  menuBar()->insertSeparator();

  QPopupMenu * help = new QPopupMenu( this );
  menuBar()->insertItem( "&Help", help );

  help->insertItem( "&About", this, SLOT(about()), Key_F1 );
  help->insertItem( "About &Qt", this, SLOT(aboutQt()) );
  help->insertSeparator();
  help->insertItem( "What's &This", this, SLOT(whatsThis()), SHIFT+Key_F1 );

  e = new QTextEdit( this, "editor" );
  e->setFocus();
  setCentralWidget( e );
  statusBar()->message( "Ready", 2000 );

  resize( 450, 600 );
}

ApplicationWindow::~ApplicationWindow()
{
  delete printer;
}

void ApplicationWindow::newDoc()
{
  ApplicationWindow *ed = new ApplicationWindow;
  ed->setCaption("Qt Example - Application");
  ed->show();
}

void ApplicationWindow::choose()
{
  QString fn = QFileDialog::getOpenFileName( QString::null, QString::null,
                     this);
  if ( !fn.isEmpty() )
   load( fn );
  else
   statusBar()->message( "Loading aborted", 2000 );
}

void ApplicationWindow::load( const QString &fileName )
{
  QFile f( fileName );
  if ( !f.open( IO_ReadOnly ) )
   return;
```

```
    QTextStream ts( &f );
    e->setText( ts.read() );
    e->setModified( FALSE );
    setCaption( fileName );
    statusBar()->message( "Loaded document " + fileName, 2000 );
}

void ApplicationWindow::save()
{
    if ( filename.isEmpty() ) {
     saveAs();
     return;
    }

    QString text = e->text();
    QFile f( filename );
    if ( !f.open( IO_WriteOnly ) ) {
     statusBar()->message( QString("Could not write to %1").arg(filename),
                      2000 );
     return;
    }

    QTextStream t( &f );
    t << text;
    f.close();

    e->setModified( FALSE );

    setCaption( filename );

    statusBar()->message( QString( "File %1 saved" ).arg( filename ), 2000 );
}

void ApplicationWindow::saveAs()
{
    QString fn = QFileDialog::getSaveFileName( QString::null, QString::null,
                               this );
    if ( !fn.isEmpty() ) {
     filename = fn;
     save();
    } else {
     statusBar()->message( "Saving aborted", 2000 );
    }
}

void ApplicationWindow::print()
{
    printer->setFullPage( TRUE );
    if ( printer->setup(this) ) {          // printer dialog
     statusBar()->message( "Printing..." );
     QPainter p;
     if( !p.begin( printer ) ) {              // paint on printer
         statusBar()->message( "Printing aborted", 2000 );
```

```
        return;
      }

      QPaintDeviceMetrics metrics( p.device() );
      int dpiy = metrics.logicalDpiY();
      int margin = (int) ( (2/2.54)*dpiy ); // 2 cm margins
      QRect view( margin, margin, metrics.width() - 2*margin, metrics.height() - 2*margin );
      QSimpleRichText richText( QStyleSheet::convertFromPlainText(e->text()),
                    QFont(),
                    e->context(),
                    e->styleSheet(),
                    e->mimeSourceFactory(),
                    view.height() );
      richText.setWidth( &p, view.width() );
      int page = 1;
      do {
        richText.draw( &p, margin, margin, view, colorGroup() );
        view.moveBy( 0, view.height() );
        p.translate( 0 , -view.height() );
        p.drawText( view.right() - p.fontMetrics().width( QString::number( page ) ),
             view.bottom() + p.fontMetrics().ascent() + 5, QString::number( page ) );
        if ( view.top() - margin >= richText.height() )
         break;
        printer->newPage();
        page++;
      } while (TRUE);

      statusBar()->message( "Printing completed", 2000 );
    } else {
      statusBar()->message( "Printing aborted", 2000 );
    }
}

void ApplicationWindow::closeEvent( QCloseEvent* ce )
{
    if ( !e->isModified() ) {
      ce->accept();
      return;
    }

    switch( QMessageBox::information( this, "Qt Application Example",
                    "Do you want to save the changes"
                    " to the document?",
                    "Yes", "No", "Cancel",
                    0, 1 ) ) {
    case 0:
      save();
      ce->accept();
      break;
    case 1:
      ce->accept();
      break;
    case 2:
    default: // just for sanity
```

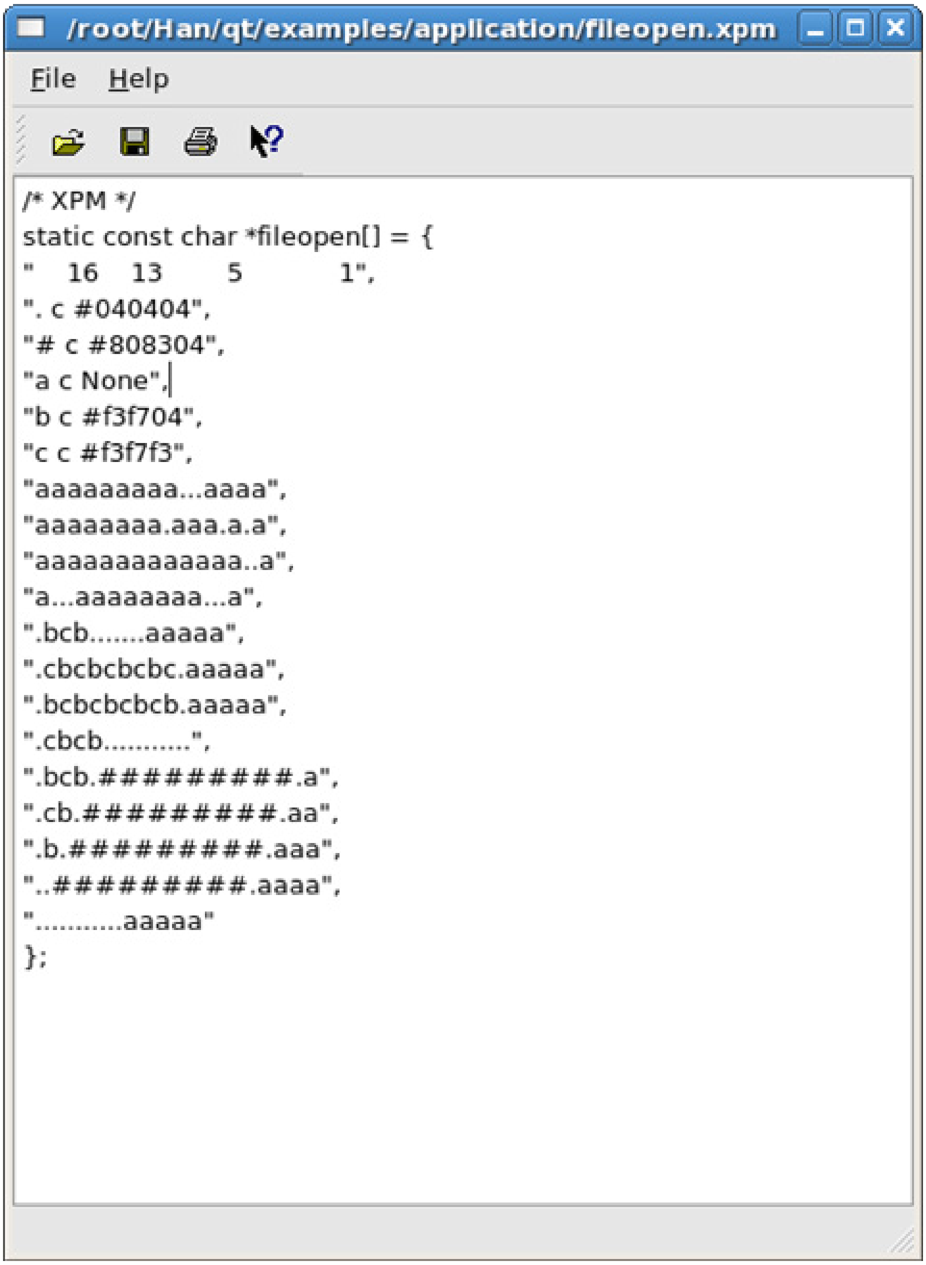
```
      ce->ignore();
      break;
    }
}

void ApplicationWindow::about()
{
    QMessageBox::about( this, "Qt Application Example",
                 "This example demonstrates simple use of "
                 "QMainWindow,\nQMenuBar and QToolBar.");
}

void ApplicationWindow::aboutQt()
{
    QMessageBox::aboutQt( this, "Qt Application Example" );
}
```

application.h

```
#ifndef APPLICATION_H
#define APPLICATION_H

#include <qmainwindow.h>

class QTextEdit;

class ApplicationWindow: public QMainWindow
{
    Q_OBJECT

public:
    ApplicationWindow();
    ~ApplicationWindow();

protected:
    void closeEvent( QCloseEvent* );

private slots:
    void newDoc();
    void choose();
    void load( const QString &fileName );
    void save();
    void saveAs();
    void print();

    void about();
    void aboutQt();

private:
    QPrinter *printer;
    QTextEdit *e;
    QString filename;
};
```

```
#endif
```

실행



```
/* XPM */
static const char *fileopen[] = {
"    16    13        5            1",
". c #040404",
"# c #808304",
"a c None",
"b c #f3f704",
"c c #f3f7f3",
"aaaaaaaaa...aaaa",
"aaaaaaaa.aaa.a.a",
"aaaaaaaaaaaaa..a",
"a...aaaaaaaa...a",
".bcb.......aaaaa",
".cbcbcbcbc.aaaaa",
".bcbcbcbcb.aaaaa",
".cbcb...........",
".bcb.#########.a",
".cb.#########.aa",
".b.#########.aaa",
"..#########.aaaa",
"...........aaaaa"
};
```

5. biff

Biff는 새로운 우편이 있는가를 가리키는 단순한 그라픽스프로그람이며 xbiff와 꼭 같아보이지만 더 간단하다.

biff.pro

```
TEMPLATE  = app
TARGET    = biff
CONFIG    += qt warn_on release
```

```
HEADERS         = biff.h
SOURCES         = biff.cpp \
            main.cpp
```

main.cpp

```
#include <qapplication.h>
#include "biff.h"

int main( int argc, char ** argv )
{
    QApplication a( argc, argv );
    Biff b;
    a.setMainWidget( &b );
    b.show();
    return a.exec();
}
```

biff.cpp

```
#include "biff.h"
#include <qstring.h>
#include <qfileinfo.h>
#include <qpainter.h>
#include <unistd.h>
#include <stdlib.h>
#include "bmp.cpp"

Biff::Biff( QWidget *parent, const char *name )
    : QWidget( parent, name, WShowModal | WType_Dialog )
{
    QFileInfo fi = QString(getenv( "MAIL" ));
    if ( !fi.exists() ) {
     QString s( "/var/spool/mail/" );
     s += getlogin();
     fi.setFile( s );
    }

    if ( fi.exists() ) {
     mailbox = fi.absFilePath();
     startTimer( 1000 );
    }

    setMinimumSize( 48, 48 );
    setMaximumSize( 48, 48 );
    resize( 48, 48 );

    hasNewMail.loadFromData( hasmail_bmp_data, hasmail_bmp_len );
    noNewMail.loadFromData( nomail_bmp_data, nomail_bmp_len );

    gotMail = FALSE;
    lastModified = fi.lastModified();
}

void Biff::timerEvent( QTimerEvent * )
{
    QFileInfo fi( mailbox );
    bool newState = ( fi.lastModified() != lastModified &&
                fi.lastModified() > fi.lastRead() );
    if ( newState != gotMail ) {
      if ( gotMail )
          lastModified = fi.lastModified();
      gotMail = newState;
      repaint( FALSE );
    }
}

void Biff::paintEvent( QPaintEvent * )
{
    if ( gotMail )
      bitBlt( this, 0, 0, &hasNewMail );
    else
      bitBlt( this, 0, 0, &noNewMail );
}

void Biff::mousePressEvent( QMouseEvent * )
{
    QFileInfo fi( mailbox );
    lastModified = fi.lastModified();
}
```

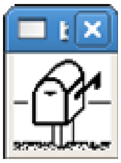
biff.h

```
#ifndef BIFF_H
#define BIFF_H
#include <qwidget.h>
#include <qdatetime.h>
#include <qpixmap.h>

class Biff : public QWidget
{
    Q_OBJECT
public:
    Biff( QWidget *parent=0, const char *name=0 );

protected:
    voidtimerEvent( QTimerEvent * );
    voidpaintEvent( QPaintEvent * );
    voidmousePressEvent( QMouseEvent * );

private:
    QDateTime      lastModified;
    QPixmap    hasNewMail;
    QPixmap    noNewMail;
    QStringmailbox;
    boolgotMail;
};

#endif // BIFF_H
```

bmp.cpp

```
static const unsigned int  hasmail_bmp_len = 1218;
static const unsigned char hasmail_bmp_data[] = {
    0x42,0x4d,0xc2,0x04,0x00,0x00,0x00,0x00,0x00,0x00,0x42,0x00,0x00,0x00,
    0x28,0x00,0x00,0x00,0x30,0x00,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,
    0x04,0x00,0x00,0x00,0x00,0x00,0x80,0x04,0x00,0x00,0x6d,0x0b,0x00,0x00,
    0x6d,0x0b,0x00,0x00,0x03,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0xff,0xff,0xff,0x00,0x51,0x61,0x30,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x10,
    0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,
    0x10,0x00,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,
    0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x00,0x00,0x10,
    0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x01,0x00,0x10,0x10,0x10,
    0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x01,0x01,0x01,0x01,0x01,0x01,
    0x01,0x01,0x01,0x01,0x00,0x01,0x00,0x01,0x01,0x01,0x01,0x01,0x01,0x01,
    0x01,0x01,0x01,0x00,0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x11,
    0x00,0x01,0x00,0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,
    0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x00,0x01,0x00,0x00,
    0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x00,0x00,0x10,0x10,0x10,
    0x10,0x10,0x10,0x10,0x10,0x11,0x00,0x01,0x00,0x00,0x01,0x10,0x10,0x10,
    0x10,0x10,0x10,0x10,0x10,0x10,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,
    0x01,0x01,0x00,0x01,0x00,0x00,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,
    0x01,0x00,0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x11,0x00,0x01,
    0x00,0x00,0x01,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x01,0x01,
    0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x00,0x01,0x00,0x00,0x01,0x01,
    0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x00,0x00,0x10,0x10,0x10,0x10,0x00,
    0x00,0x00,0x00,0x01,0x00,0x01,0x00,0x00,0x01,0x10,0x10,0x10,0x10,0x10,
    0x10,0x10,0x10,0x10,0x01,0x01,0x01,0x01,0x00,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x10,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x00,
    0x00,0x10,0x10,0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x01,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x01,0x01,0x01,0x01,
    0x01,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x11,0x11,0x01,0x01,0x01,
    0x01,0x01,0x01,0x01,0x01,0x00,0x00,0x10,0x10,0x10,0x01,0x10,0x00,0x00,
    0x00,0x00,0x00,0x00,0x01,0x10,0x11,0x00,0x10,0x10,0x10,0x10,0x10,0x10,
    0x10,0x10,0x01,0x01,0x01,0x01,0x01,0x10,0x00,0x00,0x00,0x00,0x00,0x00,
    0x01,0x10,0x01,0x10,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x00,0x00,0x10,
    0x10,0x10,0x01,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x00,0x11,
    0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x00,0x00,0x00,0x00,0x01,0x10,
    0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x00,0x01,0x10,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x10,0x10,0x10,0x10,
    0x10,0x11,0x01,0x10,0x00,0x00,0x11,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x01,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x01,0x10,
    0x00,0x00,0x01,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x01,0x10,0x10,0x10,0x11,0x10,0x10,0x10,0x01,0x10,0x00,0x00,0x00,0x10,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x11,0x10,
    0x10,0x10,0x10,0x10,0x01,0x10,0x00,0x00,0x00,0x10,0x00,0x00,0x00,0x00,
    0x00,0x00,0x11,0x11,0x11,0x10,0x01,0x10,0x10,0x10,0x01,0x00,0x10,0x10,
    0x01,0x10,0x00,0x00,0x00,0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x00,0x00,
    0x00,0x00,0x01,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x01,0x10,
    0x00,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,
    0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x01,0x11,0x00,0x10,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x00,0x00,0x00,0x00,
    0x00,0x00,0x01,0x10,0x01,0x11,0x00,0x10,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x01,0x10,0x00,0x00,0x01,0x10,0x00,0x00,0x01,0x10,
    0x01,0x11,0x00,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x01,0x10,0x00,0x01,0x11,0x11,0x10,0x00,0x01,0x11,0x01,0x11,0x00,0x10,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x11,0x00,0x00,
    0x00,0x00,0x00,0x00,0x11,0x11,0x11,0x11,0x00,0x10,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x11,0x00,0x00,0x00,0x00,0x00,0x00,
    0x11,0x01,0x11,0x11,0x00,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x01,0x10,0x00,0x00,0x00,0x00,0x01,0x10,0x00,0x11,0x11,
    0x00,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
    0x11,0x00,0x00,0x00,0x00,0x01,0x10,0x00,0x01,0x11,0x00,0x10,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x11,0x10,0x00,0x00,
    0x01,0x11,0x00,0x00,0x01,0x11,0x10,0x10,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x11,0x11,0x11,0x11,0x10,0x00,0x00,
    0x01,0x11,0x11,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x11,0x11,0x11,0x10,0x00,0x00,0x00,0x01,0x11,0x11,0x10,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
    0x10,0x00,0x00,0x00,0x00,0x00,0x01,0x11,0x01,0x10,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x11,0x00,0x00,0x00,
    0x00,0x00,0x01,0x11,0x00,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x00,0x00,0x00,0x00,0x01,0x11,
    0x00,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x11,0x00,0x00,0x00,0x00,0x01,0x11,0x11,0x11,0x11,0x10,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
    0x10,0x00,0x00,0x00,0x01,0x11,0x11,0x11,0x11,0x10,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x11,0x00,0x00,0x00,
    0x01,0x10,0x10,0x10,0x11,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x11,0x00,0x00,0x01,0x11,0x01,0x01,
    0x01,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x11,0x11,0x01,0x11,0x10,0x10,0x10,0x11,0x10,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x11,0x11,0x01,0x11,0x01,0x01,0x01,0x10,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x11,
    0x11,0x11,0x11,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x11,0x11,0x11,0x11,0x10,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};

static const unsigned int  nomail_bmp_len = 1222;
static const unsigned char nomail_bmp_data[] = {
    0x42,0x4d,0xc6,0x04,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00,0x00,
    0x28,0x00,0x00,0x00,0x30,0x00,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,
    0x04,0x00,0x00,0x00,0x00,0x00,0x80,0x04,0x00,0x00,0x6d,0x0b,0x00,0x00,
    0x6d,0x0b,0x00,0x00,0x04,0x00,0x00,0x00,0x04,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0xff,0xff,0xff,0x00,0x38,0x30,0x61,0x00,0xa6,0x8a,0xff,0x00,
    0x01,0x01,0x01,0x01,0x11,0x01,0x11,0x10,0x11,0x10,0x11,0x01,0x01,0x11,
    0x01,0x11,0x11,0x01,0x11,0x10,0x01,0x10,0x01,0x11,0x10,0x01,0x11,0x01,
    0x11,0x11,0x00,0x11,0x10,0x11,0x10,0x00,0x00,0x00,0x11,0x11,0x01,0x11,
    0x10,0x00,0x00,0x01,0x11,0x01,0x01,0x00,0x01,0x10,0x00,0x10,0x10,0x00,
    0x00,0x10,0x01,0x10,0x10,0x01,0x00,0x01,0x10,0x11,0x01,0x00,0x10,0x10,
    0x00,0x01,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x10,0x10,0x00,0x11,0x10,
    0x11,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x11,0x10,0x00,0x00,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x11,0x10,0x11,0x11,0x10,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x10,0x11,0x10,0x11,0x11,0x10,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,
    0x11,0x10,0x11,0x11,0x10,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x11,0x10,0x11,0x11,
    0x10,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x10,0x11,0x10,0x11,0x11,0x10,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x10,0x11,0x10,0x11,0x11,0x10,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x11,0x10,
    0x11,0x11,0x10,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x11,0x10,0x11,0x11,0x10,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x10,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x00,
    0x00,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,0x00,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x11,
    0x11,0x11,0x11,0x11,0x10,0x01,0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x11,
    0x10,0x01,0x11,0x00,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x10,
    0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,
    0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x00,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x11,
    0x11,0x11,0x10,0x01,0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,
    0x11,0x11,0x11,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x01,
    0x11,0x11,0x11,0x11,0x11,0x11,0x00,0x00,0x00,0x01,0x10,0x01,0x11,0x11,
    0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x01,0x10,0x00,0x00,0x00,
    0x00,0x00,0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x11,
    0x10,0x01,0x11,0x11,0x11,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x00,
    0x11,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,
    0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x00,0x01,0x01,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x10,0x01,
    0x11,0x11,0x10,0x01,0x11,0x00,0x01,0x01,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x10,0x00,0x00,0x01,0x11,0x10,0x00,
    0x11,0x10,0x00,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x00,0x11,0x11,0x11,0x11,0x11,0x11,0x00,0x00,0x01,0x11,0x00,0x00,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x00,0x11,0x11,
    0x11,0x11,0x11,0x11,0x00,0x10,0x00,0x11,0x10,0x00,0x01,0x11,0x00,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x11,0x10,
    0x01,0x11,0x00,0x01,0x11,0x00,0x00,0x11,0x00,0x01,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x10,0x00,0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x10,0x00,
    0x11,0x01,0x00,0x01,0x00,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x00,0x01,0x11,0x11,0x10,0x00,0x11,0x11,0x11,0x00,0x01,0x01,0x10,0x00,
    0x00,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x00,0x00,0x00,
    0x00,0x01,0x11,0x11,0x11,0x10,0x00,0x01,0x11,0x00,0x00,0x01,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x00,0x00,0x00,0x01,0x11,0x11,0x11,
    0x11,0x11,0x00,0x01,0x11,0x10,0x00,0x01,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,
    0x11,0x11,0x00,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x00,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x10,0x01,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x11,
    0x11,0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x01,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x00,0x11,0x11,0x11,0x11,0x11,0x11,
    0x00,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x00,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x00,0x01,0x11,0x11,0x11,0x10,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x00,0x01,0x11,
    0x10,0x00,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x10,0x00,0x00,0x00,0x01,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x10,0x00,0x01,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,
    0x11,0x11,0x11,0x11
  };
```

실행



6. 단추와 그룹칸

이 실례는 각이한 형의 그룹칸 (단추그룹 등)과 각종 단추들 (검사칸, 라지오단추, 누름단추 등)을 보여준다.

buttongroups.pro

```
TEMPLATE  = app
TARGET      = buttongroups
CONFIG      += qt warn_on release
HEADERS         = buttongroups.h
SOURCES         = buttongroups.cpp \
          main.cpp
```

buttongroups.cpp

```
#include "buttongroups.h"
#include <qpopupmenu.h>
#include <qbuttongroup.h>
#include <qlayout.h>
#include <qradiobutton.h>
#include <qcheckbox.h>
#include <qgroupbox.h>
#include <qpushbutton.h>

/*
```

```
 * Constructor
 *
 * Creates all child widgets of the ButtonGroups window
 */

ButtonsGroups::ButtonsGroups( QWidget *parent, const char *name )
    : QWidget( parent, name )
{
    // Create Widgets which allow easy layouting
    QVBoxLayout *vbox = new QVBoxLayout( this, 11, 6 );
    QHBoxLayout *box1 = new QHBoxLayout( vbox );
    QHBoxLayout *box2 = new QHBoxLayout( vbox );

    // ------- first group

    // Create an exclusive button group
    QButtonGroup *bgrp1 = new QButtonGroup( 1, QGroupBox::Horizontal, "Button Group 1
(exclusive)", this);
    box1->addWidget( bgrp1 );
    bgrp1->setExclusive( TRUE );

    // insert 3 radiobuttons
    QRadioButton *rb11 = new QRadioButton( "&Radiobutton 1", bgrp1 );
    rb11->setChecked( TRUE );
    (void)new QRadioButton( "R&adiobutton 2", bgrp1 );
    (void)new QRadioButton( "Ra&diobutton 3", bgrp1 );

    // ------- second group

    // Create a non-exclusive buttongroup
    QButtonGroup *bgrp2 = new QButtonGroup( 1, QGroupBox::Horizontal, "Button Group 2 (non-
exclusive)", this );
    box1->addWidget( bgrp2 );
    bgrp2->setExclusive( FALSE );

    // insert 3 checkboxes
    (void)new QCheckBox( "&Checkbox 1", bgrp2 );
    QCheckBox *cb12 = new QCheckBox( "C&heckbox 2", bgrp2 );
    cb12->setChecked( TRUE );
    QCheckBox *cb13 = new QCheckBox( "Triple &State Button", bgrp2 );
    cb13->setTristate( TRUE );
    cb13->setChecked( TRUE );

    // ------------ third group

    // create a buttongroup which is exclusive for radiobuttons and non-exclusive for all other buttons
    QButtonGroup *bgrp3 = new QButtonGroup( 1, QGroupBox::Horizontal, "Button Group 3
(Radiobutton-exclusive)", this );
    box2->addWidget( bgrp3 );
    bgrp3->setRadioButtonExclusive( TRUE );

    // insert three radiobuttons
    rb21 = new QRadioButton( "Rad&iobutton 1", bgrp3 );
    rb22 = new QRadioButton( "Radi&obutton 2", bgrp3 );
```

```
   rb23 = new QRadioButton( "Radio&button 3", bgrp3 );
   rb23->setChecked( TRUE );

   // insert a checkbox...
   state = new QCheckBox( "E&nable Radiobuttons", bgrp3 );
   state->setChecked( TRUE );
   // ...and connect its SIGNAL clicked() with the SLOT slotChangeGrp3State()
   connect( state, SIGNAL( clicked() ), this, SLOT( slotChangeGrp3State() ) );

   // ------------ fourth group

   // create a groupbox which layouts its childs in a columns
   QGroupBox *bgrp4 = new QButtonGroup( 1, QGroupBox::Horizontal, "Groupbox with normal
buttons", this );
   box2->addWidget( bgrp4 );

   // insert four pushbuttons...
   (void)new QPushButton( "&Push Button", bgrp4, "push" );

   // now make the second one a toggle button
   QPushButton *tb2 = new QPushButton( "&Toggle Button", bgrp4, "toggle" );
   tb2->setToggleButton( TRUE );
   tb2->setOn( TRUE );

   // ... and make the third one a flat button
   QPushButton *tb3 = new QPushButton( "&Flat Button", bgrp4, "flat" );
   tb3->setFlat(TRUE);

   // .. and the fourth a button with a menu
   QPushButton *tb4 = new QPushButton( "Popup Button", bgrp4, "popup" );
   QPopupMenu *menu = new QPopupMenu(tb4);
   menu->insertItem("Item1", 0);
   menu->insertItem("Item2", 1);
   menu->insertItem("Item3", 2);
   menu->insertItem("Item4", 3);
   tb4->setPopup(menu);
}

/*
 * SLOT slotChangeGrp3State()
* enables/disables the radiobuttons of the third buttongroup
 */

void ButtonsGroups::slotChangeGrp3State()
{
   rb21->setEnabled( state->isChecked() );
   rb22->setEnabled( state->isChecked() );
   rb23->setEnabled( state->isChecked() );
}
```

buttongroups.h

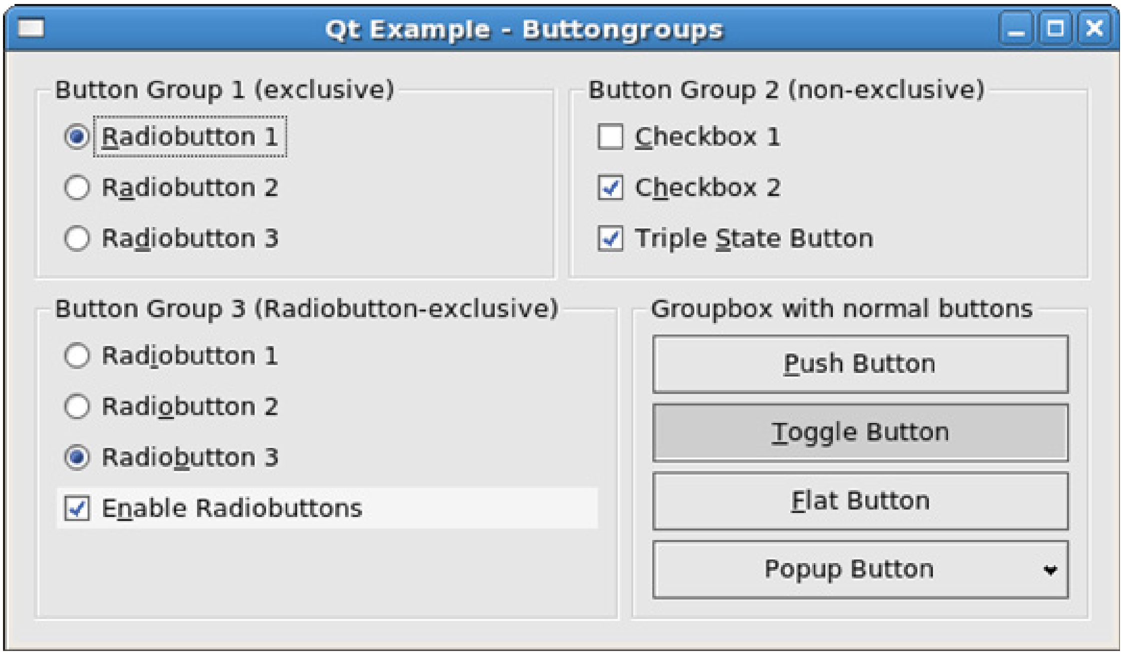
```
#ifndef BUTTONS_GROUPS_H
#define BUTTONS_GROUPS_H
```

```
#include <qwidget.h>

class QCheckBox;
class QRadioButton;

class ButtonsGroups : public QWidget
{
   Q_OBJECT

public:
   ButtonsGroups( QWidget *parent = 0, const char *name = 0 );

protected:
   QCheckBox *state;
   QRadioButton *rb21, *rb22, *rb23;

protected slots:
   void slotChangeGrp3State();

};

#endif
```

main.cpp

```
#include "buttongroups.h"
#include <qapplication.h>

int main( int argc, char **argv )
{
   QApplication a( argc, argv );

   ButtonsGroups buttonsgroups;
   buttonsgroups.resize( 500, 250 );
   buttonsgroups.setCaption( "Qt Example - Buttongroups" );
   a.setMainWidget( &buttonsgroups );
   buttonsgroups.show();

   return a.exec();
}
```

실행



7. 캔버스실례

이 실례는 동작하는 QCanvas와 일부 QCanvasItem들을 보여준다. 여기서 보여주는것보다 QCanvas로 더 많은것을 할수 있으나 실례는 무엇을 수행할수 있는가 하는 경험을 제공한다.

canvas.pro

```
TEMPLATE  = app
TARGET    = canvas
CONFIG    += qt warn_on release
HEADERS      = canvas.h
SOURCES      = canvas.cpp main.cpp
```

canvas.cpp

```
#include <qdatetime.h>
#include <qmainwindow.h>
#include <qstatusbar.h>
#include <qmessagebox.h>
#include <qmenubar.h>
#include <qapplication.h>
#include <qpainter.h>
#include <qprinter.h>
#include <qlabel.h>
#include <qimage.h>
#include <qprogressdialog.h>
#include "canvas.h"

#include <stdlib.h>

// We use a global variable to save memory - all the brushes and pens in
// the mesh are shared.
static QBrush *tb = 0;
static QPen *tp = 0;

class EdgeItem;
class NodeItem;

class EdgeItem: public QCanvasLine
{
public:
   EdgeItem( NodeItem*, NodeItem*, QCanvas *canvas );
   void setFromPoint( int x, int y ) ;
   void setToPoint( int x, int y );
   static int count() { return c; }
   void moveBy(double dx, double dy);
private:
   static int c;
};

static const int imageRTTI = 984376;

class ImageItem: public QCanvasRectangle
{
public:
   ImageItem( QImage img, QCanvas *canvas );
   int rtti () const { return imageRTTI; }
   bool hit( const QPoint&) const;
protected:
   void drawShape( QPainter & );
private:
   QImage image;
   QPixmap pixmap;
};

ImageItem::ImageItem( QImage img, QCanvas *canvas )
   : QCanvasRectangle( canvas ), image(img)
{
   setSize( image.width(), image.height() );

#if !defined(Q_WS_QWS)
   pixmap.convertFromImage(image, OrderedAlphaDither);
#endif
}

void ImageItem::drawShape( QPainter &p )
{
// On Qt/Embedded, we can paint a QImage as fast as a QPixmap,
// but on other platforms, we need to use a QPixmap.
#if defined(Q_WS_QWS)
   p.drawImage( int(x()), int(y()), image, 0, 0, -1, -1, OrderedAlphaDither );
#else
   p.drawPixmap( int(x()), int(y()), pixmap );
#endif
}

bool ImageItem::hit( const QPoint &p ) const
```

```
{
   int ix = p.x()-int(x());
   int iy = p.y()-int(y());
   if ( !image.valid( ix , iy ) )
     return FALSE;
   QRgb pixel = image.pixel( ix, iy );
   return qAlpha( pixel ) != 0;
}

class NodeItem: public QCanvasEllipse
{
public:
   NodeItem( QCanvas *canvas );
   ~NodeItem() {}

   void addInEdge( EdgeItem *edge ) { inList.append( edge ); }
   void addOutEdge( EdgeItem *edge ) { outList.append( edge ); }

   void moveBy(double dx, double dy);

   //    QPoint center() { return boundingRect().center(); }
private:
   QPtrList<EdgeItem> inList;
   QPtrList<EdgeItem> outList;
};

int EdgeItem::c = 0;

void EdgeItem::moveBy(double, double)
{
   //nothing
}

EdgeItem::EdgeItem( NodeItem *from, NodeItem *to, QCanvas *canvas )
   : QCanvasLine( canvas )
{
   c++;
   setPen( *tp );
   setBrush( *tb );
   from->addOutEdge( this );
   to->addInEdge( this );
   setPoints( int(from->x()), int(from->y()), int(to->x()), int(to->y()) );
   setZ( 127 );
}

void EdgeItem::setFromPoint( int x, int y )
{
   setPoints( x,y, endPoint().x(), endPoint().y() );
}

void EdgeItem::setToPoint( int x, int y )
{
   setPoints( startPoint().x(), startPoint().y(), x, y );
}
```

```
void NodeItem::moveBy(double dx, double dy)
{
   QCanvasEllipse::moveBy( dx, dy );

   QPtrListIterator<EdgeItem> it1( inList );
   EdgeItem *edge;
   while (( edge = it1.current() )) {
    ++it1;
    edge->setToPoint( int(x()), int(y()) );
   }
   QPtrListIterator<EdgeItem> it2( outList );
   while (( edge = it2.current() )) {
    ++it2;
    edge->setFromPoint( int(x()), int(y()) );
   }
}

NodeItem::NodeItem( QCanvas *canvas )
   : QCanvasEllipse( 6, 6, canvas )
{
   setPen( *tp );
   setBrush( *tb );
   setZ( 128 );
}

FigureEditor::FigureEditor( QCanvas& c, QWidget* parent, const char* name, WFlags f) :
   QCanvasView(&c,parent,name,f)
{
}

void FigureEditor::contentsMousePressEvent(QMouseEvent* e)
{
   QPoint p = inverseWorldMatrix().map(e->pos());
   QCanvasItemList l=canvas()->collisions(p);
   for (QCanvasItemList::Iterator it=l.begin(); it!=l.end(); ++it) {
    if ( (*it)->rtti() == imageRTTI ) {
       ImageItem *item= (ImageItem*)(*it);
       if ( !item->hit( p ) )
          continue;
    }
    moving = *it;
    moving_start = p;
    return;
   }
   moving = 0;
}

void FigureEditor::clear()
{
   QCanvasItemList list = canvas()->allItems();
   QCanvasItemList::Iterator it = list.begin();
   for (; it != list.end(); ++it) {
    if ( *it )
```

```
        delete *it;
    }
}

void FigureEditor::contentsMouseMoveEvent(QMouseEvent* e)
{
    if ( moving ) {
     QPoint p = inverseWorldMatrix().map(e->pos());
     moving->moveBy(p.x() - moving_start.x(),
                 p.y() - moving_start.y());
     moving_start = p;
     canvas()->update();
    }
}

BouncyLogo::BouncyLogo(QCanvas* canvas) :
    QCanvasSprite(0,canvas)
{
    static QCanvasPixmapArray logo("qt-trans.xpm");
    setSequence(&logo);
    setAnimated(TRUE);
    initPos();
}

const int logo_rtti = 1234;

int BouncyLogo::rtti() const
{
    return logo_rtti;
}

void BouncyLogo::initPos()
{
    initSpeed();
    int trial=1000;
    do {
     move(rand()%canvas()->width(),rand()%canvas()->height());
     advance(0);
    } while (trial-- && xVelocity()==0.0 && yVelocity()==0.0);
}

void BouncyLogo::initSpeed()
{
    const double speed = 4.0;
    double d = (double)(rand()%1024) / 1024.0;
    setVelocity( d*speed*2-speed, (1-d)*speed*2-speed );
}

void BouncyLogo::advance(int stage)
{
    switch ( stage ) {
     case 0: {
     double vx = xVelocity();
     double vy = yVelocity();
     if ( vx == 0.0 && vy == 0.0 ) {
        // stopped last turn
        initSpeed();
        vx = xVelocity();
        vy = yVelocity();
     }

     double nx = x() + vx;
     double ny = y() + vy;

     if ( nx < 0 || nx >= canvas()->width() )
        vx = -vx;
     if ( ny < 0 || ny >= canvas()->height() )
        vy = -vy;

     for (int bounce=0; bounce<4; bounce++) {
        QCanvasItemList l=collisions(FALSE);
        for (QCanvasItemList::Iterator it=l.begin(); it!=l.end(); ++it) {
         QCanvasItem *hit = *it;
         if ( hit->rtti()==logo_rtti && hit->collidesWith(this) ) {
            switch ( bounce ) {
             case 0:
             vx = -vx;
             break;
             case 1:
             vy = -vy;
             vx = -vx;
             break;
             case 2:
             vx = -vx;
             break;
             case 3:
             // Stop for this turn
             vx = 0;
             vy = 0;
             break;
            }
            setVelocity(vx,vy);
            break;
         }
        }
     }

     if ( x()+vx < 0 || x()+vx >= canvas()->width() )
        vx = 0;
     if ( y()+vy < 0 || y()+vy >= canvas()->height() )
        vy = 0;

     setVelocity(vx,vy);
     } break;
     case 1:
     QCanvasItem::advance(stage);
     break;
```

```
    }
}

static uint mainCount = 0;
static QImage *butterflyimg;
static QImage *logoimg;

Main::Main(QCanvas& c, QWidget* parent, const char* name, WFlags f) :
    QMainWindow(parent,name,f),
    canvas(c)
{
    editor = new FigureEditor(canvas,this);
    QMenuBar* menu = menuBar();

    QPopupMenu* file = new QPopupMenu( menu );
    file->insertItem("&Fill canvas", this, SLOT(init()), CTRL+Key_F);
    file->insertItem("&Erase canvas", this, SLOT(clear()), CTRL+Key_E);
    file->insertItem("&New view", this, SLOT(newView()), CTRL+Key_N);
    file->insertSeparator();
    file->insertItem("&Print...", this, SLOT(print()), CTRL+Key_P);
    file->insertSeparator();
    file->insertItem("E&xit", qApp, SLOT(quit()), CTRL+Key_Q);
    menu->insertItem("&File", file);

    QPopupMenu* edit = new QPopupMenu( menu );
    edit->insertItem("Add &Circle", this, SLOT(addCircle()), ALT+Key_C);
    edit->insertItem("Add &Hexagon", this, SLOT(addHexagon()), ALT+Key_H);
    edit->insertItem("Add &Polygon", this, SLOT(addPolygon()), ALT+Key_P);
    edit->insertItem("Add Spl&ine", this, SLOT(addSpline()), ALT+Key_I);
    edit->insertItem("Add &Text", this, SLOT(addText()), ALT+Key_T);
    edit->insertItem("Add &Line", this, SLOT(addLine()), ALT+Key_L);
    edit->insertItem("Add &Rectangle", this, SLOT(addRectangle()), ALT+Key_R);
    edit->insertItem("Add &Sprite", this, SLOT(addSprite()), ALT+Key_S);
    edit->insertItem("Create &Mesh", this, SLOT(addMesh()), ALT+Key_M );
    edit->insertItem("Add &Alpha-blended image", this, SLOT(addButterfly()), ALT+Key_A);
    menu->insertItem("&Edit", edit);

    QPopupMenu* view = new QPopupMenu( menu );
    view->insertItem("&Enlarge", this, SLOT(enlarge()), SHIFT+CTRL+Key_Plus);
    view->insertItem("Shr&ink", this, SLOT(shrink()), SHIFT+CTRL+Key_Minus);
    view->insertSeparator();
    view->insertItem("&Rotate clockwise", this, SLOT(rotateClockwise()), CTRL+Key_PageDown);
    view->insertItem("Rotate &counterclockwise", this, SLOT(rotateCounterClockwise()),
CTRL+Key_PageUp);
    view->insertItem("&Zoom in", this, SLOT(zoomIn()), CTRL+Key_Plus);
    view->insertItem("Zoom &out", this, SLOT(zoomOut()), CTRL+Key_Minus);
    view->insertItem("Translate left", this, SLOT(moveL()), CTRL+Key_Left);
    view->insertItem("Translate right", this, SLOT(moveR()), CTRL+Key_Right);
    view->insertItem("Translate up", this, SLOT(moveU()), CTRL+Key_Up);
    view->insertItem("Translate down", this, SLOT(moveD()), CTRL+Key_Down);
    view->insertItem("&Mirror", this, SLOT(mirror()), CTRL+Key_Home);
    menu->insertItem("&View", view);

    options = new QPopupMenu( menu );
    dbf_id = options->insertItem("Double buffer", this, SLOT(toggleDoubleBuffer()));
    options->setItemChecked(dbf_id, TRUE);
    menu->insertItem("&Options",options);

    menu->insertSeparator();

    QPopupMenu* help = new QPopupMenu( menu );
    help->insertItem("&About", this, SLOT(help()), Key_F1);
    help->setItemChecked(dbf_id, TRUE);
    menu->insertItem("&Help",help);

    statusBar();

    setCentralWidget(editor);

    printer = 0;

    init();
}

void Main::init()
{
    clear();

    static int r=24;
    srand(++r);

    mainCount++;
    butterflyimg = 0;
    logoimg = 0;

    int i;
    for ( i=0; i<canvas.width() / 56; i++) {
      addButterfly();
    }
    for ( i=0; i<canvas.width() / 85; i++) {
      addHexagon();
    }
    for ( i=0; i<canvas.width() / 128; i++) {
      addLogo();
    }
}

Main::~Main()
{
    delete printer;
    if ( !--mainCount ) {
      delete[] butterflyimg;
      butterflyimg = 0;
      delete[] logoimg;
      logoimg = 0;
    }
}
```

```
void Main::newView()
{
   // Open a new view... have it delete when closed.
   Main *m = new Main(canvas, 0, 0, WDestructiveClose);
   qApp->setMainWidget(m);
   m->show();
   qApp->setMainWidget(0);
}

void Main::clear()
{
   editor->clear();
}

void Main::help()
{
   static QMessageBox* about = new QMessageBox( "Qt Canvas Example",
        "<h3>The QCanvas classes example</h3>"
        "<ul>"
          "<li> Press ALT-S for some sprites."
          "<li> Press ALT-C for some circles."
          "<li> Press ALT-L for some lines."
          "<li> Drag the objects around."
          "<li> Read the code!"
        "</ul>", QMessageBox::Information, 1, 0, 0, this, 0, FALSE );
   about->setButtonText( 1, "Dismiss" );
   about->show();
}

void Main::aboutQt()
{
   QMessageBox::aboutQt( this, "Qt Canvas Example" );
}

void Main::toggleDoubleBuffer()
{
   bool s = !options->isItemChecked(dbf_id);
   options->setItemChecked(dbf_id,s);
   canvas.setDoubleBuffering(s);
}

void Main::enlarge()
{
   canvas.resize(canvas.width()*4/3, canvas.height()*4/3);
}

void Main::shrink()
{
   canvas.resize(canvas.width()*3/4, canvas.height()*3/4);
}

void Main::rotateClockwise()
{
   QWMatrix m = editor->worldMatrix();
```

```
    m.rotate( 22.5 );
    editor->setWorldMatrix( m );
}

void Main::rotateCounterClockwise()
{
    QWMatrix m = editor->worldMatrix();
    m.rotate( -22.5 );
    editor->setWorldMatrix( m );
}

void Main::zoomIn()
{
    QWMatrix m = editor->worldMatrix();
    m.scale( 2.0, 2.0 );
    editor->setWorldMatrix( m );
}

void Main::zoomOut()
{
    QWMatrix m = editor->worldMatrix();
    m.scale( 0.5, 0.5 );
    editor->setWorldMatrix( m );
}

void Main::mirror()
{
    QWMatrix m = editor->worldMatrix();
    m.scale( -1, 1 );
    editor->setWorldMatrix( m );
}

void Main::moveL()
{
    QWMatrix m = editor->worldMatrix();
    m.translate( -16, 0 );
    editor->setWorldMatrix( m );
}

void Main::moveR()
{
    QWMatrix m = editor->worldMatrix();
    m.translate( +16, 0 );
    editor->setWorldMatrix( m );
}

void Main::moveU()
{
    QWMatrix m = editor->worldMatrix();
    m.translate( 0, -16 );
    editor->setWorldMatrix( m );
}

void Main::moveD()
```

```
{
   QWMatrix m = editor->worldMatrix();
   m.translate( 0, +16 );
   editor->setWorldMatrix( m );
}

void Main::print()
{
   if ( !printer ) printer = new QPrinter;
   if ( printer->setup(this) ) {
    QPainter pp(printer);
    canvas.drawArea(QRect(0,0,canvas.width(),canvas.height()),&pp,FALSE);
   }
}

void Main::addSprite()
{
   QCanvasItem* i = new BouncyLogo(&canvas);
   i->setZ(rand()%256);
   i->show();
}

QString butterfly_fn;
QString logo_fn;

void Main::addButterfly()
{
   if ( butterfly_fn.isEmpty() )
    return;
   if ( !butterflyimg ) {
    butterflyimg = new QImage[4];
    butterflyimg[0].load( butterfly_fn );
    butterflyimg[1] = butterflyimg[0].smoothScale( int(butterflyimg[0].width()*0.75),
        int(butterflyimg[0].height()*0.75) );
    butterflyimg[2] = butterflyimg[0].smoothScale( int(butterflyimg[0].width()*0.5),
        int(butterflyimg[0].height()*0.5) );
    butterflyimg[3] = butterflyimg[0].smoothScale( int(butterflyimg[0].width()*0.25),
        int(butterflyimg[0].height()*0.25) );
   }
   QCanvasPolygonalItem* i = new ImageItem(butterflyimg[rand()%4],&canvas);
   i->move(rand()%(canvas.width()-butterflyimg->width()),
       rand()%(canvas.height()-butterflyimg->height()));
   i->setZ(rand()%256+250);
   i->show();
}

void Main::addLogo()
{
   if ( logo_fn.isEmpty() )
    return;
   if ( !logoimg ) {
    logoimg = new QImage[4];
    logoimg[0].load( logo_fn );
    logoimg[1] = logoimg[0].smoothScale( int(logoimg[0].width()*0.75),
         int(logoimg[0].height()*0.75) );
      logoimg[2] = logoimg[0].smoothScale( int(logoimg[0].width()*0.5),
         int(logoimg[0].height()*0.5) );
      logoimg[3] = logoimg[0].smoothScale( int(logoimg[0].width()*0.25),
         int(logoimg[0].height()*0.25) );
   }
   QCanvasPolygonalItem* i = new ImageItem(logoimg[rand()%4],&canvas);
   i->move(rand()%(canvas.width()-logoimg->width()),
      rand()%(canvas.height()-logoimg->width()));
   i->setZ(rand()%256+256);
   i->show();
}

void Main::addCircle()
{
   QCanvasPolygonalItem* i = new QCanvasEllipse(50,50,&canvas);
   i->setBrush( QColor(rand()%32*8,rand()%32*8,rand()%32*8) );
   i->move(rand()%canvas.width(),rand()%canvas.height());
   i->setZ(rand()%256);
   i->show();
}

void Main::addHexagon()
{
   QCanvasPolygon* i = new QCanvasPolygon(&canvas);
   const int size = canvas.width() / 25;
   QPointArray pa(6);
   pa[0] = QPoint(2*size,0);
   pa[1] = QPoint(size,-size*173/100);
   pa[2] = QPoint(-size,-size*173/100);
   pa[3] = QPoint(-2*size,0);
   pa[4] = QPoint(-size,size*173/100);
   pa[5] = QPoint(size,size*173/100);
   i->setPoints(pa);
   i->setBrush( QColor(rand()%32*8,rand()%32*8,rand()%32*8) );
   i->move(rand()%canvas.width(),rand()%canvas.height());
   i->setZ(rand()%256);
   i->show();
}

void Main::addPolygon()
{
   QCanvasPolygon* i = new QCanvasPolygon(&canvas);
   const int size = canvas.width()/2;
   QPointArray pa(6);
   pa[0] = QPoint(0,0);
   pa[1] = QPoint(size,size/5);
   pa[2] = QPoint(size*4/5,size);
   pa[3] = QPoint(size/6,size*5/4);
   pa[4] = QPoint(size*3/4,size*3/4);
   pa[5] = QPoint(size*3/4,size/4);
   i->setPoints(pa);
   i->setBrush( QColor(rand()%32*8,rand()%32*8,rand()%32*8) );
   i->move(rand()%canvas.width(),rand()%canvas.height());
```

```
    i->setZ(rand()%256);
    i->show();
}

void Main::addSpline()
{
    QCanvasSpline* i = new QCanvasSpline(&canvas);
    const int size = canvas.width()/6;
    QPointArray pa(12);
    pa[0] = QPoint(0,0);
    pa[1] = QPoint(size/2,0);
    pa[2] = QPoint(size,size/2);
    pa[3] = QPoint(size,size);
    pa[4] = QPoint(size,size*3/2);
    pa[5] = QPoint(size/2,size*2);
    pa[6] = QPoint(0,size*2);
    pa[7] = QPoint(-size/2,size*2);
    pa[8] = QPoint(size/4,size*3/2);
    pa[9] = QPoint(0,size);
    pa[10]= QPoint(-size/4,size/2);
    pa[11]= QPoint(-size/2,0);
    i->setControlPoints(pa);
    i->setBrush( QColor(rand()%32*8,rand()%32*8,rand()%32*8) );
    i->move(rand()%canvas.width(),rand()%canvas.height());
    i->setZ(rand()%256);
    i->show();
}

void Main::addText()
{
    QCanvasText* i = new QCanvasText(&canvas);
    i->setText("QCanvasText");
    i->move(rand()%canvas.width(),rand()%canvas.height());
    i->setZ(rand()%256);
    i->show();
}

void Main::addLine()
{
    QCanvasLine* i = new QCanvasLine(&canvas);
    i->setPoints( rand()%canvas.width(), rand()%canvas.height(),
            rand()%canvas.width(), rand()%canvas.height() );
    i->setPen( QPen(QColor(rand()%32*8,rand()%32*8,rand()%32*8), 6) );
    i->setZ(rand()%256);
    i->show();
}

void Main::addMesh()
{
    int x0 = 0;
    int y0 = 0;

    if ( !tb ) tb = new QBrush( Qt::red );
    if ( !tp ) tp = new QPen( Qt::black );
    int nodecount = 0;

    int w = canvas.width();
    int h = canvas.height();

    const int dist = 30;
    int rows = h / dist;
    int cols = w / dist;

#ifndef QT_NO_PROGRESSDIALOG
    QProgressDialog progress( "Creating mesh...", "Abort", rows,
                      this, "progress", TRUE );
#endif

    QMemArray<NodeItem*> lastRow(cols);
    for ( int j = 0; j < rows; j++ ) {
      int n = j%2 ? cols-1 : cols;
      NodeItem *prev = 0;
      for ( int i = 0; i < n; i++ ) {
          NodeItem *el = new NodeItem( &canvas );
          nodecount++;
          int r = rand();
          int xrand = r %20;
          int yrand = (r/20) %20;
          el->move( xrand + x0 + i*dist + (j%2 ? dist/2 : 0 ),
                  yrand + y0 + j*dist );

          if ( j > 0 ) {
            if ( i < cols-1 )
                (new EdgeItem( lastRow[i], el, &canvas ))->show();
            if ( j%2 )
                (new EdgeItem( lastRow[i+1], el, &canvas ))->show();
            else if ( i > 0 )
                (new EdgeItem( lastRow[i-1], el, &canvas ))->show();
          }
          if ( prev ) {
            (new EdgeItem( prev, el, &canvas ))->show();
          }
          if ( i > 0 ) lastRow[i-1] = prev;
          prev = el;
          el->show();
      }
      lastRow[n-1]=prev;
#ifndef QT_NO_PROGRESSDIALOG
      progress.setProgress( j );
      if ( progress.wasCancelled() )
          break;
#endif
    }
#ifndef QT_NO_PROGRESSDIALOG
    progress.setProgress( rows );
#endif
    // qDebug( "%d nodes, %d edges", nodecount, EdgeItem::count() );
```

```
}

void Main::addRectangle()
{
   QCanvasPolygonalItem *i = new QCanvasRectangle( rand()%canvas.width(),rand()%canvas.height(),
                canvas.width()/5,canvas.width()/5,&canvas);
   int z = rand()%256;
   i->setBrush( QColor(z,z,z) );
   i->setPen( QPen(QColor(rand()%32*8,rand()%32*8,rand()%32*8), 6) );
   i->setZ(z);
   i->show();
}
```

canvas.h

```
#ifndef EXAMPLE_H
#define EXAMPLE_H

#include <qpopupmenu.h>
#include <qmainwindow.h>
#include <qintdict.h>
#include <qcanvas.h>

class BouncyLogo : public QCanvasSprite {
   void initPos();
   void initSpeed();
public:
   BouncyLogo(QCanvas*);
   void advance(int);
   int rtti() const;
};


class FigureEditor : public QCanvasView {
   Q_OBJECT

public:
   FigureEditor(QCanvas&, QWidget* parent=0, const char* name=0, WFlags f=0);
   void clear();

protected:
   void contentsMousePressEvent(QMouseEvent*);
   void contentsMouseMoveEvent(QMouseEvent*);

signals:
   void status(const QString&);

private:
   QCanvasItem* moving;
   QPoint moving_start;
};

class Main : public QMainWindow {
   Q_OBJECT
public:
   Main(QCanvas&, QWidget* parent=0, const char* name=0, WFlags f=0);
   ~Main();

public slots:
   void help();

private slots:
   void aboutQt();
   void newView();
   void clear();
   void init();

   void addSprite();
   void addCircle();
   void addHexagon();
   void addPolygon();
   void addSpline();
   void addText();
   void addLine();
   void addRectangle();
   void addMesh();
   void addLogo();
   void addButterfly();

   void enlarge();
   void shrink();
   void rotateClockwise();
   void rotateCounterClockwise();
   void zoomIn();
   void zoomOut();
   void mirror();
   void moveL();
   void moveR();
   void moveU();
   void moveD();

   void print();

   void toggleDoubleBuffer();

private:
   QCanvas& canvas;
   FigureEditor *editor;

   QPopupMenu* options;
   QPrinter* printer;
   int dbf_id;
};

#endif
```

blendshadow.cpp

```
#include <qimage.h>
```

```
#include <qcolor.h>

static inline int blendComponent( int v, int av, int s, int as )
{
   return as*s + av*v -(av*as*s)/255;
}

static inline QRgb blendShade( QRgb v, QRgb s )
{
   //shadow image is already reduced and blurred
   int as = qAlpha(s);
   int av = qAlpha(v);
   if ( as == 0 || av == 255 )
     return v;

   int a = as + av -(as*av)/255;

   int r = blendComponent( qRed(v),av, qRed(s), as)/a;
   int g = blendComponent( qGreen(v),av, qGreen(s), as)/a;
   int b = blendComponent( qBlue(v),av, qBlue(s), as)/a;

   return qRgba(r,g,b,a);
}

int main( int*, char**)
{
   QImage image( "out.png" );
   image.convertDepth( 32 );
   QImage shade( "outshade.png" );
   shade.convertDepth( 32 );
   int dx = 10;
   int dy = 5;

   int w = image.width();
   int h = image.height();

   QImage img( w+dx, h+dy, 32 );
   img.setAlphaBuffer( TRUE );

   for ( int y = 0; y < h+dy; y++ ) {
     for ( int x = 0; x < w+dx; x++ ) {
        QRgb sh =  (x<dx||y<dy) ? 0 : shade.pixel( x-dx, y-dy );
        QRgb pixel = (x<w&y<h) ? image.pixel( x, y ) : 0;
        img.setPixel( x, y, blendShade( pixel, sh ) );
      }
   }
   img.save("blend.png", "PNG" );
}
```

makeimg.cpp

```
#include <qimage.h>
#include <qcolor.h>

static inline int blendComponent( int v, int av, int s, int as )
```

```
{
  //shadow gets a color inversely proportional to the
  //alpha value
  s = (s*(255-as))/255;
  //then do standard blending
  return as*s + av*v -(av*as*s)/255;
}

static inline QRgb blendShade( QRgb v, QRgb s )
{
  //pick a number: shadow is 1/3 of object
  int as = qAlpha(s)/3;
  int av = qAlpha(v);
  if ( as == 0 || av == 255 )
   return v;

  int a = as + av -(as*av)/255;


  int r = blendComponent( qRed(v),av, qRed(s), as)/a;
  int g = blendComponent( qGreen(v),av, qGreen(s), as)/a;
  int b = blendComponent( qBlue(v),av, qBlue(s), as)/a;

  return qRgba(r,g,b,a);
}

int main( int*, char**)
{
  QImage *img;

  img = new QImage( "in.png" );
  int w,h;
  int y;
  img->setAlphaBuffer( TRUE );
  *img = img->convertDepth( 32 );
  w = img->width();
  h = img->height();
#if 0
  for ( y = 0; y < h; y ++ ) {
   uint *line = (uint*)img->scanLine( y );
   for ( int x = 0; x < w; x++ ) {
     uint pixel = line[x];
     int r = qRed(pixel);
     int g = qGreen(pixel);
     int b = qBlue(pixel);
     int min = QMIN( r, QMIN( g, b ) );
     int max = QMAX( r, QMAX( g, b ) );
     r -= min;
     g -= min;
     b -= min;
     if ( max !=min ) {
      r = (r*255)/(max-min);
      g = (g*255)/(max-min);
      b = (b*255)/(max-min);
```

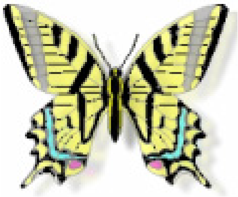
```
            }
            int a = 255-min;
            a -=  (max-min)/3; //hack more transparency for colors.
            line[x] = qRgba( r, g, b, a );
        }
      }
#endif
    *img = img->smoothScale( w/2, h/2 );

    qDebug( "saving out.png");
    img->save( "out.png", "PNG" );

    w = img->width();
    h = img->height();

    QImage *img2 = new QImage( w, h, 32 );
    img2->setAlphaBuffer( TRUE );
    for ( y = 0; y < h; y++ ) {
      for ( int x = 0; x < w; x++ ) {
            QRgb shader = img->pixel( x, y );

            int as = qAlpha(shader)/3;

            int r = (qRed(shader)*(255-as))/255;
            int g = (qGreen(shader)*(255-as))/255;
            int b = (qBlue(shader)*(255-as))/255;

            img2->setPixel( x, y, qRgba(r,g,b,as) );
        }
      }

    img2->save( "outshade.png", "PNG" );

}
```

main.cpp

```
#include <qstatusbar.h>
#include <qmessagebox.h>
#include <qmenubar.h>
#include <qapplication.h>
#include <qimage.h>

#include "canvas.h"

#include <stdlib.h>

extern QString butterfly_fn;
extern QString logo_fn;

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

    if ( argc > 1 )
```

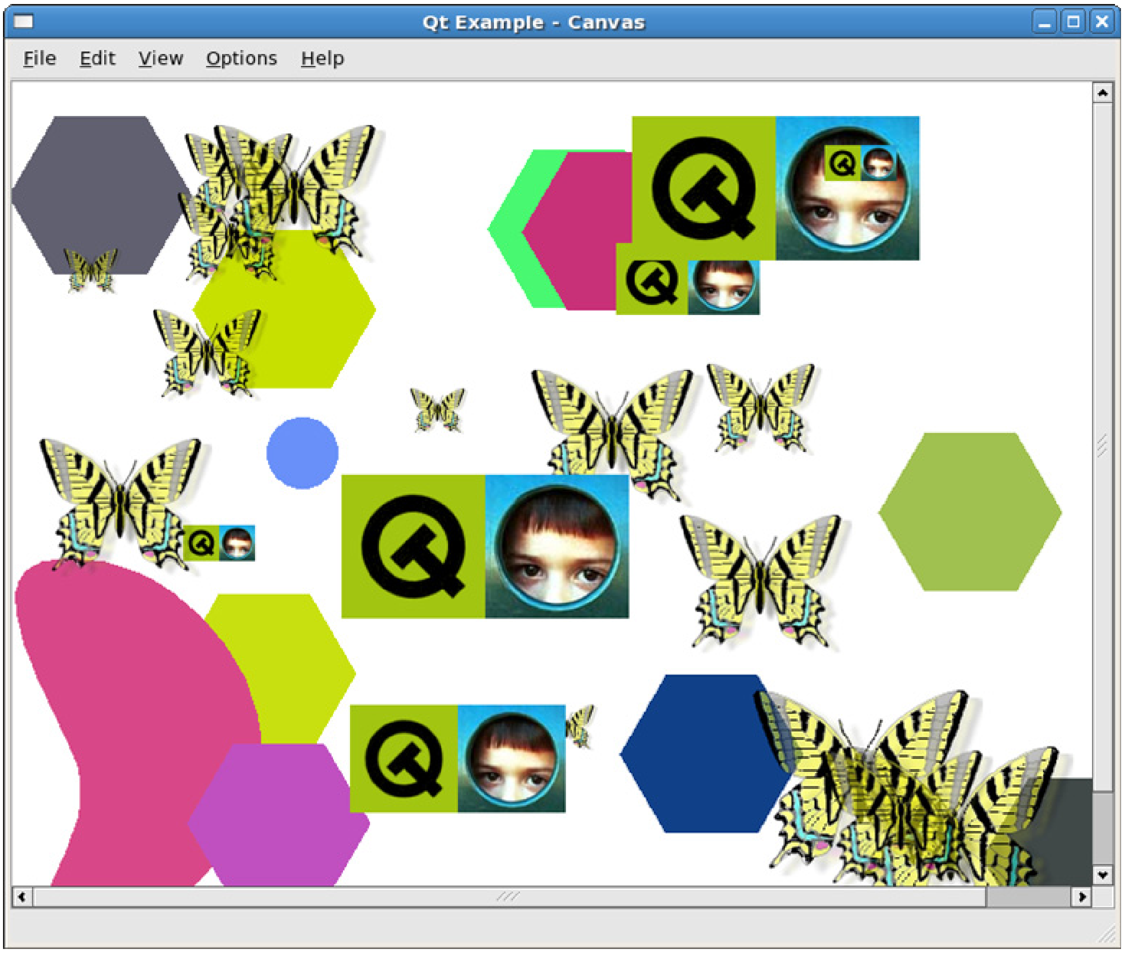
```
     butterfly_fn = argv[1];
   else
     butterfly_fn = "butterfly.png";

   if ( argc > 2 )
     logo_fn = argv[2];
   else
     logo_fn = "qtlogo.png";

   QCanvas canvas(800,600);
   canvas.setAdvancePeriod(30);
   Main m(canvas);
   m.resize(m.sizeHint());
   m.setCaption("Qt Example - Canvas");
   if ( QApplication::desktop()->width() > m.width() + 10
     && QApplication::desktop()->height() > m.height() +30 )
     m.show();
   else
     m.showMaximized();

   QObject::connect( qApp, SIGNAL(lastWindowClosed()), qApp, SLOT(quit()) );

   return app.exec();
}
```

butterfly.png



qtlogo.png

실행



8. 완전한 캔버스응용프로그람

이것은 기본창문, 차림표, 도구띠를 가지는 완전한 실례프로그람이다. 기본창문부품은 QCanvas이고 이 실례는 기본캔버스사용법을 보여준다.

chart.pro

```
TEMPLATE = app
CONFIG  += warn_on
HEADERS += element.h \
        canvastext.h \
        canvasview.h \
        chartform.h \
        optionsform.h \
        setdataform.h
SOURCES += element.cpp \
        canvasview.cpp \
        chartform.cpp \
        chartform_canvas.cpp \
        chartform_files.cpp \
        optionsform.cpp \
        setdataform.cpp \
```

main.cpp

canvastext.h

```
#ifndef CANVASTEXT_H
#define CANVASTEXT_H

#include <qcanvas.h>

class QFont;

class CanvasText : public QCanvasText
{
public:
    enum { CANVAS_TEXT = 1100 };

    CanvasText( int index, QCanvas *canvas )
      : QCanvasText( canvas ), m_index( index ) {}
    CanvasText( int index, const QString& text, QCanvas *canvas )
      : QCanvasText( text, canvas ), m_index( index ) {}
    CanvasText( int index, const QString& text, QFont font, QCanvas *canvas )
      : QCanvasText( text, font, canvas ), m_index( index ) {}

    int index() const { return m_index; }
    void setIndex( int index ) { m_index = index; }

    int rtti() const { return CANVAS_TEXT; }

private:
    int m_index;
};

#endif
```

canvasview.cpp

```
#include "canvasview.h"
#include "chartform.h"

#include <qcursor.h>
#include <qpoint.h>
#include <qpopupmenu.h>
#include <qstatusbar.h>


void CanvasView::contentsContextMenuEvent( QContextMenuEvent * )
{
    ((ChartForm*)parent())->optionsMenu->exec( QCursor::pos() );
}

void CanvasView::viewportResizeEvent( QResizeEvent *e )
{
    canvas()->resize( e->size().width(), e->size().height() );
    ((ChartForm*)parent())->drawElements();
}
```

```
void CanvasView::contentsMousePressEvent( QMouseEvent *e )
{
   QCanvasItemList list = canvas()->collisions( e->pos() );
   for ( QCanvasItemList::iterator it = list.begin(); it != list.end(); ++it )
    if ( (*it)->rtti() == CanvasText::CANVAS_TEXT ) {
        m_movingItem = *it;
        m_pos = e->pos();
        return;
    }
   m_movingItem = 0;
}

void CanvasView::contentsMouseMoveEvent( QMouseEvent *e )
{
   if ( m_movingItem ) {
    QPoint offset = e->pos() - m_pos;
    m_movingItem->moveBy( offset.x(), offset.y() );
    m_pos = e->pos();
    ChartForm *form = (ChartForm*)parent();
    form->setChanged( TRUE );
    int chartType = form->chartType();
    CanvasText *item = (CanvasText*)m_movingItem;
    int i = item->index();

    (*m_elements)[i].setProX( chartType, item->x() / canvas()->width() );
    (*m_elements)[i].setProY( chartType, item->y() / canvas()->height() );

    canvas()->update();
   }
}
```

canvasview.h

```
#ifndef CANVASVIEW_H
#define CANVASVIEW_H

#include "element.h"
#include "canvastext.h"

#include <qcanvas.h>

class QPoint;

class CanvasView : public QCanvasView
{
   Q_OBJECT
public:
   CanvasView( QCanvas *canvas, ElementVector *elements,
          QWidget* parent = 0, const char* name = "canvas view",
          WFlags f = 0 )
    : QCanvasView( canvas, parent, name, f ), m_movingItem(0),
      m_elements( elements ) {}

protected:
   void viewportResizeEvent( QResizeEvent *e );
```

```
    void contentsMousePressEvent( QMouseEvent *e );
    void contentsMouseMoveEvent( QMouseEvent *e );
    void contentsContextMenuEvent( QContextMenuEvent *e );

private:
    QCanvasItem *m_movingItem;
    QPoint m_pos;
    ElementVector *m_elements;
};

#endif
```

chartform.cpp

```
#include "canvasview.h"
#include "chartform.h"
#include "optionsform.h"
#include "setdataform.h"

#include <qaction.h>
#include <qapplication.h>
#include <qcombobox.h>
#include <qfile.h>
#include <qfiledialog.h>
#include <qfont.h>
#include <qfontdialog.h>
#include <qmenubar.h>
#include <qmessagebox.h>
#include <qpixmap.h>
#include <qpopupmenu.h>
#include <qprinter.h>
#include <qradiobutton.h>
#include <qsettings.h>
#include <qspinbox.h>
#include <qstatusbar.h>
#include <qtoolbar.h>
#include <qtoolbutton.h>

#include "images/file_new.xpm"
#include "images/file_open.xpm"
#include "images/file_save.xpm"
#include "images/file_print.xpm"
#include "images/options_setdata.xpm"
#include "images/options_setfont.xpm"
#include "images/options_setoptions.xpm"
#include "images/options_horizontalbarchart.xpm"
#include "images/options_piechart.xpm"
#include "images/options_verticalbarchart.xpm"

const QString WINDOWS_REGISTRY = "/Trolltech/QtExamples";
const QString APP_KEY = "/Chart/";

ChartForm::ChartForm( const QString& filename )
    : QMainWindow( 0, 0, WDestructiveClose )
{
    setIcon( QPixmap( options_piechart ) );

    QAction *fileNewAction;
    QAction *fileOpenAction;
    QAction *fileSaveAction;
    QAction *fileSaveAsAction;
    QAction *fileSaveAsPixmapAction;
    QAction *filePrintAction;
    QAction *fileQuitAction;
    QAction *optionsSetDataAction;
    QAction *optionsSetFontAction;
    QAction *optionsSetOptionsAction;

    fileNewAction = new QAction( "New Chart", QPixmap( file_new ),
        "&New", CTRL+Key_N, this, "new" );
    connect( fileNewAction, SIGNAL( activated() ), this, SLOT( fileNew() ) );

    fileOpenAction = new QAction( "Open Chart", QPixmap( file_open ),
        "&Open...", CTRL+Key_O, this, "open" );
    connect( fileOpenAction, SIGNAL( activated() ), this, SLOT( fileOpen() ) );

    fileSaveAction = new QAction( "Save Chart", QPixmap( file_save ),
        "&Save", CTRL+Key_S, this, "save" );
    connect( fileSaveAction, SIGNAL( activated() ), this, SLOT( fileSave() ) );

    fileSaveAsAction = new QAction( "Save Chart As", QPixmap( file_save ),
        "Save &As...", 0, this, "save as" );
    connect( fileSaveAsAction, SIGNAL( activated() ), this, SLOT( fileSaveAs() ) );

    fileSaveAsPixmapAction = new QAction( "Save Chart As Bitmap", QPixmap( file_save ),
        "Save As &Bitmap...", CTRL+Key_B, this, "save as bitmap" );
    connect( fileSaveAsPixmapAction, SIGNAL( activated() ),  this, SLOT( fileSaveAsPixmap() ) );

    filePrintAction = new QAction( "Print Chart", QPixmap( file_print ),
        "&Print Chart...", CTRL+Key_P, this, "print chart" );
    connect( filePrintAction, SIGNAL( activated() ),  this, SLOT( filePrint() ) );

    optionsSetDataAction = new QAction( "Set Data", QPixmap( options_setdata ),
        "Set &Data...", CTRL+Key_D, this, "set data" );
    connect( optionsSetDataAction, SIGNAL( activated() ), this, SLOT( optionsSetData() ) );

    QActionGroup *chartGroup = new QActionGroup( this ); // Connected later
    chartGroup->setExclusive( TRUE );

    optionsPieChartAction = new QAction( "Pie Chart", QPixmap( options_piechart ),
        "&Pie Chart", CTRL+Key_I, chartGroup, "pie chart" );
    optionsPieChartAction->setToggleAction( TRUE );

    optionsHorizontalBarChartAction = new QAction( "Horizontal Bar Chart",
QPixmap( options_horizontalbarchart ),  "&Horizontal Bar Chart", CTRL+Key_H, chartGroup,
        "horizontal bar chart" );
    optionsHorizontalBarChartAction->setToggleAction( TRUE );

    optionsVerticalBarChartAction = new QAction(
        "Vertical Bar Chart", QPixmap( options_verticalbarchart ),
        "&Vertical Bar Chart", CTRL+Key_V, chartGroup, "Vertical bar chart" );
    optionsVerticalBarChartAction->setToggleAction( TRUE );

    optionsSetFontAction = new QAction( "Set Font", QPixmap( options_setfont ),
        "Set &Font...", CTRL+Key_F, this, "set font" );
    connect( optionsSetFontAction, SIGNAL( activated() ),  this, SLOT( optionsSetFont() ) );

    optionsSetOptionsAction = new QAction( "Set Options", QPixmap( options_setoptions ),
        "Set &Options...", 0, this, "set options" );
    connect( optionsSetOptionsAction, SIGNAL( activated() ),  this, SLOT( optionsSetOptions() ) );

    fileQuitAction = new QAction( "Quit", "&Quit", CTRL+Key_Q, this, "quit" );
    connect( fileQuitAction, SIGNAL( activated() ), this, SLOT( fileQuit() ) );

    QToolBar* fileTools = new QToolBar( this, "file operations" );
    fileTools->setLabel( "File Operations" );
    fileNewAction->addTo( fileTools );
    fileOpenAction->addTo( fileTools );
    fileSaveAction->addTo( fileTools );
    fileTools->addSeparator();
    filePrintAction->addTo( fileTools );

    QToolBar *optionsTools = new QToolBar( this, "options operations" );
    optionsTools->setLabel( "Options Operations" );
    optionsSetDataAction->addTo( optionsTools );
    optionsTools->addSeparator();
    optionsPieChartAction->addTo( optionsTools );
    optionsHorizontalBarChartAction->addTo( optionsTools );
    optionsVerticalBarChartAction->addTo( optionsTools );
    optionsTools->addSeparator();
    optionsSetFontAction->addTo( optionsTools );
    optionsTools->addSeparator();
    optionsSetOptionsAction->addTo( optionsTools );

    fileMenu = new QPopupMenu( this );
    menuBar()->insertItem( "&File", fileMenu );
    fileNewAction->addTo( fileMenu );
    fileOpenAction->addTo( fileMenu );
    fileSaveAction->addTo( fileMenu );
    fileSaveAsAction->addTo( fileMenu );
    fileMenu->insertSeparator();
    fileSaveAsPixmapAction->addTo( fileMenu );
    fileMenu->insertSeparator();
    filePrintAction->addTo( fileMenu );
    fileMenu->insertSeparator();
    fileQuitAction->addTo( fileMenu );

    optionsMenu = new QPopupMenu( this );
    menuBar()->insertItem( "&Options", optionsMenu );
    optionsSetDataAction->addTo( optionsMenu );
    optionsMenu->insertSeparator();
    optionsPieChartAction->addTo( optionsMenu );
    optionsHorizontalBarChartAction->addTo( optionsMenu );
```

```
optionsVerticalBarChartAction->addTo( optionsMenu );
optionsMenu->insertSeparator();
optionsSetFontAction->addTo( optionsMenu );
optionsMenu->insertSeparator();
optionsSetOptionsAction->addTo( optionsMenu );

menuBar()->insertSeparator();

QPopupMenu *helpMenu = new QPopupMenu( this );
menuBar()->insertItem( "&Help", helpMenu );
helpMenu->insertItem( "&Help", this, SLOT(helpHelp()), Key_F1 );
helpMenu->insertItem( "&About", this, SLOT(helpAbout()) );
helpMenu->insertItem( "About &Qt", this, SLOT(helpAboutQt()) );

m_printer = 0;
m_elements.resize( MAX_ELEMENTS );

QSettings settings;
settings.insertSearchPath( QSettings::Windows, WINDOWS_REGISTRY );
int windowWidth = settings.readNumEntry( APP_KEY + "WindowWidth", 460 );
int windowHeight = settings.readNumEntry( APP_KEY + "WindowHeight", 530 );
int windowX = settings.readNumEntry( APP_KEY + "WindowX", -1 );
int windowY = settings.readNumEntry( APP_KEY + "WindowY", -1 );
setChartType( ChartType( settings.readNumEntry( APP_KEY + "ChartType", int(PIE) ) ) );
m_addValues = AddValuesType( settings.readNumEntry( APP_KEY + "AddValues", int(NO) ));
m_decimalPlaces = settings.readNumEntry( APP_KEY + "Decimals", 2 );
m_font = QFont( "Helvetica", 18, QFont::Bold );
m_font.fromString( settings.readEntry( APP_KEY + "Font", m_font.toString() ) );
for ( int i = 0; i < MAX_RECENTFILES; ++i ) {
 QString filename = settings.readEntry( APP_KEY + "File" + QString::number( i + 1 ) );
 if ( !filename.isEmpty() )
     m_recentFiles.push_back( filename );
}
if ( m_recentFiles.count() )
 updateRecentFilesMenu();

// Connect *after* we've set the chart type on so we don't call drawElements() prematurely.
connect( chartGroup, SIGNAL( selected(QAction*) ),  this, SLOT( updateChartType(QAction*) ) );

resize( windowWidth, windowHeight );
if ( windowX != -1 || windowY != -1 )
 move( windowX, windowY );

m_canvas = new QCanvas( this );
m_canvas->resize( width(), height() );
m_canvasView = new CanvasView( m_canvas, &m_elements, this );
setCentralWidget( m_canvasView );
m_canvasView->show();

if ( !filename.isEmpty() )
 load( filename );
else {
 init();
 m_elements[0].set( 20, red,    14, "Red" );
```

```
        m_elements[1].set( 70, cyan,    2, "Cyan",   darkGreen );
        m_elements[2].set( 35, blue,   11, "Blue" );
        m_elements[3].set( 55, yellow,  1, "Yellow", darkBlue );
        m_elements[4].set( 80, magenta, 1, "Magenta" );
        drawElements();
    }

    statusBar()->message( "Ready", 2000 );
}

ChartForm::~ChartForm()
{
    delete m_printer;
}

void ChartForm::init()
{
    setCaption( "Chart" );
    m_filename = QString::null;
    m_changed = FALSE;

    m_elements[0]  = Element( Element::INVALID, red );
    m_elements[1]  = Element( Element::INVALID, cyan );
    m_elements[2]  = Element( Element::INVALID, blue );
    m_elements[3]  = Element( Element::INVALID, yellow );
    m_elements[4]  = Element( Element::INVALID, green );
    m_elements[5]  = Element( Element::INVALID, magenta );
    m_elements[6]  = Element( Element::INVALID, darkYellow );
    m_elements[7]  = Element( Element::INVALID, darkRed );
    m_elements[8]  = Element( Element::INVALID, darkCyan );
    m_elements[9]  = Element( Element::INVALID, darkGreen );
    m_elements[10] = Element( Element::INVALID, darkMagenta );
    m_elements[11] = Element( Element::INVALID, darkBlue );
    for ( int i = 12; i < MAX_ELEMENTS; ++i ) {
        double x = (double(i) / MAX_ELEMENTS) * 360;
        int y = (int(x * 256) % 105) + 151;
        int z = ((i * 17) % 105) + 151;
        m_elements[i] = Element( Element::INVALID, QColor( int(x), y, z, QColor::Hsv ) );
    }
}

void ChartForm::closeEvent( QCloseEvent * )
{
    fileQuit();
}

void ChartForm::fileNew()
{
    if ( okToClear() ) {
        init();
        drawElements();
    }
}
```

```
void ChartForm::fileOpen()
{
  if ( !okToClear() )
    return;

  QString filename = QFileDialog::getOpenFileName(  QString::null, "Charts (*.cht)", this,
                  "file open", "Chart -- File Open" );
  if ( !filename.isEmpty() )
    load( filename );
  else
    statusBar()->message( "File Open abandoned", 2000 );
}

void ChartForm::fileSaveAs()
{
  QString filename = QFileDialog::getSaveFileName( QString::null, "Charts (*.cht)", this,
                  "file save as", "Chart -- File Save As" );
  if ( !filename.isEmpty() ) {
    int answer = 0;
    if ( QFile::exists( filename ) )
        answer = QMessageBox::warning( this, "Chart -- Overwrite File",
                  QString( "Overwrite\n\'%1\'?" ).arg( filename ),  "&Yes", "&No", QString::null, 1, 1 );
    if ( answer == 0 ) {
        m_filename = filename;
        updateRecentFiles( filename );
        fileSave();
        return;
    }
  }
  statusBar()->message( "Saving abandoned", 2000 );
}

void ChartForm::fileOpenRecent( int index )
{
  if ( !okToClear() )
    return;

  load( m_recentFiles[index] );
}

void ChartForm::updateRecentFiles( const QString& filename )
{
  if ( m_recentFiles.find( filename ) != m_recentFiles.end() )
    return;

  m_recentFiles.push_back( filename );
  if ( m_recentFiles.count() > MAX_RECENTFILES )
    m_recentFiles.pop_front();

  updateRecentFilesMenu();
}

void ChartForm::updateRecentFilesMenu()
{
    for ( int i = 0; i < MAX_RECENTFILES; ++i ) {
     if ( fileMenu->findItem( i ) )
        fileMenu->removeItem( i );
     if ( i < int(m_recentFiles.count()) )
        fileMenu->insertItem( QString( "&%1 %2" ). arg( i + 1 ).arg( m_recentFiles[i] ),
                   this, SLOT( fileOpenRecent(int) ), 0, i );
    }
}

void ChartForm::fileQuit()
{
    if ( okToClear() ) {
     saveOptions();
       qApp->exit( 0 );
    }
}

bool ChartForm::okToClear()
{
    if ( m_changed ) {
     QString msg;
     if ( m_filename.isEmpty() )
        msg = "Unnamed chart ";
     else
        msg = QString( "Chart '%1'\n" ).arg( m_filename );
     msg += "has been changed.";

     int x = QMessageBox::information( this, "Chart -- Unsaved Changes",
                   msg, "&Save", "Cancel", "&Abandon",  0, 1 );
     switch( x ) {
        case 0: // Save
         fileSave();
         break;
        case 1: // Cancel
        default:
         return FALSE;
        case 2: // Abandon
         break;
     }
    }

    return TRUE;
}

void ChartForm::saveOptions()
{
    QSettings settings;
    settings.insertSearchPath( QSettings::Windows, WINDOWS_REGISTRY );
    settings.writeEntry( APP_KEY + "WindowWidth", width() );
    settings.writeEntry( APP_KEY + "WindowHeight", height() );
    settings.writeEntry( APP_KEY + "WindowX", x() );
    settings.writeEntry( APP_KEY + "WindowY", y() );
    settings.writeEntry( APP_KEY + "ChartType", int(m_chartType) );
    settings.writeEntry( APP_KEY + "AddValues", int(m_addValues) );
    settings.writeEntry( APP_KEY + "Decimals", m_decimalPlaces );
    settings.writeEntry( APP_KEY + "Font", m_font.toString() );
    for ( int i = 0; i < int(m_recentFiles.count()); ++i )
      settings.writeEntry( APP_KEY + "File" + QString::number( i + 1 ),  m_recentFiles[i] );
}

void ChartForm::optionsSetData()
{
    SetDataForm *setDataForm = new SetDataForm( &m_elements, m_decimalPlaces, this );
    if ( setDataForm->exec() ) {
      m_changed = TRUE;
      drawElements();
    }
    delete setDataForm;
}

void ChartForm::setChartType( ChartType chartType )
{
    m_chartType = chartType;
    switch ( m_chartType ) {
      case PIE:
          optionsPieChartAction->setOn( TRUE );
          break;
      case VERTICAL_BAR:
          optionsVerticalBarChartAction->setOn( TRUE );
          break;
      case HORIZONTAL_BAR:
          optionsHorizontalBarChartAction->setOn( TRUE );
          break;
    }
}

void ChartForm::updateChartType( QAction *action )
{
    if ( action == optionsPieChartAction ) {
      m_chartType = PIE;
    }
    else if ( action == optionsHorizontalBarChartAction ) {
      m_chartType = HORIZONTAL_BAR;
    }
    else if ( action == optionsVerticalBarChartAction ) {
      m_chartType = VERTICAL_BAR;
    }

    drawElements();
}

void ChartForm::optionsSetFont()
{
    bool ok;
    QFont font = QFontDialog::getFont( &ok, m_font, this );
    if ( ok ) {
      m_font = font;
      drawElements();
    }
}

void ChartForm::optionsSetOptions()
{
    OptionsForm *optionsForm = new OptionsForm( this );
    optionsForm->chartTypeComboBox->setCurrentItem( m_chartType );
    optionsForm->setFont( m_font );
    switch ( m_addValues ) {
     case NO:
         optionsForm->noRadioButton->setChecked( TRUE );
         break;
     case YES:
         optionsForm->yesRadioButton->setChecked( TRUE );
         break;
     case AS_PERCENTAGE:
         optionsForm->asPercentageRadioButton->setChecked( TRUE );
         break;
    }
    optionsForm->decimalPlacesSpinBox->setValue( m_decimalPlaces );
    if ( optionsForm->exec() ) {
     setChartType( ChartType(optionsForm->chartTypeComboBox->currentItem()) );
     m_font = optionsForm->font();
     if ( optionsForm->noRadioButton->isChecked() )
         m_addValues = NO;
     else if ( optionsForm->yesRadioButton->isChecked() )
         m_addValues = YES;
     else if ( optionsForm->asPercentageRadioButton->isChecked() )
         m_addValues = AS_PERCENTAGE;
     m_decimalPlaces = optionsForm->decimalPlacesSpinBox->value();
     drawElements();
    }
    delete optionsForm;
}

void ChartForm::helpHelp()
{
    statusBar()->message( "Help is not implemented yet", 2000 );
}

void ChartForm::helpAbout()
{
    QMessageBox::about( this, "Chart -- About",
             "<center><h1><font color=blue>Chart<font></h1></center>"
             "<p>Chart your data with <i>chart</i>.</p>"
             );
}

void ChartForm::helpAboutQt()
{
    QMessageBox::aboutQt( this, "Chart -- About Qt" );
}
```

chartform.h

```
#ifndef CHARTFORM_H
#define CHARTFORM_H

#include "element.h"
#include <qmainwindow.h>
#include <qstringlist.h>

class CanvasView;
class QAction;
class QCanvas;
class QFont;
class QPrinter;
class QString;

class ChartForm: public QMainWindow
{
   Q_OBJECT
public:
   enum { MAX_ELEMENTS = 100 };
   enum { MAX_RECENTFILES = 9 }; // Must not exceed 9
   enum ChartType { PIE, VERTICAL_BAR, HORIZONTAL_BAR };
   enum AddValuesType { NO, YES, AS_PERCENTAGE };

   ChartForm( const QString& filename );
   ~ChartForm();

   int chartType() { return m_chartType; }
   void setChanged( bool changed = TRUE ) { m_changed = changed; }
   void drawElements();

   QPopupMenu *optionsMenu; // Why public? See canvasview.cpp

protected:
   virtual void closeEvent( QCloseEvent * );

private slots:
   void fileNew();
   void fileOpen();
   void fileOpenRecent( int index );
   void fileSave();
   void fileSaveAs();
   void fileSaveAsPixmap();
   void filePrint();
   void fileQuit();
   void optionsSetData();
   void updateChartType( QAction *action );
   void optionsSetFont();
   void optionsSetOptions();
   void helpHelp();
   void helpAbout();
   void helpAboutQt();
   void saveOptions();
private:
   void init();
   void load( const QString& filename );
   bool okToClear();
   void drawPieChart( const double scales[], double total, int count );
   void drawVerticalBarChart( const double scales[], double total, int count );
   void drawHorizontalBarChart( const double scales[], double total, int count );

   QString valueLabel( const QString& label, double value, double total );
   void updateRecentFiles( const QString& filename );
   void updateRecentFilesMenu();
   void setChartType( ChartType chartType );

   QPopupMenu *fileMenu;
   QAction *optionsPieChartAction;
   QAction *optionsHorizontalBarChartAction;
   QAction *optionsVerticalBarChartAction;

   QString m_filename;
   QStringList m_recentFiles;
   QCanvas *m_canvas;
   CanvasView *m_canvasView;
   bool m_changed;
   ElementVector m_elements;
   QPrinter *m_printer;
   ChartType m_chartType;
   AddValuesType m_addValues;
   int m_decimalPlaces;
   QFont m_font;
};

#endif
```

chartform_canvas.cpp

```
#include "canvastext.h"
#include "chartform.h"
#include <qbrush.h>
#include <qcanvas.h>
#include <math.h> // sin, cos

#ifndef M_PI
#define M_PI 3.1415
#endif

void ChartForm::drawElements()
{
    QCanvasItemList list = m_canvas->allItems();
    for ( QCanvasItemList::iterator it = list.begin(); it != list.end(); ++it )
      delete *it;

      // 360 * 16 for pies; Qt works with 16ths of degrees
    int scaleFactor = m_chartType == PIE ? 5760 :
                m_chartType == VERTICAL_BAR ? m_canvas->height() :  m_canvas->width();
    double biggest = 0.0;
    int count = 0;
    double total = 0.0;
    static double scales[MAX_ELEMENTS];

    for ( int i = 0; i < MAX_ELEMENTS; ++i ) {
      if ( m_elements[i].isValid() ) {
        double value = m_elements[i].value();
        count++;
        total += value;
        if ( value > biggest )
          biggest = value;
        scales[i] = m_elements[i].value() * scaleFactor;
      }
    }

    if ( count ) {
        // 2nd loop because of total and biggest
      for ( int i = 0; i < MAX_ELEMENTS; ++i )
        if ( m_elements[i].isValid() )
          if ( m_chartType == PIE )
            scales[i] = (m_elements[i].value() * scaleFactor) / total;
          else
            scales[i] = (m_elements[i].value() * scaleFactor) / biggest;

      switch ( m_chartType ) {
        case PIE:
          drawPieChart( scales, total, count );
          break;
        case VERTICAL_BAR:
          drawVerticalBarChart( scales, total, count );
          break;
        case HORIZONTAL_BAR:
          drawHorizontalBarChart( scales, total, count );
          break;
      }
    }

    m_canvas->update();
}

void ChartForm::drawPieChart( const double scales[], double total, int )
{
    double width = m_canvas->width();
    double height = m_canvas->height();
    int size = int(width > height ? height : width);
    int x = int(width / 2);
    int y = int(height / 2);
    int angle = 0;

    for ( int i = 0; i < MAX_ELEMENTS; ++i ) {
      if ( m_elements[i].isValid() ) {
        int extent = int(scales[i]);
        QCanvasEllipse *arc = new QCanvasEllipse(  size, size, angle, extent, m_canvas );
        arc->setX( x );
        arc->setY( y );
        arc->setZ( 0 );
       arc->setBrush( QBrush( m_elements[i].valueColor(), BrushStyle(m_elements[i].valuePattern()) ) );
        arc->show();
        angle += extent;
        QString label = m_elements[i].label();
        if ( !label.isEmpty() || m_addValues != NO ) {
          label = valueLabel( label, m_elements[i].value(), total );
          CanvasText *text = new CanvasText( i, label, m_font, m_canvas );
          double proX = m_elements[i].proX( PIE );
          double proY = m_elements[i].proY( PIE );
          if ( proX < 0 || proY < 0 ) {
              // Find the centre of the pie segment
              QRect rect = arc->boundingRect();
              proX = ( rect.width() / 2 ) + rect.x();
              proY = ( rect.height() / 2 ) + rect.y();
              // Centre text over the centre of the pie segment
              rect = text->boundingRect();
              proX -= ( rect.width() / 2 );
              proY -= ( rect.height() / 2 );
              // Make proportional
              proX /= width;
              proY /= height;
          }
          text->setColor( m_elements[i].labelColor() );
          text->setX( proX * width );
          text->setY( proY * height );
          text->setZ( 1 );
          text->show();
          m_elements[i].setProX( PIE, proX );
          m_elements[i].setProY( PIE, proY );
        }
      }
    }
}

void ChartForm::drawVerticalBarChart(
     const double scales[], double total, int count )
{
    double width = m_canvas->width();
    double height = m_canvas->height();
    int prowidth = int(width / count);
    int x = 0;
    QPen pen;
    pen.setStyle( NoPen );

    for ( int i = 0; i < MAX_ELEMENTS; ++i ) {
      if ( m_elements[i].isValid() ) {
        int extent = int(scales[i]);
        int y = int(height - extent);
        QCanvasRectangle *rect = new QCanvasRectangle( x, y, prowidth, extent, m_canvas );
       rect->setBrush( QBrush( m_elements[i].valueColor(),BrushStyle(m_elements[i].valuePattern()) ) );
        rect->setPen( pen );
        rect->setZ( 0 );
        rect->show();
        QString label = m_elements[i].label();
        if ( !label.isEmpty() || m_addValues != NO ) {
          double proX = m_elements[i].proX( VERTICAL_BAR );
          double proY = m_elements[i].proY( VERTICAL_BAR );
          if ( proX < 0 || proY < 0 ) {
              proX = x / width;
              proY = y / height;
          }
          label = valueLabel( label, m_elements[i].value(), total );
          CanvasText *text = new CanvasText( i, label, m_font, m_canvas );
          text->setColor( m_elements[i].labelColor() );
          text->setX( proX * width );
          text->setY( proY * height );
          text->setZ( 1 );
          text->show();
          m_elements[i].setProX( VERTICAL_BAR, proX );
          m_elements[i].setProY( VERTICAL_BAR, proY );
        }
        x += prowidth;
      }
    }
}

void ChartForm::drawHorizontalBarChart(const double scales[], double total, int count )
{
    double width = m_canvas->width();
    double height = m_canvas->height();
    int proheight = int(height / count);
    int y = 0;
    QPen pen;
    pen.setStyle( NoPen );

    for ( int i = 0; i < MAX_ELEMENTS; ++i ) {
      if ( m_elements[i].isValid() ) {
        int extent = int(scales[i]);
        QCanvasRectangle *rect = new QCanvasRectangle(  0, y, extent, proheight, m_canvas );
        rect->setBrush( QBrush( m_elements[i].valueColor(),
                        BrushStyle(m_elements[i].valuePattern()) ) );
        rect->setPen( pen );
        rect->setZ( 0 );
        rect->show();
        QString label = m_elements[i].label();
        if ( !label.isEmpty() || m_addValues != NO ) {
          double proX = m_elements[i].proX( HORIZONTAL_BAR );
          double proY = m_elements[i].proY( HORIZONTAL_BAR );
          if ( proX < 0 || proY < 0 ) {
              proX = 0;
              proY = y / height;
          }
          label = valueLabel( label, m_elements[i].value(), total );
          CanvasText *text = new CanvasText( i, label, m_font, m_canvas );
          text->setColor( m_elements[i].labelColor() );
          text->setX( proX * width );
```

```
            text->setY( proY * height );
            text->setZ( 1 );
            text->show();
            m_elements[i].setProX( HORIZONTAL_BAR, proX );
            m_elements[i].setProY( HORIZONTAL_BAR, proY );
          }
          y += proheight;
      }
    }
}

QString ChartForm::valueLabel( const QString& label, double value, double total )
{
    if ( m_addValues == NO )
      return label;

    QString newLabel = label;
    if ( !label.isEmpty() )
      if ( m_chartType == VERTICAL_BAR )
          newLabel += '\n';
      else
          newLabel += ' ';
    if ( m_addValues == YES )
      newLabel += QString::number( value, 'f', m_decimalPlaces );
    else if ( m_addValues == AS_PERCENTAGE )
      newLabel += QString::number( (value / total) * 100, 'f', m_decimalPlaces )  + '%';
    return newLabel;
}
```

chartform_files.cpp

```
#include "canvasview.h"
#include "chartform.h"
#include <qfile.h>
#include <qfiledialog.h>
#include <qpainter.h>
#include <qprinter.h>
#include <qstatusbar.h>

void ChartForm::load( const QString& filename )
{
    QFile file( filename );
    if ( !file.open( IO_ReadOnly ) ) {
      statusBar()->message( QString( "Failed to load \'%1\'" ).arg( filename ), 2000 );
      return;
    }

    init(); // Make sure we have colours
    m_filename = filename;
    QTextStream ts( &file );
    Element element;
    int errors = 0;
    int i = 0;
    while ( !ts.eof() ) {
      ts >> element;
```

```
        if ( element.isValid() )
            m_elements[i++] = element;
        else
            errors++;
        if ( i == MAX_ELEMENTS ) {
            statusBar()->message(
              QString( "Read maximum number of elements (%1)"
                       " discarding others" ).arg( i ), 2000 );
            break;
        }
    }

    file.close();

    QString bad = "";
    if ( errors ) {
        bad = QString( "; skipped " ) + QString::number( errors ) + " bad record";
        if ( errors > 1 )
            bad += "s";
    }
    statusBar()->message( QString( "Read %1 values from \'%2\'%3" ).
                arg( i ).arg( filename ).arg( bad ), 3000 );

    setCaption( QString( "Chart -- %1" ).arg( filename ) );
    updateRecentFiles( filename );

    drawElements();
    m_changed = FALSE;
}

void ChartForm::fileSave()
{
    if ( m_filename.isEmpty() ) {
        fileSaveAs();
        return;
    }

    QFile file( m_filename );
    if ( !file.open( IO_WriteOnly ) ) {
        statusBar()->message( QString( "Failed to save \'%1\'" ).arg( m_filename ), 2000 );
        return;
    }
    QTextStream ts( &file );
    for ( int i = 0; i < MAX_ELEMENTS; ++i )
        if ( m_elements[i].isValid() )
            ts << m_elements[i];

    file.close();

    setCaption( QString( "Chart -- %1" ).arg( m_filename ) );
    statusBar()->message( QString( "Saved \'%1\'" ).arg( m_filename ), 2000 );
    m_changed = FALSE;
}
```

```
void ChartForm::fileSaveAsPixmap()
{
   QString filename = QFileDialog::getSaveFileName( QString::null, "Images (*.png *.xpm *.jpg)",
                  this, "file save as bitmap",   "Chart -- File Save As Bitmap" );
   if ( QPixmap::grabWidget( m_canvasView ).
       save( filename,  filename.mid( filename.findRev( '.' ) + 1 ).upper() ) )
    statusBar()->message( QString( "Wrote \"%1\"" ).arg( filename ), 2000 );
   else
    statusBar()->message( QString( "Failed to write \"%1\"" ).arg( filename ), 2000 );
}

void ChartForm::filePrint()
{
   if ( !m_printer )
    m_printer = new QPrinter;
   if ( m_printer->setup() ) {
    QPainter painter( m_printer );
    m_canvas->drawArea( QRect( 0, 0, m_canvas->width(), m_canvas->height() ),
                &painter, FALSE );
    if ( !m_printer->outputFileName().isEmpty() )
       statusBar()->message( QString( "Printed \"%1\"" ). arg( m_printer->outputFileName() ), 2000 );
   }
}
```

element.cpp

```
#include "element.h"

#include <qstringlist.h>
#include <qtextstream.h>

const char FIELD_SEP = ':';
const char PROPOINT_SEP = ';';
const char XY_SEP = ',';

void Element::init( double value, QColor valueColor, int valuePattern,
           const QString& label, QColor labelColor )
{
   m_value = value;
   m_valueColor = valueColor;
   if ( valuePattern < Qt::SolidPattern || valuePattern > Qt::DiagCrossPattern )
    valuePattern = Qt::SolidPattern;
   m_valuePattern = valuePattern;
   m_label = label;
   m_labelColor = labelColor;
}

void Element::setValuePattern( int valuePattern )
{
   if ( valuePattern < Qt::SolidPattern || valuePattern > Qt::DiagCrossPattern )
    valuePattern = Qt::SolidPattern;
   m_valuePattern = valuePattern;
}

double Element::proX( int index ) const
```

```
{
  Q_ASSERT(index >= 0 && index < MAX_PROPOINTS);
  return m_propoints[2 * index];
}

double Element::proY( int index ) const
{
  Q_ASSERT(index >= 0 && index < MAX_PROPOINTS);
  return m_propoints[(2 * index) + 1];
}

void Element::setProX( int index, double value )
{
  Q_ASSERT(index >= 0 && index < MAX_PROPOINTS);
  m_propoints[2 * index] = value;
}

void Element::setProY( int index, double value )
{
  Q_ASSERT(index >= 0 && index < MAX_PROPOINTS);
  m_propoints[(2 * index) + 1] = value;
}

QTextStream &operator<<( QTextStream &s, const Element &element )
{
  s << element.value() << FIELD_SEP
   << element.valueColor().name() << FIELD_SEP
   << element.valuePattern() << FIELD_SEP
   << element.labelColor().name() << FIELD_SEP;

  for ( int i = 0; i < Element::MAX_PROPOINTS; ++i ) {
   s << element.proX( i ) << XY_SEP << element.proY( i );
   s << ( i == Element::MAX_PROPOINTS - 1 ? FIELD_SEP : PROPOINT_SEP );
  }

  s << element.label() << '\n';

  return s;
}

QTextStream &operator>>( QTextStream &s, Element &element )
{
  QString data = s.readLine();
  element.setValue( Element::INVALID );

  int errors = 0;
  bool ok;

  QStringList fields = QStringList::split( FIELD_SEP, data );
  if ( fields.count() >= 4 ) {
   double value = fields[0].toDouble( &ok );
   if ( !ok )
     errors++;
   QColor valueColor = QColor( fields[1] );
   if ( !valueColor.isValid() )
     errors++;
   int valuePattern = fields[2].toInt( &ok );
   if ( !ok )
     errors++;
   QColor labelColor = QColor( fields[3] );
   if ( !labelColor.isValid() )
     errors++;
   QStringList propoints = QStringList::split( PROPOINT_SEP, fields[4] );
   QString label = data.section( FIELD_SEP, 5 );

   if ( !errors ) {
     element.set( value, valueColor, valuePattern, label, labelColor );
     int i = 0;
     for ( QStringList::iterator point = propoints.begin();
      i < Element::MAX_PROPOINTS && point != propoints.end();
      ++i, ++point ) {
      errors = 0;
      QStringList xy = QStringList::split( XY_SEP, *point );
      double x = xy[0].toDouble( &ok );
      if ( !ok || x <= 0.0 || x >= 1.0 )
        errors++;
      double y = xy[1].toDouble( &ok );
      if ( !ok || y <= 0.0 || y >= 1.0 )
        errors++;
      if ( errors )
        x = y = Element::NO_PROPORTION;
      element.setProX( i, x );
      element.setProY( i, y );
     }
   }
  }

  return s;
}
```

element.h

```
#ifndef ELEMENT_H
#define ELEMENT_H

#include <qcolor.h>
#include <qnamespace.h>
#include <qstring.h>
#include <qvaluevector.h>

class Element;

typedef QValueVector<Element> ElementVector;

/*
  Elements are valid if they have a value which is > EPSILON.
*/
const double EPSILON = 0.0000001; // Must be > INVALID.
```

```
class Element
{
public:
    enum { INVALID = -1 };
    enum { NO_PROPORTION = -1 };
    enum { MAX_PROPOINTS = 3 }; // One proportional point per chart type

    Element( double value = INVALID, QColor valueColor = Qt::gray,
           int valuePattern = Qt::SolidPattern,
           const QString& label = QString::null,
           QColor labelColor = Qt::black ) {
      init( value, valueColor, valuePattern, label, labelColor );
      for ( int i = 0; i < MAX_PROPOINTS * 2; ++i )
          m_propoints[i] = NO_PROPORTION;
    }
    ~Element() {}

    bool isValid() const { return m_value > EPSILON; }

    double value() const { return m_value; }
    QColor valueColor() const { return m_valueColor; }
    int valuePattern() const { return m_valuePattern; }
    QString label() const { return m_label; }
    QColor labelColor() const { return m_labelColor; }
    double proX( int index ) const;
    double proY( int index ) const;

    void set( double value = INVALID, QColor valueColor = Qt::gray,
            int valuePattern = Qt::SolidPattern,
            const QString& label = QString::null,
            QColor labelColor = Qt::black ) {
      init( value, valueColor, valuePattern, label, labelColor );
    }
    void setValue( double value ) { m_value = value; }
    void setValueColor( QColor valueColor ) { m_valueColor = valueColor; }
    void setValuePattern( int valuePattern );
    void setLabel( const QString& label ) { m_label = label; }
    void setLabelColor( QColor labelColor ) { m_labelColor = labelColor; }
    void setProX( int index, double value );
    void setProY( int index, double value );

#ifdef Q_FULL_TEMPLATE_INSTANTIATION
    // xlC 3.x workaround
    Q_DUMMY_COMPARISON_OPERATOR(Element)
    bool operator!=( const Element& e) const {
      return ( !(e == *this) );
    }
#endif

private:
    void init( double value, QColor valueColor, int valuePattern,
             const QString& label, QColor labelColor );

    double m_value;
    QColor m_valueColor;
    int m_valuePattern;
    QString m_label;
    QColor m_labelColor;
    double m_propoints[2 * MAX_PROPOINTS];
};

QTextStream &operator<<( QTextStream&, const Element& );
QTextStream &operator>>( QTextStream&, Element& );

#endif
```

optionsform.cpp

```
#include "optionsform.h"

#include <qbuttongroup.h>
#include <qcombobox.h>
#include <qfontdialog.h>
#include <qframe.h>
#include <qimage.h>
#include <qlabel.h>
#include <qlayout.h>
#include <qpushbutton.h>
#include <qradiobutton.h>
#include <qspinbox.h>

#include "images/options_horizontalbarchart.xpm"
#include "images/options_piechart.xpm"
#include "images/options_verticalbarchart.xpm"

OptionsForm::OptionsForm( QWidget* parent, const char* name,
                  bool modal, WFlags f )
    : QDialog( parent, name, modal, f )
{
    setCaption( "Chart -- Options" );
    resize( 320, 290 );

    optionsFormLayout = new QVBoxLayout( this, 11, 6 );

    chartTypeLayout = new QHBoxLayout( 0, 0, 6 );

    chartTypeTextLabel = new QLabel( "&Chart Type", this );
    chartTypeLayout->addWidget( chartTypeTextLabel );

    chartTypeComboBox = new QComboBox( FALSE, this );
    chartTypeComboBox->insertItem( QPixmap( options_piechart ), "Pie Chart" );
    chartTypeComboBox->insertItem( QPixmap( options_verticalbarchart ),  "Vertical Bar Chart" );
    chartTypeComboBox->insertItem( QPixmap( options_horizontalbarchart ),  "Horizontal Bar Chart" );
    chartTypeLayout->addWidget( chartTypeComboBox );
    optionsFormLayout->addLayout( chartTypeLayout );

    fontLayout = new QHBoxLayout( 0, 0, 6 );

    fontPushButton = new QPushButton( "&Font...", this );
```

```
fontLayout->addWidget( fontPushButton );
QSpacerItem* spacer = new QSpacerItem( 0, 0,  QSizePolicy::Expanding, QSizePolicy::Minimum );
fontLayout->addItem( spacer );

fontTextLabel = new QLabel( this ); // Must be set by caller via setFont()
fontLayout->addWidget( fontTextLabel );
optionsFormLayout->addLayout( fontLayout );

addValuesFrame = new QFrame( this );
addValuesFrame->setFrameShape( QFrame::StyledPanel );
addValuesFrame->setFrameShadow( QFrame::Sunken );
addValuesFrameLayout = new QVBoxLayout( addValuesFrame, 11, 6 );

addValuesButtonGroup = new QButtonGroup( "Show Values", addValuesFrame );
addValuesButtonGroup->setColumnLayout(0, Qt::Vertical );
addValuesButtonGroup->layout()->setSpacing( 6 );
addValuesButtonGroup->layout()->setMargin( 11 );
addValuesButtonGroupLayout = new QVBoxLayout( addValuesButtonGroup->layout() );
addValuesButtonGroupLayout->setAlignment( Qt::AlignTop );

noRadioButton = new QRadioButton( "&No", addValuesButtonGroup );
noRadioButton->setChecked( TRUE );
addValuesButtonGroupLayout->addWidget( noRadioButton );

yesRadioButton = new QRadioButton( "&Yes", addValuesButtonGroup );
addValuesButtonGroupLayout->addWidget( yesRadioButton );

asPercentageRadioButton = new QRadioButton( "As &Percentage", addValuesButtonGroup );
addValuesButtonGroupLayout->addWidget( asPercentageRadioButton );
addValuesFrameLayout->addWidget( addValuesButtonGroup );

decimalPlacesLayout = new QHBoxLayout( 0, 0, 6 );

decimalPlacesTextLabel = new QLabel( "&Decimal Places", addValuesFrame );
decimalPlacesLayout->addWidget( decimalPlacesTextLabel );

decimalPlacesSpinBox = new QSpinBox( addValuesFrame );
decimalPlacesSpinBox->setMinValue( 0 );
decimalPlacesSpinBox->setMaxValue( 9 );
decimalPlacesLayout->addWidget( decimalPlacesSpinBox );

addValuesFrameLayout->addLayout( decimalPlacesLayout );

optionsFormLayout->addWidget( addValuesFrame );

buttonsLayout = new QHBoxLayout( 0, 0, 6 );
spacer = new QSpacerItem( 0, 0, QSizePolicy::Expanding, QSizePolicy::Minimum );
buttonsLayout->addItem( spacer );

okPushButton = new QPushButton( "OK", this );
okPushButton->setDefault( TRUE );
buttonsLayout->addWidget( okPushButton );

cancelPushButton = new QPushButton( "Cancel", this );
```

```
    buttonsLayout->addWidget( cancelPushButton );
    optionsFormLayout->addLayout( buttonsLayout );

    connect( fontPushButton, SIGNAL( clicked() ), this, SLOT( chooseFont() ) );
    connect( okPushButton, SIGNAL( clicked() ), this, SLOT( accept() ) );
    connect( cancelPushButton, SIGNAL( clicked() ), this, SLOT( reject() ) );

    chartTypeTextLabel->setBuddy( chartTypeComboBox );
    decimalPlacesTextLabel->setBuddy( decimalPlacesSpinBox );
}

void OptionsForm::chooseFont()
{
    bool ok;
    QFont font = QFontDialog::getFont( &ok, m_font, this );
    if ( ok )
      setFont( font );
}

void OptionsForm::setFont( QFont font )
{
    QString label = font.family() + " " + QString::number( font.pointSize() ) + "pt";
    if ( font.bold() )
      label += " Bold";
    if ( font.italic() )
      label += " Italic";
    fontTextLabel->setText( label );
    m_font = font;
}
```

optionsform.h

```
#ifndef OPTIONSFORM_H
#define OPTIONSFORM_H

#include <qdialog.h>

class QButtonGroup;
class QComboBox;
class QFrame;
class QGridLayout;
class QHBoxLayout;
class QLabel;
class QPushButton;
class QRadioButton;
class QSpinBox;
class QVBoxLayout;

class OptionsForm : public QDialog
{
    Q_OBJECT
public:
    OptionsForm( QWidget* parent = 0, const char* name = "options form",
          bool modal = FALSE, WFlags f = 0 );
    ~OptionsForm() {}
    QFont font() const { return m_font; }
    void setFont( QFont font );

    QLabel *chartTypeTextLabel;
    QComboBox *chartTypeComboBox;
    QPushButton *fontPushButton;
    QLabel *fontTextLabel;
    QFrame *addValuesFrame;
    QButtonGroup *addValuesButtonGroup;
    QRadioButton *noRadioButton;
    QRadioButton *yesRadioButton;
    QRadioButton *asPercentageRadioButton;
    QLabel *decimalPlacesTextLabel;
    QSpinBox *decimalPlacesSpinBox;
    QPushButton *okPushButton;
    QPushButton *cancelPushButton;

protected slots:
    void chooseFont();

protected:
    QVBoxLayout *optionsFormLayout;
    QHBoxLayout *chartTypeLayout;
    QHBoxLayout *fontLayout;
    QVBoxLayout *addValuesFrameLayout;
    QVBoxLayout *addValuesButtonGroupLayout;
    QHBoxLayout *decimalPlacesLayout;
    QHBoxLayout *buttonsLayout;

private:
    QFont m_font;
};

#endif
```

setdataform.cpp

```
#include "setdataform.h"
#include "chartform.h"

#include <qcolordialog.h>
#include <qcombobox.h>
#include <qlayout.h>
#include <qpixmap.h>
#include <qpushbutton.h>
#include <qtable.h>

#include "images/pattern01.xpm"
#include "images/pattern02.xpm"
#include "images/pattern03.xpm"
#include "images/pattern04.xpm"
#include "images/pattern05.xpm"
#include "images/pattern06.xpm"
#include "images/pattern07.xpm"
```

```
#include "images/pattern08.xpm"
#include "images/pattern09.xpm"
#include "images/pattern10.xpm"
#include "images/pattern11.xpm"
#include "images/pattern12.xpm"
#include "images/pattern13.xpm"
#include "images/pattern14.xpm"

const int MAX_PATTERNS = 14;


SetDataForm::SetDataForm( ElementVector *elements, int decimalPlaces,
                QWidget* parent,  const char* name,  bool modal, WFlags f )
    : QDialog( parent, name, modal, f )

{
    m_elements = elements;
    m_decimalPlaces = decimalPlaces;

    setCaption( "Chart -- Set Data" );
    resize( 540, 440 );

    tableButtonBox = new QVBoxLayout( this, 11, 6, "table button box layout" );

    table = new QTable( this, "data table" );
    table->setNumCols( 5 );
    table->setNumRows( ChartForm::MAX_ELEMENTS );
    table->setColumnReadOnly( 1, TRUE );
    table->setColumnReadOnly( 2, TRUE );
    table->setColumnReadOnly( 4, TRUE );
    table->setColumnWidth( 0, 80 );
    table->setColumnWidth( 1, 60 ); // Columns 1 and 4 must be equal
    table->setColumnWidth( 2, 60 );
    table->setColumnWidth( 3, 200 );
    table->setColumnWidth( 4, 60 );
    QHeader *th = table->horizontalHeader();
    th->setLabel( 0, "Value" );
    th->setLabel( 1, "Color" );
    th->setLabel( 2, "Pattern" );
    th->setLabel( 3, "Label" );
    th->setLabel( 4, "Color" );
    tableButtonBox->addWidget( table );

    buttonBox = new QHBoxLayout( 0, 0, 6, "button box layout" );

    colorPushButton = new QPushButton( this, "color button" );
    colorPushButton->setText( "&Color..." );
    colorPushButton->setEnabled( FALSE );
    buttonBox->addWidget( colorPushButton );

    QSpacerItem *spacer = new QSpacerItem( 0, 0, QSizePolicy::Expanding, QSizePolicy::Minimum );
    buttonBox->addItem( spacer );

    okPushButton = new QPushButton( this, "ok button" );
```

```
okPushButton->setText( "OK" );
okPushButton->setDefault( TRUE );
buttonBox->addWidget( okPushButton );

cancelPushButton = new QPushButton( this, "cancel button" );
cancelPushButton->setText( "Cancel" );
cancelPushButton->setAccel( Key_Escape );
buttonBox->addWidget( cancelPushButton );

tableButtonBox->addLayout( buttonBox );

connect( table, SIGNAL( clicked(int,int,int,const QPoint&) ),  this, SLOT( setColor(int,int) ) );
connect( table, SIGNAL( currentChanged(int,int) ), this, SLOT( currentChanged(int,int) ) );
connect( table, SIGNAL( valueChanged(int,int) ), this, SLOT( valueChanged(int,int) ) );
connect( colorPushButton, SIGNAL( clicked() ), this, SLOT( setColor() ) );
connect( okPushButton, SIGNAL( clicked() ), this, SLOT( accept() ) );
connect( cancelPushButton, SIGNAL( clicked() ), this, SLOT( reject() ) );

QPixmap patterns[MAX_PATTERNS];
patterns[0]  = QPixmap( pattern01 );
patterns[1]  = QPixmap( pattern02 );
patterns[2]  = QPixmap( pattern03 );
patterns[3]  = QPixmap( pattern04 );
patterns[4]  = QPixmap( pattern05 );
patterns[5]  = QPixmap( pattern06 );
patterns[6]  = QPixmap( pattern07 );
patterns[7]  = QPixmap( pattern08 );
patterns[8]  = QPixmap( pattern09 );
patterns[9]  = QPixmap( pattern10 );
patterns[10] = QPixmap( pattern11 );
patterns[11] = QPixmap( pattern12 );
patterns[12] = QPixmap( pattern13 );
patterns[13] = QPixmap( pattern14 );

QRect rect = table->cellRect( 0, 1 );
QPixmap pix( rect.width(), rect.height() );

for ( int i = 0; i < ChartForm::MAX_ELEMENTS; ++i ) {
 Element element = (*m_elements)[i];

 if ( element.isValid() )
    table->setText( i, 0, QString( "%1" ).arg( element.value(), 0, 'f',  m_decimalPlaces ) );

 QColor color = element.valueColor();
 pix.fill( color );
 table->setPixmap( i, 1, pix );
 table->setText( i, 1, color.name() );

 QComboBox *combobox = new QComboBox;
 for ( int j = 0; j < MAX_PATTERNS; ++j )
    combobox->insertItem( patterns[j] );
 combobox->setCurrentItem( element.valuePattern() - 1 );
 table->setCellWidget( i, 2, combobox );
```

```
     table->setText( i, 3, element.label() );

     color = element.labelColor();
     pix.fill( color );
     table->setPixmap( i, 4, pix );
     table->setText( i, 4, color.name() );
    }

}

void SetDataForm::currentChanged( int, int col )
{
    colorPushButton->setEnabled( col == 1 || col == 4 );
}

void SetDataForm::valueChanged( int row, int col )
{
    if ( col == 0 ) {
     bool ok;
     double d = table->text( row, col ).toDouble( &ok );
     if ( ok && d > EPSILON )
         table->setText( row, col, QString( "%1" ).arg( d, 0, 'f', m_decimalPlaces ) );
     else if ( !table->text( row, col ).isEmpty() )
         table->setText( row, col, table->text( row, col ) + "?" );
    }
}


void SetDataForm::setColor()
{
    setColor( table->currentRow(), table->currentColumn() );
    table->setFocus();
}

void SetDataForm::setColor( int row, int col )
{
    if ( !( col == 1 || col == 4 ) )
     return;

    QColor color = QColorDialog::getColor(QColor( table->text( row, col ) ), this, "color dialog" );
    if ( color.isValid() ) {
     QPixmap pix = table->pixmap( row, col );
     pix.fill( color );
     table->setPixmap( row, col, pix );
     table->setText( row, col, color.name() );
    }
}

void SetDataForm::accept()
{
    bool ok;
    for ( int i = 0; i < ChartForm::MAX_ELEMENTS; ++i ) {
     Element &element = (*m_elements)[i];
     double d = table->text( i, 0 ).toDouble( &ok );
```

```
      if ( ok )
          element.setValue( d );
      else
          element.setValue( Element::INVALID );
      element.setValueColor( QColor( table->text( i, 1 ) ) );
      element.setValuePattern(((QComboBox*)table->cellWidget( i, 2 ))->currentItem() + 1 );
      element.setLabel( table->text( i, 3 ) );
      element.setLabelColor( QColor( table->text( i, 4 ) ) );
    }
    QDialog::accept();
}
```

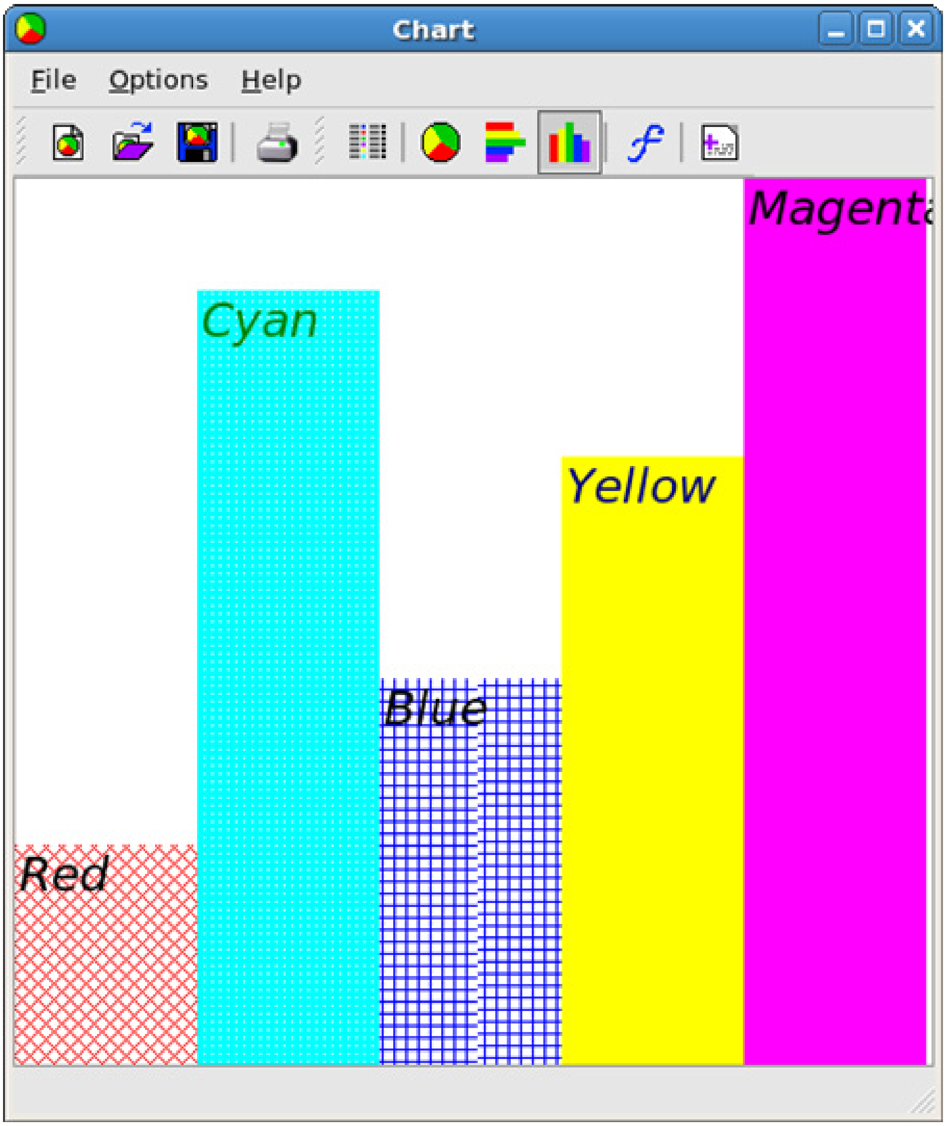
setdataform.h

```
#ifndef SETDATAFORM_H
#define SETDATAFORM_H

#include "element.h"
#include <qdialog.h>

class QHBoxLayout;
class QPushButton;
class QTable;
class QVBoxLayout;

class SetDataForm: public QDialog
{
    Q_OBJECT
public:
    SetDataForm( ElementVector *elements, int decimalPlaces,
            QWidget *parent = 0, const char *name = "set data form", bool modal = TRUE, WFlags f = 0 );
    ~SetDataForm() {}

public slots:
    void setColor();
    void setColor( int row, int col );
    void currentChanged( int row, int col );
    void valueChanged( int row, int col );

protected slots:
    void accept();

private:
    QTable *table;
    QPushButton *colorPushButton;
    QPushButton *okPushButton;
    QPushButton *cancelPushButton;

protected:
    QVBoxLayout *tableButtonBox;
    QHBoxLayout *buttonBox;

private:
    ElementVector *m_elements;
    int m_decimalPlaces;
};

#endif
```

main.cpp

```
#include <qapplication.h>
#include "chartform.h"

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

    QString filename;
    if ( app.argc() > 1 ) {
      filename = app.argv()[1];
      if ( !filename.endsWith( ".cht" ) )
          filename = QString::null;
    }

    ChartForm *cf = new ChartForm( filename );
    app.setMainWidget( cf );
    cf->show();

    return app.exec();
}
```

실행



9. 검사가능항목들을 가지는 목록보기

이 실례프로그람은 각이한 형의 검사가능항목을 가지는 목록보기의 사용법을 보여준다.

checklists.pro

```
TEMPLATE  = app
TARGET      = checklists
CONFIG      += qt warn_on release
HEADERS          = checklists.h
SOURCES          = checklists.cpp \
          main.cpp
```

checklists.cpp

```
#include "checklists.h"
#include <qlistview.h>
#include <qvbox.h>
```

```
#include <qlabel.h>
#include <qvaluelist.h>
#include <qstring.h>
#include <qpushbutton.h>
#include <qlayout.h>

/*
 * Constructor
* Create all child widgets of the CheckList Widget
 */

CheckLists::CheckLists( QWidget *parent, const char *name )  : QWidget( parent, name )
{
   QHBoxLayout *lay = new QHBoxLayout( this );
   lay->setMargin( 5 );

   // create a widget which layouts its childs in a column
   QVBoxLayout *vbox1 = new QVBoxLayout( lay );
   vbox1->setMargin( 5 );

   // First child: a Label
   vbox1->addWidget( new QLabel( "Check some items!", this ) );

   // Second child: the ListView
   lv1 = new QListView( this );
   vbox1->addWidget( lv1 );
   lv1->addColumn( "Items" );
   lv1->setRootIsDecorated( TRUE );

   // create a list with 4 ListViewItems which will be parent items of other ListViewItems
   QValueList<QListViewItem *> parentList;

   parentList.append( new QCheckListItem( lv1, "Parent Item 1",
       QCheckListItem::CheckBoxController ) );
   parentList.append( new QCheckListItem( lv1, "Parent Item 2",
       QCheckListItem::CheckBoxController ) );
   parentList.append( new QCheckListItem( lv1, "Parent Item 3",
       QCheckListItem::CheckBoxController ) );
   parentList.append( new QCheckListItem( lv1, "Parent Item 4",
       QCheckListItem::CheckBoxController ) );

   QListViewItem *item = 0;
   unsigned int num = 1;
   // go through the list of parent items...
   for ( QValueList<QListViewItem*>::Iterator it = parentList.begin(); it != parentList.end();
     ( *it )->setOpen( TRUE ), ++it, num++ ) {
    item = *it;
    // ...and create 5 checkable child ListViewItems for each parent item
    for ( unsigned int i = 1; i <= 5; i++ )
      (void)new QCheckListItem( item, QString( "%1. Child of Parent %2" ).arg( i ).arg( num ),
        QCheckListItem::CheckBox );
   }

   // Create another widget for layouting
```

```
    QVBoxLayout *tmp = new QVBoxLayout( lay );
    tmp->setMargin( 5 );

    // create a pushbutton
    QPushButton *copy1 = new QPushButton( "  ->  ", this );
    tmp->addWidget( copy1 );
    copy1->setMaximumWidth( copy1->sizeHint().width() );
    // connect the SIGNAL clicked() of the pushbutton with the SLOT copy1to2()
    connect( copy1, SIGNAL( clicked() ), this, SLOT( copy1to2() ) );

    // another widget for layouting
    QVBoxLayout *vbox2 = new QVBoxLayout( lay );
    vbox2->setMargin( 5 );

    // and another label
    vbox2->addWidget( new QLabel( "Check one item!", this ) );

    // create the second listview
    lv2 = new QListView( this );
    vbox2->addWidget( lv2 );
    lv2->addColumn( "Items" );
    lv2->setRootIsDecorated( TRUE );

    // another widget needed for layouting only
    tmp = new QVBoxLayout( lay );
    tmp->setMargin( 5 );

    // create another pushbutton...
    QPushButton *copy2 = new QPushButton( "  ->  ", this );
    lay->addWidget( copy2 );
    copy2->setMaximumWidth( copy2->sizeHint().width() );
    // ...and connect its clicked() SIGNAL to the copy2to3() SLOT
    connect( copy2, SIGNAL( clicked() ), this, SLOT( copy2to3() ) );

    tmp = new QVBoxLayout( lay );
    tmp->setMargin( 5 );

    // and create a label which will be at the right of the window
    label = new QLabel( "No Item yet...", this );
    tmp->addWidget( label );
}

/*
 * SLOT copy1to2()
* Copies all checked ListViewItems from the first ListView to
 * the second one, and inserts them as Radio-ListViewItem.
 */

void CheckLists::copy1to2()
{
    // create an iterator which operates on the first ListView
    QListViewItemIterator it( lv1 );

    lv2->clear();
    // Insert first a controller Item into the second ListView. Always if Radio-ListViewItems
    // are inserted into a Listview, the parent item of these MUST be a controller Item!
    QCheckListItem *item = new QCheckListItem( lv2, "Controller", QCheckListItem::Controller );
    item->setOpen( TRUE );

    // iterate through the first ListView...
    for ( ; it.current(); ++it )
     // ...check state of childs, and...
     if ( it.current()->parent() )
         // ...if the item is checked...
         if ( ( (QCheckListItem*)it.current() )->isOn() )
           // ...insert a Radio-ListViewItem with the same text into the second ListView
           (void)new QCheckListItem( item, it.current()->text( 0 ), QCheckListItem::RadioButton );

    if ( item->firstChild() )
      ( ( QCheckListItem* )item->firstChild() )->setOn( TRUE );
}

/*
 * SLOT copy2to3()
* Copies the checked item of the second ListView into the
 * Label at the right.
 */

void CheckLists::copy2to3()
{
    // create an iterator which operates on the second ListView
    QListViewItemIterator it( lv2 );

    label->setText( "No Item checked" );

    // iterate through the second ListView...
    for ( ; it.current(); ++it )
     // ...check state of childs, and...
     if ( it.current()->parent() )
         // ...if the item is checked...
         if ( ( (QCheckListItem*)it.current() )->isOn() )
           // ...set the text of the item to the label
           label->setText( it.current()->text( 0 ) );
}
```

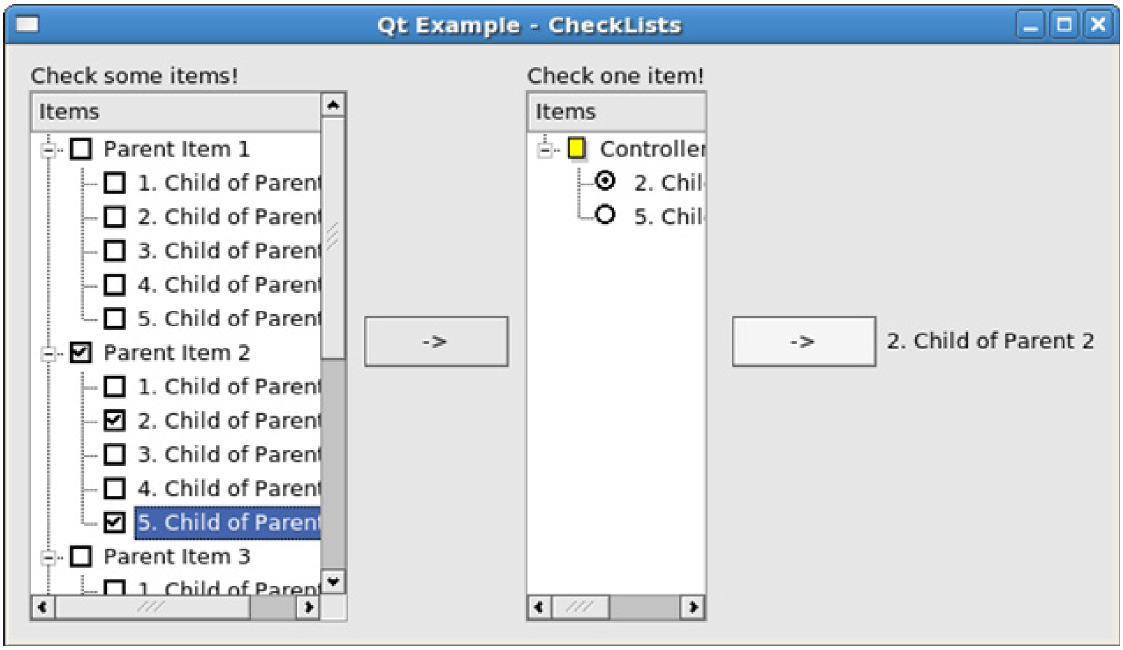
checklists.h

```
#ifndef CHECKLISTS_H
#define CHECKLISTS_H
#include <qwidget.h>

class QListView;
class QLabel;

class CheckLists : public QWidget
{
    Q_OBJECT
public:
    CheckLists( QWidget *parent = 0, const char *name = 0 );

protected:
    QListView *lv1, *lv2;
    QLabel *label;

protected slots:
    void copy1to2();
    void copy2to3();

};

#endif
```

main.cpp

```
#include "checklists.h"
#include <qapplication.h>

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    CheckLists checklists;
    checklists.resize( 650, 350 );
    checklists.setCaption( "Qt Example - CheckLists" );
    a.setMainWidget( &checklists );
    checklists.show();

    return a.exec();
}
```

실행



10.유표

다음의 실례는 창문부품용의 마우스유표를 설정하는 방법을 보여준다.

cursor.pro

```
TEMPLATE  = app
TARGET    = cursor
CONFIG    += qt warn_on release
HEADERS     =
SOURCES     = cursor.cpp
```

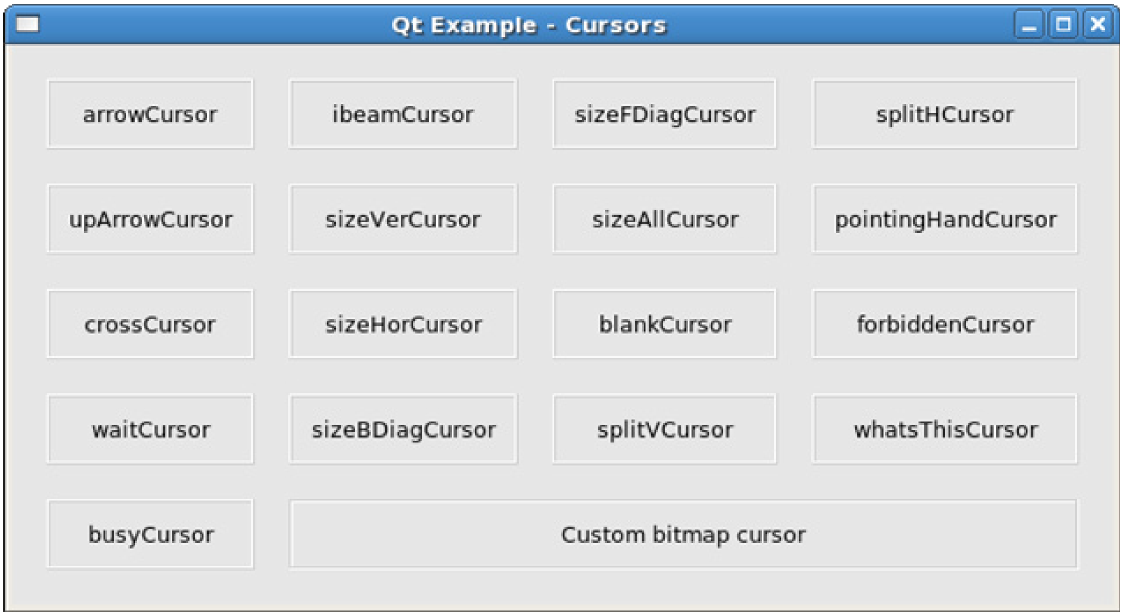
cursor.cpp

```
#include <qlabel.h>
#include <qbitmap.h>
#include <qapplication.h>
#include <qlayout.h>
#include <qcursor.h>

// cb_bits and cm_bits were generated by X bitmap program.

#define cb_width  32
#define cb_height 32

static unsigned char cb_bits[] = {        // cursor bitmap
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf8, 0x0f, 0x00,
  0x00, 0x06, 0x30, 0x00, 0x80, 0x01, 0xc0, 0x00, 0x40, 0x00, 0x00, 0x01,
  0x20, 0x00, 0x00, 0x02, 0x10, 0x00, 0x00, 0x04, 0x08, 0x3e, 0x3e, 0x08,
  0x08, 0x03, 0xe0, 0x08, 0xc4, 0x00, 0x00, 0x11, 0x04, 0x1e, 0x78, 0x10,
  0x02, 0x0c, 0x30, 0x20, 0x02, 0x40, 0x00, 0x20, 0x02, 0x40, 0x00, 0x20,
  0x02, 0x40, 0x00, 0x20, 0x02, 0x20, 0x04, 0x20, 0x02, 0x20, 0x04, 0x20,
  0x02, 0x10, 0x08, 0x20, 0x02, 0x08, 0x08, 0x20, 0x02, 0xf0, 0x07, 0x20,
  0x04, 0x00, 0x00, 0x10, 0x04, 0x00, 0x00, 0x10, 0x08, 0x00, 0xc0, 0x08,
  0x08, 0x3c, 0x30, 0x08, 0x10, 0xe6, 0x19, 0x04, 0x20, 0x00, 0x0f, 0x02,
  0x40, 0x00, 0x00, 0x01, 0x80, 0x01, 0xc0, 0x00, 0x00, 0x06, 0x30, 0x00,
  0x00, 0xf8, 0x0f, 0x00, 0x00, 0x00, 0x00, 0x00};

#define cm_width  32
#define cm_height 32

static unsigned char cm_bits[] = {        // cursor bitmap mask
  0x00, 0x00, 0x00, 0x00, 0x00, 0xf8, 0x1f, 0x00, 0x00, 0xfe, 0x3f, 0x00,
  0x80, 0x07, 0xf0, 0x00, 0xc0, 0x01, 0xc0, 0x01, 0x60, 0x00, 0x00, 0x03,
  0x30, 0x00, 0x00, 0x06, 0x18, 0x00, 0x00, 0x0c, 0x0c, 0x3e, 0x3e, 0x18,
  0x0e, 0x03, 0xe0, 0x18, 0xc6, 0x00, 0x00, 0x31, 0x07, 0x1e, 0x78, 0x30,
  0x03, 0x0c, 0x30, 0x60, 0x03, 0x40, 0x00, 0x60, 0x03, 0x40, 0x00, 0x60,
  0x03, 0x40, 0x00, 0x60, 0x03, 0x20, 0x04, 0x60, 0x03, 0x20, 0x04, 0x60,
  0x03, 0x10, 0x08, 0x60, 0x03, 0x08, 0x08, 0x60, 0x03, 0xf0, 0x07, 0x60,
  0x06, 0x00, 0x00, 0x30, 0x06, 0x00, 0x00, 0x30, 0x0c, 0x00, 0xc0, 0x18,
  0x0c, 0x3c, 0x30, 0x18, 0x18, 0xe6, 0x19, 0x0c, 0x30, 0x00, 0x0f, 0x06,
  0x60, 0x00, 0x00, 0x03, 0xc0, 0x01, 0xc0, 0x01, 0x80, 0x07, 0xf0, 0x00,
  0x00, 0xfe, 0x3f, 0x00, 0x00, 0xf8, 0x0f, 0x00};
```

```
// The CursorView contains many labels with different cursors.

class CursorView : public QWidget          // cursor view
{
public:
   CursorView();
};

// Constructs a cursor view.

CursorView::CursorView()            // construct view
{
   struct List {
    CursorShape  shape;
    const char*    name;            // cursor name
   };
   static List list[] = {
    { ArrowCursor,        "arrowCursor" },
    { UpArrowCursor,    "upArrowCursor" },
    { CrossCursor,        "crossCursor" },
    { WaitCursor,     "waitCursor" },
    { IbeamCursor,        "ibeamCursor" },
    { SizeVerCursor, "sizeVerCursor" },
    { SizeHorCursor, "sizeHorCursor" },
    { SizeBDiagCursor,  "sizeBDiagCursor" },
    { SizeFDiagCursor,   "sizeFDiagCursor" },
    { SizeAllCursor,  "sizeAllCursor" },
    { BlankCursor,        "blankCursor" },
    { SplitVCursor,       "splitVCursor" },
    { SplitHCursor,       "splitHCursor" },
    { PointingHandCursor,   "pointingHandCursor" },
    { ForbiddenCursor,   "forbiddenCursor" },
    { WhatsThisCursor,  "whatsThisCursor" },
    { BusyCursor,         "busyCursor" }
   };

   setCaption( "CursorView" );           // set window caption

   QGridLayout* grid = new QGridLayout( this, 5, 4, 20 );
   QLabel *label;

   int i=0;
   for ( int y=0; y<4; y++ ) {              // create the small labels
    for ( int x=0; x<4; x++ ) {
       label = new QLabel( this );
       label->setCursor( QCursor( list[i].shape ) );
       label->setText( list[i].name );
       label->setAlignment( AlignCenter );
       label->setMargin( 10 );
       label->setFrameStyle( QFrame::Box | QFrame::Raised );
       grid->addWidget( label, x, y );
       i++;
    }
   }
```

```
    label = new QLabel( this );
    label->setCursor( QCursor( list[i].shape ) );
    label->setText( list[i].name );
    label->setAlignment( AlignCenter );
    label->setMargin( 10 );
    label->setFrameStyle( QFrame::Box | QFrame::Raised );
    grid->addWidget( label, 4, 0 );

    QBitmap cb( cb_width, cb_height, cb_bits, TRUE );
    QBitmap cm( cm_width, cm_height, cm_bits, TRUE );
    QCursor custom( cb, cm );                // create bitmap cursor

    label = new QLabel( this );              // create the big label
    label->setCursor( custom );
    label->setText( "Custom bitmap cursor" );
    label->setAlignment( AlignCenter );
    label->setMargin( 10 );
    label->setFrameStyle( QFrame::Box | QFrame::Sunken );
    grid->addMultiCellWidget( label, 4, 4, 1, 3 );

}

// Create and display a CursorView.
int main( int argc, char **argv )
{
    QApplication a( argc, argv );       // application object
    CursorView   v;                     // cursor view
    a.setMainWidget( &v );
    v.setCaption("Qt Example - Cursors");
    v.show();
    return a.exec();
}
```

실행



11. 사용자정의배치관리기

이 실례는 카드배치, 테두리배치, 흐름배치와 같은 사용자정의배치(기하학적)관리기를 쓰는 방법을 보여준다.

customlayout.pro

```
TEMPLATE  = app
TARGET    = customlayout
CONFIG    += qt warn_on release
HEADERS       = border.h \
          card.h \
          flow.h
SOURCES       = border.cpp \
          card.cpp \
          flow.cpp \
          main.cpp
```

border.cpp

```
#include "border.h"

class BorderLayoutIterator : public QGLayoutIterator
{
public:
   BorderLayoutIterator( const QPtrList<BorderLayout::BorderLayoutStruct> *l )
    : idx( 0 ) , list( (QPtrList<BorderLayout::BorderLayoutStruct>*)l )
   {}

   uint count() const;
   QLayoutItem *current();
   BorderLayout::BorderLayoutStruct *currentStruct();
   void toFirst();
   QLayoutItem *next();
```

```
    QLayoutItem *takeCurrent();
    BorderLayoutIterator &operator++();

private:
    int idx;
    QPtrList<BorderLayout::BorderLayoutStruct> *list;

};

uint BorderLayoutIterator::count() const
{
    return list->count();
}

QLayoutItem *BorderLayoutIterator::current()
{
    return idx < (int)count() ? list->at( idx )->item : 0;
}

BorderLayout::BorderLayoutStruct *BorderLayoutIterator::currentStruct()
{
    return idx < (int)count() ? list->at( idx ) : 0;
}

void BorderLayoutIterator::toFirst()
{
    idx = 0;
}

QLayoutItem *BorderLayoutIterator::next()
{
    idx++;
    return current();
}

QLayoutItem *BorderLayoutIterator::takeCurrent()
{
    BorderLayout::BorderLayoutStruct *b
     = idx < int( list->count() ) ? list->take(  idx  ) : 0;
    QLayoutItem *item =  b ? b->item : 0;
    delete b;
    return item;
}

BorderLayoutIterator &BorderLayoutIterator::operator++()
{
    next();
    return *this;
}

BorderLayout::~BorderLayout()
{
    deleteAllItems();
}
```

```
void BorderLayout::addItem( QLayoutItem *item )
{
   add( item, West );
}

void BorderLayout::addWidget( QWidget *widget, Position pos )
{
   add( new BorderWidgetItem( widget ), pos );
}

void BorderLayout::add( QLayoutItem *item, Position pos )
{
   list.append( new BorderLayoutStruct( item, pos ) );
   sizeDirty = TRUE; msizeDirty = TRUE;
   calcSize( SizeHint ); calcSize( Minimum );
}

bool BorderLayout::hasHeightForWidth() const
{
   return FALSE;
}

QSize BorderLayout::sizeHint() const
{
   return cached;
}

QSize BorderLayout::minimumSize() const
{
   return cached;
}

QSizePolicy::ExpandData BorderLayout::expanding() const

{
   return QSizePolicy::BothDirections;
}

QLayoutIterator BorderLayout::iterator()
{
   return QLayoutIterator( new BorderLayoutIterator( &list ) );
}

void BorderLayout::setGeometry( const QRect &rct )
{
   QLayout::setGeometry( rct );
   doLayout( rct );
}

void BorderLayout::doLayout( const QRect &rct, bool /*testonly*/ )
{
   int ew = 0, ww = 0, nh = 0, sh = 0;
   int h = 0;
```

```
    BorderLayoutIterator it( &list );
    BorderLayoutStruct *o;
    BorderLayoutStruct *center = 0;
    while ( ( o = it.currentStruct() ) != 0 ) {
     ++it;

     if ( o->pos == North ) {
        o->item->setGeometry( QRect( rct.x(), nh, rct.width(), o->item->sizeHint().height() ) );
        nh += o->item->geometry().height() + spacing();
     }
     if ( o->pos == South ) {
        o->item->setGeometry( QRect( o->item->geometry().x(), o->item->geometry().y(),
                   rct.width(), o->item->sizeHint().height() ) );
        sh += o->item->geometry().height() + spacing();
        o->item->setGeometry( QRect( rct.x(), rct.y() + rct.height() - sh + spacing(),
                   o->item->geometry().width(), o->item->geometry().height() ) );
     }
     if ( o->pos == Center )
        center = o;
    }

    h = rct.height() - nh - sh;

    it.toFirst();
    while ( ( o = it.currentStruct() ) != 0 ) {
     ++it;

     if ( o->pos == West ) {
        o->item->setGeometry( QRect( rct.x() + ww, nh, o->item->sizeHint().width(), h ) );
        ww += o->item->geometry().width() + spacing();
     }
     if ( o->pos == East ) {
        o->item->setGeometry( QRect( o->item->geometry().x(), o->item->geometry().y(),
                   o->item->sizeHint().width(), h ) );
        ew += o->item->geometry().width() + spacing();
        o->item->setGeometry( QRect( rct.x() + rct.width() - ew + spacing(), nh,
                   o->item->geometry().width(), o->item->geometry().height() ) );
     }
    }

    if ( center )
     center->item->setGeometry( QRect( ww, nh, rct.width() - ew - ww, h ) );
}

void BorderLayout::calcSize( SizeType st )
{
    if ( ( st == Minimum && !msizeDirty ) ||
      ( st == SizeHint && !sizeDirty ) )
     return;

    int w = 0, h = 0;

    BorderLayoutIterator it( &list );
```

```
  BorderLayoutStruct *o;
  while ( ( o = it.currentStruct() ) != 0 ) {
   ++it;
   if ( o->pos == North ||
      o->pos == South ) {
     if ( st == Minimum )
      h += o->item->minimumSize().height();
     else
      h += o->item->sizeHint().height();
   }
   else if ( o->pos == West ||
       o->pos == East ) {
     if ( st == Minimum )
      w += o->item->minimumSize().width();
     else
      w += o->item->sizeHint().width();
   } else {
     if ( st == Minimum ) {
      h += o->item->minimumSize().height();
      w += o->item->minimumSize().width();
     }
     else {
      h += o->item->sizeHint().height();
      w += o->item->sizeHint().width();
     }
   }
  }

  if ( st == Minimum ) {
   msizeDirty = FALSE;
   mcached = QSize( w, h );
  } else {
   sizeDirty = FALSE;
   cached = QSize( w, h );
  }

  return;
}
```

border.h

```
#ifndef BORDER_H
#define BORDER_H

#include <qlayout.h>
#include <qptrlist.h>

class BorderWidgetItem : public QWidgetItem
{
public:
  BorderWidgetItem( QWidget *w )
   : QWidgetItem( w )
  {}

  void setGeometry( const QRect &r )
  { widget()->setGeometry( r ); }

};

class BorderLayout : public QLayout
{
public:
  enum Position {   West = 0,  North, South, East, Center   };

  struct BorderLayoutStruct
  {
   BorderLayoutStruct( QLayoutItem *i, Position p ) {
     item = i;
     pos = p;
   }

   QLayoutItem *item;
   Position pos;
  };

  enum SizeType {  Minimum = 0,    SizeHint   };

  BorderLayout( QWidget *parent, int border = 0, int autoBorder = -1,
       const char *name = 0 )
   : QLayout( parent, border, autoBorder, name ), cached( 0, 0 ), mcached( 0, 0 ),
     sizeDirty( TRUE ), msizeDirty( TRUE )
  {}

  BorderLayout( QLayout* parent, int autoBorder = -1, const char *name = 0 )
   : QLayout( parent, autoBorder, name  ), cached( 0, 0 ), mcached( 0, 0 ),
     sizeDirty( TRUE ), msizeDirty( TRUE )
  {}

  BorderLayout( int autoBorder = -1, const char *name = 0 )
   : QLayout( autoBorder, name ), cached( 0, 0 ), mcached( 0, 0 ),
     sizeDirty( TRUE ), msizeDirty( TRUE )
  {}

  ~BorderLayout();

  void addItem( QLayoutItem *item );

  void addWidget( QWidget *widget, Position pos );
  void add( QLayoutItem *item, Position pos );

  bool hasHeightForWidth() const;

  QSize sizeHint() const;
  QSize minimumSize() const;

  QLayoutIterator iterator();

  QSizePolicy::ExpandData expanding() const;
```

```
protected:
    void setGeometry( const QRect &rect );

private:
    void doLayout( const QRect &rect, bool testonly = FALSE );
    void calcSize( SizeType st );

    QPtrList<BorderLayoutStruct> list;
    QSize cached, mcached;
    bool sizeDirty, msizeDirty;

};

#endif
```

card.cpp

```
#include "card.h"

class CardLayoutIterator :public QGLayoutIterator
{
public:
    CardLayoutIterator( QPtrList<QLayoutItem> *l )  : idx( 0 ), list( l )  {}

    QLayoutItem *current();
    QLayoutItem *next();
    QLayoutItem *takeCurrent();

private:
    int idx;
    QPtrList<QLayoutItem> *list;
};

QLayoutItem *CardLayoutIterator::current()
{
    return idx < int( list->count() ) ? list->at( idx ) : 0;
}

QLayoutItem *CardLayoutIterator::next()
{
    idx++; return current();
}

QLayoutItem *CardLayoutIterator::takeCurrent()
{
    return idx < int( list->count() ) ?list->take( idx ) : 0;
}

QLayoutIterator CardLayout::iterator()
{
    return QLayoutIterator(  new CardLayoutIterator( &list )  );
}

CardLayout::~CardLayout()
{
    deleteAllItems();
}

void CardLayout::addItem(  QLayoutItem *item  )
{
    list.append( item );
}

void CardLayout::setGeometry( const QRect &rct )
{
    QLayout::setGeometry( rct );

    QPtrListIterator<QLayoutItem> it( list );
    if ( it.count() == 0 )
      return;

    QLayoutItem *o;

    int i = 0;

    int w = rct.width() - ( list.count() - 1 ) * spacing();
    int h = rct.height() - ( list.count() - 1 ) * spacing();

    while ( ( o=it.current() ) != 0 ) {
      ++it;
      QRect geom( rct.x() + i * spacing(), rct.y() + i * spacing(),   w, h  );
      o->setGeometry(  geom  );
      ++i;
    }
}

QSize CardLayout::sizeHint() const
{
    QSize s(0,0);
    int n = list.count();
    if ( n > 0 )
      s = QSize(100,70); //start with a nice default size
    QPtrListIterator<QLayoutItem> it(list);
    QLayoutItem *o;
    while ( (o=it.current()) != 0 ) {
      ++it;
      s = s.expandedTo( o->minimumSize() );
    }
    return s + n*QSize(spacing(),spacing());
}

QSize CardLayout::minimumSize() const
{
    QSize s(0,0);
    int n = list.count();
    QPtrListIterator<QLayoutItem> it(list);
    QLayoutItem *o;
    while ( (o=it.current()) != 0 ) {
      ++it;
      s = s.expandedTo( o->minimumSize() );
    }
    return s + n*QSize(spacing(),spacing());
}
```

card.h

```
#ifndef CARD_H
#define CARD_H

#include <qlayout.h>
#include <qptrlist.h>

class CardLayout : public QLayout
{
public:
    CardLayout( QWidget *parent, int dist ) : QLayout( parent, 0, dist ) {}
    CardLayout( QLayout* parent, int dist) : QLayout( parent, dist ) {}
    CardLayout( int dist )
      : QLayout( dist ) {}
    ~CardLayout();

    void addItem( QLayoutItem *item );
    QSize sizeHint() const;
    QSize minimumSize() const;
    QLayoutIterator iterator();
    void setGeometry( const QRect &rect );

private:
    QPtrList<QLayoutItem> list;

};

#endif
```

flow.cpp

```
#include "flow.h"

class SimpleFlowIterator :public QGLayoutIterator
{
public:
    SimpleFlowIterator( QPtrList<QLayoutItem> *l ) :idx(0), list(l)  {}
    uint count() const;
    QLayoutItem *current();
    QLayoutItem *next();
    QLayoutItem *takeCurrent();

private:
    int idx;
    QPtrList<QLayoutItem> *list;

};

uint SimpleFlowIterator::count() const
{
   return list->count();
}

QLayoutItem *SimpleFlowIterator::current()
{
   return idx < int(count()) ? list->at(idx) : 0;
}

QLayoutItem *SimpleFlowIterator::next()
{
   idx++; return current();
}

QLayoutItem *SimpleFlowIterator::takeCurrent()
{
   return idx < int(count()) ? list->take( idx ) : 0;
}

SimpleFlow::~SimpleFlow()
{
   deleteAllItems();
}


int SimpleFlow::heightForWidth( int w ) const
{
   if ( cached_width != w ) {
    //Not all C++ compilers support "mutable" yet:
    SimpleFlow * mthis = (SimpleFlow*)this;
    int h = mthis->doLayout( QRect(0,0,w,0), TRUE );
    mthis->cached_hfw = h;
    mthis->cached_width = w;
    return h;
   }
   return cached_hfw;
}

void SimpleFlow::addItem( QLayoutItem *item)
{
   list.append( item );
}

bool SimpleFlow::hasHeightForWidth() const
{
   return TRUE;
}

QSize SimpleFlow::sizeHint() const
{
   return minimumSize();
}

QSizePolicy::ExpandData SimpleFlow::expanding() const
{
   return QSizePolicy::NoDirection;
}

QLayoutIterator SimpleFlow::iterator()
{
   return QLayoutIterator( new SimpleFlowIterator( &list ) );
}

void SimpleFlow::setGeometry( const QRect &r )
{
   QLayout::setGeometry( r );
   doLayout( r );
}

int SimpleFlow::doLayout( const QRect &r, bool testonly )
{
   int x = r.x();
   int y = r.y();
   int h = 0;          //height of this line so far.
   QPtrListIterator<QLayoutItem> it(list);
   QLayoutItem *o;
   while ( (o=it.current()) != 0 ) {
    ++it;
    int nextX = x + o->sizeHint().width() + spacing();
    if ( nextX - spacing() > r.right() && h > 0 ) {
       x = r.x();
       y = y + h + spacing();
       nextX = x + o->sizeHint().width() + spacing();
       h = 0;
    }
    if ( !testonly )
       o->setGeometry( QRect( QPoint( x, y ), o->sizeHint() ) );
    x = nextX;
    h = QMAX( h,  o->sizeHint().height() );
   }
   return y + h - r.y();
}

QSize SimpleFlow::minimumSize() const
{
   QSize s(0,0);
   QPtrListIterator<QLayoutItem> it(list);
   QLayoutItem *o;
   while ( (o=it.current()) != 0 ) {
    ++it;
    s = s.expandedTo( o->minimumSize() );
   }
   return s;
}
```

flow.h

```
#ifndef FLOW_H
#define FLOW_H
```

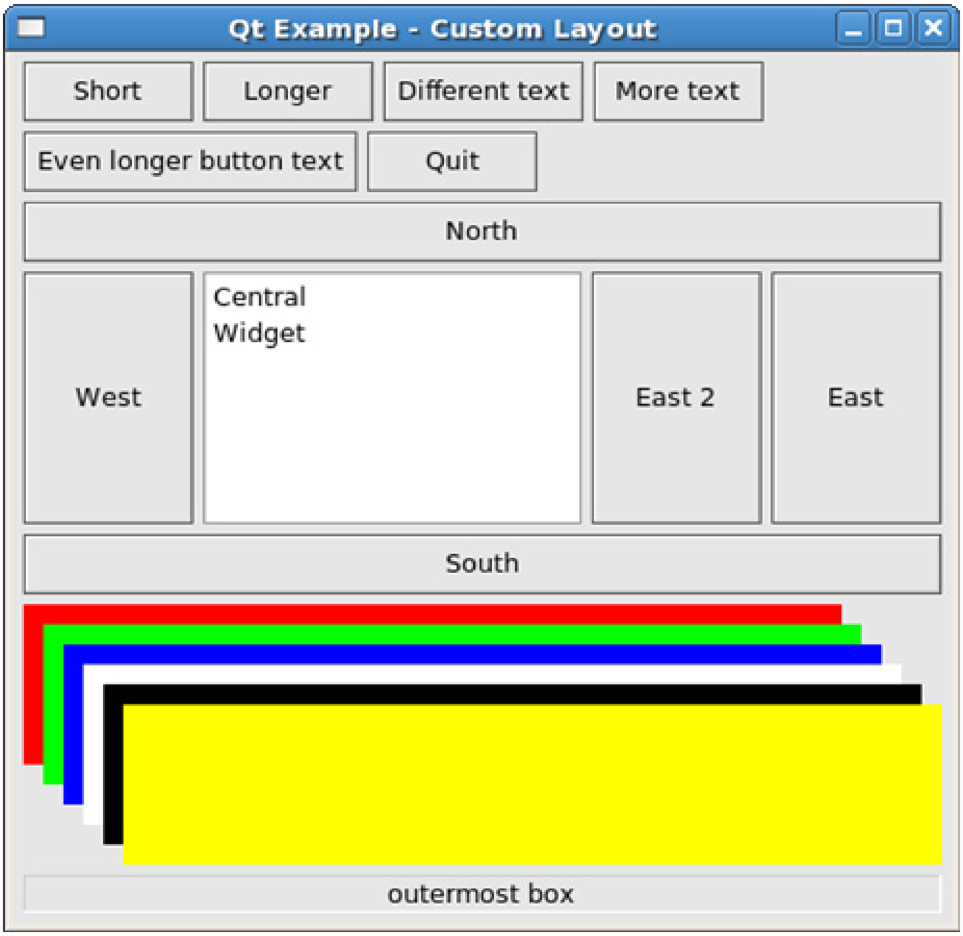
```
#include <qlayout.h>
#include <qptrlist.h>

class SimpleFlow : public QLayout
{
public:
    SimpleFlow( QWidget *parent, int border=0, int space=-1,
            const char *name=0 )
      : QLayout( parent, border, space, name ),
      cached_width(0) {}
    SimpleFlow( QLayout* parent, int space=-1, const char *name=0 )
      : QLayout( parent, space, name ),
      cached_width(0) {}
    SimpleFlow( int space=-1, const char *name=0 )
      : QLayout( space, name ),
      cached_width(0) {}

    ~SimpleFlow();

    void addItem( QLayoutItem *item);
    bool hasHeightForWidth() const;
    int heightForWidth( int ) const;
    QSize sizeHint() const;
    QSize minimumSize() const;
    QLayoutIterator iterator();
    QSizePolicy::ExpandData expanding() const;

protected:
    void setGeometry( const QRect& );

private:
    int doLayout( const QRect&, bool testonly = FALSE );
    QPtrList<QLayoutItem> list;
    int cached_width;
    int cached_hfw;

};

#endif
```

main.cpp

```
#include "flow.h"
#include "border.h"
#include "card.h"

#include <qapplication.h>
#include <qlabel.h>
#include <qcolor.h>
#include <qgroupbox.h>
#include <qpushbutton.h>
#include <qmultilineedit.h>
#include <qcolor.h>

int main( int argc, char **argv )
```

```
{
    QApplication a( argc, argv );

    QWidget *f = new QWidget;
    QBoxLayout *gm = new QVBoxLayout( f, 5 );

    SimpleFlow *b1 = new SimpleFlow( gm );

    b1->add( new QPushButton( "Short", f ) );
    b1->add( new QPushButton( "Longer", f ) );
    b1->add( new QPushButton( "Different text", f ) );
    b1->add( new QPushButton( "More text", f ) );
    b1->add( new QPushButton( "Even longer button text", f ) );
    QPushButton* qb = new QPushButton( "Quit", f );
    a.connect( qb, SIGNAL( clicked() ), SLOT( quit() ) );
    b1->add( qb );

    QWidget *wid = new QWidget( f );

    BorderLayout *large = new BorderLayout( wid );
    large->setSpacing( 5 );
    large->addWidget( new QPushButton( "North", wid ), BorderLayout::North );
    large->addWidget( new QPushButton( "West", wid ), BorderLayout::West );
    QMultiLineEdit* m = new QMultiLineEdit( wid );
    m->setText( "Central\nWidget" );
    large->addWidget( m, BorderLayout::Center );
    QWidget *east1 = new QPushButton( "East", wid );
    large->addWidget( east1, BorderLayout::East );
    QWidget *east2 = new QPushButton( "East 2", wid );
    large->addWidget( east2 , BorderLayout::East );
    large->addWidget( new QPushButton( "South", wid ), BorderLayout::South );
    //Left-to-right tab order looks better:
    QWidget::setTabOrder( east2, east1 );
    gm->addWidget( wid );

    wid = new QWidget( f );
    CardLayout *card = new CardLayout( wid, 10 );

    QWidget *crd = new QWidget( wid );
    crd->setBackgroundColor( Qt::red );
    card->add( crd );
    crd = new QWidget( wid );
    crd->setBackgroundColor( Qt::green );
    card->add( crd );
    crd = new QWidget( wid );
    crd->setBackgroundColor( Qt::blue );
    card->add( crd );
    crd = new QWidget( wid );
    crd->setBackgroundColor( Qt::white );
    card->add( crd );
    crd = new QWidget( wid );
    crd->setBackgroundColor( Qt::black );
    card->add( crd );
    crd = new QWidget( wid );
    crd->setBackgroundColor( Qt::yellow );
    card->add( crd );

    gm->addWidget( wid );

    QLabel* s = new QLabel( f );
    s->setText( "outermost box" );
    s->setFrameStyle( QFrame::Panel | QFrame::Sunken );
    s->setAlignment( Qt::AlignVCenter | Qt::AlignHCenter );
    gm->addWidget( s );
    a.setMainWidget( f );
    f->setCaption("Qt Example - Custom Layout");
    f->show();

    int result = a.exec();
    delete f;
    return result;
}
```

실행



12. 수자시계

이 실례는 시간과 날자사이를 절환할수 있는 수자형 LCD 시계를 보여준다.

dclock.pro

```
TEMPLATE  = app
TARGET      = dclock
CONFIG      += qt warn_on release
HEADERS        = dclock.h
SOURCES        = dclock.cpp \
          main.cpp
```

dclock.cpp

```
#include "dclock.h"
#include <qdatetime.h>

// Constructs a DigitalClock widget with a parent and a name.
DigitalClock::DigitalClock( QWidget *parent, const char *name )
   : QLCDNumber( parent, name )
{
   showingColon = FALSE;
   setFrameStyle( QFrame::Panel | QFrame::Raised );
   setLineWidth( 2 );               // set frame line width
   showTime();                      // display the current time
   normalTimer = startTimer( 500 );          // 1/2 second timer events
   showDateTimer = -1;                 // not showing date
}

// Handles timer events for the digital clock widget.
// There are two different timers; one timer for updating the clock
// and another one for switching back from date mode to time mode.
void DigitalClock::timerEvent( QTimerEvent *e )
{
   if ( e->timerId() == showDateTimer )    // stop showing date
    stopDate();
   else {                          // normal timer
    if ( showDateTimer == -1 )       // not showing date
       showTime();
   }
}

// Enters date mode when the left mouse button is pressed.
void DigitalClock::mousePressEvent( QMouseEvent *e )
{
   if ( e->button() == QMouseEvent::LeftButton )           // left button pressed
    showDate();
}

// Shows the current date in the internal lcd widget.
// Fires a timer to stop showing the date.
void DigitalClock::showDate()
{
   if ( showDateTimer != -1 )              // already showing date
    return;
   QDate date = QDate::currentDate();
```

```
    QString s;
    s.sprintf( "%2d %2d", date.month(), date.day() );
    display( s );                       // sets the LCD number/text
    showDateTimer = startTimer( 2000 );         // keep this state for 2 secs
}

// Stops showing the date.
void DigitalClock::stopDate()
{
    killTimer( showDateTimer );
    showDateTimer = -1;
    showTime();
}

// Shows the current time in the internal lcd widget.
void DigitalClock::showTime()
{
    showingColon = !showingColon;           // toggle/blink colon
    QString s = QTime::currentTime().toString().left(5);
    if ( !showingColon )
      s[2] = ' ';
    if ( s[0] == '0' )
      s[0] = ' ';
    display( s );                       // set LCD number/text
}
```

dclock.h

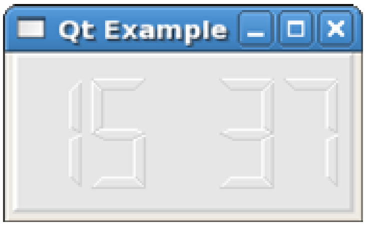
```
#ifndef DCLOCK_H
#define DCLOCK_H

#include <qlcdnumber.h>

class DigitalClock : public QLCDNumber      // digital clock widget
{
    Q_OBJECT
public:
    DigitalClock( QWidget *parent=0, const char *name=0 );

protected:                      // event handlers
    voidtimerEvent( QTimerEvent * );
    voidmousePressEvent( QMouseEvent * );

private slots:                      // internal slots
    voidstopDate();
    voidshowTime();

private:                        // internal data
    voidshowDate();

    boolshowingColon;
    int         normalTimer;
    int         showDateTimer;
};
```

```
#endif // DCLOCK_H
```

main.cpp

```
#include "dclock.h"
#include <qapplication.h>

int main( int argc, char **argv )
{
   QApplication a( argc, argv );
   DigitalClock *clock = new DigitalClock;
   clock->resize( 170, 80 );
   a.setMainWidget( clock );
   clock->setCaption("Qt Example - Digital Clock");
   clock->show();
   return a.exec();
}
```

실행



13. 탁상에 그리기

탁상프로그람은 탁상우에 그리기하는 3개의 루틴을 포함한다. 이것은 QPainter로 좋은 그림을 그리며 또한 탁상을 다른 창문부품으로서 취급하는 방법을 보여준다.

desktop.pro

```
TEMPLATE  = app
TARGET    = desktop
CONFIG    += qt warn_on release
HEADERS      =
SOURCES      = desktop.cpp
```

desktop.cpp

```
#include <qimage.h>
#include <qbitmap.h>
#include <qpainter.h>
#include <qapplication.h>
#include <qdropsite.h>
#include <qdragobject.h>
#include <stdio.h>

static double seed = 0.353535353535;
static const int KINDA_RAND_MAX = 32767;

static int kindaRand()
{
   seed = seed*147;
```

```
    seed = seed - (double) ((int) seed);
    return (int) ( seed*(KINDA_RAND_MAX + 1) );
}

static int velocity( int i )            // change velocity
{
    const int velmax = 15;
    const int velmin = 4;
    if ( i == 1 || i == 2 )
      i = (kindaRand()&0x7fff % velmax)/3 + velmin;
    else
      i = (kindaRand()&0x7fff % velmax) + velmin;
    return i;
}

// Draw polygon on desktop.
void poly()
{
    QWidget *d = QApplication::desktop();
    d->setBackgroundColor( Qt::white );         // white desktop

    const int maxpoints = 5;
    const int maxcurves = 8;
    static int xvel[maxpoints];
    static int yvel[maxpoints];
    int head = 0;
    int tail = -maxcurves + 2;
    QPointArray *a = new QPointArray[ maxcurves ];
    register QPointArray *p;
    QRect r = d->rect();              // desktop rectangle

    int i;
    for ( i=0; i<maxcurves; i++ )
      a[i].resize( maxpoints );
    p = &a[0];
    for ( i=0; i<maxpoints; i++ ) {         // setup first polygon points
      p->setPoint( i, (kindaRand()&0x7fff) % r.width(),
                (kindaRand()&0x7fff) % r.height() );
      xvel[i] = velocity(i);
      yvel[i] = velocity(i);
    }

    QPainter paint;
    paint.begin( d );                 // start painting desktop

    for ( int ntimes=0; ntimes<2000; ntimes++ ) {
      paint.setBrush( QColor(kindaRand()%360, 180, 255, QColor::Hsv) );
      paint.drawPolygon( a[head] );
      if ( ++tail >= maxcurves )
          tail = 0;

      int minx=r.left(), maxx=r.right();
      int miny=r.top(),  maxy=r.bottom();
      int x, y;
      p = &a[head];
      if ( ++head >= maxcurves )
          head = 0;
      for ( i=0; i<maxpoints; i++ ) {         // calc new curve
          p->point( i, &x, &y );
          x += xvel[i];
          y += yvel[i];
          if ( x >= maxx ) {
            x = maxx - (x - maxx + 1);
            xvel[i] = -velocity(i);
          }
          if ( x <= minx ) {
            x = minx + (minx - x + 1);
            xvel[i] = velocity(i);
          }
          if ( y >= maxy ) {
            y = maxy - (y - maxy + 1);
            yvel[i] = -velocity(i);
          }
          if ( y <= miny ) {
            y = miny + (miny - y + 1);
            yvel[i] = velocity(i);
          }
          a[head].setPoint( i, x, y );
      }
    }
    paint.end();                  // painting done
    delete[] a;
}

// Rotate pattern on desktop.
void rotate()
{
    int i;
    const int w = 64;
    const int h = 64;
    QImage image( w, h, 8, 128 );           // create image
    for ( i=0; i<128; i++ )           // build color table
      image.setColor( i, qRgb(i,0,0) );
    for ( int y=0; y<h; y++ ) {                // set image pixels
      uchar *p = image.scanLine(y);
      for ( int x=0; x<w; x++ )
          *p++ = (x+y)%128;
    }

    QPixmap pm;
    pm = image;                       // convert image to pixmap
    pm.setOptimization( QPixmap::BestOptim );   // rotation will be faster

    QWidget *d = QApplication::desktop();// w = desktop widget

    for ( i=0; i<=360; i += 2 ) {
      QWMatrix m;
      m.rotate( i );                  // rotate coordinate system
    QPixmap rpm = pm.xForm( m );         // rpm = rotated pixmap
    d->setBackgroundPixmap( rpm );       // set desktop pixmap
    d->update();                // repaint desktop
   }
}

// Generates a marble-like pattern in pm.
void generateStone( QPixmap *pm,  const QColor &c1, const QColor &c2, const QColor &c3 )
{
   QPainter p;
   QPen p1 ( c1, 0 );
   QPen p2 ( c2, 0 );
   QPen p3 ( c3, 0 );

   p.begin( pm );
   for( int i = 0 ; i < pm->width() ; i++ )
    for( int j = 0 ; j < pm->height() ; j++ ) {
       int r = kindaRand();
       if ( r < KINDA_RAND_MAX / 3 )
        p.setPen( p1 );
       else if ( r < KINDA_RAND_MAX / 3 * 2 )
        p.setPen( p2 );
       else
        p.setPen( p3 );
       p.drawPoint( i,j );
    }
   p.end();
}

void drawShadeText( QPainter *p, int x, int y, const char *text,
            const QColor &topColor, const QColor &bottomColor,   int sw = 2 )
{
   if ( !p->isActive() )
    return;

   p->setPen( bottomColor );
   p->drawText( x+sw, y+sw, text );
   p->setPen( topColor );
   p->drawText( x, y, text );
}

// NOTE: desktop drag/drop is experimental
class DesktopWidget : public QWidget, private QDropSite
{
public:
   DesktopWidget( const char *s, QWidget *parent=0, const char *name=0 );
  ~DesktopWidget();
   void paintEvent( QPaintEvent * );

   void dragEnterEvent( QDragEnterEvent *e )
   {
    if ( QImageDrag::canDecode(e) )
       e->accept();
   }
   void dragLeaveEvent( QDragLeaveEvent * )
   {
   }

   void dragMoveEvent( QDragMoveEvent *e )
   {
    e->accept();
   }

   void dropEvent( QDropEvent * e )
   {
    QPixmap pmp;
    if ( QImageDrag::decode( e, pmp ) ) {
        setBackgroundPixmap( pmp );
        update();
    }
   }

private:
   QPixmap *pm;
   QString text;
};

DesktopWidget::DesktopWidget( const char *s, QWidget *parent, const char *name )
   : QWidget( parent, name, WType_Desktop | WPaintDesktop), QDropSite(this)
{
   text = s;
   pm   = 0;
}

DesktopWidget::~DesktopWidget()
{
   delete pm;
}

void DesktopWidget::paintEvent( QPaintEvent * )
{
   QColor c1 = backgroundColor();
   QColor c2 = c1.light(104);
   QColor c3 = c1.dark(106);
   if ( !pm ) {
    pm = new QPixmap( 64, 64 );
    generateStone( pm, c1, c2, c3 );
    setBackgroundPixmap( *pm );
    update();
   }
   QRect br = fontMetrics().boundingRect( text );
   QPixmap offscreen( br.width(), br.height() );
   int x = width()/2  - br.width()/2;
   int y = height()/2 - br.height()/2;
   offscreen.fill( this, x, y );
   QPainter p;
   p.begin( &offscreen );
```

```
    drawShadeText( &p, -br.x(), -br.y(), text, c2, c3, 3 );
    p.end();
    bitBlt( this, x, y, &offscreen );
}

void desktopWidget( const char *s = "Trolltech" )
{
    DesktopWidget *t = new DesktopWidget(s);
    t->update();
    qApp->exec();
    delete t;
}

void desktopText( const char *s = "Trolltech" )
{
    const int border = 20;

    QColor c1 =      qApp->palette().inactive().background();
    QColor c2 = c1.light(104);
    QColor c3 = c1.dark(106);

    QPixmap pm(10,10);

    QPainter p;
    p.begin( &pm );
    QRect r = p.fontMetrics().boundingRect( s );
    p.end();

    int appWidth  =  qApp->desktop()->width();
    int appHeight =  qApp->desktop()->height();
    if ( r.width() > appWidth - border*2 )
      r.setWidth( appWidth - border*2 );
    if ( r.height() > appHeight - border*2 )
      r.setHeight( appHeight - border*2 );

    pm.resize( r.size() + QSize( border*2, border*2 ) );
    generateStone( &pm, c1, c2, c3 );
    p.begin( &pm );
    drawShadeText( &p, -r.x() + border, -r.y() + border, s, c2, c3 );
    p.end();

    qApp->desktop()->setBackgroundPixmap( pm );
}

// The program starts here.
int main( int argc, char **argv )
{
    QApplication app( argc, argv );

    if ( argc > 1 ) {
      QFont f( "charter", 96, QFont::Black );
      f.setStyleHint( QFont::Times );
      app.setFont( f );
    }
```

```
   bool validOptions = FALSE;

   if ( argc == 2 ) {
    validOptions = TRUE;
    if ( strcmp(argv[1],"-poly") == 0 )
       poly();
    else if ( strcmp(argv[1],"-rotate") == 0 )
       rotate();
    else if ( strcmp(argv[1],"-troll") == 0 )
       desktopText();
    else if ( strcmp(argv[1],"-trollwidget") == 0 )
       desktopWidget();
    else
       validOptions = FALSE;
   }
   if ( argc == 3 ) {
    validOptions = TRUE;
    if ( strcmp(argv[1],"-shadetext") == 0 )
       desktopText( argv[2] );
    else if ( strcmp(argv[1],"-shadewidget") == 0 )
       desktopWidget( argv[2] );
    else
       validOptions = FALSE;
   }
   if ( !validOptions ) {
    fprintf( stderr, "Usage:\n\tdesktop -poly"
                  "\n\tdesktop -rotate"
                  "\n\tdesktop -troll"
                  "\n\tdesktop -trollwidget"
                  "\n\tdesktop -shadetext <text>"
                  "\n\tdesktop -shadewidget <text>\n" );
    rotate();
   }
   return 0;
}
```

14. 등록부열람기

이 실례프로그람은 목록보기와 목록보기항목들을 사용하여 다중렬계층의 기억기 및 CPU에 효과적인 등록부열람기를 만든다. 또한 목록보기에서 끌기 및 놓기를 사용하는 방법을 보여준다.

dirview.pro

```
TEMPLATE  = app
TARGET    = dirview
CONFIG    += qt warn_on release
HEADERS       = dirview.h
SOURCES       = dirview.cpp \
          main.cpp
```

dirview.cpp

```
#include "dirview.h"
#include <qdir.h>
```

```
#include <qfile.h>
#include <qfileinfo.h>
#include <qpixmap.h>
#include <qevent.h>
#include <qpoint.h>
#include <qmessagebox.h>
#include <qdragobject.h>
#include <qmime.h>
#include <qstrlist.h>
#include <qstringlist.h>
#include <qapplication.h>
#include <qheader.h>

static const char* folder_closed_xpm[]={
    "16 16 9 1",
    "g c #808080",
    "b c #c0c000",
    "e c #c0c0c0",
    "# c #000000",
    "c c #ffff00",
    ". c None",
    "a c #585858",
    "f c #a0a0a4",
    "d c #ffffff",
    "..###...........",
    ".#abc##.........",
    ".#daabc#####....",
    ".#ddeaabbccc#...",
    ".#dedeeabbbba...",
    ".#edeeeeaaaab#..",
    ".#deeeeeeefe#ba.",
    ".#eeeeeeefef#ba.",
    ".#eeeeeefeff#ba.",
    ".#eeeeefefff#ba.",
    ".##geefeffff#ba.",
    "...##gefffff#ba.",
    ".....##fffff#ba.",
    ".......##fff#b##",
    ".........##f#b##",
    "..........####."};

static const char* folder_open_xpm[]={
    "16 16 11 1",
    "# c #000000",
    "g c #c0c0c0",
    "e c #303030",
    "a c #ffa858",
    "b c #808080",
    "d c #a0a0a4",
    "f c #585858",
    "c c #ffdca8",
    "h c #dcdcdc",
    "i c #ffffff",
    ". c None",
    "....###.........",
    "....#ab##.......",
    "....#acab####...",
    "###.#acccccca#..",
    "#ddefaaaccccca#.",
    "#bdddbaaaacccab#",
    ".eddddbbaaaacab#",
    ".#bddggdbbaaaab#",
    "..edgdggggbbaab#",
    "..#bgggghghdaab#",
    "...ebhggghicfab#",
    "....#edhhiiidab#",
    "......#egiiicfb#",
    "........#egiibb#",
    "..........#egib#",
    "............#ee#"};

static const char * folder_locked[]={
    "16 16 10 1",
    "h c #808080",
    "b c #ffa858",
    "f c #c0c0c0",
    "e c #c05800",
    "# c #000000",
    "c c #ffdca8",
    ". c None",
    "a c #585858",
    "g c #a0a0a4",
    "d c #ffffff",
    "..#a#...........",
    ".#abc####.......",
    ".#daa#eee#......",
    ".#ddf#e##b#.....",
    ".#dfd#e#bcb##...",
    ".#fdccc#daaab#..",
    ".#dfbbbccgfg#ba.",
    ".#ffb#ebbfgg#ba.",
    ".#ffbbe#bggg#ba.",
    ".#fffbbebggg#ba.",
    ".##hf#ebbggg#ba.",
    "...###e#gggg#ba.",
    ".....#e#gggg#ba.",
    "......###ggg#b##",
    ".........##g#b##",
    "...........####."};

static const char * pix_file []={
    "16 16 7 1",
    "# c #000000",
    "b c #ffffff",
    "e c #000000",
    "d c #404000",
    "c c #c0c000",
    "a c #ffffc0",
    ". c None",
    "................",
    ".........#......",
    "......#.#a##....",
    ".....#b#bbba##..",
    "....#b#bbbabbb#.",
    "...#b#bba##bb#..",
    "..#b#abb#bb##...",
    ".#a#aab#bbbab##.",
    "#a#aaa#bcbbbbbb#",
    "#ccdc#bcbbcbbb#.",
    ".##c#bcbbcabb#..",
    "...#acbacbbbe...",
    "..#aaaacaba#....",
    "...##aaaaa#.....",
    ".....##aa#......",
    ".......##......."};

QPixmap *folderLocked = 0;
QPixmap *folderClosed = 0;
QPixmap *folderOpen = 0;
QPixmap *fileNormal = 0;

/*****************************************************************************
* Class Directory
*****************************************************************************/

Directory::Directory( Directory * parent, const QString& filename )
    : QListViewItem( parent ), f(filename),
     showDirsOnly( parent->showDirsOnly ),
     pix( 0 )
{
    p = parent;
    readable = QDir( fullName() ).isReadable();

    if ( !readable )
     setPixmap( folderLocked );
    else
     setPixmap( folderClosed );
}

Directory::Directory( QListView * parent, const QString& filename )
    : QListViewItem( parent ), f(filename),
     showDirsOnly( ( (DirectoryView*)parent )->showDirsOnly() ),
     pix( 0 )
{
    p = 0;
    readable = QDir( fullName() ).isReadable();
}

void Directory::setPixmap( QPixmap *px )
{
    pix = px;
    setup();
```

```
    widthChanged( 0 );
    invalidateHeight();
    repaint();
}

const QPixmap *Directory::pixmap( int i ) const
{
    if ( i )
      return 0;
    return pix;
}

void Directory::setOpen( bool o )
{
    if ( o )
      setPixmap( folderOpen );
    else
      setPixmap( folderClosed );

    if ( o && !childCount() ) {
      QString s( fullName() );
      QDir thisDir( s );
      if ( !thisDir.isReadable() ) {
          readable = FALSE;
          setExpandable( FALSE );
          return;
      }

      listView()->setUpdatesEnabled( FALSE );
      const QFileInfoList * files = thisDir.entryInfoList();
      if ( files ) {
          QFileInfoListIterator it( *files );
          QFileInfo * fi;
          while( (fi=it.current()) != 0 ) {
            ++it;
            if ( fi->fileName() == "." || fi->fileName() == ".." )
                ; // nothing
            else if ( fi->isSymLink() && !showDirsOnly ) {
                FileItem *item = new FileItem( this, fi->fileName(),  "Symbolic Link" );
                item->setPixmap( fileNormal );
            }
            else if ( fi->isDir() )
                (void)new Directory( this, fi->fileName() );
            else if ( !showDirsOnly ) {
                FileItem *item = new FileItem( this, fi->fileName(),  fi->isFile()?"File":"Special" );
                item->setPixmap( fileNormal );
            }
          }
      }
      listView()->setUpdatesEnabled( TRUE );
    }
    QListViewItem::setOpen( o );
}
```

```
void Directory::setup()
{
   setExpandable( TRUE );
   QListViewItem::setup();
}

QString Directory::fullName()
{
   QString s;
   if ( p ) {
    s = p->fullName();
    s.append( f.name() );
    s.append( "/" );
   } else {
    s = f.name();
   }
   return s;
}

QString Directory::text( int column ) const
{
   if ( column == 0 )
    return f.name();
   else if ( readable )
    return "Directory";
   else
    return "Unreadable Directory";
}

/*****************************************************************
 * Class DirectoryView
*****************************************************************/

DirectoryView::DirectoryView( QWidget *parent, const char *name, bool sdo )
   : QListView( parent, name ), dirsOnly( sdo ), oldCurrent( 0 ),
    dropItem( 0 ), mousePressed( FALSE )
{
   autoopen_timer = new QTimer( this );
   if ( !folderLocked ) {
    folderLocked = new QPixmap( folder_locked );
    folderClosed = new QPixmap( folder_closed_xpm );
    folderOpen = new QPixmap( folder_open_xpm );
    fileNormal = new QPixmap( pix_file );
   }

   connect( this, SIGNAL( doubleClicked( QListViewItem * ) ),
        this, SLOT( slotFolderSelected( QListViewItem * ) ) );
   connect( this, SIGNAL( returnPressed( QListViewItem * ) ),
        this, SLOT( slotFolderSelected( QListViewItem * ) ) );

   setAcceptDrops( TRUE );
   viewport()->setAcceptDrops( TRUE );

   connect( autoopen_timer, SIGNAL( timeout() ),  this, SLOT( openFolder() ) );
```

```
}

void DirectoryView::slotFolderSelected( QListViewItem *i )
{
   if ( !i || !showDirsOnly() )
     return;

   Directory *dir = (Directory*)i;
   emit folderSelected( dir->fullName() );
}

void DirectoryView::openFolder()
{
   autoopen_timer->stop();
   if ( dropItem && !dropItem->isOpen() ) {
     dropItem->setOpen( TRUE );
     dropItem->repaint();
   }
}

static const int autoopenTime = 750;

void DirectoryView::contentsDragEnterEvent( QDragEnterEvent *e )
{
   if ( !QUriDrag::canDecode(e) ) {
     e->ignore();
     return;
   }

   oldCurrent = currentItem();

   QListViewItem *i = itemAt( contentsToViewport(e->pos()) );
   if ( i ) {
     dropItem = i;
     autoopen_timer->start( autoopenTime );
   }
}

void DirectoryView::contentsDragMoveEvent( QDragMoveEvent *e )
{
   if ( !QUriDrag::canDecode(e) ) {
     e->ignore();
     return;
   }

   QPoint vp = contentsToViewport( ( (QDragMoveEvent*)e )->pos() );
   QListViewItem *i = itemAt( vp );
   if ( i ) {
     setSelected( i, TRUE );
     e->accept();
     if ( i != dropItem ) {
         autoopen_timer->stop();
         dropItem = i;
         autoopen_timer->start( autoopenTime );
```

```
      }
      switch ( e->action() ) {
      case QDropEvent::Copy:
         break;
      case QDropEvent::Move:
         e->acceptAction();
         break;
      case QDropEvent::Link:
         e->acceptAction();
         break;
      default:
         ;
      }
    } else {
      e->ignore();
      autoopen_timer->stop();
      dropItem = 0;
    }
}

void DirectoryView::contentsDragLeaveEvent( QDragLeaveEvent * )
{
    autoopen_timer->stop();
    dropItem = 0;

    setCurrentItem( oldCurrent );
    setSelected( oldCurrent, TRUE );
}

void DirectoryView::contentsDropEvent( QDropEvent *e )
{
    autoopen_timer->stop();

    if ( !QUriDrag::canDecode(e) ) {
      e->ignore();
      return;
    }

    QListViewItem *item = itemAt( contentsToViewport(e->pos()) );
    if ( item ) {

      QStrList lst;

      QUriDrag::decode( e, lst );

      QString str;

      switch ( e->action() ) {
         case QDropEvent::Copy:
         str = "Copy";
         break;
         case QDropEvent::Move:
         str = "Move";
         e->acceptAction();
```

```
     break;
     case QDropEvent::Link:
     str = "Link";
     e->acceptAction();
     break;
     default:
     str = "Unknown";
   }

   str += "\n\n";

   e->accept();

   for ( uint i = 0; i < lst.count(); ++i ) {
     QString filename = QDir::convertSeparators(QUriDrag::uriToLocalFile(lst.at(i)));
     str += filename + "\n";
   }
   str += QString( "\nTo\n\n   %1" ) .arg( QDir::convertSeparators(fullPath(item)) );

   QMessageBox::information( this, "Drop target", str, "Not implemented" );
  } else
   e->ignore();

}

QString DirectoryView::fullPath(QListViewItem* item)
{
  QString fullpath = item->text(0);
  while ( (item=item->parent()) ) {
   if ( item->parent() )
     fullpath = item->text(0) + "/" + fullpath;
   else
     fullpath = item->text(0) + fullpath;
  }
#ifdef Q_WS_WIN
   if (fullpath.length() > 2 && fullpath[1] != ':') {
      QDir dir(fullpath);
      fullpath = dir.currentDirPath().left(2) + fullpath;
   }
#endif

  return fullpath;
}

void DirectoryView::contentsMousePressEvent( QMouseEvent* e )
{
  QListView::contentsMousePressEvent(e);
  QPoint p( contentsToViewport( e->pos() ) );
  QListViewItem *i = itemAt( p );
  if ( i ) {
   // if the user clicked into the root decoration of the item, don't try to start a drag!
   if ( p.x() > header()->cellPos( header()->mapToActual( 0 ) ) +
      treeStepSize() * ( i->depth() + ( rootIsDecorated() ? 1 : 0) ) + itemMargin() ||
      p.x() < header()->cellPos( header()->mapToActual( 0 ) ) ) {
        presspos = e->pos();
        mousePressed = TRUE;
     }
   }
}

void DirectoryView::contentsMouseMoveEvent( QMouseEvent* e )
{
   if ( mousePressed && ( presspos - e->pos() ).manhattanLength() >
QApplication::startDragDistance() ) {
     mousePressed = FALSE;
     QListViewItem *item = itemAt( contentsToViewport(presspos) );
     if ( item ) {
        QString source = fullPath(item);
        if ( QFile::exists(source) ) {
          QUriDrag* ud = new QUriDrag(viewport());
          ud->setFileNames( source );
          if ( ud->drag() )
             QMessageBox::information( this, "Drag source",
             QString("Delete ") + QDir::convertSeparators(source), "Not implemented" );
        }
     }
   }
}

void DirectoryView::contentsMouseReleaseEvent( QMouseEvent * )
{
   mousePressed = FALSE;
}

void DirectoryView::setDir( const QString &s )
{
   QListViewItemIterator it( this );
   ++it;
   for ( ; it.current(); ++it ) {
     it.current()->setOpen( FALSE );
   }

   QStringList lst( QStringList::split( "/", s ) );
   QListViewItem *item = firstChild();
   QStringList::Iterator it2 = lst.begin();
   for ( ; it2 != lst.end(); ++it2 ) {
     while ( item ) {
        if ( item->text( 0 ) == *it2 ) {
          item->setOpen( TRUE );
          break;
        }
        item = item->itemBelow();
     }
   }

   if ( item )
     setCurrentItem( item );
}
```

```
void FileItem::setPixmap( QPixmap *p )
{
   pix = p;
   setup();
   widthChanged( 0 );
   invalidateHeight();
   repaint();
}

const QPixmap *FileItem::pixmap( int i ) const
{
   if ( i )
     return 0;
   return pix;
}
```

dirview.h

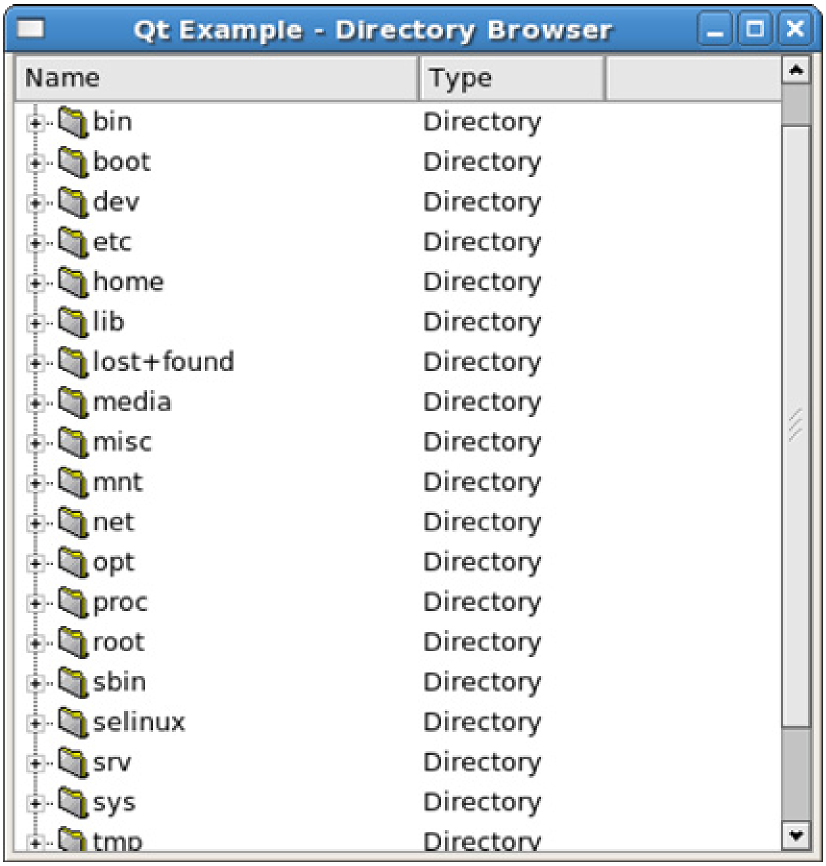
```
#ifndef DIRVIEW_H
#define DIRVIEW_H

#include <qlistview.h>
#include <qstring.h>
#include <qfile.h>
#include <qfileinfo.h>
#include <qtimer.h>

class QWidget;
class QDragEnterEvent;
class QDragMoveEvent;
class QDragLeaveEvent;
class QDropEvent;

class FileItem : public QListViewItem
{
public:
   FileItem( QListViewItem *parent, const QString &s1, const QString &s2 )
     : QListViewItem( parent, s1, s2 ), pix( 0 ) {}

   const QPixmap *pixmap( int i ) const;
#if !defined(Q_NO_USING_KEYWORD)
   using QListViewItem::setPixmap;
#endif
   void setPixmap( QPixmap *p );

private:
   QPixmap *pix;

};

class Directory : public QListViewItem
{
public:
   Directory( QListView * parent, const QString& filename );
```

```
    Directory( Directory * parent, const QString& filename, const QString &col2 )
      : QListViewItem( parent, filename, col2 ), pix( 0 ) {}
    Directory( Directory * parent, const QString& filename );

    QString text( int column ) const;

    QString fullName();

    void setOpen( bool );
    void setup();

    const QPixmap *pixmap( int i ) const;
#if !defined(Q_NO_USING_KEYWORD)
    using QListViewItem::setPixmap;
#endif
    void setPixmap( QPixmap *p );

private:
    QFile f;
    Directory * p;
    bool readable;
    bool showDirsOnly;
    QPixmap *pix;

};

class DirectoryView : public QListView
{
    Q_OBJECT

public:
    DirectoryView( QWidget *parent = 0, const char *name = 0, bool sdo = FALSE );
    bool showDirsOnly() { return dirsOnly; }

public slots:
    void setDir( const QString & );

signals:
    void folderSelected( const QString & );

protected slots:
    void slotFolderSelected( QListViewItem * );
    void openFolder();

protected:
    void contentsDragEnterEvent( QDragEnterEvent *e );
    void contentsDragMoveEvent( QDragMoveEvent *e );
    void contentsDragLeaveEvent( QDragLeaveEvent *e );
    void contentsDropEvent( QDropEvent *e );
    void contentsMouseMoveEvent( QMouseEvent *e );
    void contentsMousePressEvent( QMouseEvent *e );
    void contentsMouseReleaseEvent( QMouseEvent *e );

private:
    QString fullPath(QListViewItem* item);
    bool dirsOnly;
    QListViewItem *oldCurrent;
    QListViewItem *dropItem;
    QTimer* autoopen_timer;
    QPoint presspos;
    bool mousePressed;

};

#endif
```

main.cpp

```
#include <qapplication.h>
#include <qfileinfo.h>
#include <qdir.h>
#include "dirview.h"

int main( int argc, char ** argv )
{
    QApplication a( argc, argv );

    DirectoryView mw;

    mw.addColumn( "Name" );
    mw.addColumn( "Type" );
    mw.setTreeStepSize( 20 );

    const QFileInfoList* roots = QDir::drives();
    QPtrListIterator<QFileInfo> i(*roots);
    QFileInfo* fi;
    while ( (fi = *i) ) {
     ++i;
     Directory * root = new Directory( &mw, fi->filePath() );
     if ( roots->count() <= 1 )
         root->setOpen( TRUE ); // be interesting
    }

    mw.resize( 400, 400 );
    mw.setCaption( "Qt Example - Directory Browser" );
    mw.setAllColumnsShowFocus( TRUE );
    a.setMainWidget( &mw );
    mw.show();

    return a.exec();
}
```

실행



Name	Type
bin	Directory
boot	Directory
dev	Directory
etc	Directory
home	Directory
lib	Directory
lost+found	Directory
media	Directory
misc	Directory
mnt	Directory
net	Directory
opt	Directory
proc	Directory
root	Directory
sbin	Directory
selinux	Directory
srv	Directory
sys	Directory
tmp	Directory

15. Qt Distribution Example 배포물실례

이 실례프로그람은 Qt서고로 콤파일되는 부호화된 경로들을 변경한다. Qt서고안에서 다음의 부호화된 경로들을 수정하는데 이 실례의 코드를 리용할수 있다.

- Prefix – 보통 다른 모든 경로는 *Prefix* 와 관련된다.
- Binaries – Qt 와 함께 배포된 2 진파일들(실례로 *Qt Assistant*)의 위치.
- Documentation – Qt 문서의 위치.
- Headers – Qt 머리부의 위치.
- Libraries – Qt 와 함께 배포된 추가서고(실례로 *qui* 서고)의 위치.
- Plugins – Qt 플라그인의 위치.
- Data – Qt 와 함께 배포된 모든 프로그람을 위한 응용프로그람에 고유한 자료의 위치.

distributor.pro

```
TEMPLATE  = app
LANGUAGE= C++
TARGET     = distributor
CONFIG     += qt warn_on
SOURCES       += main.cpp
FORMS       = distributor.ui
```

distributor.ui

```
<!DOCTYPE UI><UI version="3.2" stdsetdef="1">
```

```
<class>Distributor</class>
<widget class="QWizard">
   <property name="name">
…
</functions>
<pixmapinproject/>
<layoutdefaults spacing="6" margin="11"/>
</UI>
```

distributor.ui.h

```
#include <qapplication.h>
#include <qcursor.h>
#include <qeventloop.h>
#include <qfile.h>
#include <qfiledialog.h>
#include <qfileinfo.h>
#include <qlineedit.h>
#include <qmessagebox.h>
#include <qpushbutton.h>
#include <qtimer.h>

void Distributor::init()
{
   timer = new QTimer( this );
   connect( timer, SIGNAL(timeout()), SLOT(checkLibData()) );

   cancelButton()->setAutoDefault( FALSE );
   backButton()->setAutoDefault( FALSE );

   setNextEnabled( selectLibrary, FALSE );

   setHelpEnabled( selectLibrary, FALSE );
   setHelpEnabled( modifyPaths, FALSE );
   setHelpEnabled( verifyMods, FALSE );

   setFinishEnabled( verifyMods, TRUE );
}

void Distributor::showPage( QWidget *page )
{
   if ( page == selectLibrary ) {
     nextButton()->setDefault( TRUE );
     libFilename->setFocus();
   } else if ( page == modifyPaths ) {
     nextButton()->setDefault( TRUE );
     prefixPath->selectAll();
     prefixPath->setFocus();
   } else if ( page == verifyMods ) {
     finishButton()->setDefault( TRUE );
     finishButton()->setFocus();

     QString labeltext =
        tr("<p><b>Current Library File:</b> %1</p>"
          "<table border=0>"
```

```
       "<tr><td><b>New Installation Prefix:</b></td><td>%2</td></tr>"
       "<tr><td></td><td></td></tr>"
       "<tr><td><b>Binaries Path:</b></td><td>%3</td></tr>"
       "<tr><td><b>Documentation Path:</b></td><td>%4</td></tr>"
       "<tr><td><b>Headers Path:</b></td><td>%5</td></tr>"
       "<tr><td><b>Libraries Path:</b></td><td>%6</td></tr>"
       "<tr><td><b>Plugins Path:</b></td><td>%7</td></tr>"
       "<tr><td><b>Data Path:</b></td><td>%8</td></tr>"
       "</table>"
       "<p>Please verify that these options are correct.  Press the "
       "<i>Finish</i> button to apply these modifications to the Qt "
       "library.  Use the <i>Back</i> button to make corrections.  Use "
       "the <i>Cancel</i> button to abort.</p>")
     .arg( libFilename->text() )
     .arg( prefixPath->text() )
     .arg( binPath->text() )
     .arg( docPath->text() )
     .arg( hdrPath->text() )
     .arg( libPath->text() )
     .arg( plgPath->text() )
     .arg( datPath->text() );
   textLabel4->setText( labeltext );
  }

  QWizard::showPage( page );
}

void Distributor::checkLibFilename( const QString &filename )
{
  setNextEnabled( selectLibrary, FALSE );

  QFileInfo fileinfo( filename );
  if ( ! filename.isEmpty() && fileinfo.exists() &&
    fileinfo.isReadable() && fileinfo.isWritable() &&
    fileinfo.isFile() && !fileinfo.isSymLink() )
   timer->start( 500, TRUE );
}

void Distributor::browseLibFilename()
{
  QString filename =
   QFileDialog::getOpenFileName( QString::null, QString::null, this );
  libFilename->setText( filename );
}

static char *find_pattern( char *h, const char *n, ulong hlen )
{
  if ( ! h || ! n || hlen == 0 )
   return 0;

#ifdef Q_OS_UNIX
  size_t nlen;
#else
  ulong nlen;
```

```
#endif

    char nc = *n++;
    nlen = strlen( n );
    char hc;

    do {
      do {
          hc = *h++;
          if ( hlen-- < 1 )
            return 0;
      } while ( hc != nc );

      if ( nlen > hlen )
          return 0;
    } while ( qstrncmp( h, n, nlen ) != 0 );
    return h + nlen;
}

void Distributor::checkLibData()
{
    struct step {
      const char *key;
      QCString value;
      bool done;
    } steps[7];

    steps[0].key = "qt_nstpath=";
    steps[0].done = FALSE;

    steps[1].key = "qt_binpath=";
    steps[1].done = FALSE;

    steps[2].key = "qt_docpath=";
    steps[2].done = FALSE;

    steps[3].key = "qt_hdrpath=";
    steps[3].done = FALSE;

    steps[4].key = "qt_libpath=";
    steps[4].done = FALSE;

    steps[5].key = "qt_plgpath=";
    steps[5].done = FALSE;

    steps[6].key = "qt_datpath=";
    steps[6].done = FALSE;

    uint completed = 0;
    uint total_steps = sizeof(steps) / sizeof(step);

    QFile file( libFilename->text() );
    if ( file.open( IO_ReadOnly ) ) {
      QApplication::setOverrideCursor( WaitCursor );
```

```
 // instead of reading in the entire file, do the search in chunks
 char data[60000];
 ulong offset = 0;

 while ( ! file.atEnd() && completed < total_steps ) {
    QApplication::eventLoop()->processEvents( QEventLoop::ExcludeUserInput );

    ulong len = file.readBlock( data, sizeof(data) );
    if ( len < 267 ) {
     // not enough room to make any modifications... stop
     break;
    }

    for ( uint x = 0; x < total_steps; ++x ) {
     if ( steps[x].done ) continue;

     char *s = find_pattern( data, steps[x].key, len );
     if ( s ) {
        ulong where = s - data;
        if ( len - where < 256 ) {
         // not enough space left to write the full
         // path... move the file pointer back to just
         // before the pattern and continue
         offset += where - 11;
         file.at( offset );
         len = file.readBlock( data, sizeof(data) );
         --x; // retry the current step
         continue;
        }

        steps[x].value = s;
        steps[x].done = TRUE;

        ++completed;
     }
    }

    // move to the new read position
    offset += len - 11;
    file.at( offset );
 }

 file.close();

 QApplication::restoreOverrideCursor();
}

if ( completed == total_steps ) {
 setNextEnabled( selectLibrary, TRUE );

 QString prefix = QFile::decodeName( steps[0].value );
 prefixPath->setText( prefix );
```

```
   QString def_bin = prefix + QString::fromLatin1( "/bin" );
   QString def_doc = prefix + QString::fromLatin1( "/doc" );
   QString def_hdr = prefix + QString::fromLatin1( "/include" );
   QString def_lib = prefix + QString::fromLatin1( "/lib" );
   QString def_plg = prefix + QString::fromLatin1( "/plugins" );
   QString def_dat = prefix;

   QString bin = QFile::decodeName( steps[1].value );
   QString doc = QFile::decodeName( steps[2].value );
   QString hdr = QFile::decodeName( steps[3].value );
   QString lib = QFile::decodeName( steps[4].value );
   QString plg = QFile::decodeName( steps[5].value );
   QString dat = QFile::decodeName( steps[6].value );

   autoSet->setChecked( def_bin == bin &&
             def_doc == doc &&
             def_hdr == hdr &&
             def_lib == lib &&
             def_plg == plg &&
             def_dat == dat );

   if ( ! autoSet->isChecked() ) {
     binPath->setText( bin );
     docPath->setText( doc );
     hdrPath->setText( hdr );
     libPath->setText( lib );
     plgPath->setText( plg );
     datPath->setText( dat );
   }
  }
}

void Distributor::checkInstallationPrefix( const QString &prefix )
{
  if ( autoSet->isChecked() ) {
   binPath->setText( prefix + QString::fromLatin1( "/bin" ) );
   docPath->setText( prefix + QString::fromLatin1( "/doc" ) );
   hdrPath->setText( prefix + QString::fromLatin1( "/include" ) );
   libPath->setText( prefix + QString::fromLatin1( "/lib" ) );
   plgPath->setText( prefix + QString::fromLatin1( "/plugins" ) );
   datPath->setText( prefix );
  }
}

void Distributor::browseInstallationPrefix()
{
  QString prefix =
   QFileDialog::getOpenFileName( QString::null, QString::null, this );
  prefixPath->setText( prefix );
}

void Distributor::toggleAutoSet( bool autoset )
{
  if ( autoset ) checkInstallationPrefix( prefixPath->text() );
```

```
}

void Distributor::accept()
{
   struct step {
    const char *key;
    QCString value;
    bool done;
   } steps[7];

   steps[0].key = "qt_nstpath=";
   steps[0].value = QFile::encodeName( prefixPath->text() );
   steps[0].done = FALSE;

   steps[1].key = "qt_binpath=";
   steps[1].value = QFile::encodeName( binPath->text() );
   steps[1].done = FALSE;

   steps[2].key = "qt_docpath=";
   steps[2].value = QFile::encodeName( docPath->text() );
   steps[2].done = FALSE;

   steps[3].key = "qt_hdrpath=";
   steps[3].value = QFile::encodeName( hdrPath->text() );
   steps[3].done = FALSE;

   steps[4].key = "qt_libpath=";
   steps[4].value = QFile::encodeName( libPath->text() );
   steps[4].done = FALSE;

   steps[5].key = "qt_plgpath=";
   steps[5].value = QFile::encodeName( plgPath->text() );
   steps[5].done = FALSE;

   steps[6].key = "qt_datpath=";
   steps[6].value = QFile::encodeName( datPath->text() );
   steps[6].done = FALSE;

   uint completed = 0;
   uint total_steps = sizeof(steps) / sizeof(step);

   QFile file( libFilename->text() );
   if ( file.open( IO_ReadWrite ) ) {
    QApplication::setOverrideCursor( WaitCursor );

    // instead of reading in the entire file, do the search in chunks
    char data[60000];
    ulong offset = 0;

    while ( ! file.atEnd() && completed < total_steps ) {
       QApplication::eventLoop()->processEvents( QEventLoop::ExcludeUserInput );

       ulong len = file.readBlock( data, sizeof(data) );
       if ( len < 267 ) {
```

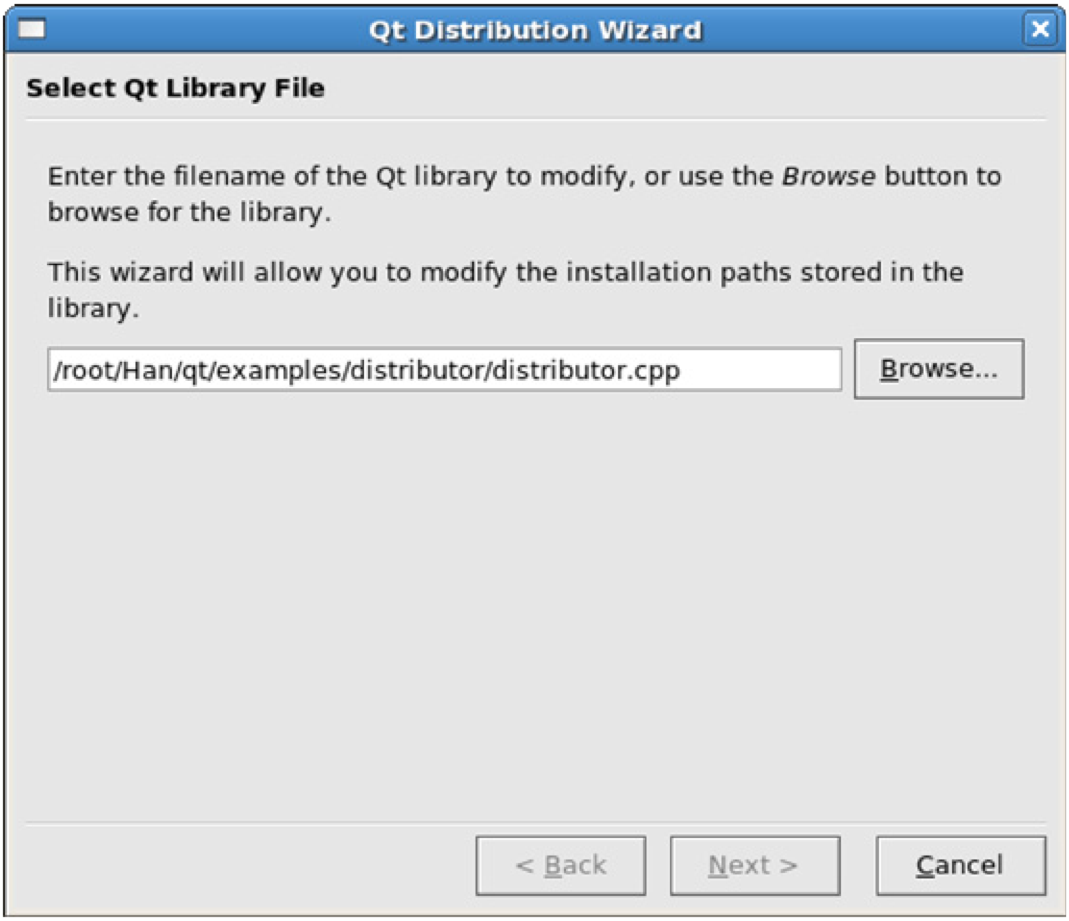
```
      // not enough room to make any modifications... stop
      break;
    }

    uint completed_save = completed;
    for ( uint x = 0; x < total_steps; ++x ) {
      if ( steps[x].done ) continue;

      char *s = find_pattern( data, steps[x].key, len );
      if ( s ) {
        ulong where = s - data;
        if ( len - where < 256 ) {
          // not enough space left to write the full
          // path... move the file pointer back to just
          // before the pattern and continue
          offset += where - 11;
          file.at( offset );
          len = file.readBlock( data, sizeof(data) );
          --x; // retry the current step
          continue;
        }

        qstrcpy( s, steps[x].value );
        steps[x].done = TRUE;

        ++completed;
      }
    }

    if ( completed != completed_save ) {
      // something changed...  move file pointer back to
      // where the data was read and write the new data
      file.at( offset );
      file.writeBlock( data, len );
    }

    // move to the new read position
    offset += len - 11;
    file.at( offset );
  }

  file.close();

  QApplication::restoreOverrideCursor();
}

if ( completed != total_steps ) {
  QMessageBox::information( this,
                tr("Qt Distribution Wizard"),
                tr("<p><h3>Modifications failed.</h3></p>"
                   "<p>Please make sure that you have permission "
                   "to write the selected file, and that the library "
                   "is properly built.</p>") );
  return;
```

```
    }

    QWizard::accept();
}
```

main.cpp

```
#include <qapplication.h>
#include "distributor.h"

int main( int argc, char ** argv )
{
    QApplication a( argc, argv );
    Distributor w;
    w.show();
    a.connect( &a, SIGNAL( lastWindowClosed() ), &a, SLOT( quit() ) );
    return a.exec();
}
```

실행



Qt Distribution Wizard

Select Qt Library File

Enter the filename of the Qt library to modify, or use the *Browse* button to browse for the library.

This wizard will allow you to modify the installation paths stored in the library.

/root/Han/qt/examples/distributor/distributor.cpp

16. 끌기와 놓기(1)

이 프로그람은 Qt 의 끌기와 놓기기능을 보여준다.

dragdrop.pro

```
TEMPLATE  = app
TARGET      = dragdrop
CONFIG       += qt warn_on release
HEADERS         = dropsite.h \
            secret.h
SOURCES         = dropsite.cpp \
            main.cpp \
            secret.cpp
```

dropsite.cpp

```
#include "dropsite.h"
#include "secret.h"
#include <qevent.h>
#include <qpixmap.h>
#include <qdragobject.h>
#include <qimage.h>
#include <qdir.h>

DropSite::DropSite( QWidget * parent, const char * name )   : QLabel( parent, name )
{
   setAcceptDrops(TRUE);
}

DropSite::~DropSite()
{
   // nothing necessary
}

void DropSite::dragMoveEvent( QDragMoveEvent *e )
{
   // Check if you want the drag at e->pos()...
   // Give the user some feedback - only copy is possible
   e->acceptAction( e->action() == QDropEvent::Copy );
}

void DropSite::dragEnterEvent( QDragEnterEvent *e )
{
   // Check if you want the drag...
   if ( SecretDrag::canDecode( e )
     || QTextDrag::canDecode( e )
     || QImageDrag::canDecode( e )
     || QUriDrag::canDecode( e ) )
   {
    e->accept();
   }

   // Give the user some feedback...
   QString t;
   const char *f;
   for( int i=0; (f=e->format( i )); i++ ) {
    if ( *(f) ) {
       if ( !t.isEmpty() )
         t += "\n";
```

```
      t += f;
    }
  }
  emit message( t );
  setBackgroundColor(white);
}

void DropSite::dragLeaveEvent( QDragLeaveEvent * )
{
  // Give the user some feedback...
  emit message("");
  setBackgroundColor(lightGray);
}

void DropSite::dropEvent( QDropEvent * e )
{
  setBackgroundColor(lightGray);

  // Try to decode to the data you understand...
  QStrList strings;
  if ( QUriDrag::decode( e, strings ) ) {
   QString m("Full URLs:\n");
   for (const char* u=strings.first(); u; u=strings.next())
      m = m + "   " + u + '\n';
   QStringList files;
   if ( QUriDrag::decodeLocalFiles( e, files ) ) {
      m += "Files:\n";
      for (QStringList::Iterator i=files.begin(); i!=files.end(); ++i)
       m = m + "   " + QDir::convertSeparators(*i) + '\n';
   }
   setText( m );
   setMinimumSize( minimumSize().expandedTo( sizeHint() ) );
   return;
  }

  QString str;
  if ( QTextDrag::decode( e, str ) ) {
   setText( str );
   setMinimumSize( minimumSize().expandedTo( sizeHint() ) );
   return;
  }

  QPixmap pm;
  if ( QImageDrag::decode( e, pm ) ) {
   setPixmap( pm );
   setMinimumSize( minimumSize().expandedTo( sizeHint() ) );
   return;
  }

  if ( SecretDrag::decode( e, str ) ) {
   setText( str );
   setMinimumSize( minimumSize().expandedTo( sizeHint() ) );
   return;
  }
```

```
}

DragMoviePlayer::DragMoviePlayer( QDragObject* p ) :
   QObject(p),    dobj(p),    movie("trolltech.gif" )
{
   movie.connectUpdate(this,SLOT(updatePixmap(const QRect&)));
}

void DragMoviePlayer::updatePixmap( const QRect& )
{
   dobj->setPixmap(movie.framePixmap());
}

void DropSite::mousePressEvent( QMouseEvent * /*e*/ )
{
   QDragObject *drobj;
   if ( pixmap() ) {
     drobj = new QImageDrag( pixmap()->convertToImage(), this );
#if 1
     QPixmap pm;
     pm.convertFromImage(pixmap()->convertToImage().smoothScale(
        pixmap()->width()/3,pixmap()->height()/3));
     drobj->setPixmap(pm,QPoint(-5,-7));
#else
     // Try it.
     (void)new DragMoviePlayer(drobj);
#endif
   } else {
     drobj = new QTextDrag( text(), this );
   }
   drobj->dragCopy();
}

void DropSite::backgroundColorChange( const QColor & )
{
   // Reduce flicker by using repaint() rather than update()
   repaint();
}
```

dropsite.h

```
#ifndef DROPSITE_H
#define DROPSITE_H

#include <qlabel.h>
#include <qmovie.h>
#include "qdropsite.h"

class QDragObject;

class DropSite: public QLabel
{
   Q_OBJECT
public:
   DropSite( QWidget * parent = 0, const char * name = 0 );
   ~DropSite();

signals:
   void message( const QString& );

protected:
   void dragEnterEvent( QDragEnterEvent * );
   void dragMoveEvent( QDragMoveEvent * );
   void dragLeaveEvent( QDragLeaveEvent * );
   void dropEvent( QDropEvent * );
   void backgroundColorChange( const QColor& );

   // this is a normal even
   void mousePressEvent( QMouseEvent * );
};

class DragMoviePlayer : public QObject {
   Q_OBJECT
   QDragObject* dobj;
   QMovie movie;
public:
   DragMoviePlayer(QDragObject*);
private slots:
   void updatePixmap( const QRect& );
};

#endif
```

secret.cpp

```
#include "secret.h"
#include <qevent.h>

//create the object withe the secret byte
SecretDrag::SecretDrag( uchar secret, QWidget * parent, const char * name )
   : QStoredDrag( "secret/magic", parent, name )
{
   QByteArray data(1);
   data[0]= secret;
   setEncodedData( data );
}

bool SecretDrag::canDecode( QDragMoveEvent* e )
{
   return e->provides( "secret/magic" );
}

//decode it into a string
bool SecretDrag::decode( QDropEvent* e, QString& str )
{
   QByteArray payload = e->data( "secret/magic" );
   if ( payload.size() ) {
    e->accept();
    QString msg;
    msg.sprintf("The secret number is %d", payload[0] );
```

```
        str = msg;
        return TRUE;
    }
    return FALSE;
}

SecretSource::SecretSource( int secret, QWidget *parent, const char * name )
    : QLabel( "Secret", parent, name )
{
    setBackgroundColor( blue.light() );
    setFrameStyle( Box | Sunken );
    setMinimumHeight( sizeHint().height()*2 );
    setAlignment( AlignCenter );
    mySecret = secret;
}

SecretSource::~SecretSource()
{
}

/* XPM */
static const char * picture_xpm[] = {
"16 16 3 1",
"   c None",
".  c #000000",
"X c #FFFF00",
"     .....      ",
"   ..XXXXX..    ",
"  .XXXXXXXXX.   ",
" .XXXXXXXXXXX.  ",
" .XX..XXX..XX.  ",
".XXXXXXXXXXXXX. ",
".XX...XXX...XX. ",
".XXX..XXX..XXX. ",
".XXXXXXXXXXXXX. ",
".XXXXXX.XXXXXX. ",
" .XX.XX.XX.XX.  ",
" .XXX..X..XXX.  ",
"  .XXXXXXXXX.   ",
"   ..XXXXX..    ",
"     .....      ",
"                "};

void SecretSource::mousePressEvent( QMouseEvent * /*e*/ )
{
    SecretDrag *sd = new SecretDrag( mySecret, this );
    sd->setPixmap(QPixmap(picture_xpm),QPoint(8,8));
    sd->dragCopy();
    mySecret++;
}
```

secret.h

```
#ifndef SECRETDRAG_H
#define SECRETDRAG_H
```

```
#include <qdragobject.h>
#include <qlabel.h>

class SecretDrag: public QStoredDrag {
public:
   SecretDrag( uchar, QWidget * parent = 0, const char * name = 0 );
   ~SecretDrag() {};

   static bool canDecode( QDragMoveEvent* e );
   static bool decode( QDropEvent* e, QString& s );
};

class SecretSource: public QLabel
{
public:
   SecretSource( int secret, QWidget *parent = 0, const char * name = 0 );
   ~SecretSource();

protected:
   void mousePressEvent( QMouseEvent * );
private:
   int mySecret;
};

#endif
```

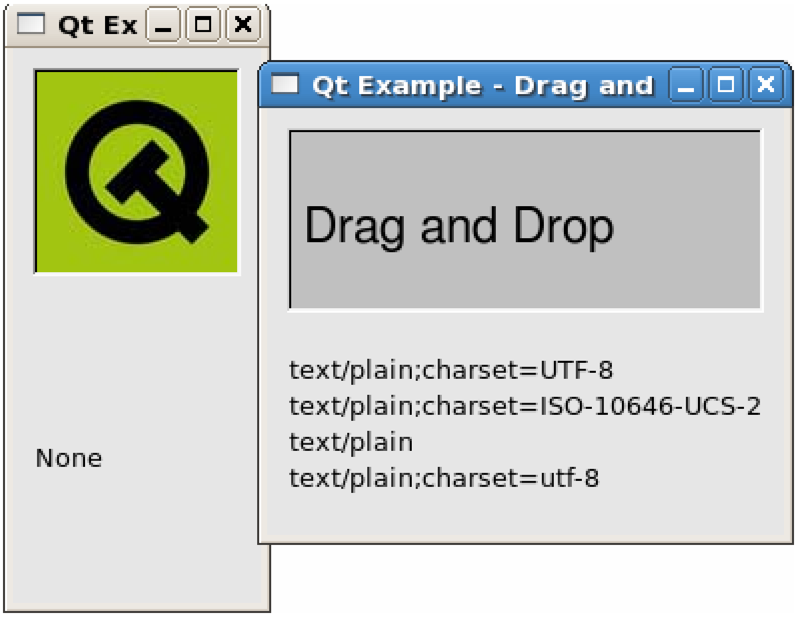
main.cpp

```
#include <qapplication.h>
#include "dropsite.h"
#include "secret.h"
#include <qlayout.h>
#include <qcombobox.h>
#include <qlabel.h>
#include <qpixmap.h>

static void addStuff( QWidget * parent, bool image, bool secret = FALSE )
{
   QVBoxLayout * tll = new QVBoxLayout( parent, 10 );
   DropSite * d = new DropSite( parent );
   d->setFrameStyle( QFrame::Sunken + QFrame::WinPanel );
   tll->addWidget( d );
   if ( image ) {
    QPixmap stuff;
    if ( !stuff.load( "trolltech.bmp" ) ) {
       stuff = QPixmap(20,20);
       stuff.fill(Qt::green);
    }
    d->setPixmap( stuff );
   } else {
    d->setText("Drag and Drop");
   }
   d->setFont(QFont("Helvetica",18));
   if ( secret ) {
    SecretSource *s = new SecretSource( 42, parent );
    tll->addWidget( s );
   }

   QLabel * format = new QLabel( "\n\n\n\nNone\n\n\n\n", parent );
   tll->addWidget( format );
   tll->activate();
   parent->resize( parent->sizeHint() );

   QObject::connect( d, SIGNAL(message(const QString&)),
             format, SLOT(setText(const QString&)) );
}

int main( int argc, char ** argv )
{
   QApplication a( argc, argv );

   QWidget mw;
   addStuff( &mw, TRUE );
   mw.setCaption( "Qt Example - Drag and Drop" );
   mw.show();

   QWidget mw2;
   addStuff( &mw2, FALSE );
   mw2.setCaption( "Qt Example - Drag and Drop" );
   mw2.show();

   QWidget mw3;
   addStuff( &mw3, TRUE, TRUE );
   mw3.setCaption( "Qt Example - Drag and Drop" );
   mw3.show();

   QObject::connect(qApp,SIGNAL(lastWindowClosed()),qApp,SLOT(quit()));
   return a.exec();
}
```

trolltec.gif

trolltec.bmp

실행



Drag and Drop

text/plain;charset=UTF-8
text/plain;charset=ISO-10646-UCS-2
text/plain
text/plain;charset=utf-8

None

17. 그리기프로그람

이 실례는 Qt의 몇가지 그리기함수와 인쇄기출력을 보여준다. 사용자정의그리기함수를 간단히 추가할수 있다.

drawdemo.pro

```
TEMPLATE  = app
TARGET    = drawdemo
CONFIG    += qt warn_on release
HEADERS      =
SOURCES      = drawdemo.cpp
```

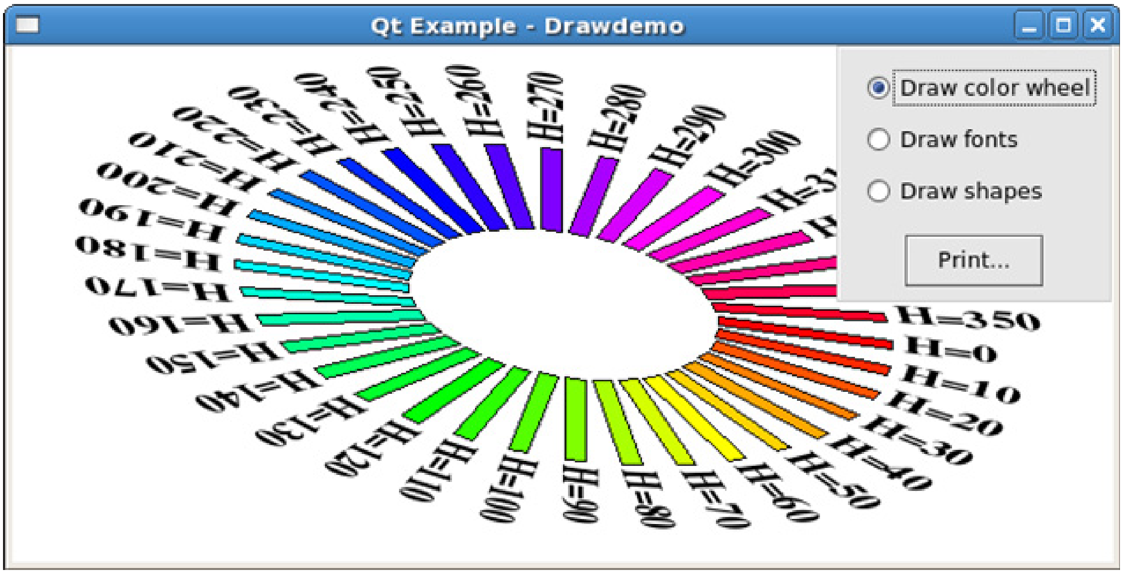
drawdemo.cpp

```
#include <qwidget.h>
#include <qpainter.h>
#include <qprinter.h>
#include <qpushbutton.h>
#include <qradiobutton.h>
#include <qbuttongroup.h>
#include <qapplication.h>
#include <math.h>

// First we define the functionality our demo should present
// to the user. You might add different demo-modes if you wish so.

// This function draws a color wheel.
// The coordinate system x=(0..500), y=(0..500) spans the paint device.

void drawColorWheel( QPainter *p )
{
   QFont f( "times", 18, QFont::Bold );
   p->setFont( f );
```

```
    p->setPen( Qt::black );
    p->setWindow( 0, 0, 500, 500 );      // defines coordinate system

    for ( int i=0; i<36; i++ ) {      // draws 36 rotated rectangles

        QWMatrix matrix;
        matrix.translate( 250.0F, 250.0F );    // move to center
        matrix.shear( 0.0F, 0.3F );      // twist it
        matrix.rotate( (float)i*10 );          // rotate 0,10,20,.. degrees
        p->setWorldMatrix( matrix );      // use this world matrix

        QColor c;
        c.setHsv( i*10, 255, 255 );      // rainbow effect
        p->setBrush( c );                // solid fill with color c
        p->drawRect( 70, -10, 80, 10 );          // draw the rectangle

        QString n;
        n.sprintf( "H=%d", i*10 );
        p->drawText( 80+70+5, 0, n );          // draw the hue number
    }
}

// This function draws a few lines of text using different fonts.
void drawFonts( QPainter *p )
{
    static const char *fonts[] = { "Helvetica", "Courier", "Times", 0 };
    static int      sizes[] = { 10, 12, 18, 24, 36, 0 };
    int f = 0;
    int y = 0;
    while ( fonts[f] ) {
        int s = 0;
        while ( sizes[s] ) {
            QFont font( fonts[f], sizes[s] );
            p->setFont( font );
            QFontMetrics fm = p->fontMetrics();
            y += fm.ascent();
            p->drawText( 10, y, "Quartz Glyph Job Vex'd Cwm Finks" );
            y += fm.descent();
            s++;
        }
        f++;
    }
}

// This function draws some shapes
void drawShapes( QPainter *p )
{
    QBrush b1( Qt::blue );
    QBrush b2( Qt::green, Qt::Dense6Pattern );          // green 12% fill
    QBrush b3( Qt::NoBrush );                  // void brush
    QBrush b4( Qt::CrossPattern );             // black cross pattern

    p->setPen( Qt::red );
    p->setBrush( b1 );
    p->drawRect( 10, 10, 200, 100 );
    p->setBrush( b2 );
    p->drawRoundRect( 10, 150, 200, 100, 20, 20 );
    p->setBrush( b3 );
    p->drawEllipse( 250, 10, 200, 100 );
    p->setBrush( b4 );
    p->drawPie( 250, 150, 200, 100, 45*16, 90*16 );
}


typedef void (*draw_func)(QPainter*);

struct DrawThing {
    draw_func   f;
    const char   *name;
};

// All previously implemented functions are collected in the following "table".
// If you implement different functionality, your new draw
// function must be assigned here with a function pointer and  description.
// Leave the zeros at the end, they will be used
// as markers referring to the end of the array.

DrawThing ourDrawFunctions[] = {
// name of the function, title presented to the user
    { drawColorWheel,     "Draw color wheel" },
    { drawFonts,    "Draw fonts" },
    { drawShapes, "Draw shapes" },
    { 0,       0 } };

class DrawView : public QWidget
{
    Q_OBJECT
public:
    DrawView();
    ~DrawView();
public slots:
    void   updateIt( int );
    void   printIt();
protected:
    void   drawIt( QPainter * );
    void   paintEvent( QPaintEvent * );
    void   resizeEvent( QResizeEvent * );
private:
    QPrinter      *printer;
    QButtonGroup *bgroup;
    QPushButton    *print;
    int           drawindex;
    int           maxindex;
};

// Construct the DrawView with buttons.
DrawView::DrawView()
{
    setCaption( "Qt Draw Demo Application" );
    setBackgroundMode(PaletteBase);

    // Create a button group to contain all buttons
    bgroup = new QButtonGroup( this );
    bgroup->resize( 200, 200 );
    connect( bgroup, SIGNAL(clicked(int)), SLOT(updateIt(int)) );

    // Calculate the size for the radio buttons
    int maxwidth = 80;
    int maxheight = 10;
    int i;
    const char *n;
    QFontMetrics fm = bgroup->fontMetrics();

    // Find out the longest function description.
    // Here we make use of the last "0,0"-entry in the
    // ourDrawFunctions-array.
    for ( i=0; (n=ourDrawFunctions[i].name) != 0; i++ ) {
        int w = fm.width( n );
        maxwidth = QMAX(w,maxwidth); // QMAX is a macro defined in qglobal.h
                                     // and returns the biggest of to values.
      // Due to its macro nature one should use it with care and with
      // constant parameters only.
    }

    maxwidth = maxwidth + 30;            // allow 30 pixels for radiobuttons

    for ( i=0; (n=ourDrawFunctions[i].name) != 0; i++ ) {
        QRadioButton *rb = new QRadioButton( n, bgroup );
        rb->setGeometry( 10, i*30+10, maxwidth, 30 );

        maxheight += 30;

      if ( i == 0 )
            rb->setChecked( TRUE );
    }

    maxheight += 10;                          // maxheight is now 10 pixels upper margin
                              // plus number_of_drawfunctions * 30 plus 10 pixels lower margin

    drawindex = 0;                // draw first thing
    maxindex  = i;

    maxwidth += 20;                  // add some margin, this results in the final width of bgroup

    bgroup->resize( maxwidth, maxheight );      // resize bgroup to its final size
                              // when no printersupport is provided

// If -- at compile time -- printer support will be disabled,
// we won't set up printing functionality.
#ifndef QT_NO_PRINTER

    printer = new QPrinter;
    // Create and setup the print button
    print = new QPushButton( "Print...", bgroup );
    print->resize( 80, 30 );
    print->move( maxwidth/2 - print->width()/2, maxindex*30+20 );
    connect( print, SIGNAL(clicked()), SLOT(printIt()) );

    // Resize bgroup to its final size when printersupport is given.
    bgroup->resize( maxwidth, print->y()+print->height()+10 );

#endif

    resize( 640,300 );
}

// Clean up.
DrawView::~DrawView()
{
#ifndef QT_NO_PRINTER
    delete printer;
#endif
}

// Called when a radio button is clicked.
void DrawView::updateIt( int index )
{
    if ( index < maxindex ) {
        drawindex = index;
        update();
    }
}

// Calls the drawing function as specified by the radio buttons.
void DrawView::drawIt( QPainter *p )
{
    (*ourDrawFunctions[drawindex].f)(p);
}

// Called when the print button is clicked.
void DrawView::printIt()
{
    if ( printer->setup( this ) ) {
      QPainter paint;
      if( !paint.begin( printer ) )
          return;
        drawIt( &paint );
    }
}

// Called when the widget needs to be updated.
void DrawView::paintEvent( QPaintEvent * )
{
    QPainter paint( this );
    drawIt( &paint );
```

```
}

// Called when the widget has been resized.
// Moves the button group to the upper right corner
// of the widget.

void DrawView::resizeEvent( QResizeEvent * )
{
   bgroup->move( width()-bgroup->width(), 0 );
}

// Create and display our widget.
#include "drawdemo.moc"

int main( int argc, char **argv )
{
   QApplication app( argc, argv );
   DrawView   draw;
   app.setMainWidget( &draw );
   draw.setCaption("Qt Example - Drawdemo");
   draw.show();
   return app.exec();
}
```

실행



Qt Example - Drawdemo

● Draw color wheel
○ Draw fonts
○ Draw shapes

18. 점들의 련결

이 실례는 아주 간단한 마우스에 기초한 사용자의 교제를 보여주며 세계변환행렬이나 다른 고급한 기능이 없이 그리기한다. 프로그람을 실행하고 단추를 선택하고 마우스를 이동하고 단추를 놓고 그려진 직선을 본다.

drawlines.pro

```
TEMPLATE  = app
TARGET      = drawlines
CONFIG      += qt warn_on release
```

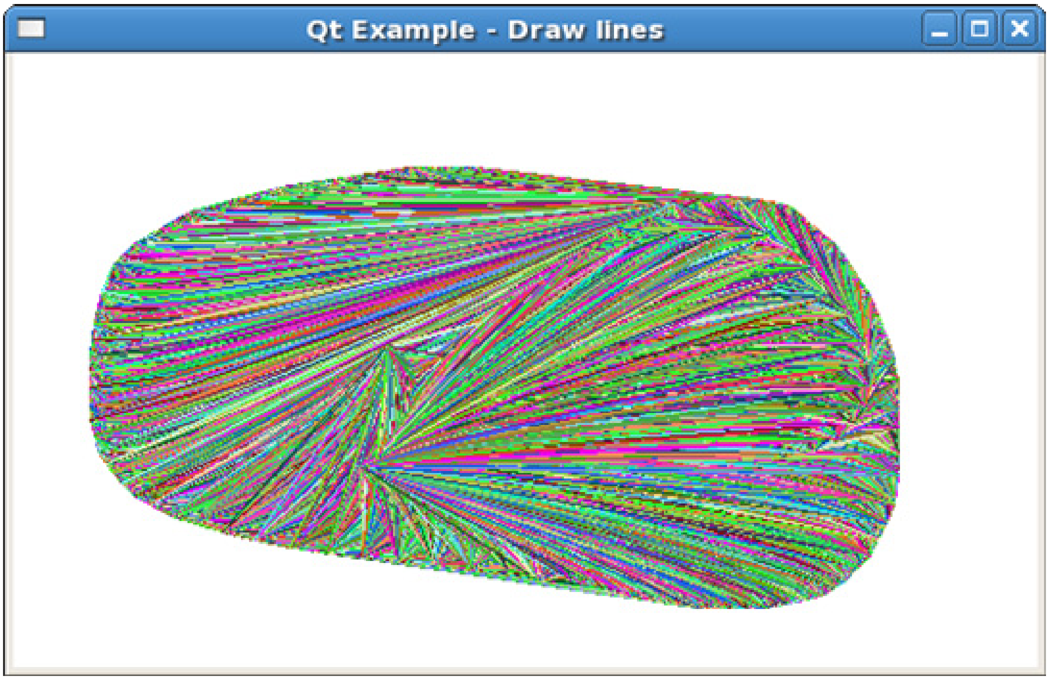
```
HEADERS         =
SOURCES         = connect.cpp
```

connect.cpp

```
#include <qwidget.h>
#include <qpainter.h>
#include <qapplication.h>
#include <stdlib.h>

const int MAXPOINTS = 2000;                 // maximum number of points
const int MAXCOLORS = 40;

// ConnectWidget - draws connected lines
class ConnectWidget : public QWidget
{
public:
    ConnectWidget( QWidget *parent=0, const char *name=0 );
   ~ConnectWidget();
protected:
    voidpaintEvent( QPaintEvent * );
    voidmousePressEvent( QMouseEvent *);
    voidmouseReleaseEvent( QMouseEvent *);
    voidmouseMoveEvent( QMouseEvent *);
private:
    QPoint     *points;                 // point array
    QColor     *colors;                 // color array
    int        count;                  // count = number of points
    booldown;                  // TRUE if mouse down
};

// Constructs a ConnectWidget.
ConnectWidget::ConnectWidget( QWidget *parent, const char *name )
    : QWidget( parent, name, WStaticContents )
{
    setBackgroundColor( white );          // white background
    count = 0;
    down = FALSE;
    points = new QPoint[MAXPOINTS];
    colors = new QColor[MAXCOLORS];
    for ( int i=0; i<MAXCOLORS; i++ )          // init color array
      colors[i] = QColor( rand()&255, rand()&255, rand()&255 );
}

ConnectWidget::~ConnectWidget()
{
    delete[] points;                // cleanup
    delete[] colors;
}

// Handles paint events for the connect widget.
void ConnectWidget::paintEvent( QPaintEvent * )
{
    QPainter paint( this );
    for ( int i=0; i<count-1; i++ ) {         // connect all points
```

```
     for ( int j=i+1; j<count; j++ ) {
        paint.setPen( colors[rand()%MAXCOLORS] ); // set random pen color
        paint.drawLine( points[i], points[j] ); // draw line
     }
   }
}

// Handles mouse press events for the connect widget.
void ConnectWidget::mousePressEvent( QMouseEvent * )
{
   down = TRUE;
   count = 0;                        // start recording points
   erase();                      // erase widget contents
}

// Handles mouse release events for the connect widget.
void ConnectWidget::mouseReleaseEvent( QMouseEvent * )
{
   down = FALSE;                         // done recording points
   update();                         // draw the lines
}

// Handles mouse move events for the connect widget.
void ConnectWidget::mouseMoveEvent( QMouseEvent *e )
{
   if ( down && count < MAXPOINTS ) {
    QPainter paint( this );
    points[count++] = e->pos();      // add point
    paint.drawPoint( e->pos() );      // plot point
   }
}

// Create and display a ConnectWidget.
int main( int argc, char **argv )
{
   QApplication a( argc, argv );
   ConnectWidget connect;
#ifndef QT_NO_WIDGET_TOPEXTRA   // for Qt/Embedded minimal build
   connect.setCaption( "Qt Example - Draw lines");
#endif
   a.setMainWidget( &connect );
   connect.show();
   return a.exec();
}
```

실행



19. 확장대화칸실례

이 실례는 확장대화칸을 창조하는 방법을 보여준다. 우선 표준대화칸을 창조하고 확장으로서 사용될 QWidget폼을 창조한다.

extension.pro

```
TEMPLATE  = app
LANGUAGE= C++
CONFIG      += qt warn_on release
SOURCES        += main.cpp
FORMS       = mainform.ui \
          dialogform.ui \
          extension.ui
DBFILE      = extension.db
```

extension.ui

```
<!DOCTYPE UI><UI version="3.0" stdsetdef="1">
<class>Extension</class>
<widget class="QWidget">
   <property name="name">
…
<pixmapinproject/>
<layoutdefaults spacing="6" margin="11"/>
</UI>
```

mainform.ui

```
<!DOCTYPE UI><UI version="3.0" stdsetdef="1">
<class>MainForm</class>
<widget class="QDialog">
   <property name="name">
```

```
…
<layoutdefaults spacing="6" margin="11"/>
</UI>
```

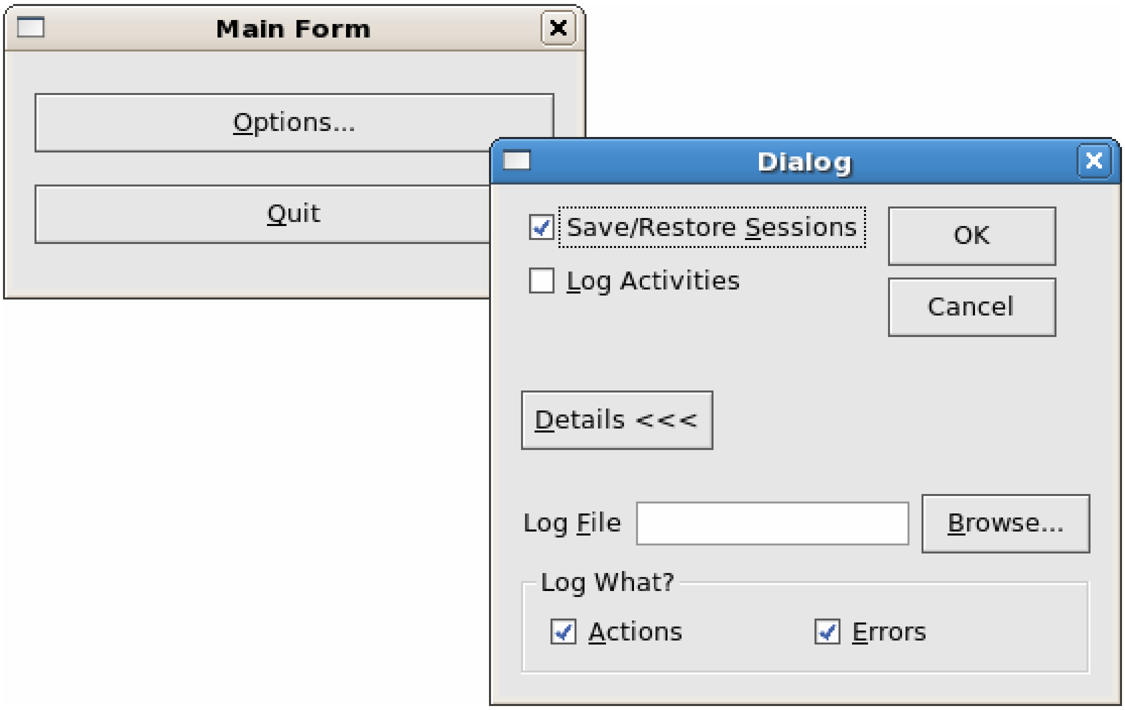
mainform.ui.h

```
#include "dialogform.h"
#include "extension.h"
#include <qapplication.h>
#include <qcheckbox.h>
#include <qlineedit.h>

void MainForm::init()
{
    sessions = FALSE;
    logging = FALSE;
    log_filename = QString::null;
    log_errors = TRUE;
    log_actions = TRUE;
}

void MainForm::optionsDlg()
{
    DialogForm *dlg = new DialogForm( this, "dialog", TRUE );
    Extension *ext = (Extension*)dlg->extension()->qt_cast( "Extension" );
    if ( !ext )
      return;
    dlg->sessionsCheckBox->setChecked( sessions );
    dlg->loggingCheckBox->setChecked( logging );
    ext->logfileLineEdit->setText( log_filename );
    ext->logErrorsCheckBox->setChecked( log_errors );

    if ( dlg->exec() ) {
      sessions = dlg->sessionsCheckBox->isChecked();
      logging = dlg->loggingCheckBox->isChecked();
      log_filename = ext->logfileLineEdit->text();
      log_errors = ext->logErrorsCheckBox->isChecked();
    }
}

void MainForm::quit()
{
    QApplication::exit( 0 );
}
```

Main.cpp

```
#include <qapplication.h>
#include "mainform.h"

int main( int argc, char ** argv )
{
    QApplication a( argc, argv );
    MainForm *w = new MainForm;
    w->show();
    a.connect( &a, SIGNAL( lastWindowClosed() ), w, SLOT( quit() ) );
    return a.exec();
}
```

실행



20. 단순한 파일관리기

이 실례는 QIconView로부터 파생된 창문부품을 리용하여 단순하면서 완전한 기능을 갖추지 못한 파일관리기를 실현하여 현재 등록부를 현시한다. 등록부나무를 현시하는데 dirview실례에서 씌여진 창문부품이 사용된다.

fileiconview.pro

```
TEMPLATE  = app
TARGET      = fileiconview
CONFIG       += qt warn_on release
HEADERS          = mainwindow.h \
          qfileiconview.h \
          ../dirview/dirview.h
SOURCES          = main.cpp \
          mainwindow.cpp \
          qfileiconview.cpp \
          ../dirview/dirview.cpp
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "qfileiconview.h"
#include "../dirview/dirview.h"

#include <qsplitter.h>
#include <qprogressbar.h>
#include <qlabel.h>
```

```
#include <qstatusbar.h>
#include <qtoolbar.h>
#include <qcombobox.h>
#include <qpixmap.h>
#include <qtoolbutton.h>
#include <qdir.h>
#include <qfileinfo.h>

static const char* cdtoparent_xpm[]={
    "15 13 3 1",
    ". c None",
    "* c #000000",
    "a c #ffff99",
    "..*****........",
    ".*aaaaa*.......",
    "***************",
    "*aaaaaaaaaaaaa*",
    "*aaaa*aaaaaaaa*",
    "*aaa***aaaaaaa*",
    "*aa*****aaaaaa*",
    "*aaaa*aaaaaaaa*",
    "*aaaa*aaaaaaaa*",
    "*aaaa******aaa*",
    "*aaaaaaaaaaaaa*",
    "*aaaaaaaaaaaaa*",
    "***************"};

static const char* newfolder_xpm[] = {
    "15 14 4 1",
    "     c None",
    ".    c #000000",
    "+    c #FFFF00",
    "@    c #FFFFFF",
    "          .    ",
    "               ",
    "          .    ",
    "       .    .  ",
    "  ....  . . .  ",
    " .+@+@.  . .   ",
    "..........  . .",
    ".@+@+@+@+@..   ",
    ".+@+@+@+@+. .  ",
    ".@+@+@+@+@.  . ",
    ".+@+@+@+@+.    ",
    ".@+@+@+@+@.    ",
    ".+@+@+@+@+.    ",
    "...........    "};

FileMainWindow::FileMainWindow()
    : QMainWindow()
{
    setup();
}
```

```
void FileMainWindow::show()
{
   QMainWindow::show();
}

void FileMainWindow::setup()
{
   QSplitter *splitter = new QSplitter( this );

   dirlist = new DirectoryView( splitter, "dirlist", TRUE );
   dirlist->addColumn( "Name" );
   dirlist->addColumn( "Type" );
   Directory *root = new Directory( dirlist, "/" );
   root->setOpen( TRUE );
   splitter->setResizeMode( dirlist, QSplitter::KeepSize );

   fileview = new QtFileIconView( "/", splitter );
   fileview->setSelectionMode( QIconView::Extended );

   setCentralWidget( splitter );

   QToolBar *toolbar = new QToolBar( this, "toolbar" );
   setRightJustification( TRUE );

   (void)new QLabel( tr( " Path: " ), toolbar );

   pathCombo = new QComboBox( TRUE, toolbar );
   pathCombo->setAutoCompletion( TRUE );
   toolbar->setStretchableWidget( pathCombo );
   connect( pathCombo, SIGNAL( activated( const QString & ) ),
        this, SLOT ( changePath( const QString & ) ) );

   toolbar->addSeparator();

   QPixmap pix;

   pix = QPixmap( cdtoparent_xpm );
   upButton = new QToolButton( pix, "One directory up", QString::null,
                 this, SLOT( cdUp() ), toolbar, "cd up" );

   pix = QPixmap( newfolder_xpm );
   mkdirButton = new QToolButton( pix, "New Folder", QString::null,
                   this, SLOT( newFolder() ), toolbar, "new folder" );

   connect( dirlist, SIGNAL( folderSelected( const QString & ) ),
        fileview, SLOT ( setDirectory( const QString & ) ) );
   connect( fileview, SIGNAL( directoryChanged( const QString & ) ),
        this, SLOT( directoryChanged( const QString & ) ) );
   connect( fileview, SIGNAL( startReadDir( int ) ),  this, SLOT( slotStartReadDir( int ) ) );
   connect( fileview, SIGNAL( readNextDir() ),  this, SLOT( slotReadNextDir() ) );
   connect( fileview, SIGNAL( readDirDone() ), this, SLOT( slotReadDirDone() ) );

   setDockEnabled( DockLeft, FALSE );
   setDockEnabled( DockRight, FALSE );
   label = new QLabel( statusBar() );
   statusBar()->addWidget( label, 2, TRUE );
   progress = new QProgressBar( statusBar() );
   statusBar()->addWidget( progress, 1, TRUE );

   connect( fileview, SIGNAL( enableUp() ),  this, SLOT( enableUp() ) );
   connect( fileview, SIGNAL( disableUp() ), this, SLOT( disableUp() ) );
   connect( fileview, SIGNAL( enableMkdir() ), this, SLOT( enableMkdir() ) );
   connect( fileview, SIGNAL( disableMkdir() ), this, SLOT( disableMkdir() ) );
}

void FileMainWindow::setPathCombo()
{
   QString dir = caption();
   int i = 0;
   bool found = FALSE;
   for ( i = 0; i < pathCombo->count(); ++i ) {
    if ( pathCombo->text( i ) == dir) {
       found = TRUE;
       break;
    }
   }

   if ( found )
    pathCombo->setCurrentItem( i );
   else {
    pathCombo->insertItem( dir );
    pathCombo->setCurrentItem( pathCombo->count() - 1 );
   }
}

void FileMainWindow::directoryChanged( const QString &dir )
{
   setCaption( dir );
   setPathCombo();
}

void FileMainWindow::slotStartReadDir( int dirs )
{
   label->setText( tr( " Reading Directory..." ) );
   progress->reset();
   progress->setTotalSteps( dirs );
}

void FileMainWindow::slotReadNextDir()
{
   int p = progress->progress();
   progress->setProgress( ++p );
}

void FileMainWindow::slotReadDirDone()
{
   label->setText( tr( " Reading Directory Done." ) );
```

```
    progress->setProgress( progress->totalSteps() );
}

void FileMainWindow::cdUp()
{
    QDir dir = fileview->currentDir();
    dir.cd( ".." );
    fileview->setDirectory( dir );
}

void FileMainWindow::newFolder()
{
    fileview->newDirectory();
}

void FileMainWindow::changePath( const QString &path )
{
    if ( QFileInfo( path ).exists() )
      fileview->setDirectory( path );
    else
      setPathCombo();
}

void FileMainWindow::enableUp()
{
    upButton->setEnabled( TRUE );
}

void FileMainWindow::disableUp()
{
    upButton->setEnabled( FALSE );
}

void FileMainWindow::enableMkdir()
{
    mkdirButton->setEnabled( TRUE );
}

void FileMainWindow::disableMkdir()
{
    mkdirButton->setEnabled( FALSE );
}
```

mainwindow.h

```
#ifndef MAINWIN_H
#define MAINWIN_H

#include <qmainwindow.h>
class QtFileIconView;
class DirectoryView;
class QProgressBar;
class QLabel;
class QComboBox;
class QToolButton;
class FileMainWindow : public QMainWindow
{
   Q_OBJECT

public:
   FileMainWindow();

   QtFileIconView *fileView() { return fileview; }
   DirectoryView *dirList() { return dirlist; }

   void show();

protected:
   void setup();
   void setPathCombo();

   QtFileIconView *fileview;
   DirectoryView *dirlist;
   QProgressBar *progress;
   QLabel *label;
   QComboBox *pathCombo;
   QToolButton *upButton, *mkdirButton;

protected slots:
   void directoryChanged( const QString & );
   void slotStartReadDir( int dirs );
   void slotReadNextDir();
   void slotReadDirDone();
   void cdUp();
   void newFolder();
   void changePath( const QString &path );
   void enableUp();
   void disableUp();
   void enableMkdir();
   void disableMkdir();

};

#endif
```

qfileiconview.cpp

```
#include "qfileiconview.h"
#include <qpainter.h>
#include <qstringlist.h>
#include <qpixmap.h>
#include <qmime.h>
#include <qstrlist.h>
#include <qdragobject.h>
#include <qmessagebox.h>
#include <qevent.h>
#include <qpopupmenu.h>
#include <qcursor.h>
#include <qapplication.h>
```

```
#include <qwmatrix.h>

#include <stdlib.h>

static const char * file_icon[]={
    "32 32 17 1",
    "# c #000000",
    "a c #ffffff",
    "j c #808080",
    "n c #a0a0a4",
    "g c #c0c0c0",
    "m c #004000",
    "o c #000000",
    "l c #004040",
    "k c #404000",
    "i c #c0c000",
    "h c #ffff00",
    "b c #ffffc0",
    "e c #ff8000",
    "f c #c05800",
    "c c #ffa858",
    "d c #ffdca8",
    ". c None",
    "................................",
    "................................",
    "................................",
    "................................",
    ".............#....###...........",
    "...###......#a##.#aba##.........",
    "..#cdb#....#aaaa#aaaaaa##.......",
    "..#ecdb#..#aaaa#aaaaaaaba##.....",
    "..#fecdb##aaaa#aaaaaaaaaaab##...",
    "...#fecdb#aaa#aaaaaaabaabaaaa##.",
    "....#fecdb#a#baaaaa#baaaaaabaaa#",
    ".....#fecdb#aaaaab#a##baaaaaaa#.",
    ".....##fecdb#bbba#aaaa##baaab#..",
    "....#bb#fecdb#ba#aaaaaaa##aa#...",
    "...#bbbb#fecdb##aaabaaaaaa##....",
    "..#bbbb#b#fecdb#aaaaaaabaaaa##..",
    ".#bbbb#bbb#fecdg#aaaaaaaaaaaba#.",
    "#hhbb#bbbbb#fegg#iiaaaaaaaaaaaa#",
    "#jhhhklibbbk#ggj#aaiiaaaaaaaaa#j",
    ".#mjhhhkmikab####aaabiiaaaaaa#j.",
    "...##jhhhmaaibbaaiibaaaiiaab#n..",
    ".....##j#baaaiiabaaiibaabaa#n...",
    "......##baibaabiibaaaiiabb#j....",
    "......#bbbbiiaabbiiaaaaabon.....",
    ".....#bbbbbbbiiabbaiiaab#n......",
    ".....#jbbbbbbbbiibaabba#n.......",
    "......##jbbbbbbbbiiaabmj........",
    "........##jbbbbbbbbbb#j.........",
    "..........##nbbbbbbbmj..........",
    "............##jbbbb#j...........",
    ".............#mjj#n.............",
    "................##n............."};

static const char * folder_icon[]={
    "32 32 11 1",
    "# c #000000",
    "b c #c0c000",
    "d c #585858",
    "a c #ffff00",
    "i c #400000",
    "h c #a0a0a4",
    "e c #000000",
    "c c #ffffff",
    "f c #303030",
    "g c #c0c0c0",
    ". c None",
    "...###..........................",
    "...#aa##........................",
    ".###baaa##......................",
    ".#cde#baaa##....................",
    ".#cccdeebaaa##..##f.............",
    ".#cccccdeebaaa##aaa##...........",
    ".#cccccccdeebaaaaaaaa##.........",
    ".#cccccccccdeebababaaa#.........",
    ".#cccccgcgghhebbbbbbbaa#........",
    ".#ccccccgcgggdebbbbbbba#........",
    ".#cccgcgcgcgghdeebiebbba#.......",
    ".#ccccgcggggggghdeddeeba#.......",
    ".#cgcgcgcggggggggghghdebb#......",
    ".#ccgcggggggggghghghghd#b#......",
    ".#cgcgcggggggggghghghhd#b#......",
    ".#gcggggggggghghghhhhhd#b#......",
    ".#cgcggggggggghghghhhhd#b#......",
    ".#gggggggggghghghhhhhhhdib#......",
    ".#gggggggggghghghhhhhhd#b#......",
    ".#hhggggghghghhhhhhhhhd#b#......",
    ".#ddhhgggghghghhhhhhhhhd#b#......",
    "..##ddhhghghhhhhhhhhhhdeb#......",
    "....##ddhhhghhhhhhhhhhd#b#......",
    "......##ddhhhhhhhhhhhhd#b#......",
    "........##ddhhhhhhhhhd#b#......",
    "..........##ddhhhhhhhhd#b#......",
    "............##ddhhhhhd#b###....",
    "..............##ddhhhhd#b#####..",
    "................##ddhhd#b######.",
    "..................##dddeb#####..",
    "....................##d#b###....",
    "......................####......"};


static const char * link_icon[]={
    "32 32 12 1",
    "# c #000000",
    "h c #a0a0a4",
    "b c #c00000",
    "d c #585858",
    "i c #400000",
    "c c #ffffff",
    "e c #000000",
    "g c #c0c0c0",
    "a c #ff0000",
    "f c #303030",
    "n c white",
    ". c None",
    "...###..........................",
    "...#aa##........................",
    ".###baaa##......................",
    ".#cde#baaa##....................",
    ".#cccdeebaaa##..##f.............",
    ".#cccccdeebaaa##aaa##...........",
    ".#cccccccdeebaaaaaaaa##.........",
    ".#cccccccccdeebababaaa#.........",
    ".#cccccgcgghhebbbbbbbaa#........",
    ".#ccccccgcgggdebbbbbbba#........",
    ".#cccgcgcgcgghdeebiebbba#.......",
    ".#ccccgcggggggghdeddeeba#.......",
    ".#cgcgcgcggggggggghghdebb#......",
    ".#ccgcggggggggghghghghd#b#......",
    ".#cgcgcgggggggggghghghhd#b#......",
    ".#gcggggggggghghghhhhhd#b#......",
    ".#cgcggggggggghghghhhhd#b#......",
    ".#gggggggghghghhhhhhhdib#......",
    ".#ggggggggghghghhhhhhd#b#......",
    ".#hhgggghghghhhhhhhhhd#b#......",
    ".#ddhhggghghghhhhhhhhd#b#......",
    "..##ddhhghghhhhhhhhhhhdeb#......",
    "###########hhhhhhhhhhd#b#......",
    "#nnnnnnnnnn#hhhhhhhhhhd#b#......",
    "#nnnnnnnnnn#hhhhhhhhhhd#b#......",
    "#nn#nn#nnnn#ddhhhhhhhhd#b#......",
    "#nn##n##nnn###ddhhhhhhd#b###....",
    "#nnn#####nn#..##ddhhhhd#b#####..",
    "#nnnnn##nnn#....##ddhhd#b######.",
    "#nnnnn#nnnn#......##dddeb#####..",
    "#nnnnnnnnnn#........##d#b###....",
    "###########..........####......"};

static const char * folder_locked_icon[]={
    "32 32 12 1",
    "# c #000000",
    "g c #808080",
    "h c #c0c0c0",
    "f c #c05800",
    "c c #ffffff",
    "d c #585858",
    "b c #ffa858",
    "a c #ffdca8",
    "e c #000000",
    "i c #a0a0a4",
    "j c #c0c0c0",
    ". c None",
    "...###..........................",
    "...#aa##........................",
    ".###baaa##......................",
    ".#cde#baaa##....................",
    ".#cccdeeba########..............",
    ".#cccccde##fffff##..............",
    ".#cccccc##fffgggg#..............",
    ".#ccccccc#ffg####a##............",
    ".#ccccchc#ffg#eebbaa##..........",
    ".#ccccccc#ffg#ddeebbba##........",
    ".#ccchccc#ffg#ihddeebbba##......",
    ".#cccccaa#ffg#ihhhddeeba##......",
    ".#chchhbbaafg#ihhhihidebb#......",
    ".#cchccbbbbaa#ihhihihid#b#......",
    ".#chchhbb#bbbaaiihihiid#b#......",
    ".#hchhcbb#fbbbafhiiiiid#b#......",
    ".#chchhbb#ffgbbfihiiiid#b#......",
    ".#hhhhhbb#ffg#bfiiiiiid#b#......",
    ".#hhhhhbbaffg#bfiiiiiid#b#......",
    ".#iihhhjbbaab#bfiiiiiid#b#......",
    ".#ddiihhh#bbbabfiiiiiid#b#......",
    "..##ddiih#ffbbbfiiiiiid#b#......",
    "....##ddi#ffg#biiiiiiid#b#......",
    "......##d#ffg#iiiiiiiid#b#......",
    "........##ffg#iiiiiiiid#b#......",
    ".........#ffg#iiiiiiiid#b#......",
    ".........#ffg#ddiiiiiid#b###....",
    ".........##fg###ddiiiid#b#####..",
    "...........####.##ddiid#b######.",
    "..................##dddeb#####..",
    "....................##d#b###....",
    "......................####......"};

static QPixmap *iconFolderLockedLarge = 0;
static QPixmap *iconFolderLarge = 0;
static QPixmap *iconFileLarge = 0;
static QPixmap *iconLinkLarge = 0;
static QPixmap *iconFolderLockedSmall = 0;
static QPixmap *iconFolderSmall = 0;
static QPixmap *iconFileSmall = 0;
static QPixmap *iconLinkSmall = 0;

static void cleanup()
{
    delete iconFolderLockedLarge;
    iconFolderLockedLarge = 0;
    delete iconFolderLarge;
    iconFolderLarge = 0;
    delete iconFileLarge;
    iconFileLarge = 0;
    delete iconLinkLarge;
    iconLinkLarge = 0;
```

```
  delete iconFolderLockedSmall;
  iconFolderLockedSmall = 0;
  delete iconFolderSmall;
  iconFolderSmall = 0;
  delete iconFileSmall;
  iconFileSmall = 0;
  delete iconLinkSmall;
  iconLinkSmall = 0;
}

// Class QtFileIconDrag
QtFileIconDrag::QtFileIconDrag( QWidget * dragSource, const char* name )
  : QIconDrag( dragSource, name )
{
}

const char* QtFileIconDrag::format( int i ) const
{
  if ( i == 0 )
   return "application/x-qiconlist";
  else if ( i == 1 )
   return "text/uri-list";
  else
   return 0;
}

QByteArray QtFileIconDrag::encodedData( const char* mime ) const
{
  QByteArray a;
  if ( QString( mime ) == "application/x-qiconlist" ) {
   a = QIconDrag::encodedData( mime );
  } else if ( QString( mime ) == "text/uri-list" ) {
   QString s = urls.join( "\r\n" );
   a.resize( s.length() );
   memcpy( a.data(), s.latin1(), s.length() );
  }
  return a;
}

bool QtFileIconDrag::canDecode( QMimeSource* e )
{
  return e->provides( "application/x-qiconlist" ) ||
   e->provides( "text/uri-list" );
}

void QtFileIconDrag::append( const QIconDragItem &item, const QRect &pr,
           const QRect &tr, const QString &url )
{
  QIconDrag::append( item, pr, tr );
  QString ourUrl = url;
#ifdef Q_WS_WIN
  if (ourUrl.length() > 2 && ourUrl[1] != ':') {
   QDir dir(ourUrl);
   ourUrl = dir.absPath();
```

```
    }
#endif
    urls << QUriDrag::localFileToUri(ourUrl);
}

// Class QtFileIconViewItem
QtFileIconViewItem::QtFileIconViewItem( QtFileIconView *parent, QFileInfo *fi )
    : QIconViewItem( parent, fi->fileName() ), itemFileName( fi->filePath() ),
      itemFileInfo( fi ), checkSetText( FALSE )
{
    vm = QtFileIconView::Large;

    if ( itemFileInfo->isDir() )
     itemType = Dir;
    else if ( itemFileInfo->isFile() )
     itemType = File;
    if ( itemFileInfo->isSymLink() )
     itemType = Link;

    viewModeChanged( ( (QtFileIconView*)iconView() )->viewMode() );

    if ( itemFileInfo->fileName() == "." ||
      itemFileInfo->fileName() == ".." )
     setRenameEnabled( FALSE );

    checkSetText = TRUE;

    QObject::connect( &timer, SIGNAL( timeout() ),
               iconView(), SLOT( openFolder() ) );
}

void QtFileIconViewItem::paintItem( QPainter *p, const QColorGroup &cg )
{
    if ( itemFileInfo->isSymLink() ) {
     QFont f( p->font() );
     f.setItalic( TRUE );
     p->setFont( f );
    }

    QIconViewItem::paintItem( p, cg );
}

void QtFileIconViewItem::viewModeChanged( QtFileIconView::ViewMode m )
{
    vm = m;
    setDropEnabled( itemType == Dir && QDir( itemFileName ).isReadable() );
    calcRect();
}

QPixmap *QtFileIconViewItem::pixmap() const
{
    switch ( itemType ) {
    case Dir:
     {
```

```
     if ( !QDir( itemFileName ).isReadable() ) {
      if ( vm == QtFileIconView::Small )
        return iconFolderLockedSmall;
      else
        return iconFolderLockedLarge;
     } else {
      if ( vm == QtFileIconView::Small )
        return iconFolderSmall;
      else
        return iconFolderLarge;
     }
   }
  case Link:
   {
     if ( vm == QtFileIconView::Small )
      return iconLinkSmall;
     else
      return iconLinkLarge;
   }
  default:
   {
     if ( vm == QtFileIconView::Small )
      return iconFileSmall;
     else
      return iconFileLarge;
   }
   }
}

QtFileIconViewItem::~QtFileIconViewItem()
{
  delete itemFileInfo;
}

void QtFileIconViewItem::setText( const QString &text )
{
  if ( checkSetText ) {
   if ( text == "." || text == "." || text.isEmpty() )
     return;
   QDir dir( itemFileInfo->dir() );
   if ( dir.rename( itemFileInfo->fileName(), text ) ) {
     itemFileName = itemFileInfo->dirPath( TRUE ) + "/" + text;
     delete itemFileInfo;
     itemFileInfo = new QFileInfo( itemFileName );
     QIconViewItem::setText( text );
   }
  } else {
   QIconViewItem::setText( text );
  }
}

bool QtFileIconViewItem::acceptDrop( const QMimeSource *e ) const
{
  if ( type() == Dir && e->provides( "text/uri-list" ) &&
    dropEnabled() )
   return TRUE;

  return FALSE;
}

void QtFileIconViewItem::dropped( QDropEvent *e, const QValueList<QIconDragItem> & )
{
  timer.stop();

  if ( !QUriDrag::canDecode( e ) ) {
   e->ignore();
   return;
  }

  QStringList lst;
  QUriDrag::decodeLocalFiles( e, lst );

  QString str;
  if ( e->action() == QDropEvent::Copy )
   str = "Copy\n\n";
  else
   str = "Move\n\n";
  for ( uint i = 0; i < lst.count(); ++i )
   str += QString( "   %1\n" ).arg( lst[i] );
  str += QString( "\n"
   "To\n\n"
   "    %1" ).arg( filename() );

  QMessageBox::information( iconView(), e->action() == QDropEvent::Copy ? "Copy" : "Move" , str,
"Not Implemented" );
  if ( e->action() == QDropEvent::Move )
   QMessageBox::information( iconView(), "Remove" , str, "Not Implemented" );
  e->acceptAction();
}

void QtFileIconViewItem::dragEntered()
{
  if ( type() != Dir ||
    type() == Dir && !QDir( itemFileName ).isReadable() )
   return;

  ( (QtFileIconView*)iconView() )->setOpenItem( this );
  timer.start( 1500 );
}

void QtFileIconViewItem::dragLeft()
{
  if ( type() != Dir ||
    type() == Dir && !QDir( itemFileName ).isReadable() )
   return;

  timer.stop();
}
```

```
// Class QtFileIconView
QtFileIconView::QtFileIconView( const QString &dir, QWidget *parent, const char *name )
  : QIconView( parent, name ), viewDir( dir ), newFolderNum( 0 )
{
  if ( !iconFolderLockedLarge ) {
   qAddPostRoutine( cleanup );
   QWMatrix m;
   m.scale( 0.6, 0.6 );
   QPixmap iconpix( folder_locked_icon );
   iconFolderLockedLarge = new QPixmap( folder_locked_icon );
   iconpix = iconpix.xForm( m );
   iconFolderLockedSmall = new QPixmap( iconpix );
   iconpix = QPixmap( folder_icon );
   iconFolderLarge = new QPixmap( folder_icon );
   iconpix = iconpix.xForm( m );
   iconFolderSmall = new QPixmap( iconpix );
   iconpix = QPixmap( file_icon );
   iconFileLarge = new QPixmap( file_icon );
   iconpix = iconpix.xForm( m );
   iconFileSmall = new QPixmap( iconpix );
   iconpix = QPixmap( link_icon );
   iconLinkLarge = new QPixmap( link_icon );
   iconpix = iconpix.xForm( m );
   iconLinkSmall = new QPixmap( iconpix );
  }

  vm = Large;

  setGridX( 75 );
  setResizeMode( Adjust );
  setWordWrapIconText( FALSE );

  connect( this, SIGNAL( doubleClicked( QIconViewItem * ) ),
      this, SLOT( itemDoubleClicked( QIconViewItem * ) ) );
  connect( this, SIGNAL( returnPressed( QIconViewItem * ) ),
      this, SLOT( itemDoubleClicked( QIconViewItem * ) ) );
  connect( this, SIGNAL( dropped( QDropEvent *, const QValueList<QIconDragItem> & ) ),
      this, SLOT( slotDropped( QDropEvent *, const QValueList<QIconDragItem> & ) ) );
  connect( this, SIGNAL( contextMenuRequested( QIconViewItem *, const QPoint & ) ),
      this, SLOT( slotRightPressed( QIconViewItem * ) ) );

  setHScrollBarMode( AlwaysOff );
  setVScrollBarMode( Auto );

  setAutoArrange( TRUE );
  setSorting( TRUE );
  openItem = 0;
}

void QtFileIconView::openFolder()
{
  if ( !openItem )
   return;
```

```
    if ( openItem->type() != QtFileIconViewItem::Dir ||
       openItem->type() == QtFileIconViewItem::Dir &&
       !QDir( openItem->itemFileName ).isReadable() )
      return;

    openItem->timer.stop();
    setDirectory( openItem->itemFileName );
}

void QtFileIconView::setDirectory( const QString &dir )
{
    viewDir = QDir( dir );
    readDir( viewDir );
}

void QtFileIconView::setDirectory( const QDir &dir )
{
    viewDir = dir;
    readDir( viewDir );
}

void QtFileIconView::newDirectory()
{
    setAutoArrange( FALSE );
    selectAll( FALSE );
    if ( viewDir.mkdir( QString( "New Folder %1" ).arg( ++newFolderNum ) ) ) {
      QFileInfo *fi = new QFileInfo( viewDir, QString( "New Folder %1" ).arg( newFolderNum ) );
      QtFileIconViewItem *item = new QtFileIconViewItem( this, new QFileInfo( *fi ) );
      item->setKey( QString( "000000%1" ).arg( fi->fileName() ) );
      delete fi;
      repaintContents( contentsX(), contentsY(), contentsWidth(), contentsHeight(), FALSE );
      ensureItemVisible( item );
      item->setSelected( TRUE, TRUE );
      setCurrentItem( item );
      repaintItem( item );
      qApp->processEvents();
      item->rename();
    }
    setAutoArrange( TRUE );
}

QDir QtFileIconView::currentDir()
{
    return viewDir;
}

static bool isRoot( const QString &s )
{
#if defined(Q_OS_UNIX)
    if ( s == "/" )
      return TRUE;
#elif defined(Q_OS_WIN32)
    QString p = s;
    if ( p.length() == 3 &&
        p.right( 2 ) == ":/" )
      return TRUE;
    if ( p[ 0 ] == '/' && p[ 1 ] == '/' ) {
      int slashes = p.contains( '/' );
      if ( slashes <= 3 )
          return TRUE;
      if ( slashes == 4 && p[ (int)p.length() - 1 ] == '/' )
          return TRUE;
    }
#endif

    return FALSE;
}

void QtFileIconView::readDir( const QDir &dir )
{
    if ( !dir.isReadable() )
      return;

    if ( isRoot( dir.absPath() ) )
      emit disableUp();
    else
      emit enableUp();

    clear();

    emit directoryChanged( dir.absPath() );

    const QFileInfoList *filist = dir.entryInfoList( QDir::DefaultFilter, QDir::DirsFirst | QDir::Name );

    emit startReadDir( filist->count() );

    QFileInfoListIterator it( *filist );
    QFileInfo *fi;
    bool allowRename = FALSE, allowRenameSet = FALSE;
    while ( ( fi = it.current() ) != 0 ) {
      ++it;
      if ( fi && fi->fileName() == ".." && ( fi->dirPath() == "/" || fi->dirPath().isEmpty() ) )
          continue;
      emit readNextDir();
      QtFileIconViewItem *item = new QtFileIconViewItem( this, new QFileInfo( *fi ) );
      if ( fi->isDir() )
          item->setKey( QString( "000000%1" ).arg( fi->fileName() ) );
      else
          item->setKey( fi->fileName() );
      if ( !allowRenameSet ) {
          if ( !QFileInfo( fi->absFilePath() ).isWritable() ||
             item->text() == "." || item->text() == ".." )
            allowRename = FALSE;
          else
            allowRename = TRUE;
          if ( item->text() == "." || item->text() == ".." )
            allowRenameSet = FALSE;
          else
      allowRenameSet = TRUE;
   }
   item->setRenameEnabled( allowRename );
  }

  if ( !QFileInfo( dir.absPath() ).isWritable() )
   emit disableMkdir();
  else
   emit enableMkdir();

  emit readDirDone();
}

void QtFileIconView::itemDoubleClicked( QIconViewItem *i )
{
  QtFileIconViewItem *item = ( QtFileIconViewItem* )i;

  if ( item->type() == QtFileIconViewItem::Dir ) {
   viewDir = QDir( item->filename() );
   readDir( viewDir );
  } else if ( item->type() == QtFileIconViewItem::Link &&
      QFileInfo( QFileInfo( item->filename() ).readLink() ).isDir() ) {
   viewDir = QDir( QFileInfo( item->filename() ).readLink() );
   readDir( viewDir );
  }
}

QDragObject *QtFileIconView::dragObject()
{
  if ( !currentItem() )
   return 0;

  QPoint orig = viewportToContents( viewport()->mapFromGlobal( QCursor::pos() ) );
  QtFileIconDrag *drag = new QtFileIconDrag( viewport() );
  drag->setPixmap( *currentItem()->pixmap(),
        QPoint( currentItem()->pixmapRect().width() / 2, currentItem()->pixmapRect().height() / 2 ) );
  for ( QtFileIconViewItem *item = (QtFileIconViewItem*)firstItem(); item;
    item = (QtFileIconViewItem*)item->nextItem() ) {
   if ( item->isSelected() ) {
     QIconDragItem id;
     id.setData( QCString( item->filename() ) );
     drag->append( id,
          QRect( item->pixmapRect( FALSE ).x() - orig.x(),
            item->pixmapRect( FALSE ).y() - orig.y(),
            item->pixmapRect().width(), item->pixmapRect().height() ),
          QRect( item->textRect( FALSE ).x() - orig.x(),
            item->textRect( FALSE ).y() - orig.y(),
            item->textRect().width(), item->textRect().height() ),
          QString( item->filename() ) );
   }
  }

  return drag;
}
```

```
void QtFileIconView::keyPressEvent( QKeyEvent *e )
{
  if ( e->key() == Key_N &&
     ( e->state() & ControlButton ) )
   newDirectory();
  else
   QIconView::keyPressEvent( e );
}

void QtFileIconView::slotDropped( QDropEvent *e, const QValueList<QIconDragItem> & )
{
  if ( openItem )
   openItem->timer.stop();
  if ( !QUriDrag::canDecode( e ) ) {
   e->ignore();
   return;
  }

  QStringList lst;
  QUriDrag::decodeLocalFiles( e, lst );

  QString str;
  if ( e->action() == QDropEvent::Copy )
   str = "Copy\n\n";
  else
   str = "Move\n\n";
  for ( uint i = 0; i < lst.count(); ++i )
   str += QString( "   %1\n" ).arg( QDir::convertSeparators(lst[i]) );
  str += QString( "\n"
   "To\n\n"
   "   %1" ).arg( viewDir.absPath() );

  QMessageBox::information( this, e->action() == QDropEvent::Copy ? "Copy" : "Move" , str, "Not
Implemented" );
  if ( e->action() == QDropEvent::Move )
   QMessageBox::information( this, "Remove" , QDir::convertSeparators(lst.join("\n")), "Not
Implemented" );
  e->acceptAction();
  openItem = 0;
}

void QtFileIconView::viewLarge()
{
  setViewMode( Large );
}

void QtFileIconView::viewSmall()
{
  setViewMode( Small );
}

void QtFileIconView::viewBottom()
{
    setItemTextPos( Bottom );
}

void QtFileIconView::viewRight()
{
    setItemTextPos( Right );
}

void QtFileIconView::flowEast()
{
    setHScrollBarMode( AlwaysOff );
    setVScrollBarMode( Auto );
    setArrangement( LeftToRight );
}

void QtFileIconView::flowSouth()
{
    setVScrollBarMode( AlwaysOff );
    setHScrollBarMode( Auto );
    setArrangement( TopToBottom );
}

void QtFileIconView::sortAscending()
{
    sort( TRUE );
}

void QtFileIconView::sortDescending()
{
    sort( FALSE );
}

void QtFileIconView::itemTextTruncate()
{
    setWordWrapIconText( FALSE );
}

void QtFileIconView::itemTextWordWrap()
{
    setWordWrapIconText( TRUE );
}

void QtFileIconView::slotRightPressed( QIconViewItem *item )
{
    if ( !item ) { // right pressed on viewport
     QPopupMenu menu( this );

     menu.insertItem( "&Large view", this, SLOT( viewLarge() ) );
     menu.insertItem( "&Small view", this, SLOT( viewSmall() ) );
     menu.insertSeparator();
     menu.insertItem( "Text at the &bottom", this, SLOT( viewBottom() ) );
     menu.insertItem( "Text at the &right", this, SLOT( viewRight() ) );
     menu.insertSeparator();
     menu.insertItem( "Arrange l&eft to right", this, SLOT( flowEast() ) );
```

```
        menu.insertItem( "Arrange t&op to bottom", this, SLOT( flowSouth() ) );
        menu.insertSeparator();
        menu.insertItem( "&Truncate item text", this, SLOT( itemTextTruncate() ) );
        menu.insertItem( "&Wordwrap item text", this, SLOT( itemTextWordWrap() ) );
        menu.insertSeparator();
        menu.insertItem( "Arrange items in &grid", this, SLOT( arrangeItemsInGrid() ) );
        menu.insertSeparator();
        menu.insertItem( "Sort &ascending", this, SLOT( sortAscending() ) );
        menu.insertItem( "Sort &descending", this, SLOT( sortDescending() ) );

        menu.setMouseTracking( TRUE );
        menu.exec( QCursor::pos() );
    } else { // on item
        QPopupMenu menu( this );

        int RENAME_ITEM = menu.insertItem( "Rename Item" );
        int REMOVE_ITEM = menu.insertItem( "Remove Item" );

        menu.setMouseTracking( TRUE );
        int id = menu.exec( QCursor::pos() );

        if ( id == -1 )
            return;

        if ( id == RENAME_ITEM && item->renameEnabled() ) {
            item->rename();
        } else if ( id == REMOVE_ITEM ) {
            delete item;
            QMessageBox::information( this, "Not implemented!", "Deleting files not implemented yet,\n"
                                      "The item has only been removed from the view! " );
        }
    }
}

void QtFileIconView::setViewMode( ViewMode m )
{
    if ( m == vm )
        return;

    vm = m;
    QtFileIconViewItem *item = (QtFileIconViewItem*)firstItem();
    for ( ; item; item = (QtFileIconViewItem*)item->nextItem() )
        item->viewModeChanged( vm );

    arrangeItemsInGrid();
}
```

qfileiconview.h

```
#ifndef QTFILEICONVIEW_H
#define QTFILEICONVIEW_H

#include <qiconset.h>
#include <qstring.h>
#include <qfileinfo.h>
```

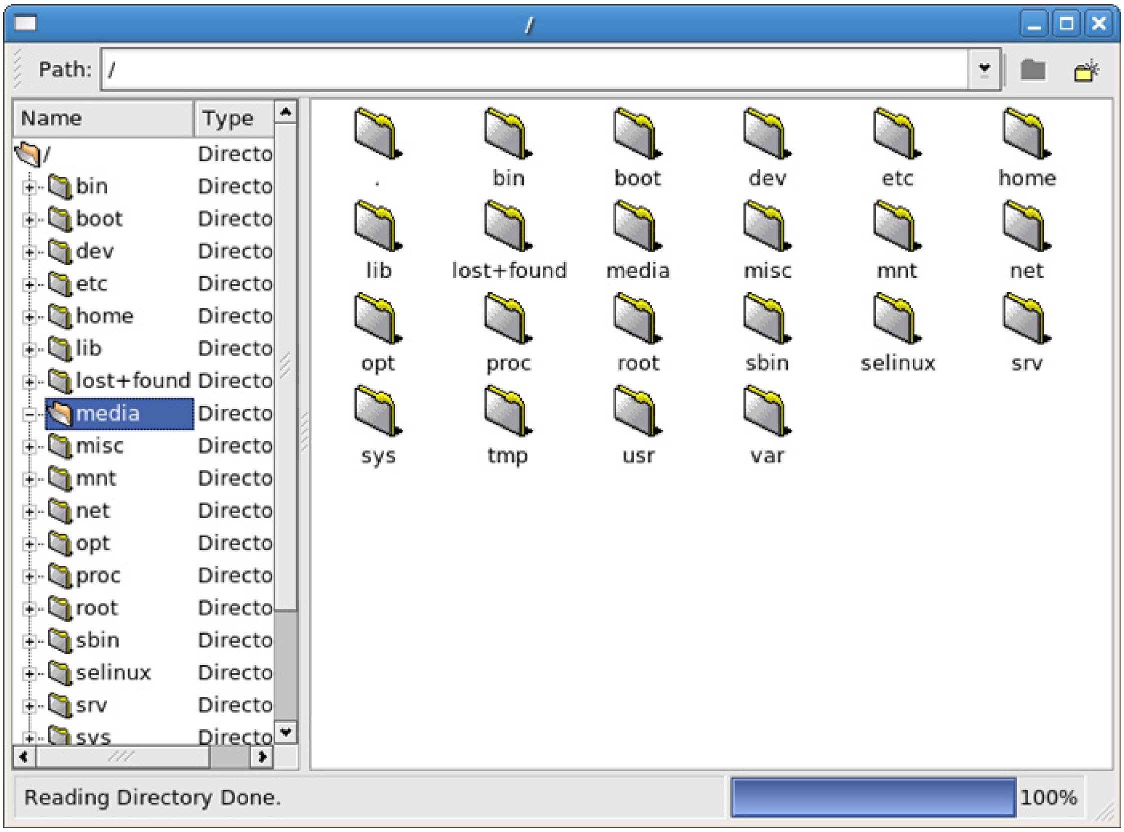
```
#include <qdir.h>
#include <qtimer.h>
#include <qiconview.h>

class QtFileIconView;
class QDragObject;
class QResizeEvent;

// Class QtFileIconDrag
class QtFileIconDrag : public QIconDrag
{
   Q_OBJECT

public:
   QtFileIconDrag( QWidget * dragSource, const char* name = 0 );

   const char* format( int i ) const;
   QByteArray encodedData( const char* mime ) const;
   static bool canDecode( QMimeSource* e );
   void append( const QIconDragItem &item, const QRect &pr, const QRect &tr, const QString &url );

private:
   QStringList urls;

};

// Class QtFileIconView
class QtFileIconViewItem;
class QtFileIconView : public QIconView
{
   Q_OBJECT

public:
   QtFileIconView( const QString &dir, QWidget *parent = 0, const char *name = 0 );

   enum ViewMode { Large, Small };

   void setViewMode( ViewMode m );
   ViewMode viewMode() const { return vm; }
   void setOpenItem( QtFileIconViewItem *i ) {
    openItem = i;
   }

public slots:
   void setDirectory( const QString &dir );
   void setDirectory( const QDir &dir );
   void newDirectory();
   QDir currentDir();

signals:
   void directoryChanged( const QString & );
   void startReadDir( int dirs );
   void readNextDir();
   void readDirDone();
```

```
    void enableUp();
    void disableUp();
    void enableMkdir();
    void disableMkdir();

protected slots:
    void itemDoubleClicked( QIconViewItem *i );
    void slotDropped( QDropEvent *e, const QValueList<QIconDragItem> & );

    void viewLarge();
    void viewSmall();
    void viewBottom();
    void viewRight();
    void flowEast();
    void flowSouth();
    void itemTextTruncate();
    void itemTextWordWrap();
    void sortAscending();
    void sortDescending();
    void arrangeItemsInGrid() {
      QIconView::arrangeItemsInGrid( TRUE );
    }

    void slotRightPressed( QIconViewItem *item );
    void openFolder();

protected:
    void readDir( const QDir &dir );
    virtual QDragObject *dragObject();

    virtual void keyPressEvent( QKeyEvent *e );

    QDir viewDir;
    int newFolderNum;
    QSize sz;
    QPixmap pix;
    ViewMode vm;
    QtFileIconViewItem *openItem;

};

// Class QtFileIconViewItem
class QtFileIconViewItem : public QIconViewItem
{
    friend class QtFileIconView;

public:
    enum ItemType {
        File = 0,
        Dir,
        Link
    };

    QtFileIconViewItem( QtFileIconView *parent, QFileInfo *fi );
    virtual ~QtFileIconViewItem();

    ItemType type() const
    { return itemType; }
    QString filename() const { return itemFileName; }

    virtual bool acceptDrop( const QMimeSource *e ) const;

    virtual void setText( const QString &text );
    virtual QPixmap *pixmap() const;

    virtual void dragEntered();
    virtual void dragLeft();

    void viewModeChanged( QtFileIconView::ViewMode m );
    void paintItem( QPainter *p, const QColorGroup &cg );

protected:
    virtual void dropped( QDropEvent *e, const QValueList<QIconDragItem> & );

    QString itemFileName;
    QFileInfo *itemFileInfo;
    ItemType itemType;
    bool checkSetText;
    QTimer timer;
    QtFileIconView::ViewMode vm;

};

#endif
```

main.cpp

```
#include "mainwindow.h"
#include "qfileiconview.h"
#include <qapplication.h>

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    FileMainWindow mw;
    mw.resize( 680, 480 );
    a.setMainWidget( &mw );
    mw.fileView()->setDirectory( "/" );
    mw.show();
    return a.exec();
}
```

실행



/
Path: /
. bin boot dev etc home
lib lost+found media misc mnt net
opt proc root sbin selinux srv
sys tmp usr var
100%

Name	Type
/	Directo
bin	Directo
boot	Directo
dev	Directo
etc	Directo
home	Directo
lib	Directo
lost+found	Directo
media	Directo
misc	Directo
mnt	Directo
net	Directo
opt	Directo
proc	Directo
root	Directo
sbin	Directo
selinux	Directo
srv	Directo
sys	Directo

Reading Directory Done.

21. 직 4 각형그리기

이 실례는 창문에 직4각형들을 련달아 그리며 초당 그려지는 직4각형의 수를 계산하는 또 하나의 창문부품을 가진다.

forever.pro

```
TEMPLATE  = app
TARGET    = forever
CONFIG    += qt warn_on release
HEADERS       = forever.h
SOURCES       = forever.cpp
```

forever.cpp

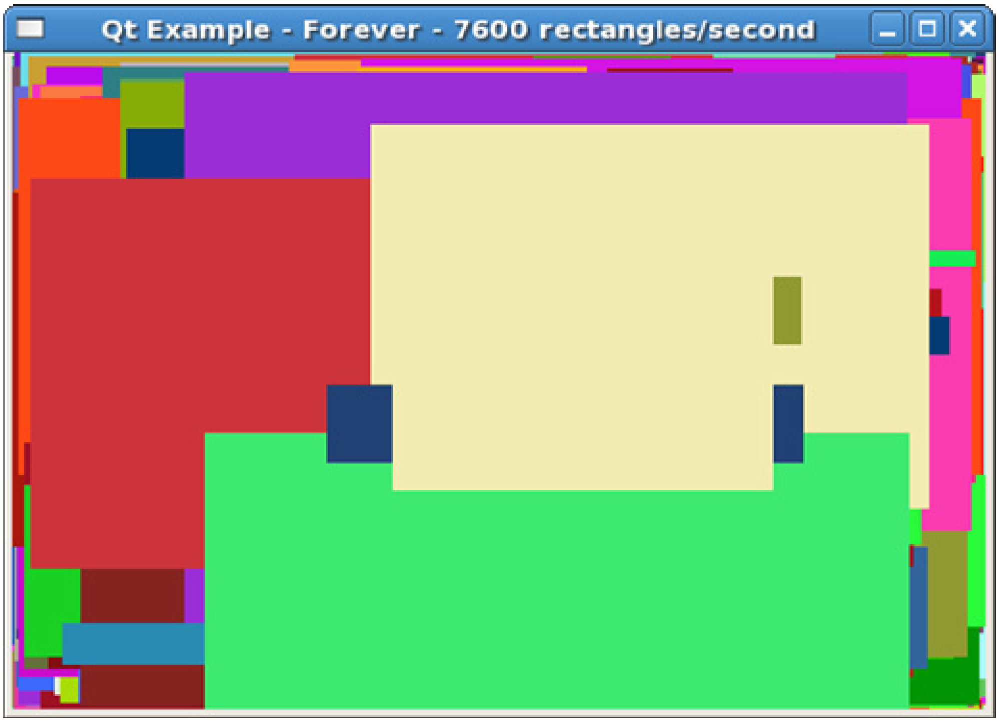
```
#include <qtimer.h>
#include <qpainter.h>
#include <qapplication.h>
#include <stdlib.h>                    // defines rand() function

#include "forever.h"

// Forever - a widget that draws rectangles forever.

// Constructs a Forever widget.
Forever::Forever( QWidget *parent, const char *name )  : QWidget( parent, name )
{
   for (int a=0; a<numColors; a++) {
    colors[a] = QColor( rand()&255,  rand()&255,  rand()&255 );
   }
   rectangles = 0;
   startTimer( 0 );                    // run continuous timer
   QTimer * counter = new QTimer( this );
   connect( counter, SIGNAL(timeout()), this, SLOT(updateCaption()) );
   counter->start( 1000 );
}

void Forever::updateCaption()
{
   QString s;
   s.sprintf( "Qt Example - Forever - %d rectangles/second", rectangles );
   rectangles = 0;
   setCaption( s );
}

// Handles paint events for the Forever widget.
void Forever::paintEvent( QPaintEvent * )
{
   QPainter paint( this );             // painter object
   int w = width();
   int h = height();
   if(w <= 0 || h <= 0)
    return;
   paint.setPen( NoPen );                // do not draw outline
   paint.setBrush( colors[rand() % numColors]);// set random brush color

   QPoint p1( rand()%w, rand()%h );    // p1 = top left
   QPoint p2( rand()%w, rand()%h );    // p2 = bottom right

   QRect r( p1, p2 );
   paint.drawRect( r );                // draw filled rectangle
}

// Handles timer events for the Forever widget.
void Forever::timerEvent( QTimerEvent * )
{
   for ( int i=0; i<100; i++ ) {
    repaint( FALSE );                  // repaint, don't erase
    rectangles++;
   }
}

// Create and display Forever widget.
int main( int argc, char **argv )
{
   QApplication a( argc, argv );       // create application object
   Forever always;                     // create widget
   always.resize( 400, 250 );            // start up with size 400x250
   a.setMainWidget( &always );                // set as main widget
   always.setCaption("Qt Example - Forever");
   always.show();                  // show widget
```

```
    return a.exec();                    // run event loop
}
```

forever.h

```
#ifndef FOREVER_H
#define FOREVER_H

#include <qwidget.h>

const int numColors = 120;

class Forever : public QWidget
{
    Q_OBJECT
public:
    Forever( QWidget *parent=0, const char *name=0 );
protected:
    voidpaintEvent( QPaintEvent * );
    voidtimerEvent( QTimerEvent * );
private slots:
    voidupdateCaption();
private:
    int         rectangles;
    QColor colors[numColors];
};

#endif
```

실행



22. 살창보기프로그람

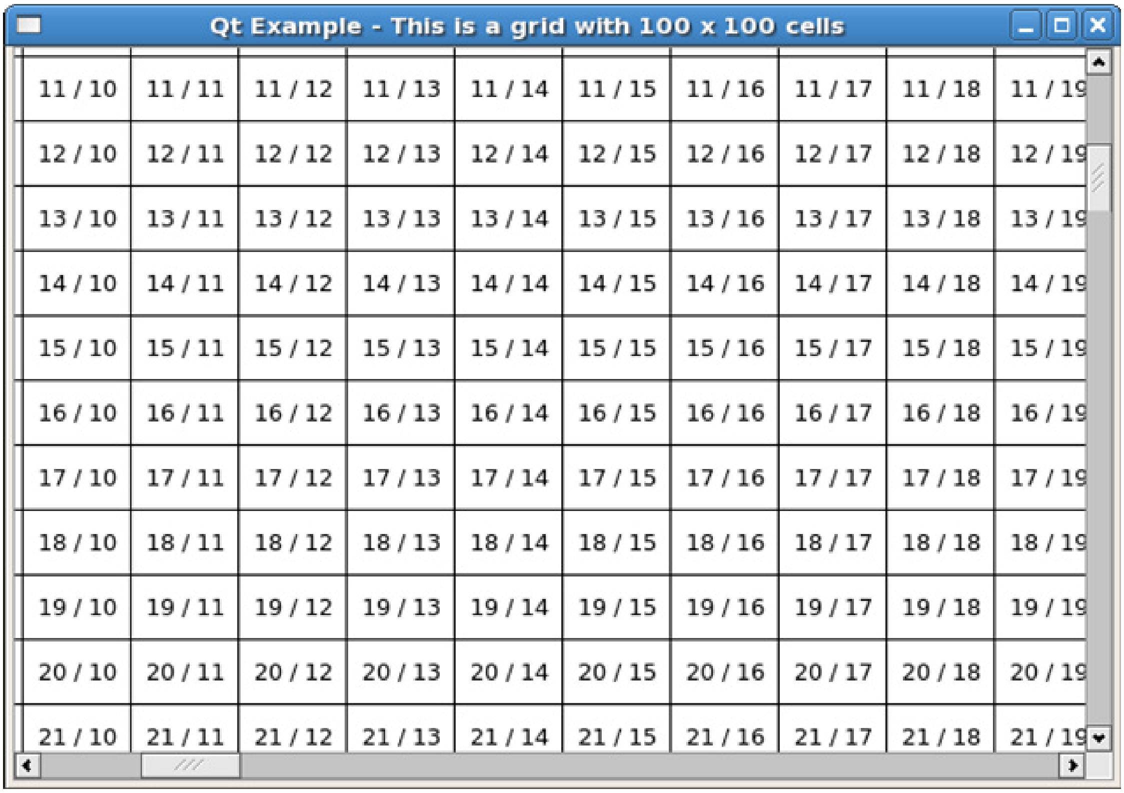
gridview.pro

```
TEMPLATE  = app
TARGET    = gridview
CONFIG    += qt warn_on release
HEADERS      =
SOURCES      = gridview.cpp
```

gridview.cpp

```
#include <qapplication.h>
#include <qgridview.h>
#include <qpainter.h>

// Grid size
const int numRows = 100;
const int numCols = 100;

class MyGridView : public QGridView
{
public:
   MyGridView() {
    setNumRows( ::numRows );
    setNumCols( ::numCols );
    setCellWidth( fontMetrics().width( QString("%1 / %2").arg(numRows()).arg(numCols())));
    setCellHeight( 2*fontMetrics().lineSpacing() );
    setCaption( tr( "Qt Example - This is a grid with 100 x 100 cells" ) );
   }

protected:
   void paintCell( QPainter *p, int row, int col ) {
    p->drawLine( cellWidth()-1, 0, cellWidth()-1, cellHeight()-1 );
    p->drawLine( 0, cellHeight()-1, cellWidth()-1, cellHeight()-1 );
    p->drawText( cellRect(), AlignCenter, QString("%1 / %1").arg(row).arg(col) );
   }
};

// The program starts here.
int main( int argc, char **argv )
{
   QApplication app( argc, argv );

   MyGridView gridview;
   app.setMainWidget( &gridview );
   gridview.show();
   return app.exec();
}
```

실행



11 / 10	11 / 11	11 / 12	11 / 13	11 / 14	11 / 15	11 / 16	11 / 17	11 / 18	11 / 19
12 / 10	12 / 11	12 / 12	12 / 13	12 / 14	12 / 15	12 / 16	12 / 17	12 / 18	12 / 19
13 / 10	13 / 11	13 / 12	13 / 13	13 / 14	13 / 15	13 / 16	13 / 17	13 / 18	13 / 19
14 / 10	14 / 11	14 / 12	14 / 13	14 / 14	14 / 15	14 / 16	14 / 17	14 / 18	14 / 19
15 / 10	15 / 11	15 / 12	15 / 13	15 / 14	15 / 15	15 / 16	15 / 17	15 / 18	15 / 19
16 / 10	16 / 11	16 / 12	16 / 13	16 / 14	16 / 15	16 / 16	16 / 17	16 / 18	16 / 19
17 / 10	17 / 11	17 / 12	17 / 13	17 / 14	17 / 15	17 / 16	17 / 17	17 / 18	17 / 19
18 / 10	18 / 11	18 / 12	18 / 13	18 / 14	18 / 15	18 / 16	18 / 17	18 / 18	18 / 19
19 / 10	19 / 11	19 / 12	19 / 13	19 / 14	19 / 15	19 / 16	19 / 17	19 / 18	19 / 19
20 / 10	20 / 11	20 / 12	20 / 13	20 / 14	20 / 15	20 / 16	20 / 17	20 / 18	20 / 19
21 / 10	21 / 11	21 / 12	21 / 13	21 / 14	21 / 15	21 / 16	21 / 17	21 / 18	21 / 19

23. 《안녕하십니까》 프로그람

이 실례는 각이한 색으로 단어 "Hello, World"를 아래우로 이동한다.

hello.pro

```
TEMPLATE  = app
TARGET    = hello
CONFIG    += qt warn_on release
HEADERS       = hello.h
SOURCES       = hello.cpp \
          main.cpp
```

hello.cpp

```
#include "hello.h"
#include <qpushbutton.h>
#include <qtimer.h>
#include <qpainter.h>
#include <qpixmap.h>

/*
  Constructs a Hello widget. Starts a 40 ms animation timer.
*/

Hello::Hello( const char *text, QWidget *parent, const char *name )
    : QWidget(parent,name), t(text), b(0)
{
    QTimer *timer = new QTimer(this);
    connect( timer, SIGNAL(timeout()), SLOT(animate()) );
    timer->start( 40 );

    resize( 260, 130 );
}

/*
  This private slot is called each time the timer fires.
*/

void Hello::animate()
{
    b = (b + 1) & 15;
    repaint( FALSE );
}

/*
  Handles mouse button release events for the Hello widget.
  We emit the clicked() signal when the mouse is released inside
  the widget.
*/

void Hello::mouseReleaseEvent( QMouseEvent *e )
{
    if ( rect().contains( e->pos() ) )
        emit clicked();
}

/*
  Handles paint events for the Hello widget.
  Flicker-free update. The text is first drawn in the pixmap and the
  pixmap is then blt'ed to the screen.
*/

void Hello::paintEvent( QPaintEvent * )
{
    static int sin_tbl[16] = {
        0, 38, 71, 92, 100, 92, 71, 38,  0, -38, -71, -92, -100, -92, -71, -38};

    if ( t.isEmpty() )
        return;

    // 1: Compute some sizes, positions etc.
    QFontMetrics fm = fontMetrics();
    int w = fm.width(t) + 20;
    int h = fm.height() * 2;
    int pmx = width()/2 - w/2;
    int pmy = height()/2 - h/2;

    // 2: Create the pixmap and fill it with the widget's background
    QPixmap pm( w, h );
    pm.fill( this, pmx, pmy );

    // 3: Paint the pixmap. Cool wave effect
    QPainter p;
    int x = 10;
    int y = h/2 + fm.descent();
    int i = 0;
    p.begin( &pm );
    p.setFont( font() );
    while ( !t[i].isNull() ) {
        int i16 = (b+i) & 15;
        p.setPen( QColor((15-i16)*16,255,255,QColor::Hsv) );
        p.drawText( x, y-sin_tbl[i16]*h/800, t.mid(i,1), 1 );
        x += fm.width( t[i] );
        i++;
    }
    p.end();

    // 4: Copy the pixmap to the Hello widget
    bitBlt( this, pmx, pmy, &pm );
}
```

hello.h

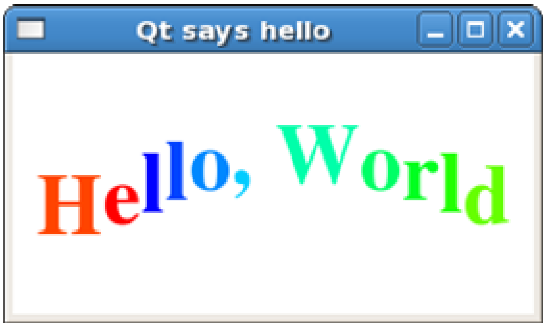
```
#ifndef HELLO_H
#define HELLO_H

#include <qwidget.h>

class Hello : public QWidget
{
    Q_OBJECT
public:
    Hello( const char *text, QWidget *parent=0, const char *name=0 );
signals:
    void clicked();
protected:
    void mouseReleaseEvent( QMouseEvent * );
    void paintEvent( QPaintEvent * );
private slots:
    void animate();
private:
    QString t;
    int     b;
};

#endif
```

main.cpp

```
#include "hello.h"
#include <qapplication.h>

/*
  The program starts here. It parses the command line and builds a message
  string to be displayed by the Hello widget.
*/

int main( int argc, char **argv )
```

```
{
   QApplication a(argc,argv);
   QString s;
   for ( int i=1; i<argc; i++ ) {
    s += argv[i];
    if ( i<argc-1 )
        s += " ";
   }
   if ( s.isEmpty() )
    s = "Hello, World";
   Hello h( s );
#ifndef QT_NO_WIDGET_TOPEXTRA  // for Qt/Embedded minimal build
   h.setCaption( "Qt says hello" );
#endif
   QObject::connect( &h, SIGNAL(clicked()), &a, SLOT(quit()) );
   h.setFont( QFont("times",32,QFont::Bold) );         // default font
   h.setBackgroundColor( Qt::white );           // default bg color
   a.setMainWidget( &h );
   h.show();
   return a.exec();
}
```

실행



24. 방조프로그람

helpdemo.pro

```
TEMPLATE = app
CONFIG  += qt warn_on
LIBS    += -lqassistantclient
unix {
  UI_DIR = .ui
  MOC_DIR = .moc
  OBJECTS_DIR = .obj
}
SOURCES += helpdemo.cpp main.cpp
HEADERS += helpdemo.h
FORMS    = helpdemobase.ui
```

helpdemo.cpp

```
#include <qassistantclient.h>
#include <qmessagebox.h>
#include <qlineedit.h>
```

```
#include <qaction.h>
#include <qpopupmenu.h>
#include <qcheckbox.h>
#include <qprocess.h>
#include <qpushbutton.h>
#include <qdir.h>

#include "helpdemo.h"

HelpDemo::HelpDemo( QWidget *parent, const char *name )  : HelpDemoBase( parent, name )
{
   leFileName->setText( "./doc/index.html" );
   assistant = new QAssistantClient( QDir( "../../bin" ).absPath(), this );
   widgets.insert( (QWidget*)openQAButton, "./doc/manual.html#openqabutton" );
   widgets.insert( (QWidget*)closeQAButton, "./doc/manual.html#closeqabutton" );
   widgets.insert( (QWidget*)checkOnlyExampleDoc, "./doc/manual.html#onlydoc" );
   widgets.insert( (QWidget*)checkHide, "./doc/manual.html#hide" );
   widgets.insert( (QWidget*)leFileName, "./doc/manual.html#lineedit" );
   widgets.insert( (QWidget*)displayButton, "./doc/manual.html#displaybutton" );
   widgets.insert( (QWidget*)closeButton, "./doc/manual.html#closebutton" );

   menu = new QPopupMenu( this );

   QAction *helpAction = new QAction( "Show Help", QKeySequence(tr("F1")), this );
   helpAction->addTo( menu );

   connect( helpAction, SIGNAL(activated()), this, SLOT(showHelp()) );
   connect( assistant, SIGNAL(assistantOpened()), this, SLOT(assistantOpened()) );
   connect( assistant, SIGNAL(assistantClosed()), this, SLOT(assistantClosed()));
   connect( assistant, SIGNAL(error(const QString&)),
        this, SLOT(showAssistantErrors(const QString&)) );
   closeQAButton->setEnabled(FALSE);
}

HelpDemo::~HelpDemo()
{
}

void HelpDemo::contextMenuEvent( QContextMenuEvent *e )
{
   QWidget *w = lookForWidget();
   if ( menu->exec( e->globalPos() ) != -1 )
    showHelp( w );
}

QWidget* HelpDemo::lookForWidget()
{
   QPtrDictIterator<char> it( widgets );
   QWidget *w;
   while ( (w = (QWidget*)(it.currentKey())) != 0 ) {
    ++it;
    if ( w->hasMouse() )
       return w;
   }
```

```
    return 0;
}

void HelpDemo::showHelp()
{
    showHelp( lookForWidget() );
}

void HelpDemo::showHelp( QWidget *w )
{
    if ( w )
      assistant->showPage( QString( widgets[w] ) );
    else
      assistant->showPage( "./doc/index.html" );
}

void HelpDemo::setAssistantArguments()
{
    QStringList cmdLst;
    if ( checkHide->isChecked() )
      cmdLst << "-hideSidebar";
    if ( checkOnlyExampleDoc->isChecked() )
        cmdLst << "-profile"
              << QString("doc") + QDir::separator() + QString("helpdemo.adp");
    assistant->setArguments( cmdLst );
}

void HelpDemo::openAssistant()
{
    if ( !assistant->isOpen() )
      assistant->openAssistant();
}

void HelpDemo::closeAssistant()
{
    if ( assistant->isOpen() )
      assistant->closeAssistant();
}

void HelpDemo::displayPage()
{
    assistant->showPage( leFileName->text() );
}

void HelpDemo::showAssistantErrors( const QString &err )
{
    QMessageBox::critical( this, "Assistant Error", err );

}

void HelpDemo::assistantOpened()
{
    closeQAButton->setEnabled( TRUE );
    openQAButton->setEnabled( FALSE );
```

```
}

void HelpDemo::assistantClosed()
{
    closeQAButton->setEnabled( FALSE );
    openQAButton->setEnabled( TRUE );
}
```

helpdemo.h

```
#ifndef HELPDEMO_H
#define HELPDEMO_H

#include <qptrdict.h>

#include "helpdemobase.h"

class QAssistantClient;
class QPopupMenu;

class HelpDemo : public HelpDemoBase
{
    Q_OBJECT

public:
    HelpDemo( QWidget *parent = 0, const char *name = 0 );
    ~HelpDemo();

protected:
    void contextMenuEvent( QContextMenuEvent *e );

private slots:
    void setAssistantArguments();
    void openAssistant();
    void closeAssistant();
    void displayPage();
    void showAssistantErrors( const QString &err );
    void assistantOpened();
    void assistantClosed();
    void showHelp();

private:
    QWidget* lookForWidget();
    void showHelp( QWidget *w );

    QPtrDict<char> widgets;
    QAssistantClient *assistant;
    QPopupMenu *menu;

};

#endif
```

helpdemobase.ui

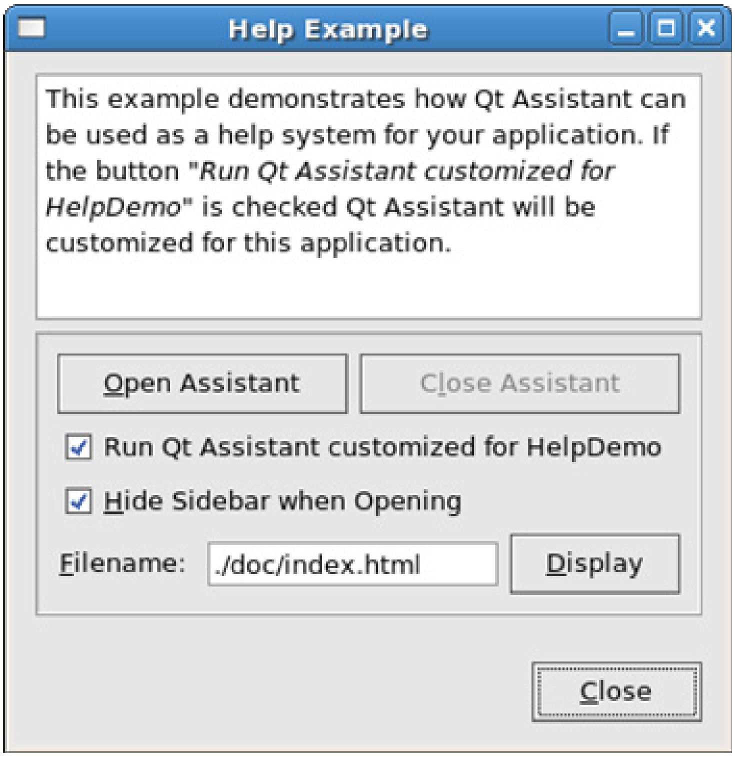
```
<!DOCTYPE UI><UI version="3.2" stdsetdef="1">
```

```
<class>HelpDemoBase</class>
<widget class="QWidget">
   <property name="name">
…
   <slot>setAssistantArguments()</slot>
</slots>
<layoutdefaults spacing="6" margin="11"/>
</UI>
```

main.cpp

```
#include <qapplication.h>
#include "helpdemo.h"

int main( int argc, char ** argv )
{
   QApplication a( argc, argv );
   HelpDemo help;
   help.show();
   a.connect( &a, SIGNAL( lastWindowClosed() ), &a, SLOT( quit() ) );
   return a.exec();
}
```

실행



This example demonstrates how Qt Assistant can be used as a help system for your application. If the button "*Run Qt Assistant customized for HelpDemo*" is checked Qt Assistant will be customized for this application.

☑ Run Qt Assistant customized for HelpDemo

☑ Hide Sidebar when Opening

Filename: ./doc/index.html

25. 방조체계

이 실례는 응용프로그람에서 문맥의존방조를 제공하는데 쓰이는 각이한 Qt클라스들을 보여준다.

QToolTip와 QWhatsThis를 사용하여 응용프로그람에서 창문부품들을 위한 정적 및 동적 고무풍선방조를 모두 제공하며 QToolTipGroup를 사용하여 상태띠에서 매개 도구암시용의 확장정보를 현시한다. QAssistantClient는 Qt Assistant를 리용하여 방조페지를 현시하는데 쓰인다.

응용프로그람은 차림표띠와 상태띠, 도구띠를 가지는 QMainWindow에 기초한 사용자대면부를 가지며 QTable을 중심창문부품으로서 사용한다.

2개의 QToolTip파생클라스들은 maybeTip()를 재실현하여 QHeader와 QTable용의 동적도구암시를 실현한다. 구성자들은 QWidget대신에 구성자의 첫 파라메터로서 QHeader와 QTable을 각각 가진다는데서 QToolTip구성자와 다르다. 이것은 오직 머리부파일과 표들만 인수로서 넘길수 있도록 담보하려고 하기때무이다. QToolTipGroup를 둘째 인수로 넘기여 가령 상태띠에서 도구암시를 표시할수 있다.

TableToolTip클라스는 후에 QTable객체를 쉽게 호출하기 위하여 QTable에로의 참고를 성원으로서 보유한다.

HeaderToolTip구성자는 QToolTip구성자에 파라메터로서 전달된다.

maybeTip()의 실현은 QHeader API를 리용하여 요구되는 위치에 절(section)을 얻으며 QToolTip::tip()를 리용하여 도구암시에 절의 표식을 현시한다. 둘째 문자렬은 QToolTipGroup에 의해 사용되며 상태띠에 표시된다.

QTable은 QScrollView이므로 모든 사용자교제는 QTable의 viewport()에서 일어난다. TableToolTip구성자는 viewport()에 넘어가고 도구암시그룹은 QToolTip구성자에 넘어가며 table성원을 QTable지적자자체로 초기화한다.

maybeTip()의 실현에서는 QTable API를 사용하여 요구된 위치에 있는 세포에 대한 정보를 얻는다. QTable API는 내용들의 좌표를 기대하며 요구하는 점은 보기구역과 관련되여있으므로 QTable의 함수들을 사용하기전에 좌표들을 변환하여야 한다.

세포의 기하학을 보기구역좌표로 변환함으로써 마우스유표가 세포를 떠날 때 도구암시가 없어지게 하고 QToolTip::tip()에 의하여 도구암시에 세포의 표식을 표시하고 사전에 QToolTipGroup의 본문을 제공한다.

WhatsThis클라스는 QObject와 QWhatsThis의 파생클라스로서 HeaderWhatsThis와 TableWhatsThis클라스들의 기초클라스로서 작업한다. (1) WhatsThis는 사용자가 What's this?창문안에서 마우스를 누를 때 호출되는 clicked()를 실현한다. 또한 초련결이 선택될 때 발행되는 신호 linkClicked()를 선언한다.

WhatsThis구성자는 두개의 파라메터를 가지는데 첫째는 WhatsThis를 제공하려고 하는 창문부품이고 둘째는 사건들을 수신하는 창문부품이다. 보통 이것은 같은 창문부품이지만 QTable과 같은 일부 창문부품들은 더 복잡하고 사건들을 수신하는 viewport()창문부품을 가진다. 그러한 창문부품을 구성자에 넘기면 그 파라메터를 QWhatsThis구성자에 넘기고 QWidget지적자자체를 그 성원변수에 넘기여 후에 QWidget API를 쉽게 사용하게 한다.

clicked()의 실현은 초련결이 선택되였으면 linkClicked()신호를 발행한다.

HeaderWhatsThis와 TableWhatsThis클라스들은 text()를 재실현하여 마우스누르기위치에 따라 본문을 돌려보낼수 있게 한다. 다른 모든 기능은 이미 일반WhatsThis기초클라스에 의해 제공된다. 도구암시클라스들에서와 같은 방법으로 여기서 형안전을 담보한다.

HeaderWhatsThis구성자는 WhatsThis구성자에로 파라메터를 전달한다.

text()의 실현에서는 uses the QHeader API를 사용하여 수평 혹은 수직머리부를 가지는가 결정하고 머리부의 방향과 절의 상태를 표시하는 문자렬을 돌려준다. (2)

QTable이 스크롤보기이고 사건들을 받아들이는 viewport()를 가지므로 표자체와 표의 viewport()를 to the WhatsThis구성자에 전달한다.

text()의 실현에서는 QTable API를 사용하여 요구하는 위치의 세포에 대한 정보를 얻는다. QTable API는 내용의 좌표를 기대하므로 처음에 도구암시클라스들에서 보여준것처럼 점을 변환하여야 한다. rtti()함수를 사용하여 항목의 형을 나타내고 문자렬을 결과로 돌려준다.

QMainWindow은 Qt Assistant가 응용프로그람에서 상황의존방조를 제공하는것과 함께 우의 클라스들을 사용하는 사용자대면부를 창조하는데 쓰인다.

MainWindow클라스는 호출시에 Qt Assistant의 실례를 창조하는 assistantSlot()라는 처리부를 선언한다. 이 클라스는 도구암시클라스들이 QObject가 아니고 명시적으로 삭제되여야 하므로 그것들에로의 참고를 성원으로 보관한다. 그 클라스는 QAssistantClient에로의 참고도 물론 성원으로 가짐으로써 후에 Qt Assistant에 대한 호출을 더 쉽게 한다.

MainWindow구성자는 체계경로를 사용하는 첫 인수로서 QString::null을 사용하여 QAssistantClient의 실례를 창조한다.

QTable은 중심창문부품으로 사용되고 표, 차림표, 도구띠가 포함된다.

정적함수 whatsThisButton()는 찰칵할 때 "What's this?"방식에 들어가는 QToolButton을 창조한다.

QToolTipGroup가 창조되고 창문부품들에 도구암시들이 현시될 때 상태띠에 도구암시들을 표시하고 제거한다.

도구암시들이 설정된다. 정적함수 add()는 Assistant도구단추에 도구암시를 설정한다. 도구암시객체들은 QToolTip파생클라스들에 의해 창조되고 구성자의 첫 파라메터는 동적암시들을 추가하려고 하는 창문부품을 지정하고 둘째인수는 그것들이 속하는 QToolTipGroup를 지정한다.

WhatsThis방조가 설정된다. 정적함수 add()는 What's This?방조를 Assistant를 여는 도구단추에 추가한다. 2개의 WhatsThis파생클라스들의 실례들은 머리부와 표에 창조된다. What's This?방조는 또한 차림표항목들에 추가된다.

신호와 처리부가 접속되여 하이퍼련결이나 방조자단추를 찰칵할 때 관련한 페지들이 현시된다.

해체자는 도구암시들을 삭제한다. 우에서 언급한것처럼 QToolTip가 QObject의 파생클라스가 아니고 창문부품이 삭제될 때 QToolTip의 실례가 삭제되지 않으므로 도구암시들을 명시적으로 삭제해야 한다.

assistantSlot()는 applicationDirPath()를 사용하여 문서파일들의 위치를 찾으며 Qt Assistant의 지정된 페지를 표시한다.

main함수는 응용프로그람의 기본창문을 여는 표준실현이다.

실례를 건설하려면 helpsystem등록부(QTDIR/examples/helpsystem)로 가서 qmake를 실행하여 makefile을 생성하고 make도구를 사용하여 서고를 건설한다.

① moc는 QObject가 첫 기초클라스일것을 요구한다.

② HeaderWhatsThis가 다중계승을 사용하므로 명시적으로 범위(QObject 혹은 QWhatsThis)를 지정해야 한다.

helpsystem.pro

```
TEMPLATE = app
LIBS    += -lqassistantclient
SOURCES += main.cpp tooltip.cpp mainwindow.cpp whatsthis.cpp
HEADERS += tooltip.h mainwindow.h whatsthis.h
```

mainwindow.cpp

```
#include <qapplication.h>
#include <qassistantclient.h>
#include <qfiledialog.h>
#include <qmenubar.h>
#include <qpopupmenu.h>
#include <qstatusbar.h>
#include <qtable.h>
#include <qtoolbar.h>
#include <qtoolbutton.h>
#include <qtooltip.h>

#include "mainwindow.h"
#include "tooltip.h"
#include "whatsthis.h"
```

```
MainWindow::MainWindow()
{
   statusBar();
   assistant = new QAssistantClient( QDir("../../bin").absPath(), this );

   QTable* table = new QTable( 2, 3, this );
   setCentralWidget( table );

   // populate table
   QStringList comboEntries;
   comboEntries << "one" << "two" << "three" << "four";
   QComboTableItem* comboItem1 = new QComboTableItem( table, comboEntries );
   QComboTableItem* comboItem2 = new QComboTableItem( table, comboEntries );
   QCheckTableItem* checkItem1 = new QCheckTableItem( table, "Check me" );
   QCheckTableItem* checkItem2 = new QCheckTableItem( table, "Check me" );

   table->setItem( 0, 0, comboItem1 );
   table->setItem( 1, 0, comboItem2 );

   table->setItem( 1, 1, checkItem1  );
   table->setItem( 0, 1, checkItem2 );

   table->setText( 1, 2, "Text" );

   table->horizontalHeader()->setLabel( 0, " Combos" );
   table->horizontalHeader()->setLabel( 1, "Checkboxes" );
   table->verticalHeader()->setLabel( 0, "1" );
   table->verticalHeader()->setLabel( 1, "2" );

   // populate menubar
   QPopupMenu* fileMenu = new QPopupMenu( this );
   QPopupMenu* helpMenu = new QPopupMenu( this );

   menuBar()->insertItem( "&File", fileMenu );
   menuBar()->insertItem( "&Help", helpMenu );

   int fileId = fileMenu->insertItem( "E&xit", this, SLOT(close()) );

   int helpId = helpMenu->insertItem( "Open Assistant", this, SLOT(assistantSlot()) );

   // populate toolbar
   QToolBar* toolbar = new QToolBar( this );
   QToolButton* assistantButton = new QToolButton( toolbar );
   assistantButton->setIconSet( QPixmap("appicon.png") );
   QWhatsThis::whatsThisButton( toolbar );

   //create tooltipgroup
   QToolTipGroup * tipGroup = new QToolTipGroup( this );
   connect( tipGroup, SIGNAL(showTip(const QString&)), statusBar(),
     SLOT(message(const QString&)) );
   connect( tipGroup, SIGNAL(removeTip()), statusBar(), SLOT(clear()) );

   // set up tooltips
```

```
    QToolTip::add( assistantButton, tr ("Open Assistant"), tipGroup, "Opens Qt Assistant" );

    horizontalTip = new HeaderToolTip( table->horizontalHeader(), tipGroup );
    verticalTip = new HeaderToolTip( table->verticalHeader(), tipGroup );

    cellTip = new TableToolTip( table, tipGroup );

    // set up whats this
    QWhatsThis::add ( assistantButton, "This is a toolbutton which opens Assistant" );

    HeaderWhatsThis *horizontalWhatsThis = new HeaderWhatsThis( table->horizontalHeader() );
    HeaderWhatsThis *verticalWhatsThis = new HeaderWhatsThis( table->verticalHeader() );

    TableWhatsThis *cellWhatsThis = new TableWhatsThis( table );

    fileMenu->setWhatsThis( fileId, "Click here to exit the application" );
    helpMenu->setWhatsThis( helpId, "Click here to open Assistant" );

    // connections
    connect( assistantButton, SIGNAL(clicked()), this, SLOT(assistantSlot()) );
    connect( horizontalWhatsThis, SIGNAL(linkClicked(const QString&)), assistant,
     SLOT(showPage(const QString&)) );
    connect( verticalWhatsThis, SIGNAL(linkClicked(const QString&)), assistant,
     SLOT(showPage(const QString&)) );
    connect( cellWhatsThis, SIGNAL(linkClicked(const QString&)), assistant,
     SLOT(showPage(const QString&)) );
}

MainWindow::~MainWindow()
{
    delete horizontalTip;
    delete verticalTip;
    delete cellTip;
}

void MainWindow::assistantSlot()
{
    QString docsPath = QDir("../../doc").absPath();
    assistant->showPage( QString("%1/html/qassistantclient.html").arg(docsPath) );
}
```

mainwindow.h

```
#include <qmainwindow.h>
class HeaderToolTip;
class TableToolTip;
class QAssistantClient;

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    MainWindow();
    ~MainWindow();
public slots:
    void assistantSlot();

private:
    HeaderToolTip *horizontalTip;
    HeaderToolTip *verticalTip;
    TableToolTip *cellTip;
    QAssistantClient *assistant;
};
```

tooltip.cpp

```
#include <qtooltip.h>
#include <qtable.h>
#include "tooltip.h"

HeaderToolTip::HeaderToolTip( QHeader *header, QToolTipGroup *group )
: QToolTip( header, group )
{
}

void HeaderToolTip::maybeTip ( const QPoint& p )
{
   QHeader *header = (QHeader*)parentWidget();

   int section = 0;

   if ( header->orientation() == Horizontal )
     section = header->sectionAt( header->offset() + p.x() );
   else
     section = header->sectionAt( header->offset() + p.y() );

   QString tipString = header->label( section );
   tip( header->sectionRect( section ), tipString, "This is a section in a header" );
}

TableToolTip::TableToolTip( QTable *tipTable, QToolTipGroup *group )
: QToolTip( tipTable->viewport(), group ), table( tipTable )
{
}

void TableToolTip::maybeTip ( const QPoint &p )
{
   QPoint cp = table->viewportToContents( p );
   int row = table->rowAt( cp.y() );
   int col = table->columnAt( cp.x() );

   QString tipString = table->text( row, col );

   QRect cr = table->cellGeometry( row, col );
   cr.moveTopLeft( table->contentsToViewport( cr.topLeft() ) );
   tip( cr, tipString, "This is a cell in a table" );
}
```

tooltip.h

```
#ifndef TOOLTIP_H
#define TOOLTIP_H

#include <qtooltip.h>

class QTable;
class QHeader;

class HeaderToolTip : public QToolTip
{
public:
   HeaderToolTip( QHeader *header, QToolTipGroup *group = 0 );

protected:
   void maybeTip ( const QPoint &p );
};

class TableToolTip : public QToolTip
{
public:
   TableToolTip( QTable* table, QToolTipGroup *group = 0  );

protected:
   void maybeTip( const QPoint &p );

private:
   QTable *table;
};

#endif
```

whatsthis.cpp

```
#include <qapplication.h>
#include <qdir.h>
#include <qheader.h>
#include <qtable.h>

#include "whatsthis.h"

WhatsThis::WhatsThis( QWidget *w, QWidget *watch ) : QWhatsThis( watch ? watch : w ), widget( w )
{
}

QWidget *WhatsThis::parentWidget() const
{
   return widget;
}

bool WhatsThis::clicked( const QString &link )
{
   if ( !link.isEmpty() )
     emit linkClicked( link );
```

```
    return TRUE;
}

HeaderWhatsThis::HeaderWhatsThis( QHeader *h )
: WhatsThis( h )
{
}

QString HeaderWhatsThis::text( const QPoint &p )
{
    QHeader *header = (QHeader*)parentWidget();

    QString orient;
    int section;
    if ( header->orientation() == QObject::Horizontal ) {
     orient = "horizontal";
     section = header->sectionAt( p.x() );
    } else {
     orient = "vertical";
     section = header->sectionAt( p.y() );
    }
    if( section == -1 )
     return "This is empty space.";
    QString docsPath = QDir("../../doc").absPath();
    return QString("This is section number %1 in the %2 <a href=%2/html/qheader.html>header</a>.").
     arg(section + 1).
     arg(orient).
     arg(docsPath);
}

TableWhatsThis::TableWhatsThis( QTable *t )
: WhatsThis( t, t->viewport() )
{
}

QString TableWhatsThis::text( const QPoint &p )
{
    QTable *table = (QTable*)parentWidget();

    QPoint cp = table->viewportToContents( p );
    int row = table->rowAt( cp.y() );
    int col = table->columnAt( cp.x() );

    if ( row == -1 || col == -1 )
     return "This is empty space.";

    QTableItem* i = table->item( row,col  );
    if ( !i )
     return "This is an empty cell.";

    QString docsPath = QDir("../../doc").absPath();

    if ( QTableItem::RTTI == i->rtti() ) {
     return QString("This is a <a href=%1/html/qtableitem.html>QTableItem</a>.").
```

```
            arg(docsPath);
    } else if ( QComboTableItem::RTTI == i->rtti() ) {
     return QString("This is a <a href=%1/html/qcombotableitem.html>QComboTableItem</a>."
            "<br>It can be used to provide multiple-choice items in a table.").
            arg(docsPath);
    } else if ( QCheckTableItem::RTTI == i->rtti() ) {
     return QString("This is a <a href=%1/html/qchecktableitem.html>QCheckTableItem</a>."
            "<br>It provide <a href=%1/html/qcheckbox.html>checkboxes</a> in tables.").
            arg(docsPath).arg(docsPath);
    }

    return "This is a user defined table item.";
}
```

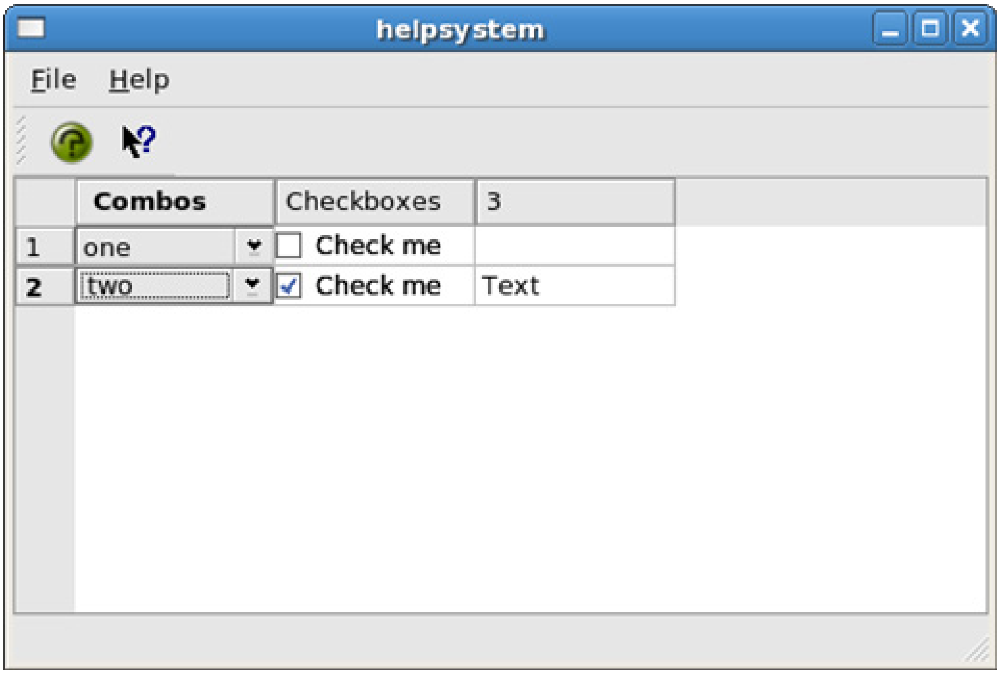
whatsthis.h

```
#ifndef WHATSTHIS_H
#define WHATSTHIS_H

#include <qwhatsthis.h>

class QHeader;
class QTable;

class WhatsThis : public QObject, public QWhatsThis
{
    Q_OBJECT
public:
    WhatsThis( QWidget *w, QWidget *watch = 0 );

    bool clicked( const QString &link );
    QWidget *parentWidget() const;

signals:
    void linkClicked( const QString &link );

private:
    QWidget *widget;
};

class HeaderWhatsThis : public WhatsThis
{
public:
    HeaderWhatsThis( QHeader *h );

    QString text( const QPoint &p );
};

class TableWhatsThis : public WhatsThis
{
public:
    TableWhatsThis( QTable *t );

    QString text( const QPoint &p );
};
```

```
#endif

main.cpp
#include <qapplication.h>
#include "mainwindow.h"

int main( int argc, char** argv )
{
   QApplication app( argc, argv );
   MainWindow main;
   main.show();
   app.setMainWidget( &main );
   return app.exec();
}
```

appicon.png

실행



	Combos	Checkboxes	3
1	one	□ Check me	
2	two	☑ Check me	Text

26. 간단한 HTML 방조열람기

이 실례는 Qt 의 풍부한 본문능력을 리용하여 간단한 HTML 방조열람기를 실현한다.

helpviewer.pro

```
TEMPLATE  = app
TARGET     = helpviewer
CONFIG     += qt warn_on release
HEADERS        = helpwindow.h
SOURCES        = helpwindow.cpp \
```

main.cpp

helpwindow.cpp

```
#include "helpwindow.h"
#include <qstatusbar.h>
#include <qpixmap.h>
#include <qpopupmenu.h>
#include <qmenubar.h>
#include <qtoolbar.h>
#include <qtoolbutton.h>
#include <qiconset.h>
#include <qfile.h>
#include <qtextstream.h>
#include <qstylesheet.h>
#include <qmessagebox.h>
#include <qfiledialog.h>
#include <qapplication.h>
#include <qcombobox.h>
#include <qevent.h>
#include <qlineedit.h>
#include <qobjectlist.h>
#include <qfileinfo.h>
#include <qfile.h>
#include <qdatastream.h>
#include <qprinter.h>
#include <qsimplerichtext.h>
#include <qpainter.h>
#include <qpaintdevicemetrics.h>

#include <ctype.h>

HelpWindow::HelpWindow( const QString& home_, const QString& _path,
            QWidget* parent, const char *name )
   : QMainWindow( parent, name, WDestructiveClose ),
     pathCombo( 0 )
{
   readHistory();
   readBookmarks();

   browser = new QTextBrowser( this );

   browser->mimeSourceFactory()->setFilePath( _path );
   browser->setFrameStyle( QFrame::Panel | QFrame::Sunken );
   connect( browser, SIGNAL( sourceChanged(const QString& ) ),
        this, SLOT( sourceChanged( const QString&) ) );

   setCentralWidget( browser );

   if ( !home_.isEmpty() )
    browser->setSource( home_ );

   connect( browser, SIGNAL( highlighted( const QString&) ),
        statusBar(), SLOT( message( const QString&)) );
```

```
resize( 640,700 );

QPopupMenu* file = new QPopupMenu( this );
file->insertItem( tr("&New Window"), this, SLOT( newWindow() ), CTRL+Key_N );
file->insertItem( tr("&Open File"), this, SLOT( openFile() ), CTRL+Key_O );
file->insertItem( tr("&Print"), this, SLOT( print() ), CTRL+Key_P );
file->insertSeparator();
file->insertItem( tr("&Close"), this, SLOT( close() ), CTRL+Key_Q );
file->insertItem( tr("E&xit"), qApp, SLOT( closeAllWindows() ), CTRL+Key_X );

// The same three icons are used twice each.
QIconSet icon_back( QPixmap("back.xpm") );
QIconSet icon_forward( QPixmap("forward.xpm") );
QIconSet icon_home( QPixmap("home.xpm") );

QPopupMenu* go = new QPopupMenu( this );
backwardId = go->insertItem( icon_back, tr("&Backward"), browser, SLOT( backward() ),
                 CTRL+Key_Left );
forwardId = go->insertItem( icon_forward, tr("&Forward"), browser, SLOT( forward() ),
                CTRL+Key_Right );
go->insertItem( icon_home, tr("&Home"), browser, SLOT( home() ) );

QPopupMenu* help = new QPopupMenu( this );
help->insertItem( tr("&About"), this, SLOT( about() ) );
help->insertItem( tr("About &Qt"), this, SLOT( aboutQt() ) );

hist = new QPopupMenu( this );
QStringList::Iterator it = history.begin();
for ( ; it != history.end(); ++it )
  mHistory[ hist->insertItem( *it ) ] = *it;
connect( hist, SIGNAL( activated( int ) ),  this, SLOT( histChosen( int ) ) );

bookm = new QPopupMenu( this );
bookm->insertItem( tr( "Add Bookmark" ), this, SLOT( addBookmark() ) );
bookm->insertSeparator();

QStringList::Iterator it2 = bookmarks.begin();
for ( ; it2 != bookmarks.end(); ++it2 )
  mBookmarks[ bookm->insertItem( *it2 ) ] = *it2;
connect( bookm, SIGNAL( activated( int ) ),  this, SLOT( bookmChosen( int ) ) );

menuBar()->insertItem( tr("&File"), file );
menuBar()->insertItem( tr("&Go"), go );
menuBar()->insertItem( tr( "History" ), hist );
menuBar()->insertItem( tr( "Bookmarks" ), bookm );
menuBar()->insertSeparator();
menuBar()->insertItem( tr("&Help"), help );

menuBar()->setItemEnabled( forwardId, FALSE);
menuBar()->setItemEnabled( backwardId, FALSE);
connect( browser, SIGNAL( backwardAvailable( bool ) ),
      this, SLOT( setBackwardAvailable( bool ) ) );
connect( browser, SIGNAL( forwardAvailable( bool ) ), this, SLOT( setForwardAvailable( bool ) ) );
```

```
    QToolBar* toolbar = new QToolBar( this );
    addToolBar( toolbar, "Toolbar");
    QToolButton* button;

    button = new QToolButton( icon_back, tr("Backward"), "", browser, SLOT(backward()), toolbar );
    connect( browser, SIGNAL( backwardAvailable(bool) ), button, SLOT( setEnabled(bool) ) );
    button->setEnabled( FALSE );
    button = new QToolButton( icon_forward, tr("Forward"), "", browser, SLOT(forward()), toolbar );
    connect( browser, SIGNAL( forwardAvailable(bool) ), button, SLOT( setEnabled(bool) ) );
    button->setEnabled( FALSE );
    button = new QToolButton( icon_home, tr("Home"), "", browser, SLOT(home()), toolbar );

    toolbar->addSeparator();

    pathCombo = new QComboBox( TRUE, toolbar );
    connect( pathCombo, SIGNAL( activated( const QString & ) ),
         this, SLOT( pathSelected( const QString & ) ) );
    toolbar->setStretchableWidget( pathCombo );
    setRightJustification( TRUE );
    setDockEnabled( DockLeft, FALSE );
    setDockEnabled( DockRight, FALSE );

    pathCombo->insertItem( home_ );
    browser->setFocus();

}

void HelpWindow::setBackwardAvailable( bool b)
{
    menuBar()->setItemEnabled( backwardId, b);
}

void HelpWindow::setForwardAvailable( bool b)
{
    menuBar()->setItemEnabled( forwardId, b);
}

void HelpWindow::sourceChanged( const QString& url )
{
    if ( browser->documentTitle().isNull() )
     setCaption( "Qt Example - Helpviewer - " + url );
    else
     setCaption( "Qt Example - Helpviewer - " + browser->documentTitle() ) ;

    if ( !url.isEmpty() && pathCombo ) {
     bool exists = FALSE;
     int i;
     for ( i = 0; i < pathCombo->count(); ++i ) {
         if ( pathCombo->text( i ) == url ) {
          exists = TRUE;
          break;
         }
     }
     if ( !exists ) {
         pathCombo->insertItem( url, 0 );
         pathCombo->setCurrentItem( 0 );
         mHistory[ hist->insertItem( url ) ] = url;
     } else
         pathCombo->setCurrentItem( i );
   }
}

HelpWindow::~HelpWindow()
{
   history =  mHistory.values();

   QFile f( QDir::currentDirPath() + "/.history" );
   f.open( IO_WriteOnly );
   QDataStream s( &f );
   s << history;
   f.close();

   bookmarks = mBookmarks.values();

   QFile f2( QDir::currentDirPath() + "/.bookmarks" );
   f2.open( IO_WriteOnly );
   QDataStream s2( &f2 );
   s2 << bookmarks;
   f2.close();
}

void HelpWindow::about()
{
   QMessageBox::about( this, "HelpViewer Example",
              "<p>This example implements a simple HTML help viewer "
              "using Qt's rich text capabilities</p>"
              "<p>It's just about 400 lines of C++ code, so don't expect too much :-)</p>"
              );
}

void HelpWindow::aboutQt()
{
   QMessageBox::aboutQt( this, "QBrowser" );
}

void HelpWindow::openFile()
{
#ifndef QT_NO_FILEDIALOG
   QString fn = QFileDialog::getOpenFileName( QString::null, QString::null, this );
   if ( !fn.isEmpty() )
     browser->setSource( fn );
#endif
}

void HelpWindow::newWindow()
{
   ( new HelpWindow(browser->source(), "qbrowser") )->show();
```

```
}

void HelpWindow::print()
{
#ifndef QT_NO_PRINTER
   QPrinter printer( QPrinter::HighResolution );
   printer.setFullPage(TRUE);
   if ( printer.setup( this ) ) {
    QPainter p( &printer );
    if( !p.isActive() ) // starting printing failed
       return;
    QPaintDeviceMetrics metrics(p.device());
    int dpiy = metrics.logicalDpiY();
    int margin = (int) ( (2/2.54)*dpiy ); // 2 cm margins
    QRect view( margin, margin, metrics.width() - 2*margin, metrics.height() - 2*margin );
    QSimpleRichText richText( browser->text(), QFont(),  browser->context(),
               browser->styleSheet(),  browser->mimeSourceFactory(), view.height() );
    richText.setWidth( &p, view.width() );
    int page = 1;
    do {
       richText.draw( &p, margin, margin, view, colorGroup() );
       view.moveBy( 0, view.height() );
       p.translate( 0 , -view.height() );
       p.drawText( view.right() - p.fontMetrics().width( QString::number(page) ),
            view.bottom() + p.fontMetrics().ascent() + 5, QString::number(page) );
       if ( view.top() - margin >= richText.height() )
        break;
       printer.newPage();
       page++;
    } while (TRUE);
   }
#endif
}

void HelpWindow::pathSelected( const QString &_path )
{
   browser->setSource( _path );
   if ( mHistory.values().contains(_path) )
    mHistory[ hist->insertItem( _path ) ] = _path;
}

void HelpWindow::readHistory()
{
   if ( QFile::exists( QDir::currentDirPath() + "/.history" ) ) {
    QFile f( QDir::currentDirPath() + "/.history" );
    f.open( IO_ReadOnly );
    QDataStream s( &f );
    s >> history;
    f.close();
    while ( history.count() > 20 )
       history.remove( history.begin() );
   }
}
```

```
void HelpWindow::readBookmarks()
{
   if ( QFile::exists( QDir::currentDirPath() + "/.bookmarks" ) ) {
    QFile f( QDir::currentDirPath() + "/.bookmarks" );
    f.open( IO_ReadOnly );
    QDataStream s( &f );
    s >> bookmarks;
    f.close();
   }
}

void HelpWindow::histChosen( int i )
{
   if ( mHistory.contains( i ) )
    browser->setSource( mHistory[ i ] );
}

void HelpWindow::bookmChosen( int i )
{
   if ( mBookmarks.contains( i ) )
    browser->setSource( mBookmarks[ i ] );
}

void HelpWindow::addBookmark()
{
   mBookmarks[ bookm->insertItem( caption() ) ] = browser->context();
}
```

helpwindow.h

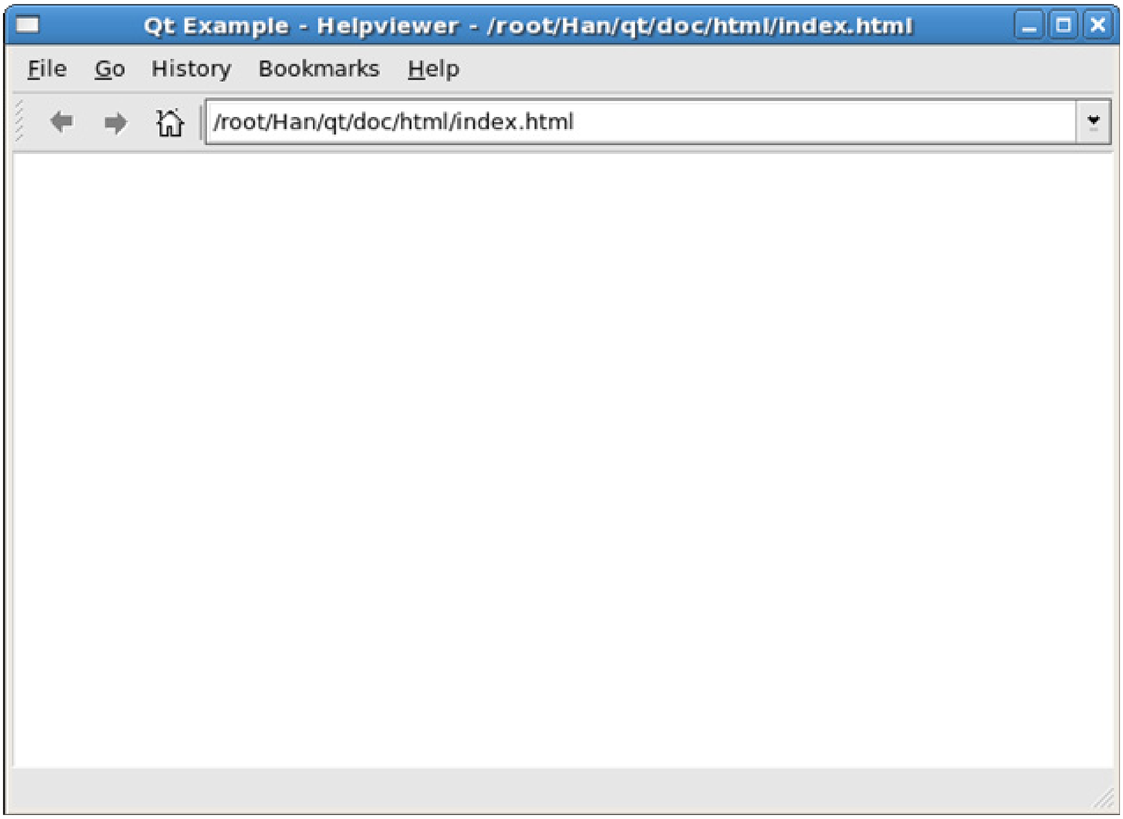
```
#ifndef HELPWINDOW_H
#define HELPWINDOW_H

#include <qmainwindow.h>
#include <qtextbrowser.h>
#include <qstringlist.h>
#include <qmap.h>
#include <qdir.h>

class QComboBox;
class QPopupMenu;

class HelpWindow : public QMainWindow
{
   Q_OBJECT
public:
   HelpWindow( const QString& home_,  const QString& path, QWidget* parent = 0, const char
*name=0 );
   ~HelpWindow();

private slots:
   void setBackwardAvailable( bool );
   void setForwardAvailable( bool );

   void sourceChanged( const QString& );
```

```
    void about();
    void aboutQt();
    void openFile();
    void newWindow();
    void print();

    void pathSelected( const QString & );
    void histChosen( int );
    void bookmChosen( int );
    void addBookmark();

private:
    void readHistory();
    void readBookmarks();

    QTextBrowser* browser;
    QComboBox *pathCombo;
    int backwardId, forwardId;
    QStringList history, bookmarks;
    QMap<int, QString> mHistory, mBookmarks;
    QPopupMenu *hist, *bookm;

};

#endif
```

main.cpp

```
#include "helpwindow.h"
#include <qapplication.h>
#include <qdir.h>
#include <stdlib.h>

int main( int argc, char ** argv )
{
    QApplication::setColorSpec( QApplication::ManyColor );
    QApplication a(argc, argv);

    QString home;
    if (argc > 1) {
        home = argv[1];
    } else {
      // Use a hard coded path. It is only an example.
      home = QDir( "../../doc/html/index.html" ).absPath();
    }

    HelpWindow *help = new HelpWindow(home, ".", 0, "help viewer");
    help->setCaption("Qt Example - Helpviewer");
    if ( QApplication::desktop()->width() > 400
       && QApplication::desktop()->height() > 500 )
      help->show();
    else
      help->showMaximized();

    QObject::connect( &a, SIGNAL(lastWindowClosed()),
                        &a, SLOT(quit()) );

    return a.exec();
}
```

실행



Qt Example - Helpviewer - /root/Han/qt/doc/html/index.html
File Go History Bookmarks Help
/root/Han/qt/doc/html/index.html

27. 국제화

이 실례는 응용프로그람들을 국제화하는 방법을 보여준다. 도이췰란드판을 얻으려면 다음과 같이 시작한다.

```
# i18n de
```

영문판을 얻으려면 다음과 같이 시작한다.

```
# i18n en
```

i18n.pro

```
TEMPLATE  = app
TARGET    = i18n
CONFIG    += qt warn_on release
HEADERS       = mywidget.h
SOURCES       = main.cpp \
          mywidget.cpp
TRANSLATIONS    = mywidget_cs.ts \
          mywidget_de.ts \
          mywidget_el.ts \
          mywidget_en.ts \
          mywidget_eo.ts \
          mywidget_fr.ts \
          mywidget_it.ts \
          mywidget_jp.ts \
          mywidget_ko.ts \
          mywidget_no.ts \
          mywidget_ru.ts \
          mywidget_zh.ts
```

mywidget.cpp

```
#include <qbuttongroup.h>
#include <qradiobutton.h>
#include <qlabel.h>
#include <qlistbox.h>
#include <qcombobox.h>
#include <qlabel.h>
#include <qhbox.h>
#include <qvbox.h>
#include <qaccel.h>
#include <qpopupmenu.h>
#include <qmenubar.h>
#include <qstatusbar.h>
#include <qapplication.h>

#include "mywidget.h"

MyWidget::MyWidget( QWidget* parent, const char* name )
     : QMainWindow( parent, name )
{
   QVBox* central = new QVBox(this);
   central->setMargin( 5 );
   central->setSpacing( 5 );
   setCentralWidget(central);

   QPopupMenu* file = new QPopupMenu(this);
   file->insertItem( tr("E&xit"), qApp, SLOT(quit()), QAccel::stringToKey(tr("Ctrl+Q")) );
   menuBar()->insertItem( tr("&File"), file );

   setCaption( tr( "Internationalization Example" ) );

   QString l;
   statusBar()->message( tr("Language: English") );

   ( void )new QLabel( tr( "The Main Window" ), central );

   QButtonGroup* gbox = new QButtonGroup( 1, QGroupBox::Horizontal,  tr( "View" ), central );
   (void)new QRadioButton( tr( "Perspective" ), gbox );
   (void)new QRadioButton( tr( "Isometric" ), gbox );
   (void)new QRadioButton( tr( "Oblique" ), gbox );

   initChoices(central);
}

static const char* choices[] = {
    QT_TRANSLATE_NOOP( "MyWidget", "First" ),
    QT_TRANSLATE_NOOP( "MyWidget", "Second" ),
    QT_TRANSLATE_NOOP( "MyWidget", "Third" ),
    0
};

void MyWidget::initChoices(QWidget* parent)
{
    QListBox* lb = new QListBox( parent );
    for ( int i = 0; choices[i]; i++ )
      lb->insertItem( tr( choices[i] ) );
}

void MyWidget::closeEvent(QCloseEvent* e)
{
    QWidget::closeEvent(e);
    emit closed();
}
```

mywidget.h

```
#ifndef MYWIDGET_H
#define MYWIDGET_H

#include <qmainwindow.h>
#include <qstring.h>

class MyWidget : public QMainWindow
{
    Q_OBJECT

public:
    MyWidget( QWidget* parent=0, const char* name = 0 );

signals:
    void closed();

protected:
    void closeEvent(QCloseEvent*);

private:
    static void initChoices(QWidget* parent);
};

#endif
```

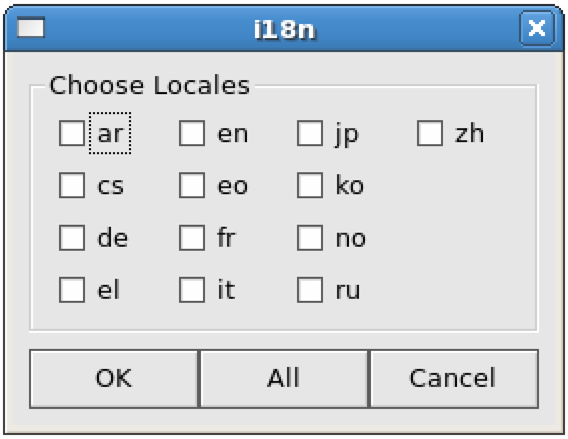
main.cpp

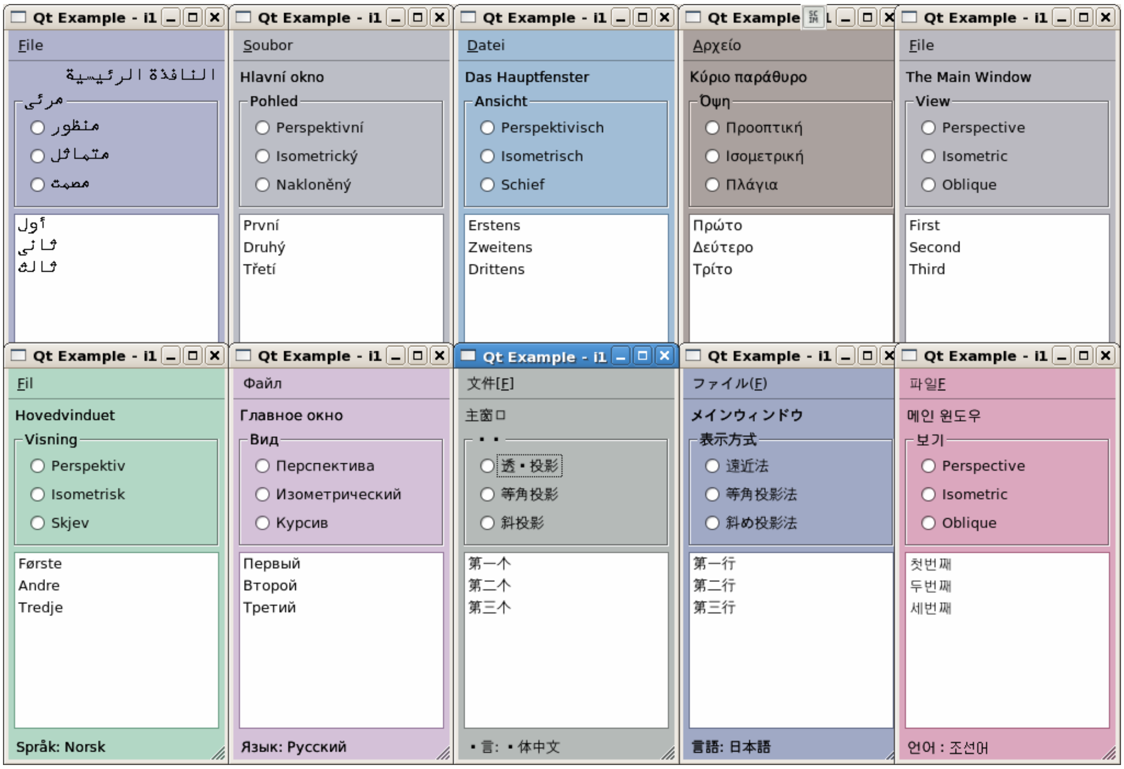
```
#include <qapplication.h>
#include <qtranslator.h>
#include <qfileinfo.h>
#include <qmessagebox.h>
#include <qcheckbox.h>
#include <qvbox.h>
#include <qlayout.h>
#include <qbuttongroup.h>
```

```
#include <qpushbutton.h>
#include <qsignalmapper.h>
#include <qtextcodec.h>
#include <stdlib.h>

#if defined(Q_OS_UNIX)
#include <unistd.h>
#endif

#include "mywidget.h"

//#define USE_I18N_FONT

class QVDialog : public QDialog {
public:
   QVDialog(QWidget *parent=0, const char *name=0, bool modal=FALSE,
          WFlags f=0) : QDialog(parent,name,modal,f)
   {
    QVBoxLayout* vb = new QVBoxLayout(this,8);
    vb->setAutoAdd(TRUE);
    hb = 0;
    sm = new QSignalMapper(this);
    connect(sm,SIGNAL(mapped(int)),this,SLOT(done(int)));
   }
   void addButtons( const QString& cancel=QString::null,    const QString& ok=QString::null,
            const QString& mid1=QString::null,    const QString& mid2=QString::null,
            const QString& mid3=QString::null)
   {
    addButton(ok.isNull() ? QObject::tr("OK") : ok, 1);
    if ( !mid1.isNull() ) addButton(mid1,2);
    if ( !mid2.isNull() ) addButton(mid2,3);
    if ( !mid3.isNull() ) addButton(mid3,4);
    addButton(cancel.isNull() ? QObject::tr("Cancel") : cancel, 0);
   }

   void addButton( const QString& text, int result )
   {
    if ( !hb )
       hb = new QHBox(this);
    QPushButton *c = new QPushButton(text, hb);
    sm->setMapping(c,result);
    connect(c,SIGNAL(clicked()),sm,SLOT(map()));
   }

private:
   QSignalMapper *sm;
   QHBox *hb;
};

MyWidget* showLang(QString lang)
{

   static QTranslator *translator = 0;
```

```
    qApp->setPalette(QPalette(QColor(220-rand()%64,220-rand()%64,220-rand()%64)));

    lang = "mywidget_" + lang + ".qm";
    QFileInfo fi( lang );

    if ( !fi.exists() ) {
      QMessageBox::warning( 0, "File error", QString("Cannot find translation for language: "+lang+
                          "\n(try eg. 'de', 'ko' or 'no')") );
      return 0;
    }
    if ( translator ) {
      qApp->removeTranslator( translator );
      delete translator;
    }
    translator = new QTranslator( 0 );
    translator->load( lang, "." );
    qApp->installTranslator( translator );
    MyWidget *m = new MyWidget;
    m->setCaption("Qt Example - i18n - " + m->caption() );
    return m;
}

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

    const char* qm[]= { "ar", "cs", "de", "el", "en", "eo", "fr", "it", "jp", "ko", "no", "ru", "zh", 0 };

#if defined(Q_OS_UNIX)
    srand( getpid() << 2 );
#endif

    QString lang;
    if ( argc == 2 )
        lang = argv[1];

    if ( argc != 2 || lang == "all" ) {
      QVDialog dlg(0,0,TRUE);
      QCheckBox* qmb[sizeof(qm)/sizeof(qm[0])];
      int r;
      if ( lang == "all" ) {
          r = 2;
      } else {
          QButtonGroup *bg = new QButtonGroup(4,Qt::Vertical,"Choose Locales",&dlg);
          QString loc = QTextCodec::locale();
          for ( int i=0; qm[i]; i++ ) {
            qmb[i] = new QCheckBox((const char*)qm[i],bg);
            qmb[i]->setChecked( loc == qm[i] );
          }
          dlg.addButtons("Cancel","OK","All");
          r = dlg.exec();
      }
      if ( r ) {
          QRect screen = qApp->desktop()->availableGeometry();
      bool tight = screen.width() < 1024;
      int x=screen.left()+5;
      int y=screen.top()+25;
      for ( int i=0; qm[i]; i++ ) {
       if ( r == 2 || qmb[i]->isChecked() ) {
         MyWidget* w = showLang((const char*)qm[i]);

         if( w == 0 ) exit( 0 );
         QObject::connect(w, SIGNAL(closed()), qApp, SLOT(quit()));
         w->setGeometry(x,y,197,356);
         w->show();
         if ( tight ) {
          x += 8;
          y += 8;
         } else {
          x += 205;
          if ( x > 1000 ) {
             x = 5;
             y += 384;
          }
         }
        }
       }
    } else {
        exit( 0 );
     }
   } else {
    QString lang = argv[1];
    QWidget* m = showLang(lang);
    app.setMainWidget( m );
    m->setCaption("Qt Example - i18n");
    m->show();
   }

#ifdef USE_I18N_FONT
   memorymanager->savePrerenderedFont(font.handle(),FALSE);
#endif

   // While we run "all", kill them all
   return app.exec();

}
```

실행





28. 그림기호보기

이 실례는 수많은 그림기호항목들을 보관할수 있는 유연한 그림기호보기를 실현한다. 이것은 끌기 및 놓기, 각이한 선택방식, 보기방식, 선택창선택 등을 유지한다.

iconview.pro

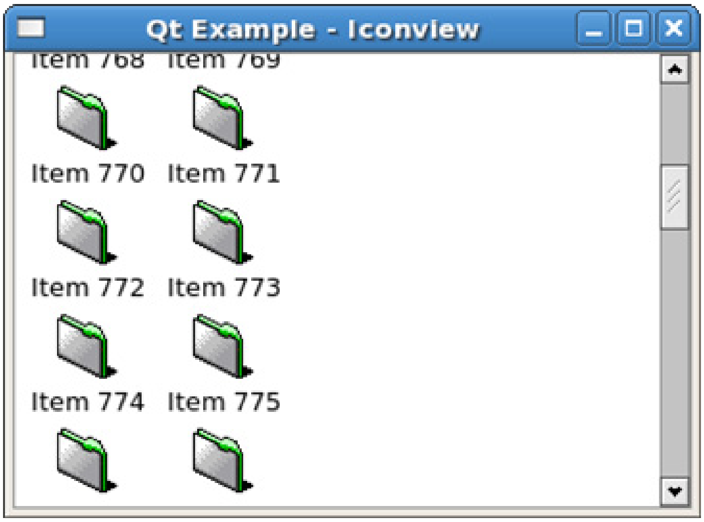
```
TEMPLATE  = app
TARGET    = iconview
CONFIG    += qt warn_on release
HEADERS       =
SOURCES       = main.cpp
```

main.cpp

```
#include <qiconview.h>
#include <qapplication.h>
#include <qdragobject.h>
#include <qpixmap.h>
#include <qiconset.h>

#include <qmime.h>
#include <stdio.h>

class ListenDND : public QObject
{
   Q_OBJECT

public:
   ListenDND( QWidget *w ) : view( w )
   {}

public slots:
   void dropped( QDropEvent *mime ) {
      qDebug( "Dropped Mimesource %p into the view %p", mime, view );
      qDebug( "  Formats:" );
      int i = 0;
      const char *str = mime->format( i );
      qDebug( "    %s", str );
      while ( str ) {
         qDebug( "    %s", str );
         str = mime->format( ++i );
      }
   };
   void moved() {
      qDebug( "All selected items were moved to another widget" );
   }

protected:
   QWidget *view;

};

int main( int argc, char **argv )
{
   QApplication a( argc, argv );

   QIconView qiconview;
   qiconview.setSelectionMode( QIconView::Extended );

   for ( unsigned int i = 0; i < 3000; i++ ) {
    QIconViewItem *item = new QIconViewItem( &qiconview, QString( "Item %1" ).arg( i + 1 ) );
    item->setRenameEnabled( TRUE );
   }

   qiconview.setCaption( "Qt Example - Iconview" );

   ListenDND listen_dnd( &qiconview );
```

```
    QObject::connect( &qiconview, SIGNAL( dropped( QDropEvent *,
            const QValueList<QIconDragItem> & ) ), &listen_dnd, SLOT( dropped( QDropEvent * ) ) );
    QObject::connect( &qiconview, SIGNAL( moved() ), &listen_dnd, SLOT( moved() ) );

    a.setMainWidget( &qiconview );
    qiconview.show();
    qiconview.resize( qiconview.sizeHint() );

    return a.exec();
}

#include "main.moc"
```

실행



29. 배치관리기

이 실례는 Qt의 배치클라스들인 QGridLayout, QBoxLayout 등의 간단하면서도 중급의 사용법을 보여준다.

layout.pro

```
TEMPLATE  = app
TARGET    = layout
CONFIG    += qt warn_on release
HEADERS       =
SOURCES       = layout.cpp
```

layout.cpp

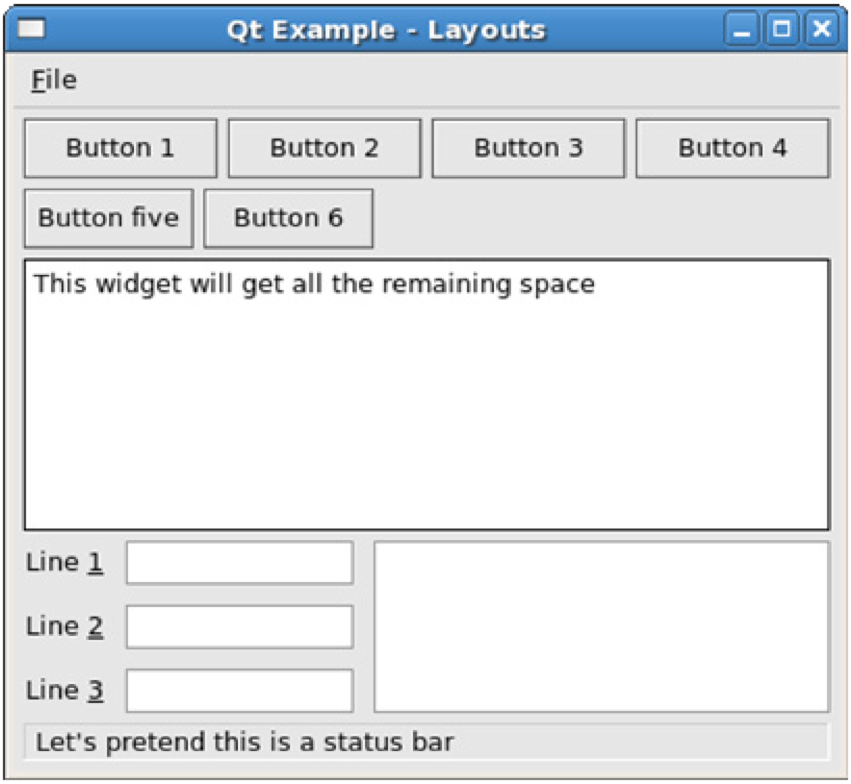
```
#include <qapplication.h>
#include <qlabel.h>
#include <qcolor.h>
#include <qpushbutton.h>
#include <qlayout.h>
#include <qlineedit.h>
#include <qmultilineedit.h>
#include <qmenubar.h>
#include <qpopupmenu.h>

class ExampleWidget : public QWidget
```

```
{
public:
   ExampleWidget( QWidget *parent = 0, const char *name = 0 );
   ~ExampleWidget();
};

ExampleWidget::ExampleWidget( QWidget *parent, const char *name ) : QWidget( parent, name )
{
   // Make the top-level layout; a vertical box to contain all widgets
   // and sub-layouts.
   QBoxLayout *topLayout = new QVBoxLayout( this, 5 );

   // Create a menubar...
   QMenuBar *menubar = new QMenuBar( this );
   menubar->setSeparator( QMenuBar::InWindowsStyle );
   QPopupMenu* popup;
   popup = new QPopupMenu( this );
   popup->insertItem( "&Quit", qApp, SLOT(quit()) );
   menubar->insertItem( "&File", popup );

   // ...and tell the layout about it.
   topLayout->setMenuBar( menubar );

   // Make an hbox that will hold a row of buttons.
   QBoxLayout *buttons = new QHBoxLayout( topLayout );
   int i;
   for ( i = 1; i <= 4; i++ ) {
    QPushButton* but = new QPushButton( this );
    QString s;
    s.sprintf( "Button %d", i );
    but->setText( s );

    // Set horizontal stretch factor to 10 to let the buttons
    // stretch horizontally. The buttons will not stretch
    // vertically, since bigWidget below will take up vertical stretch.
    buttons->addWidget( but, 10 );
    // (Actually, the result would have been the same with a
    // stretch factor of 0; if no items in a layout have non-zero
    // stretch, the space is divided equally between members.)
   }

   // Make another hbox that will hold a left-justified row of buttons.
   QBoxLayout *buttons2 = new QHBoxLayout( topLayout );

   QPushButton* but = new QPushButton( "Button five", this );
   buttons2->addWidget( but );

   but = new QPushButton( "Button 6", this );
   buttons2->addWidget( but );

   // Fill up the rest of the hbox with stretchable space, so that
   // the buttons get their minimum width and are pushed to the left.
   buttons2->addStretch( 10 );
```

```
// Make  a big widget that will grab all space in the middle.
QMultiLineEdit *bigWidget = new QMultiLineEdit( this );
bigWidget->setText( "This widget will get all the remaining space" );
bigWidget->setFrameStyle( QFrame::Panel | QFrame::Plain );

// Set vertical stretch factor to 10 to let the bigWidget stretch
// vertically. It will stretch horizontally because there are no
// widgets beside it to take up horizontal stretch.
//    topLayout->addWidget( bigWidget, 10 );
topLayout->addWidget( bigWidget );

// Make a grid that will hold a vertical table of QLabel/QLineEdit
// pairs next to a large QMultiLineEdit.

// Don't use hard-coded row/column numbers in QGridLayout, you'll
// regret it when you have to change the layout.
const int numRows = 3;
const int labelCol = 0;
const int linedCol = 1;
const int multiCol = 2;

// Let the grid-layout have a spacing of 10 pixels between
// widgets, overriding the default from topLayout.
QGridLayout *grid = new QGridLayout( topLayout, 0, 0, 10 );
int row;

for ( row = 0; row < numRows; row++ ) {
  QLineEdit *ed = new QLineEdit( this );
  // The line edit goes in the second column
  grid->addWidget( ed, row, linedCol );

  // Make a label that is a buddy of the line edit
  QString s;
  s.sprintf( "Line &%d", row+1 );
  QLabel *label = new QLabel( ed, s, this );
  // The label goes in the first column.
  grid->addWidget( label, row, labelCol );
}

// The multiline edit will cover the entire vertical range of the
// grid (rows 0 to numRows) and stay in column 2.

QMultiLineEdit *med = new QMultiLineEdit( this );
grid->addMultiCellWidget( med, 0, -1, multiCol, multiCol );

// The labels will take the space they need. Let the remaining
// horizontal space be shared so that the multiline edit gets
// twice as much as the line edit.
grid->setColStretch( linedCol, 10 );
grid->setColStretch( multiCol, 20 );

// Add a widget at the bottom.
QLabel* sb = new QLabel( this );
sb->setText( "Let's pretend this is a status bar" );
```

```
    sb->setFrameStyle( QFrame::Panel | QFrame::Sunken );
    // This widget will use all horizontal space, and have a fixed height.

    // we should have made a subclass and implemented sizePolicy there...
    sb->setFixedHeight( sb->sizeHint().height() );

    sb->setAlignment( AlignVCenter | AlignLeft );
    topLayout->addWidget( sb );

    topLayout->activate();
}

ExampleWidget::~ExampleWidget()
{
    // All child widgets are deleted by Qt.
    // The top-level layout and all its sub-layouts are deleted by Qt.
}

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    ExampleWidget f;
    a.setMainWidget(&f);
    f.setCaption("Qt Example - Layouts");
    f.show();

    return a.exec();
}
```

실행



This widget will get all the remaining space

Let's pretend this is a status bar

30. Conway 의 생명게임

life.pro

```
TEMPLATE  = app
TARGET     = life
CONFIG     += qt warn_on release
HEADERS         = life.h \
          lifedlg.h
SOURCES         = life.cpp \
          lifedlg.cpp \
          main.cpp
```

life.cpp

```
#include "life.h"
#include <qpainter.h>
#include <qdrawutil.h>
#include <qcheckbox.h>
#include <qevent.h>
#include <qapplication.h>

// The main game of life widget

LifeWidget::LifeWidget( int s, QWidget *parent, const char *name )  : QFrame( parent, name )
{
   SCALE = s;
   maxi = maxj = 50;
   setMinimumSize( MINSIZE * SCALE + 2 * BORDER,  MINSIZE * SCALE + 2 * BORDER );
   setMaximumSize( MAXSIZE * SCALE + 2 * BORDER,  MAXSIZE * SCALE + 2 * BORDER );
   setSizeIncrement( SCALE, SCALE);

   clear();
   resize( maxi * SCALE + 2 * BORDER , maxj * SCALE + 2 * BORDER );

}

void LifeWidget::clear()
{
   current = 0;
   for ( int t = 0; t < 2; t++ )
    for ( int i = 0; i < MAXSIZE + 2; i++ )
      for ( int j = 0; j < MAXSIZE + 2; j++ )
       cells[t][i][j] = FALSE;

   repaint();
}

// We assume that the size will never be beyond the maximum size set
// this is not in general TRUE, but in practice it's good enough for this program

void LifeWidget::resizeEvent( QResizeEvent * e )
{
   maxi = (e->size().width()  - 2 * BORDER) / SCALE;
   maxj = (e->size().height() - 2 * BORDER) / SCALE;
}

void LifeWidget::setPoint( int i, int j )
{
   if ( i < 1 || i > maxi || j < 1 || j > maxi )
    return;
   cells[current][i][j] = TRUE;
   repaint( index2pos(i), index2pos(j), SCALE, SCALE, FALSE );
}

void LifeWidget::mouseHandle( const QPoint &pos )
{
   int i = pos2index( pos.x() );
   int j = pos2index( pos.y() );
   setPoint( i, j );
}

void LifeWidget::mouseMoveEvent( QMouseEvent *e )
{
   mouseHandle( e->pos() );
}

void LifeWidget::mousePressEvent( QMouseEvent *e )
{
   if ( e->button() == QMouseEvent::LeftButton )
    mouseHandle( e->pos() );
```

```
}

void LifeWidget::nextGeneration()
{
   for ( int i = 1; i <= MAXSIZE; i++ ) {
    for ( int j = 1; j <= MAXSIZE; j++ ) {
       int t = cells[current][i - 1][j - 1]
       + cells[current][i - 1][j]
       + cells[current][i - 1][j + 1]
       + cells[current][i][j - 1]
       + cells[current][i][j + 1]
       + cells[current][i + 1][j - 1]
       + cells[current][i + 1][j]
       + cells[current][i + 1][j + 1];

       cells[!current][i][j] = ( t == 3 ||
                    t == 2 && cells[current][i][j] );
    }
   }
   current = !current;
   repaint( FALSE );      // repaint without erase
}

void LifeWidget::paintEvent( QPaintEvent * e )
{
   int starti = pos2index( e->rect().left() );
   int stopi  = pos2index( e->rect().right() );
   int startj = pos2index( e->rect().top() );
   int stopj  = pos2index( e->rect().bottom() );

   if (stopi > maxi)
    stopi = maxi;
   if (stopj > maxj)
    stopj = maxj;

   QPainter paint( this );
   for ( int i = starti; i <= stopi; i++ ) {
    for ( int j = startj; j <= stopj; j++ ) {
       if ( cells[current][i][j] )
        qDrawShadePanel( &paint, index2pos( i ), index2pos( j ),
                SCALE - 1, SCALE - 1, colorGroup() );
       else if ( cells[!current][i][j] )
        erase(index2pos( i ), index2pos( j ), SCALE - 1, SCALE - 1);
    }
   }
   drawFrame( &paint );
}
```

life.h

```
#ifndef LIFE_H
#define LIFE_H

#include <qframe.h>
```

```
class LifeWidget : public QFrame
{
   Q_OBJECT
public:
   LifeWidget( int s = 10, QWidget *parent = 0, const char *name = 0 );

   voidsetPoint( int i, int j );

   int        maxCol() { return maxi; }
   int        maxRow() { return maxj; }

public slots:
   voidnextGeneration();
   voidclear();

protected:
   virtual void paintEvent( QPaintEvent * );
   virtual void mouseMoveEvent( QMouseEvent * );
   virtual void mousePressEvent( QMouseEvent * );
   virtual void resizeEvent( QResizeEvent * );
   void mouseHandle( const QPoint &pos );

private:
   enum { MAXSIZE = 50, MINSIZE = 10, BORDER = 5 };

   boolcells[2][MAXSIZE + 2][MAXSIZE + 2];
   int        current;
   int        maxi, maxj;

   int pos2index( int x )
   {
    return ( x - BORDER ) / SCALE + 1;
   }
   int index2pos( int i )
   {
    return  ( i - 1 ) * SCALE + BORDER;
   }

   int SCALE;
};

#endif // LIFE_H
```

lifedlg.cpp

```
#include "lifedlg.h"
#include <qapplication.h>
#include <qpushbutton.h>
#include <qlabel.h>
#include <qslider.h>
#include <qcombobox.h>
#include <qdatetime.h>
#include <stdlib.h>
#include "patterns.cpp"
```

```
// A simple timer which has a pause and a setSpeed slot

LifeTimer::LifeTimer( QWidget *parent ) : QTimer( parent ), interval( 500 )
{
   start( interval );
}

void LifeTimer::pause( bool stopIt )
{
   if ( stopIt )
    stop();
   else
    start( interval );
}

void LifeTimer::setSpeed( int speed )
{
   interval = MAXSPEED - speed;
   if ( isActive() )
    changeInterval( interval );
}

// A top-level container widget to organize the others

LifeDialog::LifeDialog( int scale, QWidget * parent, const char * name )
   : QWidget( parent, name )
{
   qb = new QPushButton( "Quit!", this );
   cb = new QComboBox( this, "comboBox" );
   life = new LifeWidget(scale, this);
   life->move( SIDEBORDER, TOPBORDER );

   connect( qb, SIGNAL(clicked()), qApp, SLOT(quit()) );
   qb->setGeometry( SIDEBORDER, SIDEBORDER, qb->sizeHint().width(), 25 );
   timer = new LifeTimer( this );

   connect( timer, SIGNAL(timeout()), life, SLOT(nextGeneration()) );
   pb = new QPushButton( "Pause", this );
   pb->setToggleButton( TRUE );
   connect( pb, SIGNAL(toggled(bool)), timer, SLOT(pause(bool)) );
   pb->resize( pb->sizeHint().width(), 25 );
   pb->move( width() - SIDEBORDER - pb->width(), SIDEBORDER );

   sp = new QLabel( "Speed:", this );
   sp->adjustSize();
   sp->move( SIDEBORDER, 45 );
   scroll = new QSlider( 0, LifeTimer::MAXSPEED, 50,
                 LifeTimer::MAXSPEED / 2,
                 QSlider::Horizontal, this );
   connect( scroll, SIGNAL(valueChanged(int)),
        timer,  SLOT(setSpeed(int)) );

   scroll->move( sp->width() + 2 * SIDEBORDER, 45 );
   scroll->resize( 200, 15 );
```

```
    life->setFrameStyle( QFrame::Panel | QFrame::Sunken );
    life->show();

    srand( QTime(0,0,0).msecsTo(QTime::currentTime()) );
    int sel =  rand() % NPATS;
    getPattern( sel );

    cb->move( 2*SIDEBORDER + qb->width(), SIDEBORDER);
    cb->insertItem( "Glider Gun " );
    cb->insertItem( "Figure Eight " );
    cb->insertItem( "Pulsar " );
    cb->insertItem( "Barber Pole P2 " );
    cb->insertItem( "Achim P5 " );
    cb->insertItem( "Hertz P4 " );
    cb->insertItem( "Tumbler " );
    cb->insertItem( "Pulse1 P4" );
    cb->insertItem( "Shining Flower P5 " );
    cb->insertItem( "Pulse2 P6 " );
    cb->insertItem( "Pinwheel, Clock P4 " );
    cb->insertItem( "Pentadecatholon " );
    cb->insertItem( "Piston " );
    cb->insertItem( "Piston2 " );
    cb->insertItem( "Switch Engine " );
    cb->insertItem( "Gears (Gear, Flywheel, Blinker) " );
    cb->insertItem( "Turbine8 " );
    cb->insertItem( "P16 " );
    cb->insertItem( "Puffer " );
    cb->insertItem( "Escort " );
    cb->insertItem( "Dart Speed 1/3 " );
    cb->insertItem( "Period 4 Speed 1/2 " );
    cb->insertItem( "Another Period 4 Speed 1/2 " );
    cb->insertItem( "Smallest Known Period 3 Spaceship Speed 1/3 " );
    cb->insertItem( "Turtle Speed 1/3 " );
    cb->insertItem( "Smallest Known Period 5 Speed 2/5 " );
    cb->insertItem( "Sym Puffer " );
    cb->insertItem( "], Near Ship, Pi Heptomino " );
    cb->insertItem( "R Pentomino " );
    cb->setAutoResize( FALSE );
    cb->setCurrentItem( sel );
    cb->show();
    connect( cb, SIGNAL(activated(int)), SLOT(getPattern(int)) );

    QSize s;
    s = life->minimumSize();
    setMinimumSize( s.width() + 2 * SIDEBORDER,  s.height() + TOPBORDER + SIDEBORDER );
    s = life->maximumSize();
    setMaximumSize( s.width() + 2 * SIDEBORDER,  s.height() + TOPBORDER + SIDEBORDER );
    s = life->sizeIncrement();
    setSizeIncrement( s.width(), s.height() );

    resize( QMIN(512, qApp->desktop()->width()),
        QMIN(480, qApp->desktop()->height()) );
}
```

```
void LifeDialog::resizeEvent( QResizeEvent * e )
{
   life->resize( e->size() - QSize( 2 * SIDEBORDER, TOPBORDER + SIDEBORDER ));
   pb->move( e->size().width() - SIDEBORDER - pb->width(), SIDEBORDER );
   scroll->resize( e->size().width() - sp->width() - 3 * SIDEBORDER,  scroll->height() );
   cb->resize( width() - 4*SIDEBORDER - qb->width() - pb->width()  , 25 );
}

// Adapted from xlock, see pattern.cpp for copyright info.

void LifeDialog::getPattern( int pat )
{
   life->clear();
   int i = pat % NPATS;
   int col;
   int * patptr = &patterns[i][0];
   while ( (col = *patptr++) != 127 ) {
    int row = *patptr++;
    col += life->maxCol() / 2;
    row += life->maxRow() / 2;
    life->setPoint( col, row );
   }
}
```

lifedlg.h

```
#ifndef LIFEDLG_H
#define LIFEDLG_H

#include <qtimer.h>
#include <qwidget.h>

class QSlider;
class QPushButton;
class QLabel;
class QComboBox;

#include "life.h"

class LifeTimer : public QTimer
{
   Q_OBJECT
public:
   LifeTimer( QWidget *parent );
   enum { MAXSPEED = 1000 };

public slots:
   voidsetSpeed( int speed );
   voidpause( bool );

private:
   int        interval;
};
```

```
class LifeDialog : public QWidget
{
   Q_OBJECT
public:
   LifeDialog( int scale = 10, QWidget *parent = 0, const char *name = 0 );
public slots:
   voidgetPattern( int );

protected:
   virtual void resizeEvent( QResizeEvent * e );

private:
   enum { TOPBORDER = 70, SIDEBORDER = 10 };

   LifeWidget*life;
   QPushButton *qb;
   LifeTimer   *timer;
   QPushButton *pb;
   QComboBox    *cb;
   QLabel *sp;
   QSlider*scroll;
};

#endif // LIFEDLG_H
```

patterns.cpp

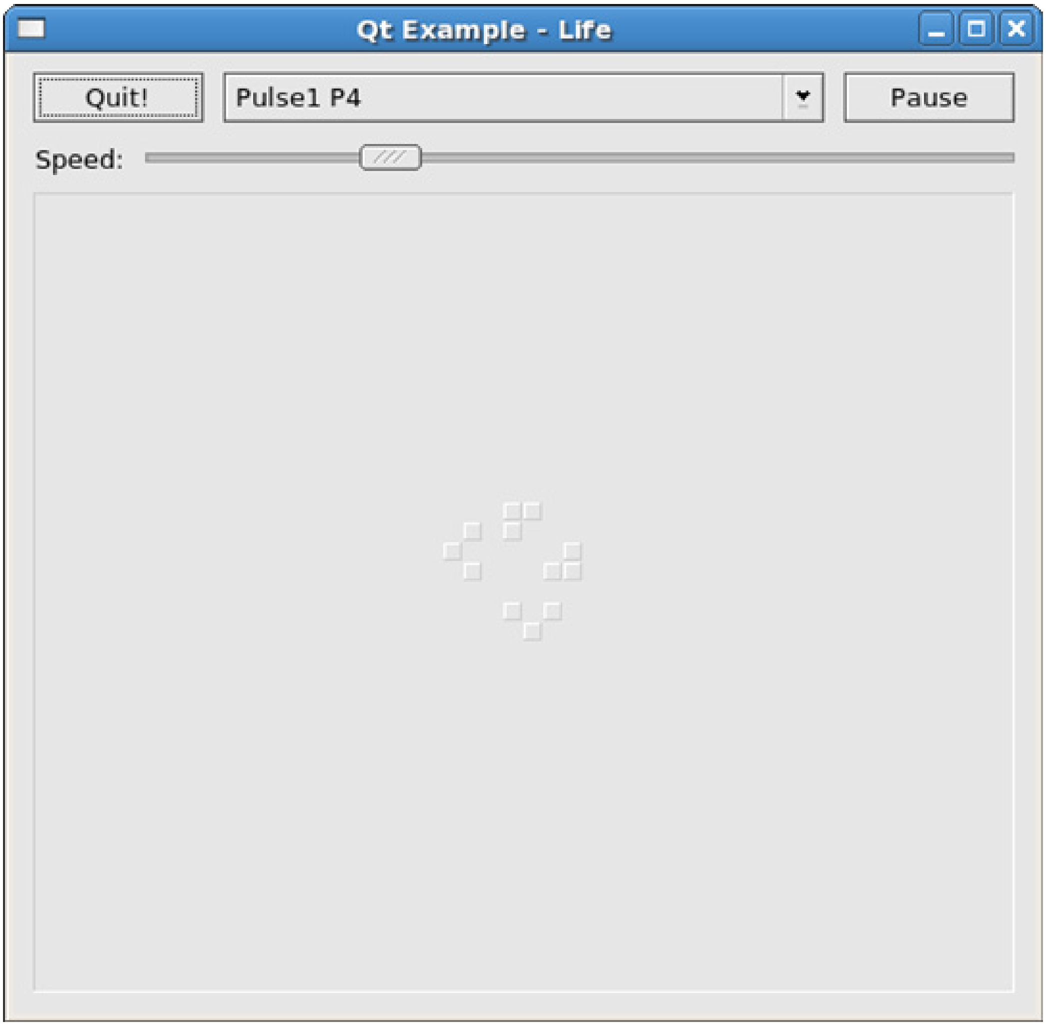
```
//#include <qglobal.h>
#define NUMPTS     63
/* Patterns have < NUMPTS pts (and should have a size of <= 32x32,
  the Gun is an exception) */
static int  patterns[][2 * NUMPTS + 1] = {
   {                  /* GLIDER GUN */
    6, -4,
    5, -3, 6, -3,
    -6, -2, -5, -2, 8, -2, 9, -2, 16, -2,
    -7, -1, 8, -1, 9, -1, 10, -1, 16, -1, 17, -1,
    -18, 0, -17, 0, -8, 0, 8, 0, 9, 1,
    -17, 1, -8, 1, 5, 1, 6, 1,
    -8, 2, 6, 2,
    -7, 3,
    -6, 4, -5, 4,
    127
   },
   {                  /* FIGURE EIGHT */
    -3, -3, -2, -3, -1, -3,
    -3, -2, -2, -2, -1, -2,
    -3, -1, -2, -1, -1, -1,
    0, 0, 1, 0, 2, 0,
    0, 1, 1, 1, 2, 1,
    0, 2, 1, 2, 2, 2,
    127
   },
   {                  /* PULSAR */
    -2, -1, -1, -1, 0, -1, 1, -1, 2, -1,
    -2, 0, 2, 0,
    127
   },
   {                    /* BARBER POLE P2 */
    -6, -6, -5, -6,
    -6, -5, -4, -5,
    -4, -3, -2, -3,
    -2, -1, 0, -1,
    0, 1, 2, 1,
    2, 3, 4, 3,
    5, 4,
    4, 5, 5, 5,
    127
   },
   {                    /* ACHIM P5 */
    -6, -6, -5, -6,
    -6, -5,
    -4, -4,
    -4, -3, -2, -3,
    -2, -1, 0, -1,
    0, 1, 2, 1,
    2, 3, 3, 3,
    5, 4,
    4, 5, 5, 5,
    127
   },
   {                    /* HERTZ P4 */
    -2, -5, -1, -5,
    -2, -4, -1, -4,
    -7, -2, -6, -2, -2, -2, -1, -2, 0, -2, 1, -2, 5, -2, 6, -2,
    -7, -1, -5, -1, -3, -1, 2, -1, 4, -1, 6, -1,
    -5, 0, -3, 0, -2, 0, 2, 0, 4, 0,
    -7, 1, -5, 1, -3, 1, 2, 1, 4, 1, 6, 1,
    -7, 2, -6, 2, -2, 2, -1, 2, 0, 2, 1, 2, 5, 2, 6, 2,
    -2, 4, -1, 4,
    -2, 5, -1, 5,
    127
   },
   {                    /* TUMBLER */
    -2, -3, -1, -3, 1, -3, 2, -3,
    -2, -2, -1, -2, 1, -2, 2, -2,
    -1, -1, 1, -1,
    -3, 0, -1, 0, 1, 0, 3, 0,
    -3, 1, -1, 1, 1, 1, 3, 1,
    -3, 2, -2, 2, 2, 2, 3, 2,
    127
   },
   {                    /* PULSE1 P4*/
    0, -3, 1, -3,
    -2, -2, 0, -2,
    -3, -1, 3, -1,
    -2, 0, 2, 0, 3, 0,
    0, 2, 2, 2,
    1, 3,
    127
   },
   {                    /* SHINING FLOWER P5 */
    -1, -4, 0, -4,
    -2, -3, 1, -3,
    -3, -2, 2, -2,
    -4, -1, 3, -1,
    -4, 0, 3, 0,
    -3, 1, 2, 1,
    -2, 2, 1, 2,
    -1, 3, 0, 3,
    127
   },
   {                    /* PULSE2 P6 */
    0, -4, 1, -4,
    -4, -3, -3, -3, -1, -3,
    -4, -2, -3, -2, 0, -2, 3, -2,
    1, -1, 3, -1,
    2, 0,
    1, 2, 2, 2,
    1, 3, 2, 3,
    127
   },
   {                    /* PINWHEEL, CLOCK P4 */
    -2, -6, -1, -6,
    -2, -5, -1, -5,
    -2, -3, -1, -3, 0, -3, 1, -3,
    -3, -2, -1, -2, 2, -2, 4, -2, 5, -2,
    -3, -1, 1, -1, 2, -1, 4, -1, 5, -1,
    -6, 0, -5, 0, -3, 0, 0, 0, 2, 0,
    -6, 1, -5, 1, -3, 1, 2, 1,
    -2, 2, -1, 2, 0, 2, 1, 2,
    0, 4, 1, 4,
    0, 5, 1, 5,
    127
   },
   {                    /* PENTADECATHOLON */
    -5, 0, -4, 0, -3, 0, -2, 0, -1, 0, 0, 0, 1, 0, 2, 0, 3, 0, 4, 0,
    127
   },
   {                    /* PISTON */
    1, -3, 2, -3,
    0, -2,
    -10, -1, -1, -1,
    -11, 0, -10, 0, -1, 0,   9, 0, 10, 0,
    -1, 1, 9, 1,
    0, 2,
    1, 3, 2, 3,
    127
   },
   {                    /* PISTON2 */
     -3, -5,
     -14, -4, -13, -4, -4, -4, -3, -4, 13, -4, 14, -4,
     -14, -3, -13, -3, -5, -3, -4, -3, 13, -3, 14, -3,
     -4, -2, -3, -2, 0, -2, 1, -2,
     -4,  2, -3, 2, 0, 2, 1, 2,
     -14, 3, -13, 3, -5, 3, -4, 3, 13, 3, 14, 3,
     -14, 4, -13, 4, -4, 4, -3, 4, 13, 4, 14, 4,
     -3, 5,
    127
   },
   {                    /* SWITCH ENGINE */
    -12, -3, -10, -3,
    -13, -2,
    -12, -1, -9, -1,
    -10, 0, -9,  0, -8,  0,
    13, 2, 14,  2,
    13, 3,
    127
   },
   {                    /* GEARS (gear, flywheel, blinker) */
    -1, -4,
    -1, -3, 1, -3,
    -3, -2,
    2, -1, 3, -1,
    -4, 0, -3, 0,
    2, 1,
    -2, 2, 0, 2,
    0, 3,

    5, 3,
    3, 4, 4, 4,
    5, 5, 6, 5,
    4, 6,

    8, 0,
    8, 1,
    8, 2,
    127
   },
   {                    /* TURBINE8 */
    -4, -4, -3, -4, -2, -4, -1, -4, 0, -4, 1, -4, 3, -4, 4, -4,
    -4, -3, -3, -3, -2, -3, -1, -3, 0, -3, 1, -3, 3, -3, 4, -3,
    3, -2, 4, -2,
    -4, -1, -3, -1, 3, -1, 4, -1,
    -4, 0, -3, 0, 3, 0, 4, 0,
    -4, 1, -3, 1, 3, 1, 4, 1,
    -4, 2, -3, 2,
    -4, 3, -3, 3, -1, 3, 0, 3, 1, 3, 2, 3, 3, 3, 4, 3,
    -4, 4, -3, 4, -1, 4, 0, 4, 1, 4, 2, 4, 3, 4, 4, 4,
    127
   },
   {                     /* P16 */
    -3, -6, 1, -6, 2, -6,
    -3, -5, 0, -5, 3, -5,
    3, -4,
    -5, -3, -4, -3, 1, -3, 2, -3, 5, -3, 6, -3,
    -6, -2, -3, -2,
    -6, -1, -3, -1,
    -5, 0, 5, 0,
    3, 1, 6, 1,
    3, 2, 6, 2,
    -6, 3, -5, 3, -2, 3, -1, 3, 4, 3, 5, 3,
    -3, 4,
    -3, 5, 0, 5, 3, 5,
    -2, 6, -1, 6, 3, 6,
    127
   },
   {                    /* PUFFER */
    1, -9,
    2, -8,
    -2, -7, 2, -7,
    -1, -6, 0, -6, 1, -6, 2, -6,
    -2, -2,
    -1, -1, 0, -1,
    0, 0,
    0, 1,
    -1, 2,
    1, 5,
    2, 6,
    -2, 7, 2, 7,
    -1, 8, 0, 8, 1, 8, 2, 8,
    127
   },
   {                    /* ESCORT */
    3, -8,
    4, -7,
    -2, -6, 4, -6,
    -1, -5, 0, -5, 1, -5, 2, -5, 3, -5, 4, -5,
    -5, -1, -4, -1, -3, -1, -2, -1, -1, -1, 0, -1,
    1, -1, 2, -1, 3, -1, 4, -1, 5, -1, 6, -1,
    -6, 0, 6, 0,
    6, 1,
    5, 2,
    3, 4,
    4, 5,
    -2, 6, 4, 6,
    -1, 7, 0, 7, 1, 7, 2, 7, 3, 7, 4, 7,
    127
   },
   {                    /* DART SPEED 1/3 */
    3, -7,
    2, -6, 4, -6,
    1, -5, 2, -5,
    4, -4,
    0, -3, 4, -3,
    -3, -2, 0, -2,
    -4, -1, -2, -1, 1, -1, 2, -1, 3, -1, 4, -1,
    -5, 0, -2, 0,
    -4, 1, -2, 1, 1, 1, 2, 1, 3, 1, 4, 1,
    -3, 2, 0, 2,
    0, 3, 4, 3,
    4, 4,
    1, 5, 2, 5,
    2, 6, 4, 6,
    3, 7,
    127
   },
   {                    /* PERIOD 4 SPEED 1/2 */
    -3, -5,
    -4, -4, -3, -4, -2, -4, -1, -4, 0, -4,
    -5, -3, -4, -3, 0, -3, 1, -3, 3, -3,
    -4, -2, 4, -2,
    -3, -1, -2, -1, 1, -1, 3, -1,
    -3, 1, -2, 1, 1, 1, 3, 1,
    -4, 2, 4, 2,
    -5, 3, -4, 3, 0, 3, 1, 3, 3, 3,
    -4, 4, -3, 4, -2, 4, -1, 4, 0, 4,
    -3, 5,
    127
   },
   {                    /* ANOTHER PERIOD 4 SPEED 1/2 */
    -4, -7, -3, -7, -1, -7, 0, -7, 1, -7, 2, -7, 3, -7, 4, -7,
    -5, -6, -4, -6, -3, -6, -2, -6, 5, -6,
    -6, -5, -5, -5,
    -5, -4, 5, -4,
    -4, -3, -3, -3, -2, -3, 0, -3,
    -2, -2,
    -2, -1,
    -1, 0,
    -2, 1,
    -2, 2,
    -4, 3, -3, 3, -2, 3, 0, 3,
    -5, 4, 5, 4,
    -6, 5, -5, 5,
    -5, 6, -4, 6, -3, 6, -2, 6, 5, 6,
    -4, 7, -3, 7, -1, 7, 0, 7, 1, 7, 2, 7, 3, 7, 4, 7,
    127
   },
   {                    /* SMALLEST KNOWN PERIOD 3 SPACESHIP SPEED 1/3 */
    0, -8,
    -1, -7, 1, -7,
    -1, -6, 1, -6,
    -1, -5,
    -2, -3, -1, -3,
    -1, -2, 1, -2,
    -2, -1, 0, -1,
    -2, 0, -1, 0, 0, 0,
    -1, 2, 1, 2,
    -1, 3, 0, 3,
    0, 4,
    0, 5, 2, 5,
    0, 6, 2, 6,
    1, 7,
    127
   },
       {                    /* TURTLE SPEED 1/3 */
        -4, -5, -3, -5, -2, -5, 6, -5,
        -4, -4, -3, -4, 0, -4, 2, -4, 3, -4, 5, -4, 6, -4,
        -2, -3, -1, -3, 0, -3, 5, -3,
        -4, -2, -1, -2, 1, -2, 5, -2,
        -5, -1, 0, -1, 5, -1,
        -5, 0, 0, 0, 5, 0,
        -4, 1, -1, 1, 1, 1, 5, 1,
        -2, 2, -1, 2, 0, 2, 5, 2,
        -4, 3, -3, 3, 0, 3, 2, 3, 3, 3, 5, 3, 6, 3,
        -4, 4, -3, 4, -2, 4, 6, 4,
        127
       },
       {                    /* SMALLEST KNOWN PERIOD 5 SPEED 2/5 */
        1, -7, 3, -7,
        -2, -6, 3, -6,
        -3, -5, -2, -5, -1, -5, 4, -5,
        -4, -4, -2, -4,
        -5, -3, -4, -3, -1, -3, 0, -3, 5, -3,
        -4, -2, -3, -2, 0, -2, 1, -2, 2, -2, 3, -2, 4, -2,
        -4, 2, -3, 2, 0, 2, 1, 2, 2, 2, 3, 2, 4, 2,
        -5, 3, -4, 3, -1, 3, 0, 3, 5, 3,
        -4, 4, -2, 4,
        -3, 5, -2, 5, -1, 5, 4, 5,
        -2, 6, 3, 6,
        1, 7, 3, 7,
        127
       },
       {                    /* SYM PUFFER */
        1, -4, 2, -4, 3, -4, 4, -4,
        0, -3, 4, -3,
        4, -2,
        -4, -1, -3, -1, 0, -1, 3, -1,
        -4, 0, -3, 0, -2, 0,
        -4, 1, -3, 1, 0, 1, 3, 1,
        4, 2,
        0, 3, 4, 3,
        1, 4, 2, 4, 3, 4, 4, 4,
        127
       },
       {                    /* ], NEAR SHIP, PI HEPTOMINO */
        -2, -1, -1, -1, 0, -1,
        1, 0,
        -2, 1, -1, 1, 0, 1,
        127
       },
       {                    /* R PENTOMINO */
        0, -1, 1, -1,
        -1, 0, 0, 0,
        0, 1,
        127
       }
   };
```

```
#define NPATS   (sizeof patterns / sizeof patterns[0])
```

main.cpp

```
#include "lifedlg.h"
#include <qapplication.h>
#include <stdlib.h>

void usage()
{
   qWarning( "Usage: life [-scale scale]" );
}

int main( int argc, char **argv )
{
   QApplication a( argc, argv );

   int scale = 10;

   for ( int i = 1; i < argc; i++ ){
      QString arg = argv[i];
    if ( arg == "-scale" )
       scale = atoi( argv[++i] );
    else {
       usage();
       exit(1);
    }
   }

   if ( scale < 2 )
    scale = 2;

   LifeDialog *life = new LifeDialog( scale );
   a.setMainWidget( life );
   life->setCaption("Qt Example - Life");
   life->show();

   return a.exec();
}
```

실행



31. 행편집

이 실례는 단일행편집창문부품과의 작업방법과 각이한 echo방식과 유효자(validator)들의 사용법을 설명한다.

lineedits.pro

```
TEMPLATE  = app
TARGET    = lineedits
CONFIG    += qt warn_on release
HEADERS       = lineedits.h
SOURCES       = lineedits.cpp \
        main.cpp
```

lineedits.cpp

```
#include "lineedits.h"
#include <qlineedit.h>
#include <qcombobox.h>
#include <qframe.h>
#include <qvalidator.h>
```

```
#include <qlabel.h>
#include <qlayout.h>
#include <qhbox.h>

/*
 * Constructor
* Creates child widgets of the LineEdits widget
 */

LineEdits::LineEdits( QWidget *parent, const char *name )
    : QGroupBox( 0, Horizontal, "Line edits", parent, name )
{
    setMargin( 10 );

    QVBoxLayout* box = new QVBoxLayout( layout() );

    QHBoxLayout *row1 = new QHBoxLayout( box );
    row1->setMargin( 5 );

    // Create a Label
    QLabel* label = new QLabel( "Echo Mode: ", this);
    row1->addWidget( label );

    // Create a Combobox with three items...
    combo1 = new QComboBox( FALSE, this );
    row1->addWidget( combo1 );
    combo1->insertItem( "Normal" );
    combo1->insertItem( "Password" );
    combo1->insertItem( "No Echo" );
    // ...and connect the activated() SIGNAL with the slotEchoChanged() SLOT to be able
    // to react when an item is selected
    connect( combo1, SIGNAL( activated( int ) ), this, SLOT( slotEchoChanged( int ) ) );

    // insert the first LineEdit
    lined1 = new QLineEdit( this );
    box->addWidget( lined1 );

    // another widget which is used for layouting
    QHBoxLayout *row2 = new QHBoxLayout( box );
    row2->setMargin( 5 );

    // and the second label
    label = new QLabel( "Validator: ", this );
    row2->addWidget( label );

    // A second Combobox with again three items...
    combo2 = new QComboBox( FALSE, this );
    row2->addWidget( combo2 );
    combo2->insertItem( "No Validator" );
    combo2->insertItem( "Integer Validator" );
    combo2->insertItem( "Double Validator" );
    // ...and again the activated() SIGNAL gets connected with a SLOT
    connect( combo2, SIGNAL( activated( int ) ), this, SLOT( slotValidatorChanged( int ) ) );
```

```
// and the second LineEdit
lined2 = new QLineEdit( this );
box->addWidget( lined2 );

// yet another widget which is used for layouting
QHBoxLayout *row3 = new QHBoxLayout( box );
row3->setMargin( 5 );

// we need a label for this too
label = new QLabel( "Alignment: ", this );
row3->addWidget( label );

// A combo box for setting alignment
combo3 = new QComboBox( FALSE, this );
row3->addWidget( combo3 );
combo3->insertItem( "Left" );
combo3->insertItem( "Centered" );
combo3->insertItem( "Right" );
// ...and again the activated() SIGNAL gets connected with a SLOT
connect( combo3, SIGNAL( activated( int ) ), this, SLOT( slotAlignmentChanged( int ) ) );

// and the third lineedit
lined3 = new QLineEdit( this );
box->addWidget( lined3 );

// exactly the same for the fourth
QHBoxLayout *row4 = new QHBoxLayout( box );
row4->setMargin( 5 );

// we need a label for this too
label = new QLabel( "Input mask: ", this );
row4->addWidget( label );

// A combo box for choosing an input mask
combo4 = new QComboBox( FALSE, this );
row4->addWidget( combo4 );
combo4->insertItem( "No mask" );
combo4->insertItem( "Phone number" );
combo4->insertItem( "ISO date" );
combo4->insertItem( "License key" );

// ...this time we use the activated( const QString & ) signal
connect( combo4, SIGNAL( activated( int ) ),
     this, SLOT( slotInputMaskChanged( int ) ) );

// and the fourth lineedit
lined4 = new QLineEdit( this );
box->addWidget( lined4 );

// last widget used for layouting
QHBox *row5 = new QHBox( this );
box->addWidget( row5 );
row5->setMargin( 5 );
```

```
    // last label
    (void)new QLabel( "Read-Only: ", row5 );

    // A combo box for setting alignment
    combo5 = new QComboBox( FALSE, row5 );
    combo5->insertItem( "False" );
    combo5->insertItem( "True" );
    // ...and again the activated() SIGNAL gets connected with a SLOT
    connect( combo5, SIGNAL( activated( int ) ), this, SLOT( slotReadOnlyChanged( int ) ) );

    // and the last lineedit
    lined5 = new QLineEdit( this );
    box->addWidget( lined5 );

    // give the first LineEdit the focus at the beginning
    lined1->setFocus();
}

/*
 * SLOT slotEchoChanged( int i )
* i contains the number of the item which the user has been chosen in the
 * first Combobox. According to this value, we set the Echo-Mode for the
 * first LineEdit.
 */

void LineEdits::slotEchoChanged( int i )
{
    switch ( i ) {
    case 0:
     lined1->setEchoMode( QLineEdit::Normal );
       break;
    case 1:
     lined1->setEchoMode( QLineEdit::Password );
       break;
    case 2:
     lined1->setEchoMode( QLineEdit::NoEcho );
       break;
    }

    lined1->setFocus();
}

/*
 * SLOT slotValidatorChanged( int i )
* i contains the number of the item which the user has been chosen in the
 * second Combobox. According to this value, we set a validator for the
 * second LineEdit. A validator checks in a LineEdit each character which
 * the user enters and accepts it if it is valid, else the character gets
 * ignored and not inserted into the lineedit.
 */

void LineEdits::slotValidatorChanged( int i )
{
    switch ( i ) {
```

```
    case 0:
      lined2->setValidator( 0 );
        break;
    case 1:
      lined2->setValidator( new QIntValidator( lined2 ) );
        break;
    case 2:
      lined2->setValidator( new QDoubleValidator( -999.0, 999.0, 2,
                                      lined2 ) );
        break;
    }

    lined2->setText( "" );
    lined2->setFocus();
}

/*
 * SLOT slotAlignmentChanged( int i )
* i contains the number of the item which the user has been chosen in
 * the third Combobox.  According to this value, we set an alignment third LineEdit.
 */

void LineEdits::slotAlignmentChanged( int i )
{
    switch ( i ) {
    case 0:
      lined3->setAlignment( QLineEdit::AlignLeft );
        break;
    case 1:
      lined3->setAlignment( QLineEdit::AlignCenter );
        break;
    case 2:
      lined3->setAlignment( QLineEdit::AlignRight );
        break;
    }

    lined3->setFocus();
}

/*
 * SLOT slotInputMaskChanged( const QString &mask )
* i contains the number of the item which the user has been chosen in
 * the third Combobox.  According to this value, we set an input mask on
 * third LineEdit.
 */

void LineEdits::slotInputMaskChanged( int i )
{
    switch( i ) {
    case 0:
      lined4->setInputMask( QString::null );
      break;
    case 1:
      lined4->setInputMask( "+99 99 99 99 99;_" );
```

```
     break;
   case 2:
     lined4->setInputMask( "0000-00-00" );
     lined4->setText( "00000000" );
     lined4->setCursorPosition( 0 );
     break;
   case 3:
     lined4->setInputMask( ">AAAAA-AAAAA-AAAAA-AAAAA-AAAAA;#" );
     break;
   }
   lined4->setFocus();
}

/*
 * SLOT slotReadOnlyChanged( int i )
* i contains the number of the item which the user has been chosen in
 * the fourth Combobox.  According to this value, we toggle read-only.
 */

void LineEdits::slotReadOnlyChanged( int i )
{
   switch ( i ) {
   case 0:
     lined5->setReadOnly( FALSE );
       break;
   case 1:
     lined5->setReadOnly( TRUE );
       break;
   }

   lined5->setFocus();
}
```

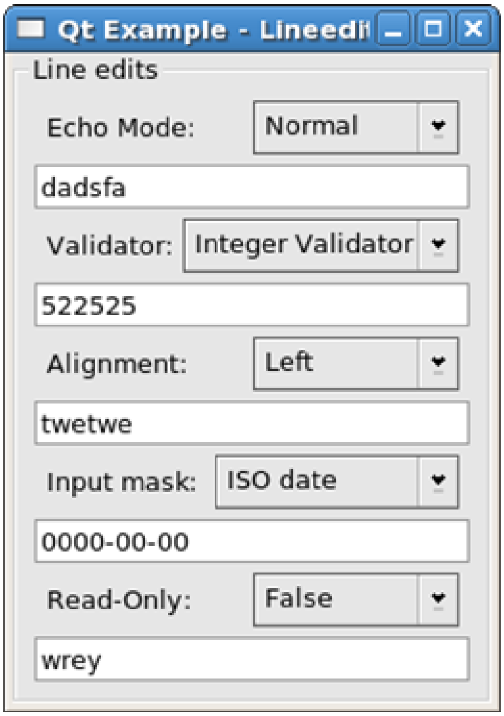
lineedits.h

```
#ifndef LINEDITS_H
#define LINEDITS_H

#include <qgroupbox.h>

class QLineEdit;
class QComboBox;

class LineEdits : public QGroupBox
{
   Q_OBJECT

public:
   LineEdits( QWidget *parent = 0, const char *name = 0 );

protected:
   QLineEdit *lined1, *lined2, *lined3, *lined4, *lined5;
   QComboBox *combo1, *combo2, *combo3, *combo4, *combo5;

protected slots:
   void slotEchoChanged( int );
   void slotValidatorChanged( int );
   void slotAlignmentChanged( int );
   void slotInputMaskChanged( int );
   void slotReadOnlyChanged( int );
};

#endif
```

main.cpp

```
#include "lineedits.h"
#include <qapplication.h>

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    LineEdits lineedits;
    lineedits.setCaption( "Qt Example - Lineedits" );
    a.setMainWidget( &lineedits );
    lineedits.show();

    return a.exec();
}
```

실행



Qt Example - Lineedi

Line edits

Echo Mode: Normal

dadsfa

Validator: Integer Validator

522525

Alignment: Left

twetwe

Input mask: ISO date

0000-00-00

Read-Only: False

wrey

32. 목록칸실례

이 실례는 QListBox 의 각이한 방식(단일칸, 여러칸, 고정개수행 등)의 사용법을 보여준다.

listbox.pro

```
TEMPLATE  = app
TARGET      = listbox
CONFIG       += qt warn_on release
HEADERS         = listbox.h
SOURCES         = listbox.cpp \
            main.cpp
```

listbox.cpp

```
#include "listbox.h"
#include <qlabel.h>
#include <qradiobutton.h>
#include <qcheckbox.h>
#include <qspinbox.h>
#include <qlistbox.h>
#include <qbuttongroup.h>
#include <qlayout.h>
#include <qpushbutton.h>

ListBoxDemo::ListBoxDemo()  : QWidget( 0, 0 )
{
   QGridLayout * g = new QGridLayout( this, 2, 2, 6 );

   g->addWidget( new QLabel( "<b>Configuration:</b>", this ), 0, 0 );
   g->addWidget( new QLabel( "<b>Result:</b>", this ), 0, 1 );

   l = new QListBox( this );
   g->addWidget( l, 1, 1 );
   l->setFocusPolicy( QWidget::StrongFocus );

   QVBoxLayout * v = new QVBoxLayout;
   g->addLayout( v, 1, 0 );

   QRadioButton * b;
   bg = new QButtonGroup( 0 );

   b = new QRadioButton( "Fixed number of columns,\n"
                  "as many rows as needed.",
                  this );
   bg->insert( b );
   v->addWidget( b );
   b->setChecked( TRUE );
   connect( b, SIGNAL(clicked()), this, SLOT(setNumCols()) );
   QHBoxLayout * h = new QHBoxLayout;
   v->addLayout( h );
   h->addSpacing( 30 );
   h->addSpacing( 100 );
   h->addWidget( new QLabel( "Columns:", this ) );
   columns = new QSpinBox( this );
   h->addWidget( columns );

   v->addSpacing( 12 );

   b = new QRadioButton( "As many columns as fit on-screen,\n"
                  "as many rows as needed.",
                  this );
   bg->insert( b );
   v->addWidget( b );
   connect( b, SIGNAL(clicked()), this, SLOT(setColsByWidth()) );

   v->addSpacing( 12 );

   b = new QRadioButton( "Fixed number of rows,\n"
                  "as many columns as needed.",
                  this );
   bg->insert( b );
   v->addWidget( b );
   connect( b, SIGNAL(clicked()), this, SLOT(setNumRows()) );
   h = new QHBoxLayout;
   v->addLayout( h );
   h->addSpacing( 30 );
   h->addSpacing( 100 );
   h->addWidget( new QLabel( "Rows:", this ) );
   rows = new QSpinBox( this );
   rows->setEnabled( FALSE );
   h->addWidget( rows );

   v->addSpacing( 12 );

   b = new QRadioButton( "As many rows as fit on-screen,\n"
                  "as many columns as needed.",  this );
   bg->insert( b );
   v->addWidget( b );
   connect( b, SIGNAL(clicked()), this, SLOT(setRowsByHeight()) );

   v->addSpacing( 12 );

   QCheckBox * cb = new QCheckBox( "Variable-height rows", this );
   cb->setChecked( TRUE );
   connect( cb, SIGNAL(toggled(bool)), this, SLOT(setVariableHeight(bool)) );
   v->addWidget( cb );
   v->addSpacing( 6 );

   cb = new QCheckBox( "Variable-width columns", this );
   connect( cb, SIGNAL(toggled(bool)), this, SLOT(setVariableWidth(bool)) );
   v->addWidget( cb );

   cb = new QCheckBox( "Extended-Selection", this );
   connect( cb, SIGNAL(toggled(bool)), this, SLOT(setMultiSelection(bool)) );
   v->addWidget( cb );

   QPushButton *pb = new QPushButton( "Sort ascending", this );
   connect( pb, SIGNAL( clicked() ), this, SLOT( sortAscending() ) );
   v->addWidget( pb );

   pb = new QPushButton( "Sort descending", this );
   connect( pb, SIGNAL( clicked() ), this, SLOT( sortDescending() ) );
   v->addWidget( pb );
```

```
    v->addStretch( 100 );

    int i = 0;
    while( ++i <= 2560 )
        l->insertItem( QString::fromLatin1( "Item " ) + QString::number( i ),  i );
    columns->setRange( 1, 256 );
    columns->setValue( 1 );
    rows->setRange( 1, 256 );
    rows->setValue( 256 );

    connect( columns, SIGNAL(valueChanged(int)), this, SLOT(setNumCols()) );
    connect( rows, SIGNAL(valueChanged(int)), this, SLOT(setNumRows()) );
}

ListBoxDemo::~ListBoxDemo()
{
    delete bg;
}

void ListBoxDemo::setNumRows()
{
    columns->setEnabled( FALSE );
    rows->setEnabled( TRUE );
    l->setRowMode( rows->value() );
}

void ListBoxDemo::setNumCols()
{
    columns->setEnabled( TRUE );
    rows->setEnabled( FALSE );
    l->setColumnMode( columns->value() );
}

void ListBoxDemo::setRowsByHeight()
{
    columns->setEnabled( FALSE );
    rows->setEnabled( FALSE );
    l->setRowMode( QListBox::FitToHeight );
}

void ListBoxDemo::setColsByWidth()
{
    columns->setEnabled( FALSE );
    rows->setEnabled( FALSE );
    l->setColumnMode( QListBox::FitToWidth );
}

void ListBoxDemo::setVariableWidth( bool b )
{
    l->setVariableWidth( b );
}

void ListBoxDemo::setVariableHeight( bool b )
```

```
{
   l->setVariableHeight( b );
}

void ListBoxDemo::setMultiSelection( bool b )
{
   l->clearSelection();
   l->setSelectionMode( b ? QListBox::Extended : QListBox::Single );
}

void ListBoxDemo::sortAscending()
{
   l->sort( TRUE );
}

void ListBoxDemo::sortDescending()
{
   l->sort( FALSE );
}
```

listbox.h

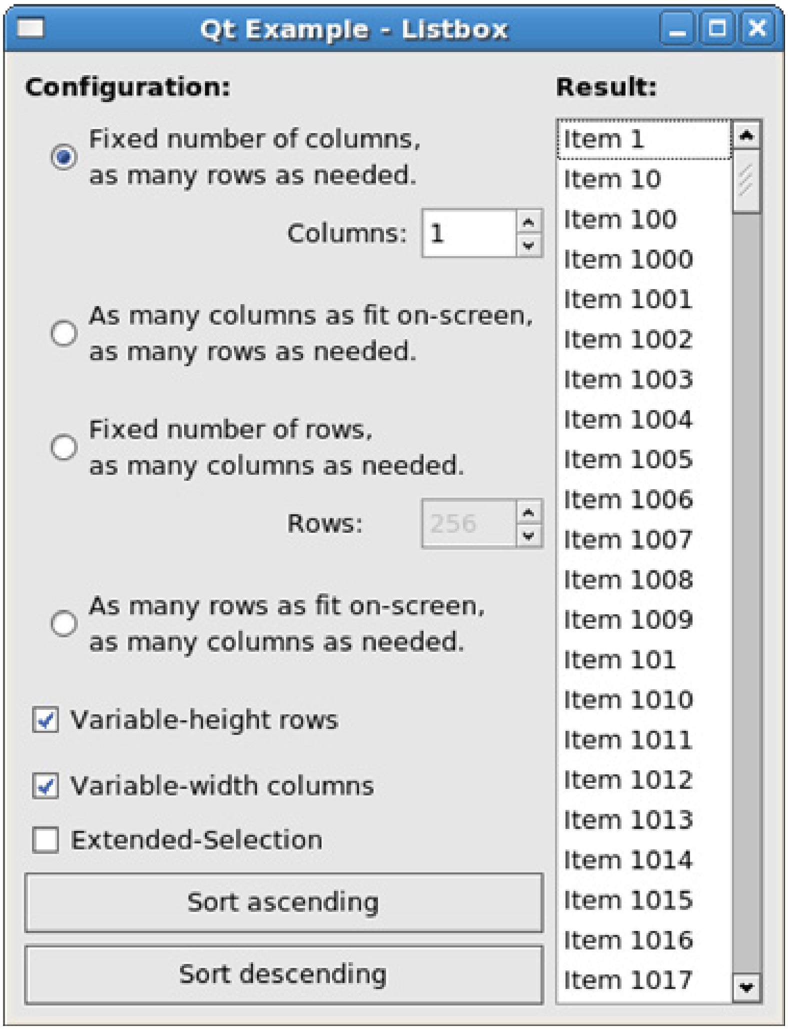
```
#ifndef LISTBOX_H
#define LISTBOX_H

class QSpinBox;
class QListBox;
class QButtonGroup;

#include <qwidget.h>

class ListBoxDemo: public QWidget
{
   Q_OBJECT
public:
   ListBoxDemo();
   ~ListBoxDemo();

private slots:
   void setNumRows();
   void setNumCols();
   void setRowsByHeight();
   void setColsByWidth();
   void setVariableWidth( bool );
   void setVariableHeight( bool );
   void setMultiSelection( bool );
   void sortAscending();
   void sortDescending();

private:
   QListBox * l;
   QSpinBox * columns;
   QSpinBox * rows;
   QButtonGroup * bg;
};
```

```
#endif
```

main.cpp

```
#include "listbox.h"
#include <qapplication.h>

int main( int argc, char **argv )
{
   QApplication a( argc, argv );

   ListBoxDemo t;
   t.setCaption( "Qt Example - Listbox" );
   a.setMainWidget( &t );
   t.show();

   return a.exec();
}
```

실행



Configuration:

● Fixed number of columns, as many rows as needed.

Columns: 1

○ As many columns as fit on-screen, as many rows as needed.

○ Fixed number of rows, as many columns as needed.

Rows: 256

○ As many rows as fit on-screen, as many columns as needed.

☑ Variable-height rows

☑ Variable-width columns

☐ Extended-Selection

Result:

```
Item 1
Item 10
Item 100
Item 1000
Item 1001
Item 1002
Item 1003
Item 1004
Item 1005
Item 1006
Item 1007
Item 1008
Item 1009
Item 101
Item 1010
Item 1011
Item 1012
Item 1013
Item 1014
Item 1015
Item 1016
Item 1017
```

33. 목록칸과 복합칸

이 실례프로그람은 목록칸(단일선택 및 여러선택)과 복합칸(편집가능 및 비편집가능)의 사용법을 보여준다.

listboxcombo.pro

```
TEMPLATE  = app
TARGET    = listboxcombo
CONFIG    += qt warn_on release
HEADERS      = listboxcombo.h
SOURCES      = listboxcombo.cpp \
        main.cpp
```

listboxcombo.cpp

```
#include "listboxcombo.h"
#include <qcombobox.h>
#include <qlistbox.h>
#include <qhbox.h>
#include <qpushbutton.h>
#include <qstring.h>
#include <qpixmap.h>
#include <qlabel.h>
#include <qimage.h>
#include <qpainter.h>
#include <qstyle.h>

class MyListBoxItem : public QListBoxItem
{
public:
   MyListBoxItem()
     : QListBoxItem()
   {
     setCustomHighlighting( TRUE );
   }

protected:
   virtual void paint( QPainter * );
   virtual int width( const QListBox* ) const { return 100; }
   virtual int height( const QListBox* ) const { return 16; }

};

void MyListBoxItem::paint( QPainter *painter )
{
   // evil trick: find out whether we are painted onto our listbox
   bool in_list_box = listBox() && listBox()->viewport() == painter->device();

   QRect r ( 0, 0, width( listBox() ), height( listBox() ) );
   if ( in_list_box && isSelected() )
     painter->eraseRect( r );
   painter->fillRect( 5, 5, width( listBox() ) - 10, height( listBox() ) - 10, Qt::red );
   if ( in_list_box && isCurrent() )
     listBox()->style().drawPrimitive( QStyle::PE_FocusRect, painter, r, listBox()->colorGroup() );
```

```
}

/*
 * Constructor
* Creates child widgets of the ListBoxCombo widget
 */

ListBoxCombo::ListBoxCombo( QWidget *parent, const char *name )
   : QVBox( parent, name )
{
   setMargin( 5 );
   setSpacing( 5 );

   unsigned int i;
   QString str;

   QHBox *row1 = new QHBox( this );
   row1->setSpacing( 5 );

   // Create a multi-selection ListBox...
   lb1 = new QListBox( row1 );
   lb1->setSelectionMode( QListBox::Multi );

   // ...insert a pixmap item...
   lb1->insertItem( QPixmap( "qtlogo.png" ) );
   // ...and 100 text items
   for ( i = 0; i < 100; i++ ) {
    str = QString( "Listbox Item %1" ).arg( i );
    if ( !( i % 4 ) )
       lb1->insertItem( QPixmap( "fileopen.xpm" ), str );
    else
       lb1->insertItem( str );
   }

   // Create a pushbutton...
   QPushButton *arrow1 = new QPushButton( " -> ", row1 );
   // ...and connect the clicked SIGNAL with the SLOT slotLeft2Right
   connect( arrow1, SIGNAL( clicked() ), this, SLOT( slotLeft2Right() ) );

   // create an empty single-selection ListBox
   lb2 = new QListBox( row1 );

   QHBox *row2 = new QHBox( this );
   row2->setSpacing( 5 );

   QVBox *box1 = new QVBox( row2 );
   box1->setSpacing( 5 );

   // Create a non-editable Combobox and a label below...
   QComboBox *cb1 = new QComboBox( FALSE, box1 );
   label1 = new QLabel( "Current Item: Combobox Item 0", box1 );
   label1->setMaximumHeight( label1->sizeHint().height() * 2 );
   label1->setFrameStyle( QFrame::Panel | QFrame::Sunken );
   //...and insert 50 items into the Combobox
   for ( i = 0; i < 50; i++ ) {
    str = QString( "Combobox Item %1" ).arg( i );
    if ( i % 9 )
       cb1->insertItem( str );
    else
       cb1->listBox()->insertItem( new MyListBoxItem );
   }

   QVBox *box2 = new QVBox( row2 );
   box2->setSpacing( 5 );

   // Create an editable Combobox and a label below...
   QComboBox *cb2 = new QComboBox( TRUE, box2 );
   label2 = new QLabel( "Current Item: Combobox Item 0", box2 );
   label2->setMaximumHeight( label2->sizeHint().height() * 2 );
   label2->setFrameStyle( QFrame::Panel | QFrame::Sunken );

   // ... and insert 50 items into the Combobox
   for ( i = 0; i < 50; i++ ) {
    str = QString( "Combobox Item %1" ).arg( i );
    if ( !( i % 4 ) )
       cb2->insertItem( QPixmap( "fileopen.xpm" ), str );
    else
       cb2->insertItem( str );
   }

   // Connect the activated SIGNALs of the Comboboxes with SLOTs
   connect( cb1, SIGNAL( activated( const QString & ) ), this, SLOT( slotCombo1Activated( const
QString & ) ) );
   connect( cb2, SIGNAL( activated( const QString & ) ), this, SLOT( slotCombo2Activated( const
QString & ) ) );
}

/*
 * SLOT slotLeft2Right
* Copies all selected items of the first ListBox into the
 * second ListBox
 */

void ListBoxCombo::slotLeft2Right()
{
   // Go through all items of the first ListBox
   for ( unsigned int i = 0; i < lb1->count(); i++ ) {
    QListBoxItem *item = lb1->item( i );
    // if the item is selected...
    if ( item->isSelected() ) {
       // ...and it is a text item...
       if ( item->pixmap() && !item->text().isEmpty() )
        lb2->insertItem( *item->pixmap(), item->text() );
       else if ( !item->pixmap() )
        lb2->insertItem( item->text() );
       else if ( item->text().isEmpty() )
        lb2->insertItem( *item->pixmap() );
```

```
      }
    }
}

/*
 * SLOT slotCombo1Activated( const QString &s )
* Sets the text of the item which the user just selected
 * in the first Combobox (and is now the value of s) to
 * the first Label.
 */

void ListBoxCombo::slotCombo1Activated( const QString &s )
{
    label1->setText( QString( "Current Item: %1" ).arg( s ) );
}

/*
 * SLOT slotCombo2Activated( const QString &s )
* Sets the text of the item which the user just selected
 * in the second Combobox (and is now the value of s) to
 * the second Label.
 */

void ListBoxCombo::slotCombo2Activated( const QString &s )
{
    label2->setText( QString( "Current Item: %1" ).arg( s ) );
}
```

listboxcombo.h

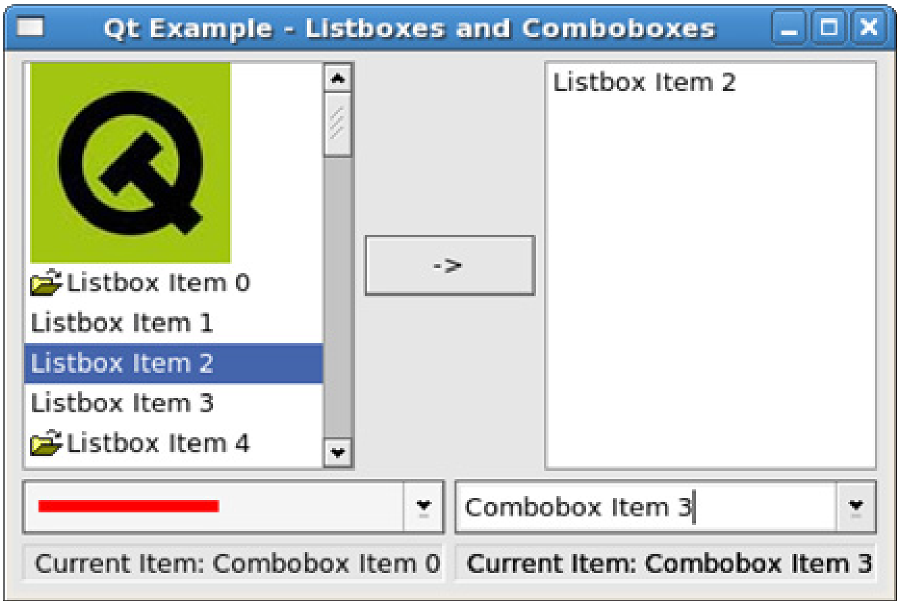
```
#ifndef LISTBOX_COMBO_H
#define LISTBOX_COMBO_H

#include <qvbox.h>

class QListBox;
class QLabel;

class ListBoxCombo : public QVBox
{
    Q_OBJECT

public:
    ListBoxCombo( QWidget *parent = 0, const char *name = 0 );

protected:
    QListBox *lb1, *lb2;
    QLabel *label1, *label2;

protected slots:
    void slotLeft2Right();
    void slotCombo1Activated( const QString &s );
    void slotCombo2Activated( const QString &s );

};
```

```
#endif
```

main.cpp

```
#include "listboxcombo.h"
#include <qapplication.h>

int main( int argc, char **argv )
{
   QApplication a( argc, argv );

   ListBoxCombo listboxcombo;
   listboxcombo.resize( 400, 270 );
   listboxcombo.setCaption( "Qt Example - Listboxes and Comboboxes" );
   a.setMainWidget( &listboxcombo );
   listboxcombo.show();

   return a.exec();
}
```

실행



34. 목록보기

이 실례는 목록보기(계층 및 여러칸)의 작업방법을 보여준다. 또한 특별한 리유로 목록보기 항목들을 파생클라스화하는 방법을 보여준다. 우편의뢰기의 기본창문처럼 표시하고 작업한다.

listviews.pro

```
TEMPLATE   = app
TARGET     = listviews
CONFIG     += qt warn_on release
HEADERS       = listviews.h
SOURCES       = listviews.cpp \
          main.cpp
```

listviews.cpp

```
#include "listviews.h"
#include <qlabel.h>
#include <qpainter.h>
#include <qpalette.h>
#include <qobjectlist.h>
#include <qpopupmenu.h>
#include <qheader.h>
#include <qregexp.h>

// -----------------------------------------------------------------
MessageHeader::MessageHeader( const MessageHeader &mh )
{
    msender = mh.msender;
    msubject = mh.msubject;
    mdatetime = mh.mdatetime;
}

MessageHeader &MessageHeader::operator=( const MessageHeader &mh )
{
    msender = mh.msender;
    msubject = mh.msubject;
    mdatetime = mh.mdatetime;

    return *this;
}

// -----------------------------------------------------------------
Folder::Folder( Folder *parent, const QString &name )
    : QObject( parent, name ), fName( name )
{
    lstMessages.setAutoDelete( TRUE );
}

// -----------------------------------------------------------------
FolderListItem::FolderListItem( QListView *parent, Folder *f )
    : QListViewItem( parent )
{
    myFolder = f;
    setText( 0, f->folderName() );

    if ( myFolder->children() )
      insertSubFolders( myFolder->children() );
}

FolderListItem::FolderListItem( FolderListItem *parent, Folder *f )
    : QListViewItem( parent )
{
    myFolder = f;

    setText( 0, f->folderName() );

    if ( myFolder->children() )
      insertSubFolders( myFolder->children() );
```

```
}

void FolderListItem::insertSubFolders( const QObjectList *lst )
{
   Folder *f;
   for ( f = ( Folder* )( ( QObjectList* )lst )->first(); f; f = ( Folder* )( ( QObjectList* )lst )->next() )
    (void)new FolderListItem( this, f );
}

// ----------------------------------------------------
MessageListItem::MessageListItem( QListView *parent, Message *m )
   : QListViewItem( parent )
{
   myMessage = m;
   setText( 0, myMessage->header().sender() );
   setText( 1, myMessage->header().subject() );
   setText( 2, myMessage->header().datetime().toString() );
}

void MessageListItem::paintCell( QPainter *p, const QColorGroup &cg,
                   int column, int width, int alignment )
{
   QColorGroup _cg( cg );
   QColor c = _cg.text();

   if ( myMessage->state() == Message::Unread )
    _cg.setColor( QColorGroup::Text, Qt::red );

   QListViewItem::paintCell( p, _cg, column, width, alignment );

   _cg.setColor( QColorGroup::Text, c );
}

// ----------------------------------------------------
ListViews::ListViews( QWidget *parent, const char *name )
   : QSplitter( Qt::Horizontal, parent, name )
{
   lstFolders.setAutoDelete( TRUE );

   folders = new QListView( this );
   folders->header()->setClickEnabled( FALSE );
   folders->addColumn( "Folder" );

   initFolders();
   setupFolders();

   folders->setRootIsDecorated( TRUE );
   setResizeMode( folders, QSplitter::KeepSize );

   QSplitter *vsplitter = new QSplitter( Qt::Vertical, this );

   messages = new QListView( vsplitter );
   messages->addColumn( "Sender" );
   messages->addColumn( "Subject" );
```

```
  messages->addColumn( "Date" );
  messages->setColumnAlignment( 1, Qt::AlignRight );
  messages->setAllColumnsShowFocus( TRUE );
  messages->setShowSortIndicator( TRUE );
  menu = new QPopupMenu( messages );
  for( int i = 1; i <= 10; i++ )
   menu->insertItem( QString( "Context Item %1" ).arg( i ) );
  connect(messages, SIGNAL( contextMenuRequested( QListViewItem *, const QPoint& , int ) ),
     this, SLOT( slotRMB( QListViewItem *, const QPoint &, int ) ) );
  vsplitter->setResizeMode( messages, QSplitter::KeepSize );

  message = new QLabel( vsplitter );
  message->setAlignment( Qt::AlignTop );
  message->setBackgroundMode( PaletteBase );

  connect( folders, SIGNAL( selectionChanged( QListViewItem* ) ),
      this, SLOT( slotFolderChanged( QListViewItem* ) ) );
  connect( messages, SIGNAL( selectionChanged() ),
      this, SLOT( slotMessageChanged() ) );
  connect( messages, SIGNAL( currentChanged( QListViewItem * ) ),
      this, SLOT( slotMessageChanged() ) );

  messages->setSelectionMode( QListView::Extended );
  // some preparations
  folders->firstChild()->setOpen( TRUE );
  folders->firstChild()->firstChild()->setOpen( TRUE );
  folders->setCurrentItem( folders->firstChild()->firstChild()->firstChild() );
  folders->setSelected( folders->firstChild()->firstChild()->firstChild(), TRUE );

  messages->setSelected( messages->firstChild(), TRUE );
  messages->setCurrentItem( messages->firstChild() );
  message->setMargin( 5 );

  QValueList<int> lst;
  lst.append( 170 );
  setSizes( lst );
}

void ListViews::initFolders()
{
  unsigned int mcount = 1;

  for ( unsigned int i = 1; i < 20; i++ ) {
   QString str;
   str = QString( "Folder %1" ).arg( i );
   Folder *f = new Folder( 0, str );
   for ( unsigned int j = 1; j < 5; j++ ) {
     QString str2;
     str2 = QString( "Sub Folder %1" ).arg( j );
     Folder *f2 = new Folder( f, str2 );
     for ( unsigned int k = 1; k < 3; k++ ) {
      QString str3;
      str3 = QString( "Sub Sub Folder %1" ).arg( k );
      Folder *f3 = new Folder( f2, str3 );
```

```
            initFolder( f3, mcount );
          }
       }
       lstFolders.append( f );
    }
}

void ListViews::initFolder( Folder *folder, unsigned int &count )
{
    for ( unsigned int i = 0; i < 15; i++, count++ ) {
      QString str;
      str = QString( "Message %1  " ).arg( count );
      QDateTime dt = QDateTime::currentDateTime();
      dt = dt.addSecs( 60 * count );
      MessageHeader mh( "Trolltech <info@trolltech.com>  ", str, dt );

      QString body;
      body = QString( "This is the message number %1 of this application, \n"
              "which shows how to use QListViews, QListViewItems, \n"
              "QSplitters and so on. The code should show how easy\n"
              "this can be done in Qt." ).arg( count );
      Message *msg = new Message( mh, body );
      folder->addMessage( msg );
    }
}

void ListViews::setupFolders()
{
    folders->clear();

    for ( Folder* f = lstFolders.first(); f; f = lstFolders.next() )
      (void)new FolderListItem( folders, f );
}

void ListViews::slotRMB( QListViewItem* Item, const QPoint & point, int )
{
    if( Item )
      menu->popup( point );
}

void ListViews::slotFolderChanged( QListViewItem *i )
{
    if ( !i )
      return;
    messages->clear();
    message->setText( "" );

    FolderListItem *item = ( FolderListItem* )i;

    for ( Message* msg = item->folder()->firstMessage(); msg;
        msg = item->folder()->nextMessage() )
      (void)new MessageListItem( messages, msg );
}
```

```
void ListViews::slotMessageChanged()
{
    QListViewItem *i = messages->currentItem();
    if ( !i )
      return;

    if ( !i->isSelected() ) {
      message->setText( "" );
      return;
    }

    MessageListItem *item = ( MessageListItem* )i;
    Message *msg = item->message();

    QString text;
    QString tmp = msg->header().sender();
    tmp = tmp.replace( "<", "&lt;" );
    tmp = tmp.replace( ">", "&gt;" );
    text = QString( "<b><i>From:</i></b> <a href=\"mailto:info@trolltech.com\">%1</a><br>"
              "<b><i>Subject:</i></b> <big><big><b>%2</b></big></big><br>"
              "<b><i>Date:</i></b> %3<br><br>"
              "%4" ).
         arg( tmp ).arg( msg->header().subject() ).
         arg( msg->header().datetime().toString() ).arg( msg->body() );

    message->setText( text );

    msg->setState( Message::Read );
}
```

listviews.h

```
#ifndef LISTVIEWS_H
#define LISTVIEWS_H

#include <qsplitter.h>
#include <qstring.h>
#include <qobject.h>
#include <qdatetime.h>
#include <qptrlist.h>
#include <qlistview.h>

class QListView;
class QLabel;
class QPainter;
class QColorGroup;
class QObjectList;
class QPopupMenu;

// -----------------------------------------------------
class MessageHeader
{
public:
    MessageHeader( const QString &_sender, const QString &_subject, const QDateTime &_datetime )
      : msender( _sender ), msubject( _subject ), mdatetime( _datetime )
```

```
   {}

   MessageHeader( const MessageHeader &mh );
   MessageHeader &operator=( const MessageHeader &mh );

   QString sender() { return msender; }
   QString subject() { return msubject; }
   QDateTime datetime() { return mdatetime; }

protected:
   QString msender, msubject;
   QDateTime mdatetime;

};

// ----------------------------------------------------
class Message
{
public:
   enum State { Read = 0,
         Unread};

   Message( const MessageHeader &mh, const QString &_body )
    : mheader( mh ), mbody( _body ), mstate( Unread )
   {}

   Message( const Message &m )
    : mheader( m.mheader ), mbody( m.mbody ), mstate( m.mstate )
   {}

   MessageHeader header() { return mheader; }
   QString body() { return mbody; }

   void setState( const State &s ) { mstate = s; }
   State state() { return mstate; }

protected:
   MessageHeader mheader;
   QString mbody;
   State mstate;

};

// ----------------------------------------------------
class Folder : public QObject
{
   Q_OBJECT

public:
   Folder( Folder *parent, const QString &name );
   ~Folder()
   {}

   void addMessage( Message *m )
```

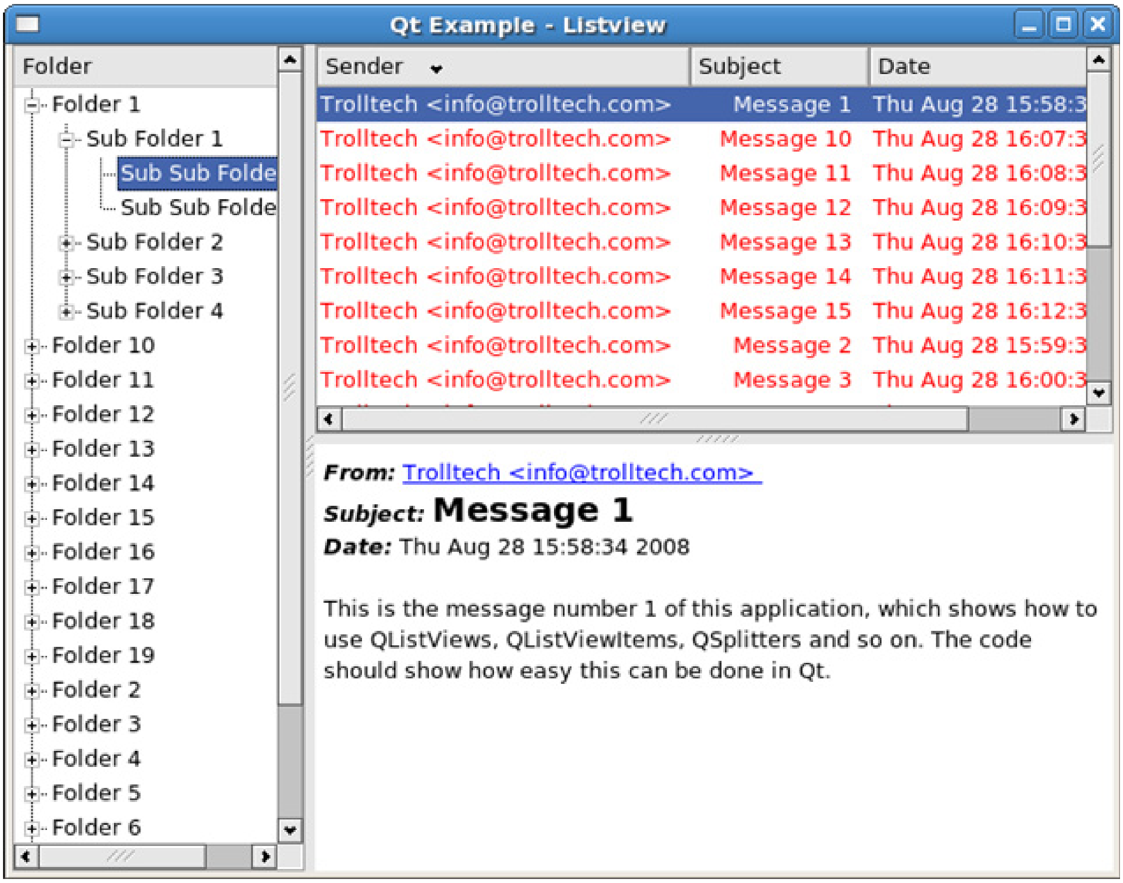
```
    { lstMessages.append( m ); }

    QString folderName() { return fName; }

    Message *firstMessage() { return lstMessages.first(); }
    Message *nextMessage() { return lstMessages.next(); }

protected:
    QString fName;
    QPtrList<Message> lstMessages;

};

// -----------------------------------------------------------------
class FolderListItem : public QListViewItem
{
public:
    FolderListItem( QListView *parent, Folder *f );
    FolderListItem( FolderListItem *parent, Folder *f );

    void insertSubFolders( const QObjectList *lst );

    Folder *folder() { return myFolder; }

protected:
    Folder *myFolder;

};

// -----------------------------------------------------------------
class MessageListItem : public QListViewItem
{
public:
    MessageListItem( QListView *parent, Message *m );

    virtual void paintCell( QPainter *p, const QColorGroup &cg,
                     int column, int width, int alignment );

    Message *message() { return myMessage; }

protected:
    Message *myMessage;

};

// -----------------------------------------------------------------
class ListViews : public QSplitter
{
    Q_OBJECT

public:
    ListViews( QWidget *parent = 0, const char *name = 0 );
    ~ListViews()
    {}
```

```
protected:
    void initFolders();
    void initFolder( Folder *folder, unsigned int &count );
    void setupFolders();

    QListView *messages, *folders;
    QLabel *message;
    QPopupMenu* menu;

    QPtrList<Folder> lstFolders;

protected slots:
    void slotFolderChanged( QListViewItem* );
    void slotMessageChanged();
    void slotRMB( QListViewItem*, const QPoint &, int );

};

#endif
```

main.cpp

```
#include "listviews.h"
#include <qapplication.h>

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    ListViews listViews;
    listViews.resize( 640, 480 );
    listViews.setCaption( "Qt Example - Listview" );
    a.setMainWidget( &listViews );
    listViews.show();

    return a.exec();
}
```

실행



Folder
- Folder 1
  - Sub Folder 1
    - Sub Sub Folde
    - Sub Sub Folde
  - Sub Folder 2
  - Sub Folder 3
  - Sub Folder 4
- Folder 10
- Folder 11
- Folder 12
- Folder 13
- Folder 14
- Folder 15
- Folder 16
- Folder 17
- Folder 18
- Folder 19
- Folder 2
- Folder 3
- Folder 4
- Folder 5
- Folder 6

Sender	Subject	Date
Trolltech <info@trolltech.com>	Message 1	Thu Aug 28 15:58:3
Trolltech <info@trolltech.com>	Message 10	Thu Aug 28 16:07:3
Trolltech <info@trolltech.com>	Message 11	Thu Aug 28 16:08:3
Trolltech <info@trolltech.com>	Message 12	Thu Aug 28 16:09:3
Trolltech <info@trolltech.com>	Message 13	Thu Aug 28 16:10:3
Trolltech <info@trolltech.com>	Message 14	Thu Aug 28 16:11:3
Trolltech <info@trolltech.com>	Message 15	Thu Aug 28 16:12:3
Trolltech <info@trolltech.com>	Message 2	Thu Aug 28 15:59:3
Trolltech <info@trolltech.com>	Message 3	Thu Aug 28 16:00:3

From: Trolltech <info@trolltech.com>

Subject: **Message 1**

Date: Thu Aug 28 15:58:34 2008

This is the message number 1 of this application, which shows how to use QListViews, QListViewItems, QSplitters and so on. The code should show how easy this can be done in Qt.

35. MDI 응용프로그람

이 실례프로그람은 MDI 를 제공하는것을 제외하면 실례 4 와 거의 같다.

mdi.pro

```
TEMPLATE  = app
TARGET    = mdi
CONFIG    += qt warn_on release
HEADERS        = application.h
SOURCES        = application.cpp \
          main.cpp
```

application.cpp

```
#include "application.h"
#include <qworkspace.h>
#include <qimage.h>
#include <qpixmap.h>
#include <qtoolbar.h>
#include <qtoolbutton.h>
#include <qpopupmenu.h>
#include <qmenubar.h>
#include <qmovie.h>
#include <qfile.h>
#include <qfiledialog.h>
```

```
#include <qlabel.h>
#include <qstatusbar.h>
#include <qmessagebox.h>
#include <qprinter.h>
#include <qapplication.h>
#include <qpushbutton.h>
#include <qaccel.h>
#include <qtextstream.h>
#include <qtextedit.h>
#include <qpainter.h>
#include <qpaintdevicemetrics.h>
#include <qwhatsthis.h>
#include <qobjectlist.h>
#include <qvbox.h>
#include <qsimplerichtext.h>

#include "filesave.xpm"
#include "fileopen.xpm"
#include "fileprint.xpm"

const char * fileOpenText = "Click this button to open a <em>new file</em>. <br><br>"
"You can also select the <b>Open command</b> from the File menu.";
const char * fileSaveText = "Click this button to save the file you are "
"editing.  You will be prompted for a file name.\n\n"
"You can also select the Save command from the File menu.\n\n"
"Note that implementing this function is left as an exercise for the reader.";
const char * filePrintText = "Click this button to print the file you "
"are editing.\n\n"
"You can also select the Print command from the File menu.";

ApplicationWindow::ApplicationWindow()
    : QMainWindow( 0, "example application main window", WDestructiveClose )
{
    int id;

    QPixmap openIcon, saveIcon;

    fileTools = new QToolBar( this, "file operations" );
    addToolBar( fileTools, tr( "File Operations" ), DockTop, TRUE );

    openIcon = QPixmap( fileopen );
    QToolButton * fileOpen
      = new QToolButton( openIcon, "Open File", QString::null,
                this, SLOT(load()), fileTools, "open file" );

    saveIcon = QPixmap( filesave );
    QToolButton * fileSave
      = new QToolButton( saveIcon, "Save File", QString::null,
                this, SLOT(save()), fileTools, "save file" );

#ifndef QT_NO_PRINTER
    printer = new QPrinter( QPrinter::HighResolution );
    QPixmap printIcon;
    printIcon = QPixmap( fileprint );
    QToolButton * filePrint   = new QToolButton( printIcon, "Print File", QString::null,
                   this, SLOT(print()), fileTools, "print file" );
    QWhatsThis::add( filePrint, filePrintText );
#endif

    (void)QWhatsThis::whatsThisButton( fileTools );

    QWhatsThis::add( fileOpen, fileOpenText );
    QWhatsThis::add( fileSave, fileSaveText );

    QPopupMenu * file = new QPopupMenu( this );
    menuBar()->insertItem( "&File", file );

    file->insertItem( "&New", this, SLOT(newDoc()), CTRL+Key_N );

    id = file->insertItem( openIcon, "&Open...",  this, SLOT(load()), CTRL+Key_O );
    file->setWhatsThis( id, fileOpenText );

    id = file->insertItem( saveIcon, "&Save",  this, SLOT(save()), CTRL+Key_S );
    file->setWhatsThis( id, fileSaveText );
    id = file->insertItem( "Save &As...", this, SLOT(saveAs()) );
    file->setWhatsThis( id, fileSaveText );
#ifndef QT_NO_PRINTER
    file->insertSeparator();
    id = file->insertItem( printIcon, "&Print...",  this, SLOT(print()), CTRL+Key_P );
    file->setWhatsThis( id, filePrintText );
#endif
    file->insertSeparator();
    file->insertItem( "&Close", this, SLOT(closeWindow()), CTRL+Key_W );
    file->insertItem( "&Quit", qApp, SLOT( closeAllWindows() ), CTRL+Key_Q );

    windowsMenu = new QPopupMenu( this );
    windowsMenu->setCheckable( TRUE );
    connect( windowsMenu, SIGNAL( aboutToShow() ),  this, SLOT( windowsMenuAboutToShow() ) );
    menuBar()->insertItem( "&Windows", windowsMenu );

    menuBar()->insertSeparator();
    QPopupMenu * help = new QPopupMenu( this );
    menuBar()->insertItem( "&Help", help );

    help->insertItem( "&About", this, SLOT(about()), Key_F1);
    help->insertItem( "About &Qt", this, SLOT(aboutQt()));
    help->insertSeparator();
    help->insertItem( "What's &This", this, SLOT(whatsThis()), SHIFT+Key_F1);

    QVBox* vb = new QVBox( this );
    vb->setFrameStyle( QFrame::StyledPanel | QFrame::Sunken );
    ws = new QWorkspace( vb );
    ws->setScrollBarsEnabled( TRUE );
    setCentralWidget( vb );

    statusBar()->message( "Ready", 2000 );
}
```

```
ApplicationWindow::~ApplicationWindow()
{
#ifndef QT_NO_PRINTER
   delete printer;
#endif
}

MDIWindow* ApplicationWindow::newDoc()
{
   MDIWindow* w = new MDIWindow( ws, 0, WDestructiveClose );
   connect( w, SIGNAL( message(const QString&, int) ), statusBar(), SLOT( message(const QString&,
int )) );
   w->setCaption("unnamed document");
   w->setIcon( QPixmap("document.xpm") );
   // show the very first window in maximized mode
   if ( ws->windowList().isEmpty() )
    w->showMaximized();
   else
    w->show();
   return w;
}

void ApplicationWindow::load()
{
   QString fn = QFileDialog::getOpenFileName( QString::null, QString::null, this );
   if ( !fn.isEmpty() ) {
    MDIWindow* w = newDoc();
    w->load( fn );
   }  else {
    statusBar()->message( "Loading aborted", 2000 );
   }
}

void ApplicationWindow::save()
{
   MDIWindow* m = (MDIWindow*)ws->activeWindow();
   if ( m )
    m->save();
}

void ApplicationWindow::saveAs()
{
   MDIWindow* m = (MDIWindow*)ws->activeWindow();
   if ( m )
    m->saveAs();
}

void ApplicationWindow::print()
{
#ifndef QT_NO_PRINTER
   MDIWindow* m = (MDIWindow*)ws->activeWindow();
   if ( m )
    m->print( printer );
```

```
#endif
}

void ApplicationWindow::closeWindow()
{
   MDIWindow* m = (MDIWindow*)ws->activeWindow();
   if ( m )
    m->close();
}

void ApplicationWindow::about()
{
   QMessageBox::about( this, "Qt Application Example",
             "This example demonstrates simple use of\n "
             "Qt's Multiple Document Interface (MDI).");
}

void ApplicationWindow::aboutQt()
{
   QMessageBox::aboutQt( this, "Qt Application Example" );
}

void ApplicationWindow::windowsMenuAboutToShow()
{
   windowsMenu->clear();
   int cascadeId = windowsMenu->insertItem("&Cascade", ws, SLOT(cascade() ) );
   int tileId = windowsMenu->insertItem("&Tile", ws, SLOT(tile() ) );
   int horTileId = windowsMenu->insertItem("Tile &Horizontally", this, SLOT(tileHorizontal() ) );
   if ( ws->windowList().isEmpty() ) {
    windowsMenu->setItemEnabled( cascadeId, FALSE );
    windowsMenu->setItemEnabled( tileId, FALSE );
    windowsMenu->setItemEnabled( horTileId, FALSE );
   }
   windowsMenu->insertSeparator();
   QWidgetList windows = ws->windowList();
   for ( int i = 0; i < int(windows.count()); ++i ) {
    int id = windowsMenu->insertItem(windows.at(i)->caption(),
                    this, SLOT( windowsMenuActivated( int ) ) );
    windowsMenu->setItemParameter( id, i );
    windowsMenu->setItemChecked( id, ws->activeWindow() == windows.at(i) );
   }
}

void ApplicationWindow::windowsMenuActivated( int id )
{
   QWidget* w = ws->windowList().at( id );
   if ( w )
    w->showNormal();
   w->setFocus();
}

void ApplicationWindow::tileHorizontal()
{
   // primitive horizontal tiling
```

```
  QWidgetList windows = ws->windowList();
  if ( !windows.count() )
   return;

  int heightForEach = ws->height() / windows.count();
  int y = 0;
  for ( int i = 0; i < int(windows.count()); ++i ) {
   QWidget *window = windows.at(i);
   if ( window->testWState( WState_Maximized ) ) {
     // prevent flicker
     window->hide();
     window->showNormal();
   }
   int preferredHeight = window->minimumHeight()+window->parentWidget()->baseSize().height();
   int actHeight = QMAX(heightForEach, preferredHeight);

   window->parentWidget()->setGeometry( 0, y, ws->width(), actHeight );
   y += actHeight;
  }
}

void ApplicationWindow::closeEvent( QCloseEvent *e )
{
  QWidgetList windows = ws->windowList();
  if ( windows.count() ) {
   for ( int i = 0; i < int(windows.count()); ++i ) {
     QWidget *window = windows.at( i );
     if ( !window->close() ) {
      e->ignore();
      return;
     }
   }
  }

  QMainWindow::closeEvent( e );
}

MDIWindow::MDIWindow( QWidget* parent, const char* name, int wflags )
  : QMainWindow( parent, name, wflags )
{
  mmovie = 0;
  medit = new QTextEdit( this );
  setFocusProxy( medit );
  setCentralWidget( medit );
}

MDIWindow::~MDIWindow()
{
  delete mmovie;
}

void MDIWindow::closeEvent( QCloseEvent *e )
{
  if ( medit->isModified() ) {
   switch( QMessageBox::warning( this, "Save Changes",
     tr("Save changes to %1?").arg( caption() ),
     tr("Yes"), tr("No"), tr("Cancel") ) ) {
   case 0:
     {
      save();
      if ( !filename.isEmpty() )
        e->accept();
      else
        e->ignore();
     }
     break;
   case 1:
     e->accept();
     break;
   default:
     e->ignore();
     break;
   }
  } else {
   e->accept();
  }
}

void MDIWindow::load( const QString& fn )
{
  filename  = fn;
  QFile f( filename );
  if ( !f.open( IO_ReadOnly ) )
   return;

  if(fn.contains(".gif")) {
   QWidget * tmp=new QWidget(this);
   setFocusProxy(tmp);
   setCentralWidget(tmp);
   medit->hide();
   delete medit;
   QMovie * qm=new QMovie(fn);
#ifdef Q_WS_QWS // temporary speed-test hack
   qm->setDisplayWidget(tmp);
#endif
   tmp->setBackgroundMode(QWidget::NoBackground);
   tmp->show();
   mmovie=qm;
  } else {
   mmovie = 0;

   QTextStream t(&f);
   QString s = t.read();
   medit->setText( s );
   f.close();

  }
  setCaption( filename );
```

```
   emit message( QString("Loaded document %1").arg(filename), 2000 );
}

void MDIWindow::save()
{
   if ( filename.isEmpty() ) {
      saveAs();
      return;
   }

   QString text = medit->text();
   QFile f( filename );
   if ( !f.open( IO_WriteOnly ) ) {
      emit message( QString("Could not write to %1").arg(filename),
              2000 );
      return;
   }

   QTextStream t( &f );
   t << text;
   f.close();

   setCaption( filename );

   emit message( QString( "File %1 saved" ).arg( filename ), 2000 );
}

void MDIWindow::saveAs()
{
   QString fn = QFileDialog::getSaveFileName( filename, QString::null, this );
   if ( !fn.isEmpty() ) {
      filename = fn;
      save();
   } else {
      emit message( "Saving aborted", 2000 );
   }
}

void MDIWindow::print( QPrinter* printer)
{
#ifndef QT_NO_PRINTER
   int pageNo = 1;

   if ( printer->setup(this) ) {          // printer dialog
      printer->setFullPage( TRUE );
    emit message( "Printing...", 0 );
    QPainter p;
    if ( !p.begin( printer ) )
       return;                   // paint on printer
    QPaintDeviceMetrics metrics( p.device() );
    int dpiy = metrics.logicalDpiY();
    int margin = (int) ( (2/2.54)*dpiy ); // 2 cm margins
    QRect view( margin, margin, metrics.width() - 2*margin, metrics.height() - 2*margin );
    QSimpleRichText richText( QStyleSheet::convertFromPlainText(medit->text()),
                   QFont(),  medit->context(),  medit->styleSheet(), medit->mimeSourceFactory(),
                   view.height() );
    richText.setWidth( &p, view.width() );
    int page = 1;
    do {
       richText.draw( &p, margin, margin, view, colorGroup() );
       view.moveBy( 0, view.height() );
       p.translate( 0 , -view.height() );
       p.drawText( view.right() - p.fontMetrics().width( QString::number( page ) ),
             view.bottom() + p.fontMetrics().ascent() + 5, QString::number( page ) );
       if ( view.top() - margin >= richText.height() )
         break;
       QString msg( "Printing (page " );
       msg += QString::number( ++pageNo );
       msg += ")...";
       emit message( msg, 0 );
       printer->newPage();
       page++;
    } while (TRUE);
   }
#endif
}
```

application.h

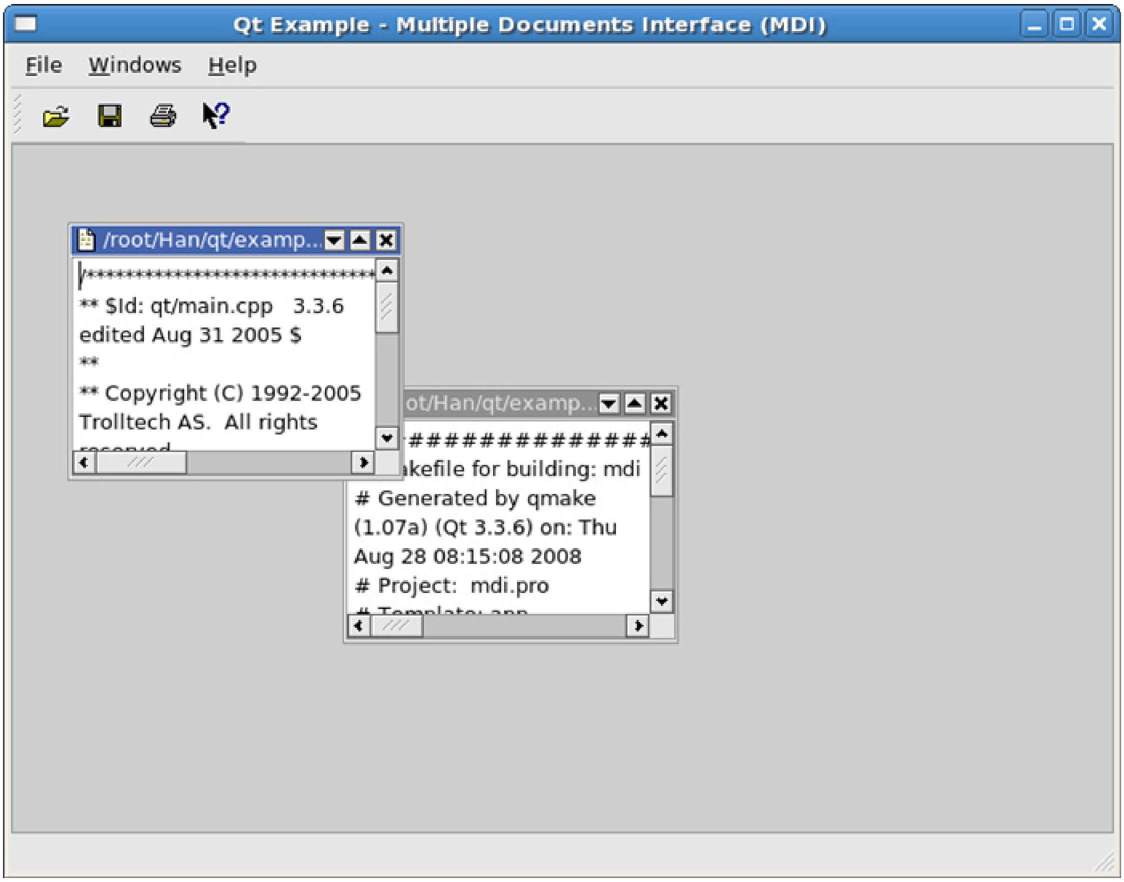
```
#ifndef APPLICATION_H
#define APPLICATION_H

#include <qmainwindow.h>
#include <qptrlist.h>

class QTextEdit;
class QToolBar;
class QPopupMenu;
class QWorkspace;
class QPopupMenu;
class QMovie;

class MDIWindow: public QMainWindow
{
   Q_OBJECT
public:
   MDIWindow( QWidget* parent, const char* name, int wflags );
   ~MDIWindow();

   void load( const QString& fn );
   void save();
   void saveAs();
   void print( QPrinter* );

protected:
   void closeEvent( QCloseEvent * );

signals:
   void message(const QString&, int );
private:
   QTextEdit* medit;
   QMovie * mmovie;
   QString filename;
};

class ApplicationWindow: public QMainWindow
{
   Q_OBJECT
public:
   ApplicationWindow();
   ~ApplicationWindow();

protected:
   void closeEvent( QCloseEvent * );

private slots:
   MDIWindow* newDoc();
   void load();
   void save();
   void saveAs();
   void print();
   void closeWindow();
   void tileHorizontal();

   void about();
   void aboutQt();

   void windowsMenuAboutToShow();
   void windowsMenuActivated( int id );

private:
   QPrinter *printer;
   QWorkspace* ws;
   QToolBar *fileTools;
   QPopupMenu* windowsMenu;
};

#endif
```

main.cpp

```
#include <qapplication.h>
#include "application.h"

int main( int argc, char ** argv ) {
   QApplication a( argc, argv );
   ApplicationWindow * mw = new ApplicationWindow();
   a.setMainWidget(mw);
   mw->setCaption( "Qt Example - Multiple Documents Interface (MDI)" );
   mw->show();
   a.connect( &a, SIGNAL(lastWindowClosed()), &a, SLOT(quit()) );
   int res = a.exec();
   return res;
```

}

실행



```
/*******************************
** $Id: qt/main.cpp   3.3.6
edited Aug 31 2005 $
**
** Copyright (C) 1992-2005
Trolltech AS.  All rights
reserved
```

```
##############
akefile for building: mdi
# Generated by qmake
(1.07a) (Qt 3.3.6) on: Thu
Aug 28 08:15:08 2008
# Project:  mdi.pro
# Template: app
```

36. 차림표리용

이 실례는 기본차림표, 보조차림표, 전용차림표항목들을 가지는 차림표띠를 보여준다. 또한 튀여나오기차림표를 보여준다.

menu.pro

```
TEMPLATE  = app
TARGET    = menu
CONFIG    += qt warn_on release
HEADERS       = menu.h
SOURCES       = menu.cpp
```

menu.cpp

```
#include "menu.h"
#include <qcursor.h>
#include <qpopupmenu.h>
#include <qapplication.h>
#include <qmessagebox.h>
#include <qpixmap.h>
#include <qpainter.h>

/* XPM */
```

```
static const char * p1_xpm[] = {
"16 16 3 1",
"   c None",
".  c #000000000000",
"X c #FFFFFFFF0000",
"                ",
"                ",
"         ....   ",
"        .XXXX.  ",
" .............. ",
" .XXXXXXXXXXXX. ",
" .XXXXXXXXXXXX. ",
" .XXXXXXXXXXXX. ",
" .XXXXXXXXXXXX. ",
" .XXXXXXXXXXXX. ",
" .XXXXXXXXXXXX. ",
" .XXXXXXXXXXXX. ",
" .XXXXXXXXXXXX. ",
" .XXXXXXXXXXXX. ",
" .............. ",
"                "};

/* XPM */
static const char * p2_xpm[] = {
"16 16 3 1",
"   c None",
".  c #000000000000",
"X c #FFFFFFFFFFFF",
"                ",
"   ......       ",
"   .XXX.X.      ",
"   .XXX.XX.     ",
"   .XXX.XXX.    ",
"   .XXX.....    ",
"   .XXXXXXX.    ",
"   .XXXXXXX.    ",
"   .XXXXXXX.    ",
"   .XXXXXXX.    ",
"   .XXXXXXX.    ",
"   .XXXXXXX.    ",
"   .XXXXXXX.    ",
"   .........    ",
"                ",
"                "};

/* XPM */
static const char * p3_xpm[] = {
"16 16 3 1",
"   c None",
".  c #000000000000",
"X c #FFFFFFFFFFFF",
"                ",
"                ",
"   .........    ",
"  ...........   ",
"  ........ ..   ",
"  ...........   ",
"  ...........   ",
"  ...........   ",
"  ...........   ",
"  ...XXXXX...   ",
"  ...XXXXX...   ",
"  ...XXXXX...   ",
"  ...XXXXX...   ",
"   .........    ",
"                ",
"                "};

/*
  Auxiliary class to provide fancy menu items with different
  fonts. Used for the "bold" and "underline" menu items in the options
  menu.
 */
class MyMenuItem : public QCustomMenuItem
{
public:
    MyMenuItem( const QString& s, const QFont& f )
      : string( s ), font( f ){};
    ~MyMenuItem(){}

    void paint( QPainter* p, const QColorGroup& /*cg*/, bool /*act*/, bool /*enabled*/, int x, int y, int w,
int h )
    {
      p->setFont ( font );
      p->drawText( x, y, w, h, AlignLeft | AlignVCenter | DontClip | ShowPrefix, string );
    }

    QSize sizeHint()
    {
      return QFontMetrics( font ).size( AlignLeft | AlignVCenter | ShowPrefix | DontClip,  string );
    }
private:
    QString string;
    QFont font;
};

MenuExample::MenuExample( QWidget *parent, const char *name )
    : QWidget( parent, name )
{
    QPixmap p1( p1_xpm );
    QPixmap p2( p2_xpm );
    QPixmap p3( p3_xpm );
    QPopupMenu *print = new QPopupMenu( this );
    Q_CHECK_PTR( print );
    print->insertTearOffHandle();
    print->insertItem( "&Print to printer", this, SLOT(printer()) );
    print->insertItem( "Print to &file", this, SLOT(file()) );
    print->insertItem( "Print to fa&x", this, SLOT(fax()) );
```

```
print->insertSeparator();
print->insertItem( "Printer &Setup", this, SLOT(printerSetup()) );

QPopupMenu *file = new QPopupMenu( this );
Q_CHECK_PTR( file );
file->insertItem( p1, "&Open",  this, SLOT(open()), CTRL+Key_O );
file->insertItem( p2, "&New", this, SLOT(news()), CTRL+Key_N );
file->insertItem( p3, "&Save", this, SLOT(save()), CTRL+Key_S );
file->insertItem( "&Close", this, SLOT(closeDoc()), CTRL+Key_W );
file->insertSeparator();
file->insertItem( "&Print", print, CTRL+Key_P );
file->insertSeparator();
file->insertItem( "E&xit",  qApp, SLOT(quit()), CTRL+Key_Q );

QPopupMenu *edit = new QPopupMenu( this );
Q_CHECK_PTR( edit );
int undoID = edit->insertItem( "&Undo", this, SLOT(undo()) );
int redoID = edit->insertItem( "&Redo", this, SLOT(redo()) );
edit->setItemEnabled( undoID, FALSE );
edit->setItemEnabled( redoID, FALSE );

QPopupMenu* options = new QPopupMenu( this );
Q_CHECK_PTR( options );
options->insertTearOffHandle();
options->setCaption("Options");
options->insertItem( "&Normal Font", this, SLOT(normal()) );
options->insertSeparator();

options->polish(); // adjust system settings
QFont f = options->font();
f.setBold( TRUE );
boldID = options->insertItem( new MyMenuItem( "Bold", f ) );
options->setAccel( CTRL+Key_B, boldID );
options->connectItem( boldID, this, SLOT(bold()) );
f = font();
f.setUnderline( TRUE );
underlineID = options->insertItem( new MyMenuItem( "Underline", f ) );
options->setAccel( CTRL+Key_U, underlineID );
options->connectItem( underlineID, this, SLOT(underline()) );

isBold = FALSE;
isUnderline = FALSE;
options->setCheckable( TRUE );

QPopupMenu *help = new QPopupMenu( this );
Q_CHECK_PTR( help );
help->insertItem( "&About", this, SLOT(about()), CTRL+Key_H );
help->insertItem( "About &Qt", this, SLOT(aboutQt()) );

// If we used a QMainWindow we could use its built-in menuBar().
menu = new QMenuBar( this );
Q_CHECK_PTR( menu );
menu->insertItem( "&File", file );
menu->insertItem( "&Edit", edit );
```

```
    menu->insertItem( "&Options", options );
    menu->insertSeparator();
    menu->insertItem( "&Help", help );
    menu->setSeparator( QMenuBar::InWindowsStyle );

    QLabel *msg = new QLabel( this );
    Q_CHECK_PTR( msg );
    msg->setText( "A context menu is available.\n"
            "Invoke it by right-clicking or by"
            " pressing the 'context' button." );
    msg->setGeometry( 0, height() - 60, width(), 60 );
    msg->setAlignment( AlignCenter );

    label = new QLabel( this );
    Q_CHECK_PTR( label );
    label->setGeometry( 20, rect().center().y()-20, width()-40, 40 );
    label->setFrameStyle( QFrame::Box | QFrame::Raised );
    label->setLineWidth( 1 );
    label->setAlignment( AlignCenter );

    connect( this,  SIGNAL(explain(const QString&)),  label, SLOT(setText(const QString&)) );

    setMinimumSize( 100, 80 );
    setFocusPolicy( QWidget::ClickFocus );
}

void MenuExample::contextMenuEvent( QContextMenuEvent * )
{
    QPopupMenu*      contextMenu = new QPopupMenu( this );
    Q_CHECK_PTR( contextMenu );
    QLabel *caption = new QLabel( "<font color=darkblue><u><b>"
     "Context Menu</b></u></font>", this );
    caption->setAlignment( Qt::AlignCenter );
    contextMenu->insertItem( caption );
    contextMenu->insertItem( "&New",  this, SLOT(news()), CTRL+Key_N );
    contextMenu->insertItem( "&Open...", this, SLOT(open()), CTRL+Key_O );
    contextMenu->insertItem( "&Save", this, SLOT(save()), CTRL+Key_S );
    QPopupMenu *submenu = new QPopupMenu( this );
    Q_CHECK_PTR( submenu );
    submenu->insertItem( "&Print to printer", this, SLOT(printer()) );
    submenu->insertItem( "Print to &file", this, SLOT(file()) );
    submenu->insertItem( "Print to fa&x", this, SLOT(fax()) );
    contextMenu->insertItem( "&Print", submenu );
    contextMenu->exec( QCursor::pos() );
    delete contextMenu;
}

void MenuExample::open()
{
    emit explain( "File/Open selected" );
}

void MenuExample::news()
{
    emit explain( "File/New selected" );
}

void MenuExample::save()
{
    emit explain( "File/Save selected" );
}

void MenuExample::closeDoc()
{
    emit explain( "File/Close selected" );
}

void MenuExample::undo()
{
    emit explain( "Edit/Undo selected" );
}

void MenuExample::redo()
{
    emit explain( "Edit/Redo selected" );
}

void MenuExample::normal()
{
    isBold = FALSE;
    isUnderline = FALSE;
    QFont font;
    label->setFont( font );
    menu->setItemChecked( boldID, isBold );
    menu->setItemChecked( underlineID, isUnderline );
    emit explain( "Options/Normal selected" );
}

void MenuExample::bold()
{
    isBold = !isBold;
    QFont font;
    font.setBold( isBold );
    font.setUnderline( isUnderline );
    label->setFont( font );
    menu->setItemChecked( boldID, isBold );
    emit explain( "Options/Bold selected" );
}

void MenuExample::underline()
{
    isUnderline = !isUnderline;
    QFont font;
    font.setBold( isBold );
    font.setUnderline( isUnderline );
    label->setFont( font );
    menu->setItemChecked( underlineID, isUnderline );
    emit explain( "Options/Underline selected" );
```

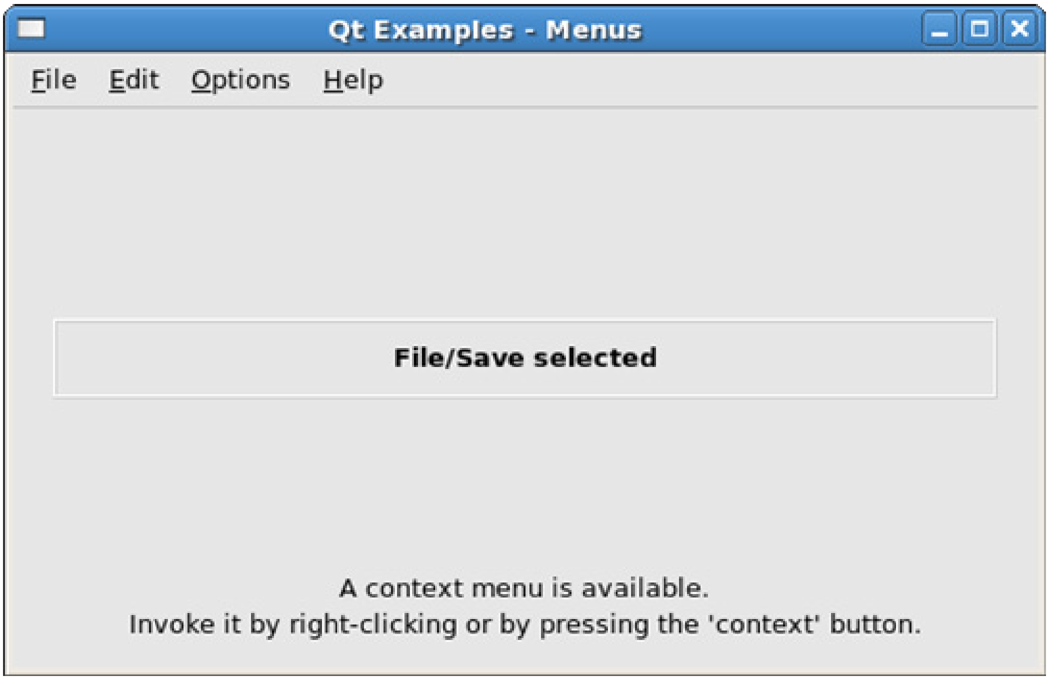
```
}

void MenuExample::about()
{
   QMessageBox::about( this, "Qt Menu Example",
              "This example demonstrates simple use of Qt menus.\n"
              "You can cut and paste lines from it to your own\n"
              "programs." );
}

void MenuExample::aboutQt()
{
   QMessageBox::aboutQt( this, "Qt Menu Example" );
}

void MenuExample::printer()
{
   emit explain( "File/Printer/Print selected" );
}

void MenuExample::file()
{
   emit explain( "File/Printer/Print To File selected" );
}

void MenuExample::fax()
{
   emit explain( "File/Printer/Print To Fax selected" );
}

void MenuExample::printerSetup()
{
   emit explain( "File/Printer/Printer Setup selected" );
}

void MenuExample::resizeEvent( QResizeEvent * )
{
   label->setGeometry( 20, rect().center().y()-20, width()-40, 40 );
}

int main( int argc, char ** argv )
{
   QApplication a( argc, argv );
   MenuExample m;
   m.setCaption("Qt Examples - Menus");
   a.setMainWidget( &m );
   m.show();
   return a.exec();
}
```

menu.h

```
#ifndef MENU_H
#define MENU_H
```

```
#include <qwidget.h>
#include <qmenubar.h>
#include <qlabel.h>

class MenuExample : public QWidget
{
   Q_OBJECT
public:
   MenuExample( QWidget *parent=0, const char *name=0 );

public slots:
   void open();
   void news();
   void save();
   void closeDoc();
   void undo();
   void redo();
   void normal();
   void bold();
   void underline();
   void about();
   void aboutQt();
   void printer();
   void file();
   void fax();
   void printerSetup();

protected:
   void    resizeEvent( QResizeEvent * );

signals:
   void    explain( const QString& );

private:
   void contextMenuEvent ( QContextMenuEvent * );

   QMenuBar *menu;
   QLabel   *label;
   bool isBold;
   bool isUnderline;
   int boldID, underlineID;
};

#endif // MENU_H
```

실행



File/Save selected

A context menu is available.
Invoke it by right-clicking or by pressing the 'context' button.

37. 영화 혹은 동화상 GIF 파일의 재생

Movies실례는 MNG와 동화상GIF파일들을 QMovie와 QLabel클라스들을 사용하여 보여준다. 영화는 Qt가 구축되였을 때 읽기가 허용되면 동화상GIF만 읽어들인다.

movies.pro

```
TEMPLATE  = app
TARGET    = movies
CONFIG    += qt warn_on release
HEADERS      =
SOURCES      = main.cpp
```

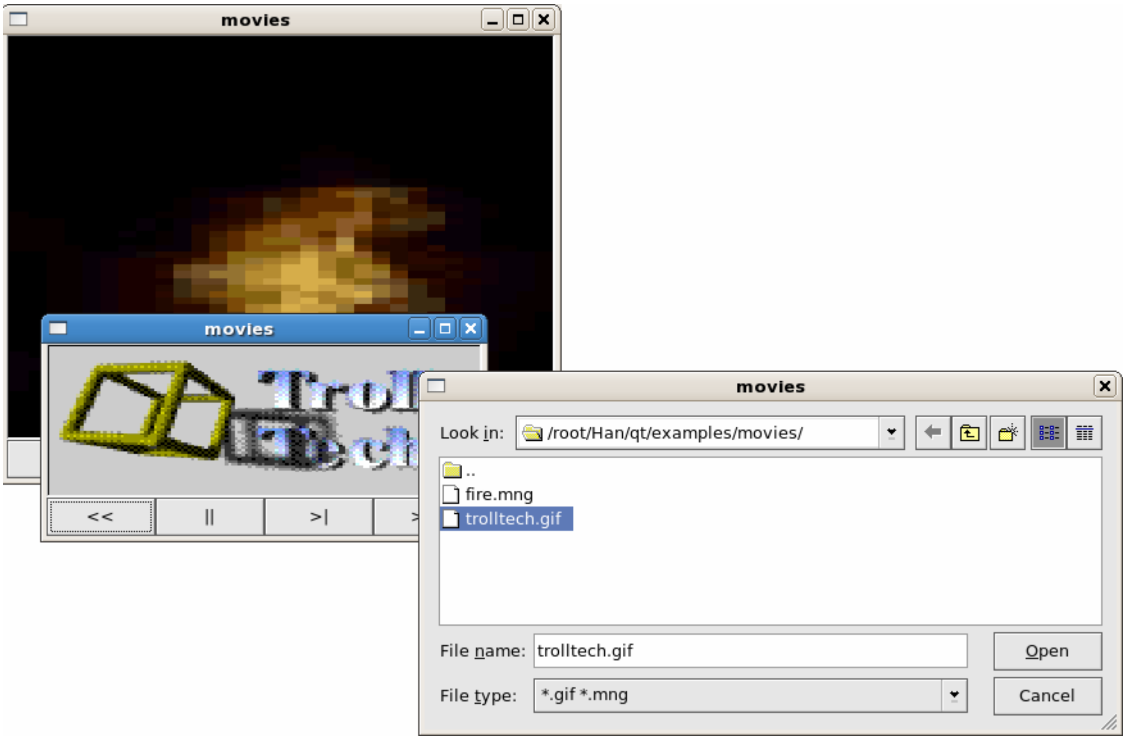
main.cpp

```
#include <qapplication.h>
#include <qfiledialog.h>
#include <qpushbutton.h>
#include <qlabel.h>
#include <qpainter.h>
#include <qmessagebox.h>
#include <qmovie.h>
#include <qvbox.h>

class MovieScreen : public QFrame {
   Q_OBJECT
   QMovie movie;
   QString filename;
   QSize sh;

public:
   MovieScreen(const char* fname, QMovie m, QWidget* p=0, const char* name=0, WFlags f=0) :
    QFrame(p, name, f),
   sh(100,100)
  {
    setCaption(fname);
    filename = fname;
    movie = m;

    // Set a frame around the movie.
    setFrameStyle(QFrame::WinPanel|QFrame::Sunken);

    // No background needed, since we draw on the whole widget.
    movie.setBackgroundColor(backgroundColor());
    setBackgroundMode(NoBackground);

    // Get the movie to tell use when interesting things happen.
    movie.connectUpdate(this, SLOT(movieUpdated(const QRect&)));
    movie.connectResize(this, SLOT(movieResized(const QSize&)));
    movie.connectStatus(this, SLOT(movieStatus(int)));

   setSizePolicy(QSizePolicy(QSizePolicy::Expanding,QSizePolicy::Expanding));
  }

  QSize sizeHint() const
  {
   return sh;
  }

protected:

  // Draw the contents of the QFrame - the movie and on-screen-display
  void drawContents(QPainter* p)
  {
    // Get the current movie frame.
    QPixmap pm = movie.framePixmap();

    // Get the area we have to draw in.
    QRect r = contentsRect();

   if ( !pm.isNull() ) {
     // Only rescale is we need to - it can take CPU!
     if ( r.size() != pm.size() ) {
      QWMatrix m;
      m.scale((double)r.width()/pm.width(), (double)r.height()/pm.height());
      pm = pm.xForm(m);
     }

     // Draw the [possibly scaled] frame.  movieUpdated() below calls
     // repaint with only the changed area, so clipping will ensure we
     // only do the minimum amount of rendering.
     //
     p->drawPixmap(r.x(), r.y(), pm);
   }
```

```
        // The on-screen display

        const char* message = 0;

        if (movie.paused()) {
            message = "PAUSED";
        } else if (movie.finished()) {
            message = "THE END";
        } else if (movie.steps() > 0) {
            message = "FF >>";
        }

        if (message) {
            // Find a good font size...
            p->setFont(QFont("Helvetica", 24));

            QFontMetrics fm = p->fontMetrics();
            if ( fm.width(message) > r.width()-10 )
                p->setFont(QFont("Helvetica", 18));

            fm = p->fontMetrics();
            if ( fm.width(message) > r.width()-10 )
                p->setFont(QFont("Helvetica", 14));

            fm = p->fontMetrics();
            if ( fm.width(message) > r.width()-10 )
                p->setFont(QFont("Helvetica", 12));

            fm = p->fontMetrics();
            if ( fm.width(message) > r.width()-10 )
                p->setFont(QFont("Helvetica", 10));

            // "Shadow" effect.
            p->setPen(black);
            p->drawText(1, 1, width()-1, height()-1, AlignCenter, message);
            p->setPen(white);
            p->drawText(0, 0, width()-1, height()-1, AlignCenter, message);
        }
    }

public slots:
    void restart()
    {
     movie.restart();
        repaint();
    }

    void togglePause()
    {
     if ( movie.paused() )
         movie.unpause();
     else
         movie.pause();
        repaint();
    }

    void step()
    {
     movie.step();
       repaint();
    }

    void step10()
    {
     movie.step(10);
       repaint();
    }

private slots:
    void movieUpdated(const QRect& area)
    {
       if (!isVisible())
          show();

       // The given area of the movie has changed.

       QRect r = contentsRect();

       if ( r.size() != movie.framePixmap().size() ) {
          // Need to scale - redraw whole frame.
          repaint( r );
       } else {
          // Only redraw the changed area of the frame
          repaint( area.x()+r.x(), area.y()+r.x(), area.width(), area.height() );
       }
    }

    void movieResized(const QSize& size)
    {
       // The movie changed size, probably from its initial zero size.

       int fw = frameWidth();
       sh = QSize( size.width() + fw*2, size.height() + fw*2 );
     updateGeometry();
     if ( parentWidget() && parentWidget()->isHidden() )
        parentWidget()->show();
    }

    void movieStatus(int status)
    {
       // The movie has sent us a status message.

       if (status < 0) {
        QString msg;
        msg.sprintf("Could not play movie \"%s\"", (const char*)filename);
        QMessageBox::warning(this, "movies", msg);
        parentWidget()->close();
       } else if (status == QMovie::Paused || status == QMovie::EndOfMovie) {
            repaint(); // Ensure status text is displayed
        }
    }
};

class MoviePlayer : public QVBox {
    MovieScreen* movie;
public:
    MoviePlayer(const char* fname, QMovie m, QWidget* p=0, const char* name=0, WFlags f=0) :
     QVBox(p,name,f)
    {
     movie = new MovieScreen(fname, m, this);
     QHBox* hb = new QHBox(this);
     QPushButton* btn;
     btn = new QPushButton("<<", hb);
     connect(btn, SIGNAL(clicked()), movie, SLOT(restart()));
     btn = new QPushButton("||", hb);
     connect(btn, SIGNAL(clicked()), movie, SLOT(togglePause()));
     btn = new QPushButton(">|", hb);
     connect(btn, SIGNAL(clicked()), movie, SLOT(step()));
     btn = new QPushButton(">>|", hb);
     connect(btn, SIGNAL(clicked()), movie, SLOT(step10()));
    }
};


// A QFileDialog that chooses movies.
//
class MovieStarter: public QFileDialog {
    Q_OBJECT
public:
    MovieStarter(const char *dir);

public slots:
    void startMovie(const QString& filename);
    // QDialog's method - normally closes the file dialog.
    // We want it left open, and we want Cancel to quit everything.
    void done( int r );
};


MovieStarter::MovieStarter(const char *dir)  : QFileDialog(dir, "*.gif *.mng")
{
    //behave as in getOpenFilename
    setMode( ExistingFile );
    // When a file is selected, show it as a movie.
    connect(this, SIGNAL(fileSelected(const QString&)),
          this, SLOT(startMovie(const QString&)));
}

void MovieStarter::startMovie(const QString& filename)
{
    if ( filename ) // Start a new movie - have it delete when closed.
     (new MoviePlayer( filename, QMovie(filename), 0, 0,  WDestructiveClose))->show();
```

```
}

void MovieStarter::done( int r )
{
   if (r != Accepted)
    qApp->quit(); // end on Cancel
   setResult( r );

   // And don't hide.
}


int main(int argc, char **argv)
{
   QApplication a(argc, argv);

   if (argc > 1) {
      // Commandline mode - show movies given on the command line
      //
    bool gui=TRUE;
      for (int arg=1; arg<argc; arg++) {
       if ( QString(argv[arg]) == "-i" )
        gui = !gui;
       else if ( gui )
        (void)new MoviePlayer(argv[arg], QMovie(argv[arg]), 0, 0,  Qt::WDestructiveClose);
       else
        (void)new MovieScreen(argv[arg], QMovie(argv[arg]), 0, 0, Qt::WDestructiveClose);
     }
      QObject::connect(qApp, SIGNAL(lastWindowClosed()), qApp, SLOT(quit()));
   } else {
      // "GUI" mode - open a chooser for movies
      //
      MovieStarter* fd = new MovieStarter(".");
      fd->show();
   }

   // Go!
   return a.exec();
}

#include "main.moc"
```

실행



38. 망

다음의 실례프로그람들은 Qt망모듈의 사용법을 보여준다.

1) qt 관련우편목록파일탐색

다음의 실례는 qt관련우편목록파일에 대한 탐색을 수행한다. 여기서는 QHttp를 리용하여 탐색지령을 발행하고 결과를 추출한다. GUI부분은 Qt Designer를 리용하여 작성하였다.

archivesearch.pro

```
TEMPLATE  = app
CONFIG      += qt warn_on release
HEADERS       += archivedialog.ui.h
INTERFACES   += archivedialog.ui
SOURCES       += main.cpp
```

archivedialog.ui

```
<!DOCTYPE UI><UI version="3.1" stdsetdef="1">
<class>ArchiveDialog</class>
<widget class="QDialog">
  <property name="name">
…
  <function access="private" specifier="non virtual">init()</function>
</functions>
<layoutdefaults spacing="6" margin="11"/>
</UI>
```

archivedialog.ui.h

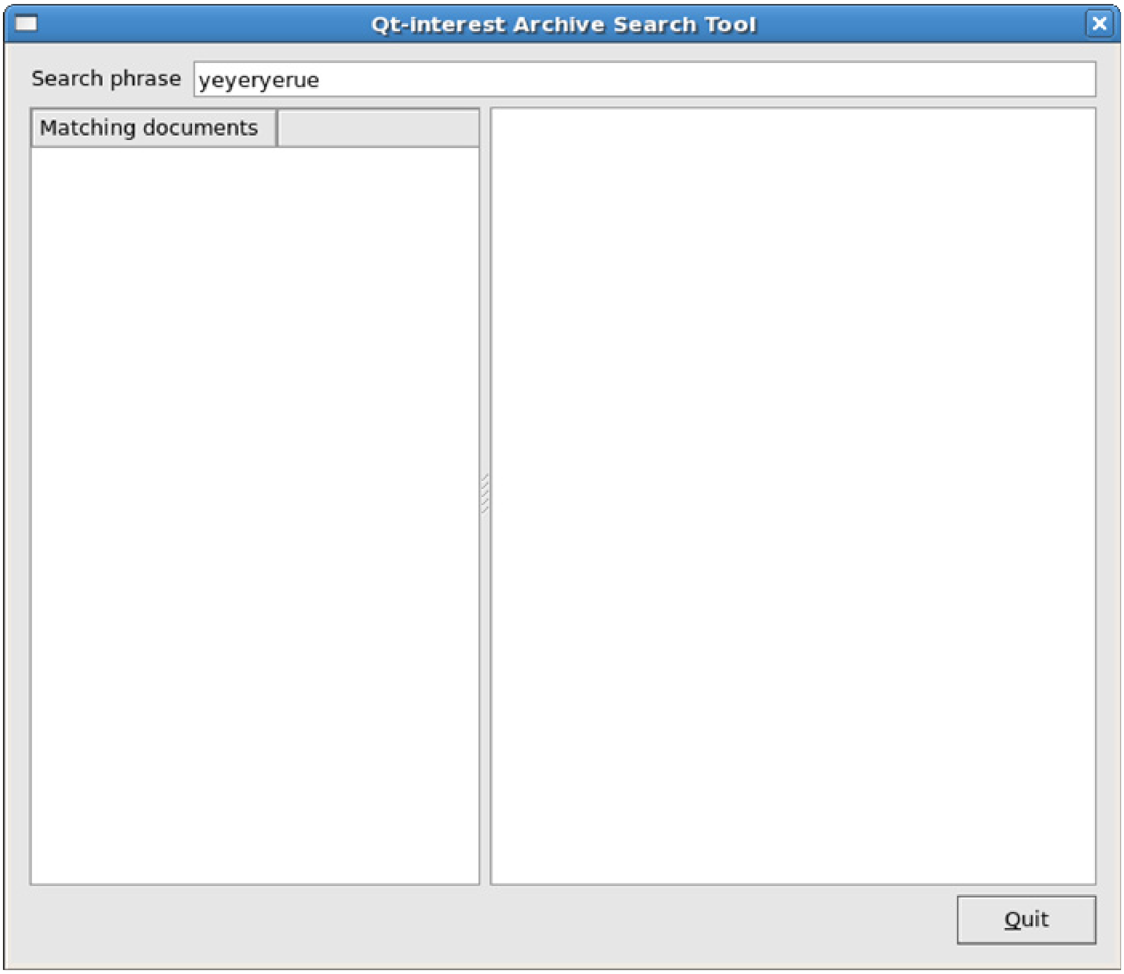
```
void ArchiveDialog::init()
```

```
{
   connect(&articleSearcher, SIGNAL(done(bool)), this, SLOT(searchDone(bool)));
   connect(&articleFetcher, SIGNAL(done(bool)), this, SLOT(fetchDone(bool)));
   connect(myListView, SIGNAL(selectionChanged(QListViewItem*)), this,
        SLOT(fetch(QListViewItem*)));
   connect(myLineEdit, SIGNAL(returnPressed()), this, SLOT(search()));
   connect(myListView, SIGNAL(returnPressed(QListViewItem*)), this,
SLOT(fetch(QListViewItem*)));
   connect(myPushButton, SIGNAL(clicked()), this, SLOT(close()));
}

void ArchiveDialog::fetch( QListViewItem *it )
{
   QUrl u(it->text(1));
   articleFetcher.setHost(u.host());
   articleFetcher.get(it->text(1));
}

void ArchiveDialog::fetchDone( bool error )
{
   if (error) {
    QMessageBox::critical(this, "Error fetching",
                 "An error occurred when fetching this document: "
                 + articleFetcher.errorString(),
                 QMessageBox::Ok, QMessageBox::NoButton);
   } else {
    myTextBrowser->setText(articleFetcher.readAll());
   }
}

void ArchiveDialog::search()
{
   if (articleSearcher.state() == QHttp::HostLookup
    || articleSearcher.state() == QHttp::Connecting
    || articleSearcher.state() == QHttp::Sending
    || articleSearcher.state() == QHttp::Reading) {
    articleSearcher.abort();
   }

   if (myLineEdit->text() == "") {
    QMessageBox::critical(this, "Empty query",
                 "Please type a search string.",
                 QMessageBox::Ok, QMessageBox::NoButton);
   } else {
    QApplication::setOverrideCursor(QCursor(Qt::WaitCursor));

    articleSearcher.setHost("www.trolltech.com");

    QHttpRequestHeader header("POST", "/search.html");
    header.setValue("Host", "www.trolltech.com");
    header.setContentType("application/x-www-form-urlencoded");

    QString encodedTopic = myLineEdit->text();
    QUrl::encode(encodedTopic);
   QString searchString = "qt-interest=on&search=" + encodedTopic;

   articleSearcher.request(header, searchString.utf8());
  }

}

void ArchiveDialog::searchDone( bool error )
{
  if (error) {
   QMessageBox::critical(this, "Error searching",
              "An error occurred when searching: "
              + articleSearcher.errorString(),
              QMessageBox::Ok, QMessageBox::NoButton);
  } else {
   QString result(articleSearcher.readAll());

   QRegExp rx("<a href=\"(http://lists\\.trolltech\\.com/qt-interest/.*)\">(.*)</a>");
   rx.setMinimal(TRUE);
   int pos = 0;
   while (pos >= 0) {
     pos = rx.search(result, pos);
     if (pos > -1) {
      pos += rx.matchedLength();
      new QListViewItem(myListView, rx.cap(2), rx.cap(1));
     }
   }
  }

  QApplication::restoreOverrideCursor();
}
```

main.cpp

```
#include "archivedialog.h"
#include <qapplication.h>

int main(int argc, char **argv)
{
  QApplication a( argc, argv );
  ArchiveDialog ad;
  ad.show();

  QObject::connect( &a, SIGNAL(lastWindowClosed()),  &a, SLOT(quit()) );

  return a.exec();
}
```

실행



yeyeryerue

2) 간단한 의뢰기-봉사기실례

이 실례는 소케트를 사용하여 두개의 프로그람이 통신하는 방법을 보여준다.

2개의 간단한 실례프로그람 즉 의뢰기프로그람과 봉사기프로그람이 제공된다. 둘다 QSocket클라스를 사용하고 봉사기는 또한 QServerSocket클라스를 사용한다.

봉사기는 포구번호 4242를 감시하며 들어오는 접속을 받아들인다. 의뢰기로부터 들어오는 번호가 달린 매개 행을 되돌려보낸다.

의뢰기는 지령행에서 지정된 호스트의 봉사기에, 혹은 지령행인수가 지정되지 않으면 국부호스트의 봉사기에 접속하려고 시도한다. 봉사기에 한개 행씩 보낼수 있다.

(1) 의뢰기

client.pro

```
TEMPLATE  = app
TARGET    = client
CONFIG    += qt warn_on release
HEADERS     =
SOURCES     = client.cpp
```

client.cpp

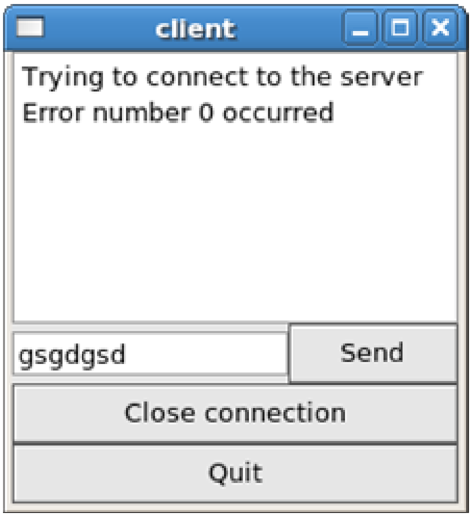
```
#include <qsocket.h>
```

```
#include <qapplication.h>
#include <qvbox.h>
#include <qhbox.h>
#include <qtextview.h>
#include <qlineedit.h>
#include <qlabel.h>
#include <qpushbutton.h>
#include <qtextstream.h>

class Client : public QVBox
{
   Q_OBJECT
public:
   Client( const QString &host, Q_UINT16 port )
   {
    // GUI layout
    infoText = new QTextView( this );
    QHBox *hb = new QHBox( this );
    inputText = new QLineEdit( hb );
    QPushButton *send = new QPushButton( tr("Send") , hb );
    QPushButton *close = new QPushButton( tr("Close connection") , this );
    QPushButton *quit = new QPushButton( tr("Quit") , this );

    connect( send, SIGNAL(clicked()), SLOT(sendToServer()) );
    connect( close, SIGNAL(clicked()), SLOT(closeConnection()) );
    connect( quit, SIGNAL(clicked()), qApp, SLOT(quit()) );

    // create the socket and connect various of its signals
    socket = new QSocket( this );
    connect( socket, SIGNAL(connected()), SLOT(socketConnected()) );
    connect( socket, SIGNAL(connectionClosed()), SLOT(socketConnectionClosed()) );
    connect( socket, SIGNAL(readyRead()), SLOT(socketReadyRead()) );
    connect( socket, SIGNAL(error(int)), SLOT(socketError(int)) );

    // connect to the server
    infoText->append( tr("Trying to connect to the server\n") );
    socket->connectToHost( host, port );
   }

   ~Client()
   {
   }

private slots:
   void closeConnection()
   {
    socket->close();
    if ( socket->state() == QSocket::Closing ) {
       // We have a delayed close.
       connect( socket, SIGNAL(delayedCloseFinished()),
            SLOT(socketClosed()) );
    } else {
       // The socket is closed.
       socketClosed();
```

```
     }
    }

    void sendToServer()
    {
     // write to the server
     QTextStream os(socket);
     os << inputText->text() << "\n";
     inputText->setText( "" );
    }

    void socketReadyRead()
    {
     // read from the server
     while ( socket->canReadLine() ) {
        infoText->append( socket->readLine() );
     }
    }

    void socketConnected()
    {
     infoText->append( tr("Connected to server\n") );
    }

    void socketConnectionClosed()
    {
     infoText->append( tr("Connection closed by the server\n") );
    }

    void socketClosed()
    {
     infoText->append( tr("Connection closed\n") );
    }

    void socketError( int e )
    {
     infoText->append( tr("Error number %1 occurred\n").arg(e) );
    }

private:
    QSocket *socket;
    QTextView *infoText;
    QLineEdit *inputText;
};

int main( int argc, char** argv )
{
    QApplication app( argc, argv );
    Client client( argc<2 ? "localhost" : argv[1], 4242 );
    app.setMainWidget( &client );
    client.show();
    return app.exec();
}
```

```
#include "client.moc"
```

실행



```
Trying to connect to the server
Error number 0 occurred
```

(2) 봉사기

server.pro

```
TEMPLATE  = app
TARGET      = server
CONFIG      += qt warn_on release
HEADERS         =
SOURCES         = server.cpp
```

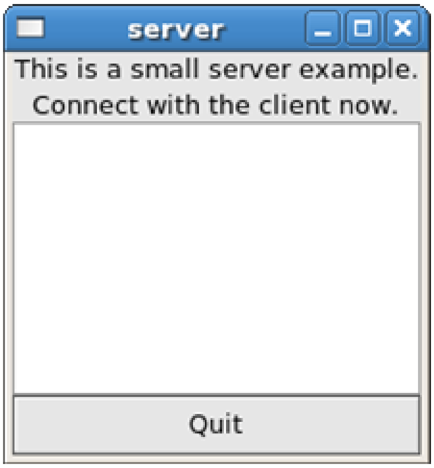
server.cpp

```
#include <qsocket.h>
#include <qserversocket.h>
#include <qapplication.h>
#include <qvbox.h>
#include <qtextview.h>
#include <qlabel.h>
#include <qpushbutton.h>
#include <qtextstream.h>
#include <stdlib.h>


/*
  The ClientSocket class provides a socket that is connected with a client.
  For every client that connects to the server, the server creates a new
  instance of this class.
*/
class ClientSocket : public QSocket
{
    Q_OBJECT
public:
    ClientSocket( int sock, QObject *parent=0, const char *name=0 ) :
     QSocket( parent, name )
    {
     line = 0;
```

```
      connect( this, SIGNAL(readyRead()), SLOT(readClient()) );
      connect( this, SIGNAL(connectionClosed()),  SLOT(deleteLater()) );
      setSocket( sock );
    }

    ~ClientSocket()
    {
    }

signals:
    void logText( const QString& );

private slots:
    void readClient()
    {
      QTextStream ts( this );
      while ( canReadLine() ) {
          QString str = ts.readLine();
          emit logText( tr("Read: '%1'\n").arg(str) );

          ts << line << ": " << str << endl;
          emit logText( tr("Wrote: '%1: %2'\n").arg(line).arg(str) );

          line++;
      }
    }

private:
    int line;
};

/*
  The SimpleServer class handles new connections to the server. For every
  client that connects, it creates a new ClientSocket -- that instance is now
  responsible for the communication with that client.
*/
class SimpleServer : public QServerSocket
{
    Q_OBJECT
public:
    SimpleServer( QObject* parent=0 ) :
      QServerSocket( 4242, 1, parent )
    {
      if ( !ok() ) {
          qWarning("Failed to bind to port 4242");
          exit(1);
      }
    }

    ~SimpleServer()
    {
    }

    void newConnection( int socket )
    {
     ClientSocket *s = new ClientSocket( socket, this );
     emit newConnect( s );
    }

signals:
    void newConnect( ClientSocket* );
};

/*
  The ServerInfo class provides a small GUI for the server. It also creates the
  SimpleServer and as a result the server.
*/
class ServerInfo : public QVBox
{
    Q_OBJECT
public:
    ServerInfo()
    {
     SimpleServer *server = new SimpleServer( this );

     QString itext = tr("This is a small server example.\n"
          "Connect with the client now."
          );
     QLabel *lb = new QLabel( itext, this );
     lb->setAlignment( AlignHCenter );
     infoText = new QTextView( this );
     QPushButton *quit = new QPushButton( tr("Quit") , this );

     connect( server, SIGNAL(newConnect(ClientSocket*)),      SLOT(newConnect(ClientSocket*)) );
     connect( quit, SIGNAL(clicked()), qApp,  SLOT(quit()) );
    }

    ~ServerInfo()
    {
    }

private slots:
    void newConnect( ClientSocket *s )
    {
     infoText->append( tr("New connection\n") );
     connect( s, SIGNAL(logText(const QString&)),  infoText, SLOT(append(const QString&)) );
     connect( s, SIGNAL(connectionClosed()), SLOT(connectionClosed()) );
    }

    void connectionClosed()
    {
     infoText->append( tr("Client closed connection\n") );
    }

private:
    QTextView *infoText;
};
```

```
int main( int argc, char** argv )
{
   QApplication app( argc, argv );
   ServerInfo info;
   app.setMainWidget( &info );
   info.show();
   return app.exec();
}

#include "server.moc"
```

실행



This is a small server example.
Connect with the client now.

3) FTP 의뢰기

이 실례는 FTP의뢰기를 실현한다. 여기서는 It uses QFtp를 사용하여 그 FTP지령들을 수행한다. GUI부분들은 Designer에서 작성된다.

ftpclient.pro

```
TEMPLATE  = app
TARGET      = ftpclient
CONFIG      += qt warn_on release
HEADERS         = ftpviewitem.h
SOURCES         = main.cpp \
          ftpviewitem.cpp
FORMS         = ftpmainwindow.ui \
          connectdialog.ui
IMAGES       = images/file.png \
          images/folder.png
```

ftpviewitem.cpp

```
#include <qpixmap.h>
#include "ftpviewitem.h"

FtpViewItem::FtpViewItem( QListView *parent, Type t, const QString &name, const QString &size,
const QString &lastModified )  : QListViewItem(parent,name,size,lastModified), type(t)
{
   // the pixmaps for folders and files are in an image collection
```

```
    if ( type == Directory )
      setPixmap( 0, QPixmap::fromMimeSource( "folder.png" ) );
    else
      setPixmap( 0, QPixmap::fromMimeSource( "file.png" ) );
}

int FtpViewItem::compare( QListViewItem * i, int col, bool ascending ) const
{
    // The entry ".." is always the first one.
    if ( text(0) == ".." ) {
      if ( ascending )
          return -1;
      else
          return 1;
    }
    if ( i->text(0) == ".." ) {
      if ( ascending )
          return 1;
      else
          return -1;
    }

    // Directories are before files.
    if ( type != ((FtpViewItem*)i)->type ) {
      if ( type == Directory ) {
          if ( ascending )
            return -1;
          else
            return 1;
      } else {
          if ( ascending )
            return 1;
          else
            return -1;
      }
    }

    // Use default sorting otherwise.
    return QListViewItem::compare( i, col, ascending );
}
```

ftpviewitem.h

```
#ifndef FTPVIEWITEM_H
#define FTPVIEWITEM_H

#include <qlistview.h>
#include <qdatetime.h>

class FtpViewItem : public QListViewItem
{
public:
    enum Type {
      Directory,
      File
    };

    FtpViewItem( QListView *parent, Type t, const QString &name, const QString &size, const QString
&lastModified );

    int compare( QListViewItem * i, int col, bool ascending ) const;

    bool isDir()
    { return type==Directory; }

private:
    Type type;
};

#endif
```

connectdialog.ui

```
<!DOCTYPE UI><UI version="3.0" stdsetdef="1">
<class>ConnectDialog</class>
<widget class="QDialog">
    <property name="name">
        <cstring>ConnectDialog</cstring>
    </property>
…
    <tabstop>username</tabstop>
    <tabstop>password</tabstop>
    <tabstop>buttonOk</tabstop>
    <tabstop>buttonCancel</tabstop>
</tabstops>
<layoutdefaults spacing="6" margin="11"/>
</UI>
```

ftpmainwindow.ui

```
<!DOCTYPE UI><UI version="3.0" stdsetdef="1">
<class>FtpMainWindow</class>
<widget class="QMainWindow">
    <property name="name">
        <cstring>FtpMainWindow</cstring>
…
    <function access="private">init()</function>
    <function access="private">destroy()</function>
</functions>
<layoutdefaults spacing="6" margin="11"/>
</UI>
```

ftpmainwindow.ui.h

```
#include <qftp.h>
#include <qlineedit.h>
#include <qspinbox.h>
#include <qstatusbar.h>
#include <qmessagebox.h>
#include <qfiledialog.h>
#include <qprogressdialog.h>
#include <qapplication.h>
```

```
#include "connectdialog.h"
#include "ftpviewitem.h"

void FtpMainWindow::init()
{
   stateFtp = new QLabel( tr("Unconnected"), statusBar() );
   statusBar()->addWidget( stateFtp, 0, TRUE );

   ftp = new QFtp( this );
   connect( ftp, SIGNAL(commandStarted(int)), SLOT(ftp_commandStarted()) );
   connect( ftp, SIGNAL(commandFinished(int,bool)), SLOT(ftp_commandFinished()) );
   connect( ftp, SIGNAL(done(bool)), SLOT(ftp_done(bool)) );
   connect( ftp, SIGNAL(stateChanged(int)), SLOT(ftp_stateChanged(int)) );
   connect( ftp, SIGNAL(listInfo(const QUrlInfo &)), SLOT(ftp_listInfo(const QUrlInfo &)) );
   connect( ftp, SIGNAL(rawCommandReply(int, const QString &)),
       SLOT(ftp_rawCommandReply(int, const QString &)) );
}

void FtpMainWindow::destroy()
{
   if ( ftp->state() != QFtp::Unconnected )
    ftp->close();
}

void FtpMainWindow::uploadFile()
{
   QString fileName = QFileDialog::getOpenFileName( QString::null,  QString::null,  this,
       "upload file dialog",  tr("Choose a file to upload") );
   if ( fileName.isNull() )
    return;

   QFile *file = new QFile( fileName );
   if ( !file->open( IO_ReadOnly ) ) {
   QMessageBox::critical( this, tr("Upload error"),tr("Can't open file '%1' for reading.").arg(fileName) );
    delete file;
    return;
   }

   QProgressDialog progress( tr("Uploading file..."),  tr("Cancel"),  0,  this,  "upload progress dialog",
       TRUE );
   connect( ftp, SIGNAL(dataTransferProgress(int,int)),  &progress, SLOT(setProgress(int,int)) );
   connect( ftp, SIGNAL(commandFinished(int,bool)),  &progress, SLOT(reset()) );
   connect( &progress, SIGNAL(cancelled()),  ftp, SLOT(abort()) );

   QFileInfo fi( fileName );
   ftp->put( file, fi.fileName() );
   progress.exec(); // ### takes a lot of time!!!

   ftp->list();
}

void FtpMainWindow::downloadFile()
{
   FtpViewItem *item = (FtpViewItem*)remoteView->selectedItem();
   if ( !item || item->isDir() )
    return;

   QString fileName = QFileDialog::getSaveFileName( item->text(0), QString::null,  this,
       "download file dialog",  tr("Save downloaded file as") );
   if ( fileName.isNull() )
    return;

   // create file on the heap because it has to be valid throughout the whole
   // asynchronous download operation
   QFile *file = new QFile( fileName );
   if ( !file->open( IO_WriteOnly ) ) {
    QMessageBox::critical( this, tr("Download error"),
         tr("Can't open file '%1' for writing.").arg(fileName) );
    delete file;
    return;
   }

   QProgressDialog progress( tr("Downloading file..."), tr("Cancel"), 0, this,
       "download progress dialog", TRUE );
   connect( ftp, SIGNAL(dataTransferProgress(int,int)), &progress, SLOT(setProgress(int,int)) );
   connect( ftp, SIGNAL(commandFinished(int,bool)), &progress, SLOT(reset()) );
   connect( &progress, SIGNAL(cancelled()), ftp, SLOT(abort()) );

   ftp->get( item->text(0), file );
   progress.exec(); // ### takes a lot of time!!!
}

void FtpMainWindow::removeFile()
{
   FtpViewItem *item = (FtpViewItem*)remoteView->selectedItem();
   if ( !item || item->isDir() )
    return;

   ftp->remove( item->text(0) );
   ftp->list();
}

void FtpMainWindow::connectToHost()
{
   ConnectDialog connectDialog;
   if ( connectDialog.exec() == QDialog::Rejected )
    return;

   remotePath->clear();
   remoteView->clear();

   if ( ftp->state() != QFtp::Unconnected )
    ftp->close();

   ftp->connectToHost( connectDialog.host->text(), connectDialog.port->value() );
   ftp->login( connectDialog.username->text(), connectDialog.password->text() );
   ftp->rawCommand( "PWD" );
    ftp->list();
}

// This slot is connected to the QComboBox::activated() signal of the remotePath.
void FtpMainWindow::changePath( const QString &newPath )
{
    ftp->cd( newPath );
    ftp->rawCommand( "PWD" );
    ftp->list();
}

// This slot is connected to the QListView::doubleClicked() and
// QListView::returnPressed() signals of the remoteView.
void FtpMainWindow::changePathOrDownload( QListViewItem *item )
{
    if ( ((FtpViewItem*)item)->isDir() )
     changePath( item->text(0) );
    else
     downloadFile();
}

// Slots connected to signals of the QFtp class**

void FtpMainWindow::ftp_commandStarted()
{
    QApplication::setOverrideCursor( QCursor(Qt::WaitCursor) );
    if ( ftp->currentCommand() == QFtp::List ) {
     remoteView->clear();
     if ( currentFtpDir != "/" )
         new FtpViewItem( remoteView, FtpViewItem::Directory, "..", "", "" );
    }
}

void FtpMainWindow::ftp_commandFinished()
{
    QApplication::restoreOverrideCursor();
    delete ftp->currentDevice();
}

void FtpMainWindow::ftp_done( bool error )
{
    if ( error ) {
     QMessageBox::critical( this, tr("FTP Error"), ftp->errorString() );

     // If we are connected, but not logged in, it is not meaningful to stay
     // connected to the server since the error is a really fatal one (login
     // failed).
     if ( ftp->state() == QFtp::Connected )
         ftp->close();
    }
}

void FtpMainWindow::ftp_stateChanged( int state )
{
    switch ( (QFtp::State)state ) {
     case QFtp::Unconnected:
         stateFtp->setText( tr("Unconnected") );
         break;
     case QFtp::HostLookup:
         stateFtp->setText( tr("Host lookup") );
         break;
     case QFtp::Connecting:
         stateFtp->setText( tr("Connecting") );
         break;
     case QFtp::Connected:
         stateFtp->setText( tr("Connected") );
         break;
     case QFtp::LoggedIn:
         stateFtp->setText( tr("Logged in") );
         break;
     case QFtp::Closing:
         stateFtp->setText( tr("Closing") );
         break;
    }
}

void FtpMainWindow::ftp_listInfo( const QUrlInfo &i )
{
    FtpViewItem::Type type;
    if ( i.isDir() )
     type = FtpViewItem::Directory;
    else
     type = FtpViewItem::File;

    new FtpViewItem( remoteView, type,
         i.name(), QString::number(i.size()), i.lastModified().toString() );
}

void FtpMainWindow::ftp_rawCommandReply( int code, const QString &text )
{
    if ( code == 257 ) {
     currentFtpDir = text.section( '"', 1, 1 );

     for ( int i = 0; i<remotePath->count(); i++ ) {
         // make sure that we don't insert duplicates
         if ( remotePath->text( i ) == currentFtpDir )
           remotePath->removeItem( i );
     }
     remotePath->insertItem( currentFtpDir, 0 );
     remotePath->setCurrentItem( 0 );
    }
}
```

main.cpp

```
#include <qapplication.h>
#include "ftpmainwindow.h"

int main( int argc, char **argv )
```

```
{
   QApplication a( argc, argv );

   FtpMainWindow m;
   a.setMainWidget( &m );
   m.show();
   a.processEvents();
   m.connectToHost();
   return a.exec();
}
```

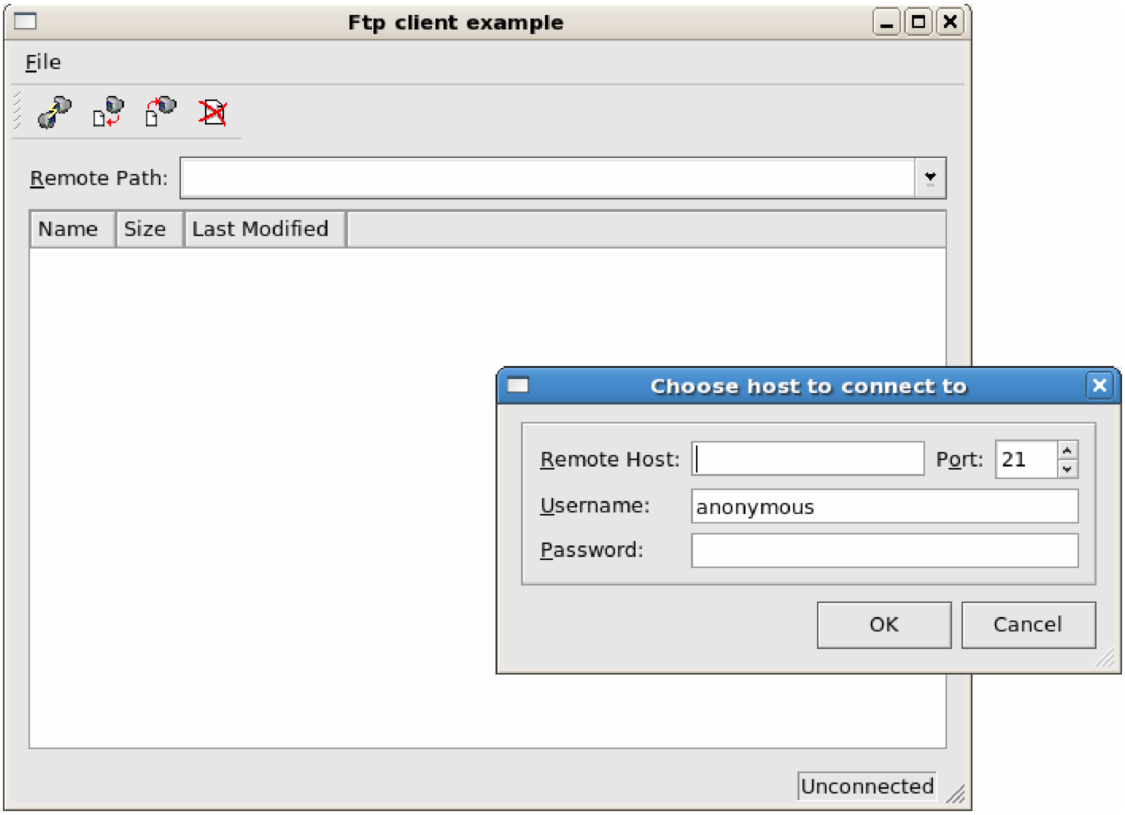
images/file.png

🗋

images/folder.png

📁

실행



Remote Host:
Port: 21
Username: anonymous
Password:

Unconnected

4) 간단한 HTTP 데몬

이 실례는 QServerSocket클라스의 사용법을 보여준다. 이것은 포구 8080을 열어놓고 GET요구를 얻을 때마다 단순한 HTML페지를 송신하는 HTTP데몬의 아주 단순한 실현이다. 페지를 송신한 후에 접속을 닫는다.

httpd.pro

```
TEMPLATE  = app
TARGET     = httpd
```

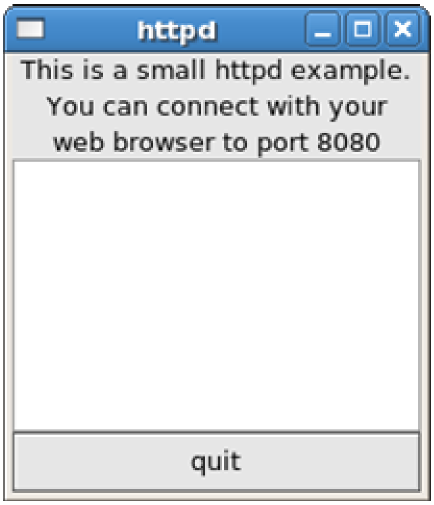
```
CONFIG       += qt warn_on release
HEADERS        =
SOURCES        = httpd.cpp
```

httpd.cpp

```
#include <stdlib.h>
#include <qsocket.h>
#include <qregexp.h>
#include <qserversocket.h>
#include <qapplication.h>
#include <qmainwindow.h>
#include <qtextstream.h>
#include <qvbox.h>
#include <qlabel.h>
#include <qtextview.h>
#include <qpushbutton.h>

// HttpDaemon is the the class that implements the simple HTTP server.
class HttpDaemon : public QServerSocket
{
   Q_OBJECT
public:
   HttpDaemon( QObject* parent=0 ) :
    QServerSocket(8080,1,parent)
   {
    if ( !ok() ) {
       qWarning("Failed to bind to port 8080");
       exit( 1 );
    }
   }

   void newConnection( int socket )
   {
    // When a new client connects, the server constructs a QSocket and all
    // communication with the client is done over this QSocket. QSocket
    // works asynchronouslyl, this means that all the communication is done
    // in the two slots readClient() and discardClient().
    QSocket* s = new QSocket( this );
    connect( s, SIGNAL(readyRead()), this, SLOT(readClient()) );
    connect( s, SIGNAL(delayedCloseFinished()), this, SLOT(discardClient()) );
    s->setSocket( socket );
    emit newConnect();
   }

signals:
   void newConnect();
   void endConnect();
   void wroteToClient();

private slots:
   void readClient()
   {
    // This slot is called when the client sent data to the server. The
    // server looks if it was a get request and sends a very simple HTML
   // document back.
   QSocket* socket = (QSocket*)sender();
   if ( socket->canReadLine() ) {
     QStringList tokens = QStringList::split( QRegExp("[ \r\n][ \r\n]*"), socket->readLine() );
     if ( tokens[0] == "GET" ) {
      QTextStream os( socket );
      os.setEncoding( QTextStream::UnicodeUTF8 );
      os << "HTTP/1.0 200 Ok\r\n"
        "Content-Type: text/html; charset=\"utf-8\"\r\n"
        "\r\n"
        "<h1>Nothing to see here</h1>\n";
      socket->close();
      emit wroteToClient();
     }
   }
  }
  void discardClient()
  {
   QSocket* socket = (QSocket*)sender();
   delete socket;
   emit endConnect();
  }
};

// HttpInfo provides a simple graphical user interface to the server and shows
// the actions of the server.
class HttpInfo : public QVBox
{
  Q_OBJECT
public:
  HttpInfo()
  {
   HttpDaemon *httpd = new HttpDaemon( this );

   QString itext = QString(
       "This is a small httpd example.\n"
       "You can connect with your\n"
       "web browser to port %1"
     ).arg( httpd->port() );
   QLabel *lb = new QLabel( itext, this );
   lb->setAlignment( AlignHCenter );
   infoText = new QTextView( this );
   QPushButton *quit = new QPushButton( "quit" , this );

   connect( httpd, SIGNAL(newConnect()), SLOT(newConnect()) );
   connect( httpd, SIGNAL(endConnect()), SLOT(endConnect()) );
   connect( httpd, SIGNAL(wroteToClient()), SLOT(wroteToClient()) );
   connect( quit, SIGNAL(pressed()), qApp, SLOT(quit()) );
  }

  ~HttpInfo()
  {
  }
```

```
private slots:
    void newConnect()
    {
     infoText->append( "New connection" );
    }
    void endConnect()
    {
     infoText->append( "Connection closed\n\n" );
    }
    void wroteToClient()
    {
     infoText->append( "Wrote to client" );
    }

private:
    QTextView *infoText;
};


int main( int argc, char** argv )
{
    QApplication app( argc, argv );
    HttpInfo info;
    app.setMainWidget( &info );
    info.show();
    return app.exec();
}

#include "httpd.moc"
```

실행



This is a small httpd example.
You can connect with your
web browser to port 8080

5) InfoProtocol

(1) InfoClient

infoclient.pro

```
TEMPLATE  = app
```

```
TARGET       = infoclient
CONFIG       += qt warn_on release
HEADERS          = client.h
SOURCES          = main.cpp \
            client.cpp
INTERFACES    = clientbase.ui
```

client.cpp

```
#include <qsocket.h>
#include <qapplication.h>
#include <qtextedit.h>
#include <qlineedit.h>
#include <qlabel.h>
#include <qpushbutton.h>
#include <qtextstream.h>
#include <qlistbox.h>

#include "client.h"

ClientInfo::ClientInfo( QWidget *parent, const char *name ) :
   ClientInfoBase( parent, name ), socket( 0 )
{
   edHost->setText( "localhost" );
   edPort->setText( QString::number( (uint)infoPort ) );

   connect( infoList, SIGNAL(selected(const QString&)), SLOT(selectItem(const QString&)) );
   connect( btnConnect, SIGNAL(clicked()), SLOT(connectToServer()) );
   connect( btnBack, SIGNAL(clicked()), SLOT(stepBack()) );
   connect( btnQuit, SIGNAL(clicked()), qApp, SLOT(quit()) );
}

void ClientInfo::connectToServer()
{
   delete socket;
   socket = new QSocket( this );
   connect( socket, SIGNAL(connected()), SLOT(socketConnected()) );
   connect( socket, SIGNAL(connectionClosed()), SLOT(socketConnectionClosed()) );
   connect( socket, SIGNAL(readyRead()), SLOT(socketReadyRead()) );
   connect( socket, SIGNAL(error(int)), SLOT(socketError(int)) );

   socket->connectToHost( edHost->text(), edPort->text().toInt() );
}

void ClientInfo::selectItem( const QString& item )
{
   // item in listBox selected, use LIST or GET depending of the node type.
   if ( item.endsWith( "/" ) ) {
    sendToServer( List, infoPath->text() + item );
    infoPath->setText( infoPath->text() + item );
   } else
    sendToServer( Get, infoPath->text() + item );
}

void ClientInfo::stepBack()
```

```
{
   // go back (up) in path hierarchy
   int i = infoPath->text().findRev( '/', -2 );
   if ( i > 0 )
    infoPath->setText( infoPath->text().left( i + 1 ) );
   else
    infoPath->setText( "/" );
   infoList->clear();
   sendToServer( List, infoPath->text() );
}

void ClientInfo::socketConnected()
{
   sendToServer( List, "/" );
}

void ClientInfo::sendToServer( Operation op, const QString& location )
{
   QString line;
   switch (op) {
    case List:
       infoList->clear();
       line = "LIST " + location;
       break;
    case Get:
       line = "GET " + location;
       break;
   }
   infoText->clear();
   QTextStream os(socket);
   os << line << "\r\n";
}

void ClientInfo::socketReadyRead()
{
   QTextStream stream( socket );
   QString line;
   while ( socket->canReadLine() ) {
    line = stream.readLine();
    if ( line.startsWith( "500" ) || line.startsWith( "550" ) ) {
       infoText->append( tr( "error: " ) + line.mid( 4 ) );
    } else if ( line.startsWith( "212+" ) ) {
       infoList->insertItem( line.mid( 6 ) + QString( ( line[ 4 ] == 'D' ) ? "/" : "" ) );
    } else if ( line.startsWith( "213+" ) ) {
       infoText->append( line.mid( 4 ) );
    }
   }
}

void ClientInfo::socketConnectionClosed()
{
   infoText->clear();
   infoText->append( tr( "error: Connection closed by the server\n" ) );
}
```

```
void ClientInfo::socketError( int code )
{
    infoText->clear();
    infoText->append( tr( "error: Error number %1 occurred\n" ).arg( code ) );
}
```

client.h

```
#ifndef CLIENT_H
#define CLIENT_H

#include "clientbase.h"

class QSocket;
class QTextEdit;
class QLineEdit;
class QListBox;
class QLabel;

static const Q_UINT16 infoPort = 42417;

class ClientInfo : public ClientInfoBase
{
    Q_OBJECT
public:
    ClientInfo( QWidget *parent = 0, const char *name = 0 );

private:
    enum Operation { List, Get };

private slots:
    void connectToServer();
    void selectItem( const QString& item );
    void stepBack();
    void sendToServer( Operation op, const QString& location );
    void socketConnected();
    void socketReadyRead();
    void socketConnectionClosed();
    void socketError( int code );

private:
    QSocket *socket;
};

#endif // CLIENT_H
```

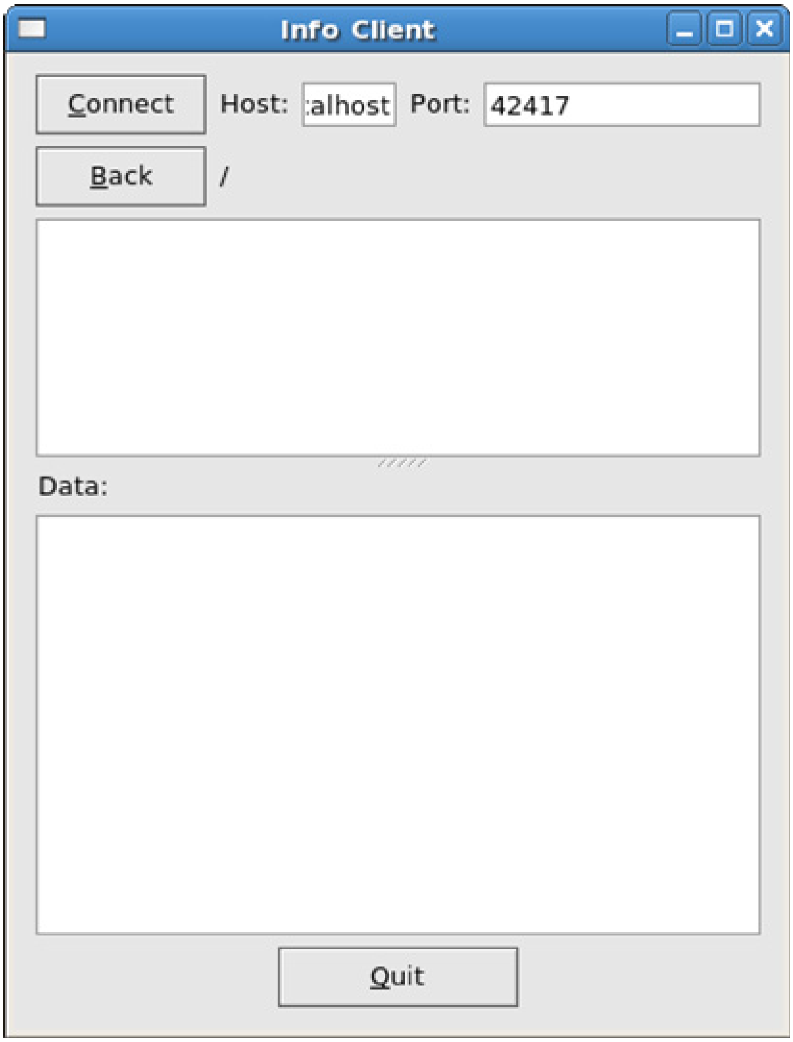
main.cpp

```
#include <qapplication.h>
#include "client.h"

int main( int argc, char** argv )
{
    QApplication app( argc, argv );
    ClientInfo info;
    app.setMainWidget( &info );
    info.show();
    return app.exec();
}
```

clientbase.ui

```
<!DOCTYPE UI><UI version="3.0" stdsetdef="1">
<class>ClientInfoBase</class>
<widget class="QWidget">
    <property name="name">
        <cstring>ClientInfoBase</cstring>
    </property>
…
                </spacer>
            </hbox>
        </widget>
    </vbox>
</widget>
<layoutdefaults spacing="6" margin="11"/>
</UI>
```

실행



(2) Info Server

infoserver.pro

```
TEMPLATE  = app
TARGET      = infoserver
CONFIG      += qt warn_on release
HEADERS         = server.h \
            infodata.h
SOURCES         = main.cpp \
            server.cpp \
            infodata.cpp
INTERFACES     = serverbase.ui
```

infodata.cpp

```
#include "infodata.h"

// we hard code all nodes and data in InfoData class
InfoData::InfoData() :
   nodes( 17, TRUE ), data( 17, TRUE )
```

```
{
  nodes.setAutoDelete(TRUE);
  data.setAutoDelete(TRUE);
  QStringList *item;

  nodes.insert( "/", item = new QStringList( ) );
  (*item) << "D network";
  nodes.insert( "/network/", item = new QStringList() );
  (*item) << "D workstations" << "D printers" << "D fax";
  nodes.insert( "/network/workstations/", item = new QStringList() );
  (*item) << "D nibble" << "D douglas";
  nodes.insert( "/network/workstations/nibble/", item = new QStringList() );
  (*item) << "F os" << "F cpu" << "F memory";
  nodes.insert( "/network/workstations/douglas/", item = new QStringList() );
  (*item) << "F os" << "F cpu" << "F memory";
  nodes.insert( "/network/printers/", item = new QStringList() );
  (*item) << "D overbitt" << "D kroksleiven";
  nodes.insert( "/network/printers/overbitt/", item = new QStringList() );
  (*item) << "D jobs" << "F type";
  nodes.insert( "/network/printers/overbitt/jobs/", item = new QStringList() );
  (*item) << "F job1" << "F job2";
  nodes.insert( "/network/printers/kroksleiven/", item = new QStringList() );
  (*item) << "D jobs" << "F type";
  nodes.insert( "/network/printers/kroksleiven/jobs/", item = new QStringList() );
  nodes.insert( "/network/fax/", item = new QStringList() );
  (*item) << "F last_number";

  data.insert( "/network/workstations/nibble/os", new QString( "Linux" ) );
  data.insert( "/network/workstations/nibble/cpu", new QString( "AMD Athlon 1000" ) );
  data.insert( "/network/workstations/nibble/memory", new QString( "256 MB" ) );
  data.insert( "/network/workstations/douglas/os", new QString( "Windows 2000" ) );
  data.insert( "/network/workstations/douglas/cpu", new QString( "2 x Intel Pentium III 800" ) );
  data.insert( "/network/workstations/douglas/memory", new QString( "256 MB" ) );
  data.insert( "/network/printers/overbitt/type", new QString( "Lexmark Optra S 1255 PS" ) );
  data.insert( "/network/printers/overbitt/jobs/job1",
         new QString( "Qt manual\n" "A4 size\n" "3000 pages" ) );
  data.insert( "/network/printers/overbitt/jobs/job2",
      new QString( "Monthly report\n" "Letter size\n" "24 pages\n" "8 copies" ) );
  data.insert( "/network/printers/kroksleiven/type", new QString( "HP C LaserJet 4500-PS" ) );
  data.insert( "/network/fax/last_number", new QString( "22 22 22 22" ) );
}

QStringList InfoData::list( QString path, bool *found ) const
{
  if ( !path.endsWith( "/" ) )
   path += "/";
  if ( !path.startsWith( "/" ) )
   path = "/" + path;
  QStringList *list = nodes[ path ];
  if ( list ) {
   *found = TRUE;
   return *list;
  } else {
   *found = FALSE;
```

```
     QStringList empty;
     return empty;
   }
}

QString InfoData::get( QString path, bool *found ) const
{
   if ( !path.startsWith( "/" ) )
     path = "/" + path;
   QString *file = data[ path ];
   if ( file ) {
     *found = TRUE;
     return *file;
   } else {
     *found = FALSE;
     QString empty;
     return empty;
   }
}
```

infodata.h

```
#ifndef INFODATA_H
#define INFODATA_H
#include <qdict.h>
#include <qstringlist.h>

// The InfoData class manages data, organized in tree structure.
class InfoData
{
public:
   InfoData();
   QStringList list( QString path, bool *found ) const;
   QString get( QString path, bool *found ) const;

private:
   QDict< QStringList > nodes;
   QDict< QString > data;
};

#endif // INFODATA_H
```

server.cpp

```
#include <qtextview.h>
#include <qpushbutton.h>
#include <qtextstream.h>
#include <qapplication.h>
#include <qmessagebox.h>
#include <stdlib.h>
#include "server.h"

ServerInfo::ServerInfo( Q_UINT16 port, QWidget *parent, const char *name ) :
   ServerInfoBase( parent, name )
{
   SimpleServer *server = new SimpleServer( port, this, "simple server" );
```

```
    connect( server, SIGNAL(newConnect()), SLOT(newConnect()) );
    connect( btnQuit, SIGNAL(clicked()), qApp, SLOT(quit()) );
}

void ServerInfo::newConnect()
{
    infoText->append( tr( "New connection\n" ) );
}

SimpleServer::SimpleServer( Q_UINT16 port, QObject* parent, const char *name ) :
    QServerSocket( port, 1, parent, name )
{
    if ( !ok() ) {
     QMessageBox::critical( 0, tr( "Error" ), tr( "Failed to bind to port %1" ).arg( port ) );
     exit(1);
    }
}

void SimpleServer::newConnection( int socket )
{
    (void)new ClientSocket( socket, &info, this, "client socket" );
    emit newConnect();
}


ClientSocket::ClientSocket( int sock, InfoData *i, QObject *parent, const char *name ) :
    QSocket( parent, name ), info( i )
{
    connect( this, SIGNAL(readyRead()), SLOT(readClient()) );
    connect( this, SIGNAL(connectionClosed()), SLOT(connectionClosed()) );
    setSocket( sock );
}

void ClientSocket::readClient()
{
    QTextStream stream( this );
    QStringList answer;
    while ( canReadLine() ) {
     stream << processCommand( stream.readLine() );
    }
}

QString ClientSocket::processCommand( const QString& command )
{
    QString answer;
    QString com = command.simplifyWhiteSpace ();
    if ( com.startsWith( "LIST" ) ) {
     bool ok;
     QStringList nodes = info->list( com.mid( 5 ), &ok );
     if ( ok ) {
         for ( QStringList::Iterator it = nodes.begin(); it != nodes.end(); ++it )
          answer += "212+" + *it + "\r\n";
         answer += "212 \r\n";
     } else
```

```
        answer += "550 Invalid path\r\n";
    } else if ( com.startsWith( "GET " ) ) {
      bool ok;
      QStringList data = QStringList::split( '\n', info->get( com.mid( 4 ), &ok ), TRUE );
      if ( ok ) {
        for ( QStringList::Iterator it = data.begin(); it != data.end(); ++it )
          answer += "213+" + *it + "\r\n";
        answer += "213 \r\n";
      } else
        answer += "550 Info not found\r\n";
    } else
      answer += "500 Syntax error\r\n";

    return answer;
}

void ClientSocket::connectionClosed()
{
    delete this;
}
```

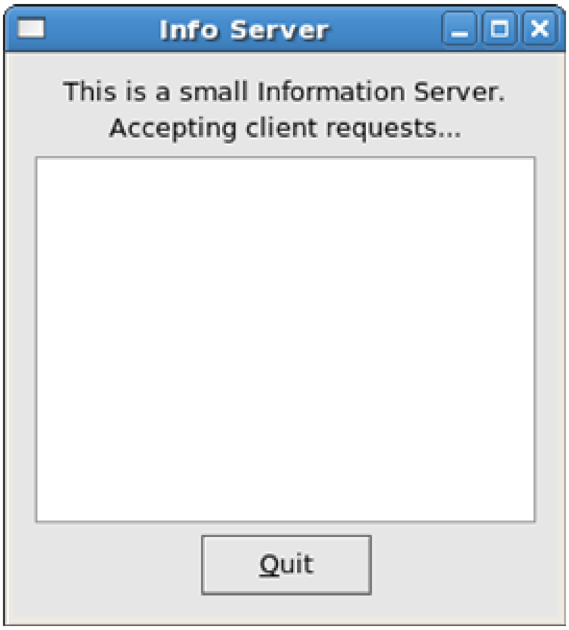
server.h

```
#ifndef SERVER_H
#define SERVER_H
#include <qsocket.h>
#include <qserversocket.h>
#include "infodata.h"
#include "serverbase.h"

static const Q_UINT16 infoPort = 42417;

/*
  The ServerInfo class provides a small GUI for the server. It also creates the
  SimpleServer and as a result the server.
*/
class ServerInfo : public ServerInfoBase
{
    Q_OBJECT
public:
    ServerInfo( Q_UINT16 port = infoPort, QWidget *parent = 0, const char *name = 0 );

private slots:
    void newConnect();
};

class SimpleServer : public QServerSocket
{
    Q_OBJECT
public:
    SimpleServer( Q_UINT16 port = infoPort, QObject* parent = 0, const char *name = 0 );
    void newConnection( int socket );

signals:
    void newConnect();
private:
   InfoData info;
};

class ClientSocket : public QSocket
{
   Q_OBJECT
public:
   ClientSocket( int sock, InfoData *i, QObject *parent = 0, const char *name = 0 );

private slots:
   void readClient();
   void connectionClosed();

private:
   QString processCommand( const QString& command );
   InfoData *info;
};

#endif //SERVER_H
```

serverbase.ui

```
<!DOCTYPE UI><UI version="3.0" stdsetdef="1">
<class>ServerInfoBase</class>
<widget class="QWidget">
   <property name="name">
      <cstring>ServerInfoBase</cstring>
…
   </vbox>
</widget>
<layoutdefaults spacing="6" margin="11"/>
</UI>
```

실행



Info Server
This is a small Information Server.
Accepting client requests...
Quit

(3) infourlclient

infourlclient.pro

```
TEMPLATE  = app
TARGET    = infourlclient
CONFIG    += qt warn_on release
HEADERS       = client.h \
          qip.h
SOURCES       = main.cpp \
          client.cpp \
          qip.cpp
INTERFACES    = clientbase.ui
```

client.cpp

```
#include <qapplication.h>
#include <qtextedit.h>
#include <qpushbutton.h>
#include <qfiledialog.h>
#include "qip.h"
#include "client.h"

ClientInfo::ClientInfo( QWidget *parent, const char *name ) :
   ClientInfoBase( parent, name )
{
   connect( btnOpen, SIGNAL(clicked()), SLOT(downloadFile()) );
   connect( btnQuit, SIGNAL(clicked()), qApp, SLOT(quit()) );
   connect( &op, SIGNAL( data( const QByteArray &, QNetworkOperation * ) ),
        this, SLOT( newData( const QByteArray & ) ) );
}

void ClientInfo::downloadFile()
{
   // under Windows you must not use the native file dialog
   QString file = getOpenFileName();
   if ( !file.isEmpty() ) {
    infoText->clear();
    // download the data
    op = file;
    op.get();
   }
}

QString ClientInfo::getOpenFileName()
{
   static QString workingDirectory( "qip://localhost/" );

   QFileDialog dlg( workingDirectory, QString::null, 0, 0, TRUE );
   dlg.setCaption( QFileDialog::tr( "Open" ) );
   dlg.setMode( QFileDialog::ExistingFile );
   QString result;
   if ( dlg.exec() == QDialog::Accepted ) {
    result = dlg.selectedFile();
    workingDirectory = dlg.url();
   }
   return result;
}

void ClientInfo::newData( const QByteArray &ba )
{
   infoText->append( QString::fromUtf8( ba ) );
}
```

client.h

```
#ifndef CLIENT_H
#define CLIENT_H

#include <qurloperator.h>
#include "clientbase.h"

class ClientInfo : public ClientInfoBase
{
    Q_OBJECT

public:
    ClientInfo( QWidget *parent = 0, const char *name = 0 );

private slots:
    void downloadFile();
    void newData( const QByteArray &ba );

private:
    QUrlOperator op;
    QString getOpenFileName();
};

#endif // CLIENT_H
```

qip.cpp

```
#include <qsocket.h>
#include <qurlinfo.h>
#include <qurloperator.h>
#include <qtextstream.h>
#include "qip.h"

Qip::Qip()
{
   state = Start;
   socket = new QSocket( this );
   connect( socket, SIGNAL(connected()), SLOT(socketConnected()) );
   connect( socket, SIGNAL(connectionClosed()), SLOT(socketConnectionClosed()) );
   connect( socket, SIGNAL(readyRead()), SLOT(socketReadyRead()) );
   connect( socket, SIGNAL(error(int)), SLOT(socketError(int)) );
}

int Qip::supportedOperations() const
{
   return OpListChildren | OpGet;
}

bool Qip::checkConnection( QNetworkOperation * )
{
   if ( socket->isOpen() )
    return TRUE;

   // don't call connectToHost() if we are already trying to connect
   if ( socket->state() == QSocket::Connecting )
    return FALSE;

   socket->connectToHost( url()->host(), url()->port() != -1 ? url()->port() : infoPort );
   return FALSE;
}

void Qip::operationListChildren( QNetworkOperation * )
{
   QTextStream os(socket);
   os << "LIST " + url()->path() + "\r\n";
   operationInProgress()->setState( StInProgress );
}

void Qip::operationGet( QNetworkOperation * )
{
   QTextStream os(socket);
   os << "GET " + url()->path() + "\r\n";
   operationInProgress()->setState( StInProgress );
}

void Qip::socketConnected()
{
   if ( url() )
    emit connectionStateChanged( ConConnected, tr( "Connected to host %1" ).arg( url()->host() ) );
   else
    emit connectionStateChanged( ConConnected, tr ("Connected to host" ) );
}

void Qip::socketConnectionClosed()
{
   if ( url() )
    emit connectionStateChanged( ConClosed, tr( "Connection to %1 closed" ).arg( url()->host() ) );
   else
    emit connectionStateChanged( ConClosed, tr ("Connection closed" ) );
}

void Qip::socketError( int code )
{
   if ( code == QSocket::ErrHostNotFound ||
     code == QSocket::ErrConnectionRefused ) {
    error( ErrHostNotFound, tr( "Host not found or couldn't connect to: %1\n" ).arg( url()->host() ) );
   } else
    error( ErrUnsupported, tr( "Error" ) );
}

void Qip::socketReadyRead()
{
   // read from the server
   QTextStream stream( socket );
   QString line;
   while ( socket->canReadLine() ) {
    line = stream.readLine();
    if ( line.startsWith( "500" ) ) {
       error( ErrValid, line.mid( 4 ) );
    } else if ( line.startsWith( "550" ) ) {
       error( ErrFileNotExisting, line.mid( 4 ) );
    } else if ( line.startsWith( "212+" ) ) {
       if ( state != List ) {
        state = List;
          emit start( operationInProgress() );
       }
       QUrlInfo inf;
       inf.setName( line.mid( 6 ) + QString( ( line[ 4 ] == 'D' ) ? "/" : "" ) );
       inf.setDir( line[ 4 ] == 'D' );
       inf.setSymLink( FALSE );
       inf.setFile( line[ 4 ] == 'F' );
       inf.setWritable( FALSE );
       inf.setReadable( TRUE );
       emit newChild( inf, operationInProgress() );
    } else if ( line.startsWith( "213+" ) ) {
       state = Data;
       emit data( line.mid( 4 ).utf8(), operationInProgress() );
    }
    if( line[3] == ' ' && state != Start) {
       state = Start;
       operationInProgress()->setState( StDone );
       emit finished( operationInProgress() );
    }
```

```
  }
}

void Qip::error( int code, const QString& msg )
{
  if ( operationInProgress() ) {
    operationInProgress()->setState( StFailed );
    operationInProgress()->setErrorCode( code );
    operationInProgress()->setProtocolDetail( msg );
    clearOperationQueue();
    emit finished( operationInProgress() );
  }
  state = Start;
}
```

qip.h

```
#ifndef QIP_H
#define QIP_H
#include <qnetworkprotocol.h>

class QSocket;
static const Q_UINT16 infoPort = 42417;

class Qip : public QNetworkProtocol
{
  Q_OBJECT

public:
  Qip();
  virtual int supportedOperations() const;

protected:
  virtual void operationListChildren( QNetworkOperation *op );
  virtual void operationGet( QNetworkOperation *op );
  virtual bool checkConnection( QNetworkOperation *op );

private slots:
  void socketConnected();
  void socketReadyRead();
  void socketConnectionClosed();
  void socketError( int code );

private:
  QSocket *socket;
  enum State { Start, List, Data } state;
  void error( int code, const QString& msg );
};

#endif // QIP_H
```

main.cpp

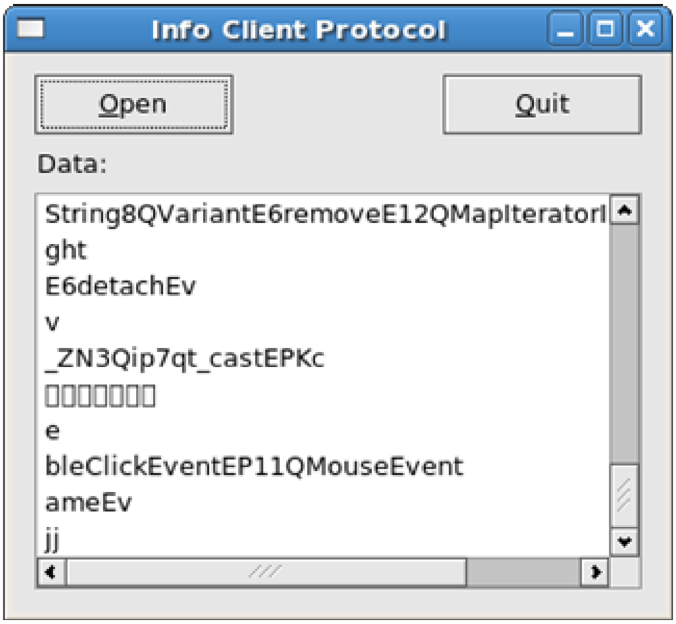
```
#include <qapplication.h>
#include "qip.h"
#include "client.h"
```

```
int main( int argc, char** argv )
{
   QApplication app( argc, argv );
   QNetworkProtocol::registerNetworkProtocol( "qip", new QNetworkProtocolFactory< Qip > );
   ClientInfo info;
   app.setMainWidget( &info );
   info.show();
   return app.exec();
}
```

clientbase.ui

```
<!DOCTYPE UI><UI version="3.0" stdsetdef="1">
<class>ClientInfoBase</class>
<widget class="QWidget">
   <property name="name">
…
      </widget>
   </vbox>
</widget>
<layoutdefaults spacing="6" margin="11"/>
</UI>
```

실행



```
String8QVariantE6removeE12QMapIteratorI
ght
E6detachEv
v
_ZN3Qip7qt_castEPKc
□□□□□□□□
e
bleClickEventEP11QMouseEvent
ameEv
jj
```

6) 간단한 우편의뢰기

이 실례는 QSocket클라스의 사용법을 보여준다. 의뢰기는 우편을 송신하는데만 사용될수 있다. 흥미있는 부분은 SMTP통신규약의 실현이다.

mail.pro

```
TEMPLATE  = app
TARGET    = mail
CONFIG    += qt warn_on release
```

```
HEADERS         = composer.h \
            smtp.h
SOURCES         = composer.cpp \
            main.cpp \
            smtp.cpp
```

composer.cpp

```
#include "composer.h"
#include "smtp.h"
#include <qlineedit.h>
#include <qmultilineedit.h>
#include <qpushbutton.h>
#include <qlabel.h>
#include <qlayout.h>

Composer::Composer( QWidget *parent )  : QWidget( parent )
{
   QGridLayout * layout = new QGridLayout( this, 1, 1, 6 );

   layout->addWidget( new QLabel( tr( "From:" ), this ), 0, 0 );
   from = new QLineEdit( this );
   layout->addWidget( from, 0, 1 );

   layout->addWidget( new QLabel( tr( "To:" ), this ), 1, 0 );
   to = new QLineEdit( this );
   layout->addWidget( to, 1, 1 );

   layout->addWidget( new QLabel( tr( "Subject:" ), this ), 2, 0 );
   subject = new QLineEdit( this );
   layout->addWidget( subject, 2, 1 );

   message = new QMultiLineEdit( this );
   layout->addMultiCellWidget( message, 3, 3, 0, 1 );

   send = new QPushButton( tr( "&Send" ), this );
   layout->addWidget( send, 4, 0 );
   connect( send, SIGNAL( clicked() ), this, SLOT( sendMessage() ) );

   sendStatus = new QLabel( this );
   layout->addWidget( sendStatus, 4, 1 );
}

void Composer::sendMessage()
{
   send->setEnabled( FALSE );
   sendStatus->setText( tr( "Looking up mail servers" ) );
   Smtp *smtp = new Smtp( from->text(), to->text(),
                subject->text(),
                message->text() );
   connect( smtp, SIGNAL(destroyed()),
        this, SLOT(enableSend()) );
   connect( smtp, SIGNAL(status(const QString &)),
        sendStatus, SLOT(setText(const QString &)) );
}
```

```
void Composer::enableSend()
{
   send->setEnabled( TRUE );
}
```

composer.h

```
#ifndef COMPOSER_H
#define COMPOSER_H

#include <qwidget.h>

class QLineEdit;
class QMultiLineEdit;
class QLabel;
class QPushButton;

class Composer : public QWidget
{
   Q_OBJECT

public:
   Composer( QWidget *parent = 0 );

private slots:
   void sendMessage();
   void enableSend();

private:
   QLineEdit *from, *to, *subject;
   QMultiLineEdit *message;
   QLabel * sendStatus;
   QPushButton * send;
};

#endif
```

smtp.cpp

```
#include "smtp.h"
#include <qtextstream.h>
#include <qsocket.h>
#include <qdns.h>
#include <qtimer.h>
#include <qapplication.h>
#include <qmessagebox.h>
#include <qregexp.h>

Smtp::Smtp( const QString &from, const QString &to,
        const QString &subject,
        const QString &body )
{
   socket = new QSocket( this );
   connect ( socket, SIGNAL( readyRead() ),  this, SLOT( readyRead() ) );
   connect ( socket, SIGNAL( connected() ),  this, SLOT( connected() ) );
```

```
    mxLookup = new QDns( to.mid( to.find( '@' )+1 ), QDns::Mx );
    connect( mxLookup, SIGNAL(resultsReady()), this, SLOT(dnsLookupHelper()) );

    message = QString::fromLatin1( "From: " ) + from +  QString::fromLatin1( "\nTo: " ) + to +
          QString::fromLatin1( "\nSubject: " ) + subject + QString::fromLatin1( "\n\n" ) + body + "\n";
    message.replace( QString::fromLatin1( "\n" ),  QString::fromLatin1( "\r\n" ) );
    message.replace( QString::fromLatin1( "\r\n.\r\n" ),  QString::fromLatin1( "\r\n..\r\n" ) );

    this->from = from;
    rcpt = to;

    state = Init;
}

Smtp::~Smtp()
{
    delete t;
    delete socket;
}

void Smtp::dnsLookupHelper()
{
    QValueList<QDns::MailServer> s = mxLookup->mailServers();
    if ( s.isEmpty() ) {
     if ( !mxLookup->isWorking() )
         emit status( tr( "Error in MX record lookup" ) );
     return;
    }

    emit status( tr( "Connecting to %1" ).arg( s.first().name ) );

    socket->connectToHost( s.first().name, 25 );
    t = new QTextStream( socket );
}

void Smtp::connected()
{
    emit status( tr( "Connected to %1" ).arg( socket->peerName() ) );
}

void Smtp::readyRead()
{
    // SMTP is line-oriented
    if ( !socket->canReadLine() )
     return;

    QString responseLine;
    do {
     responseLine = socket->readLine();
     response += responseLine;
    } while( socket->canReadLine() && responseLine[3] != ' ' );
    responseLine.truncate( 3 );
```

```
   if ( state == Init && responseLine[0] == '2' ) {
    // banner was okay, let's go on
    *t << "HELO there\r\n";
    state = Mail;
   } else if ( state == Mail && responseLine[0] == '2' ) {
    // HELO response was okay (well, it has to be)
    *t << "MAIL FROM: <" << from << ">\r\n";
    state = Rcpt;
   } else if ( state == Rcpt && responseLine[0] == '2' ) {
    *t << "RCPT TO: <" << rcpt << ">\r\n";
    state = Data;
   } else if ( state == Data && responseLine[0] == '2' ) {
    *t << "DATA\r\n";
    state = Body;
   } else if ( state == Body && responseLine[0] == '3' ) {
    *t << message << ".\r\n";
    state = Quit;
   } else if ( state == Quit && responseLine[0] == '2' ) {
    *t << "QUIT\r\n";
    // here, we just close.
    state = Close;
    emit status( tr( "Message sent" ) );
   } else if ( state == Close ) {
    deleteLater();
    return;
   } else {
    // something broke.
    QMessageBox::warning( qApp->activeWindow(),  tr( "Qt Mail Example" ),
                tr( "Unexpected reply from SMTP server:\n\n" ) +  response );
    state = Close;
   }

   response = "";
}
```

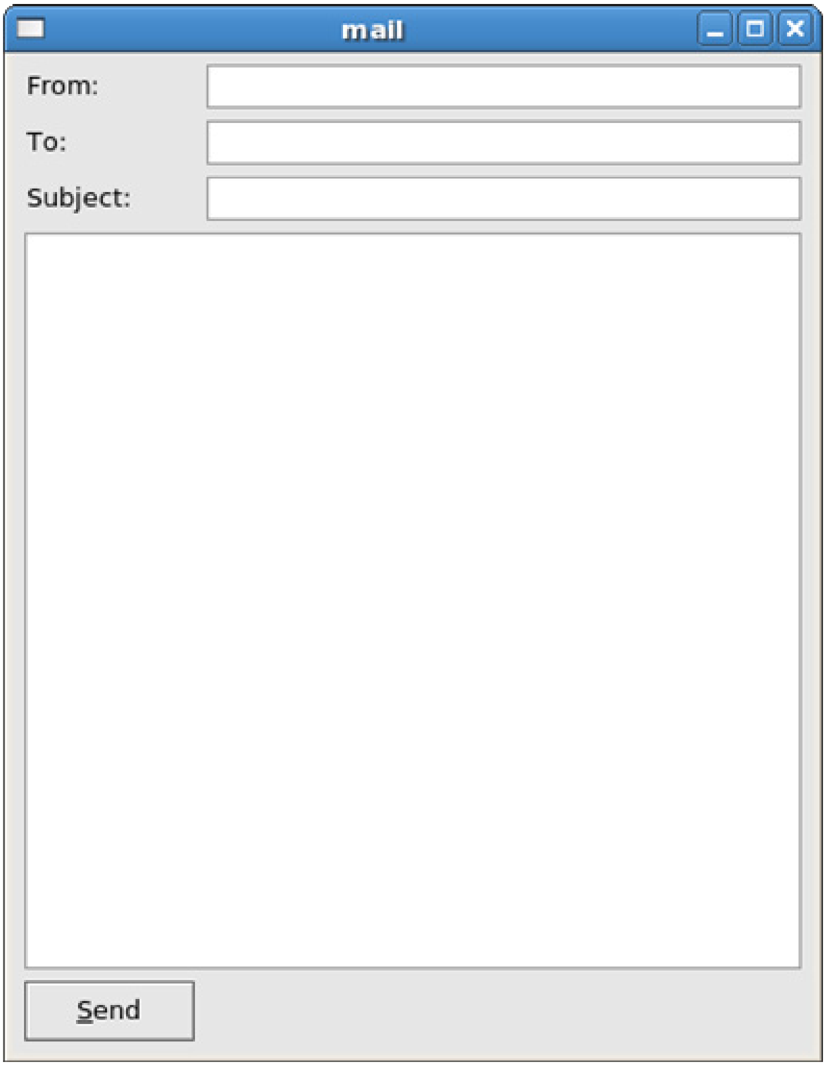
smtp.h

```
#ifndef SMTP_H
#define SMTP_H

#include <qobject.h>
#include <qstring.h>

class QSocket;
class QTextStream;
class QDns;

class Smtp : public QObject
{
   Q_OBJECT

public:
   Smtp( const QString &from, const QString &to, const QString &subject, const QString &body );
   ~Smtp();
signals:
   void status( const QString & );

private slots:
   void dnsLookupHelper();
   void readyRead();
   void connected();

private:
   enum State {   Init,    Mail,   Rcpt,   Data,   Body,  Quit,   Close   };

   QString message;
   QString from;
   QString rcpt;
   QSocket *socket;
   QTextStream * t;
   int state;
   QString response;
   QDns * mxLookup;
};

#endif
```

main.cpp

```
#include <qapplication.h>
#include "composer.h"

int main( int argc, char **argv )
{
   QApplication a( argc, argv );

   Composer c;
   a.setMainWidget( &c );
   c.resize( 400, 500 );
   c.show();
   return a.exec();
}
```

실행



7) 간단한 NNTP 실현

이 실례는 자체의 QNetworkProtocol을 실현하는 방법을 보여준다. 이 실례에서 선택한 통신규약은 NTTP이다. 이 실현은 실례에 맞게 설계되였으므로 아주 단순하다. 실제의 NNTP 실현으로서 사용하지 말아야 한다.

networkprotocol.pro

```
TEMPLATE  = app
TARGET    = networkprotocol
CONFIG    += qt warn_on release
HEADERS       = nntp.h view.h
SOURCES       = main.cpp \
        nntp.cpp view.cpp
```

nntp.h

```
#ifndef NNTP_H
#define NNTP_H
```

```
#include <qsocket.h>
#include <qnetworkprotocol.h>

class Nntp : public QNetworkProtocol
{
   Q_OBJECT

public:
   Nntp();
   virtual ~Nntp();
   virtual int supportedOperations() const;

protected:
   virtual void operationListChildren( QNetworkOperation *op );
   virtual void operationGet( QNetworkOperation *op );

   QSocket *commandSocket;
   bool connectionReady;
   bool readGroups;
   bool readArticle;

private:
   bool checkConnection( QNetworkOperation *op );
   void close();
   void parseGroups();
   void parseArticle();

protected slots:
   void hostFound();
   void connected();
   void closed();
   void readyRead();
   void error( int );

};

#endif
```

nntp.cpp

```
#include "nntp.h"
#include <qurlinfo.h>
#include <stdlib.h>
#include <qurloperator.h>
#include <qstringlist.h>
#include <qregexp.h>

Nntp::Nntp()
   : QNetworkProtocol(), connectionReady( FALSE ),
     readGroups( FALSE ), readArticle( FALSE )
{
   // create the command socket and connect to its signals
   commandSocket = new QSocket( this );
   connect( commandSocket, SIGNAL( hostFound() ),  this, SLOT( hostFound() ) );
```

```
    connect( commandSocket, SIGNAL( connected() ),  this, SLOT( connected() ) );
    connect( commandSocket, SIGNAL( connectionClosed() ),  this, SLOT( closed() ) );
    connect( commandSocket, SIGNAL( readyRead() ),  this, SLOT( readyRead() ) );
    connect( commandSocket, SIGNAL( error( int ) ),  this, SLOT( error( int ) ) );
}

Nntp::~Nntp()
{
    close();
    delete commandSocket;
}

void Nntp::operationListChildren( QNetworkOperation * )
{
    // create a command
    QString path = url()->path(), cmd;
    if ( path.isEmpty() || path == "/" ) {
     // if the path is empty or we are in the root dir,
     // we want to read the list of available newsgroups
     cmd = "list newsgroups\r\n";
    } else if ( url()->isDir() ) {
     // if the path is a directory (in our case a news group)
     // we want to list the articles of this group
     path = path.replace( "/", "" );
     cmd = "listgroup " + path + "\r\n";
    } else
     return;

    // write the command to the socket
    commandSocket->writeBlock( cmd.latin1(), cmd.length() );
    readGroups = TRUE;
}

void Nntp::operationGet( QNetworkOperation *op )
{
    // get the dirPath of the URL (this is our news group)
    // and the filename (which is the article we want to read)
    QUrl u( op->arg( 0 ) );
    QString dirPath = u.dirPath(), file = u.fileName();
    dirPath = dirPath.replace( "/", "" );

    // go to the group in which the article is
    QString cmd;
    cmd = "group " + dirPath + "\r\n";
    commandSocket->writeBlock( cmd.latin1(), cmd.length() );

    // read the head of the article
    cmd = "article " + file + "\r\n";
    commandSocket->writeBlock( cmd.latin1(), cmd.length() );
    readArticle = TRUE;
}

bool Nntp::checkConnection( QNetworkOperation * )
{
    // we are connected, return TRUE
    if ( commandSocket->isOpen() && connectionReady )
      return TRUE;

    // seems that there is no chance to connect
    if ( commandSocket->isOpen() )
      return FALSE;

    // don't call connectToHost() if we are already trying to connect
    if ( commandSocket->state() == QSocket::Connecting )
      return FALSE;

    // start connecting
    connectionReady = FALSE;
    commandSocket->connectToHost( url()->host(),
                      url()->port() != -1 ? url()->port() : 119 );
    return FALSE;
}

void Nntp::close()
{
    // close the command socket
    if ( commandSocket->isOpen() ) {
      commandSocket->writeBlock( "quit\r\n", strlen( "quit\r\n" ) );
      commandSocket->close();
    }
}

int Nntp::supportedOperations() const
{
    // we only support listing children and getting data
    return OpListChildren | OpGet;
}

void Nntp::hostFound()
{
    if ( url() )
      emit connectionStateChanged( ConHostFound, tr( "Host %1 found" ).arg( url()->host() ) );
    else
      emit connectionStateChanged( ConHostFound, tr( "Host found" ) );
}

void Nntp::connected()
{
    if ( url() )
      emit connectionStateChanged( ConConnected, tr( "Connected to host %1" ).arg( url()->host() ) );
    else
      emit connectionStateChanged( ConConnected, tr( "Connected to host" ) );
}

void Nntp::closed()
{
    if ( url() )
      emit connectionStateChanged( ConClosed, tr( "Connection to %1 closed" ).arg( url()->host() ) );
    else
     emit connectionStateChanged( ConClosed, tr( "Connection closed" ) );
}

void Nntp::readyRead()
{
    // new data arrived on the command socket

    // of we should read the list of available groups, let's do so
    if ( readGroups ) {
     parseGroups();
     return;
    }

    // of we should read an article, let's do so
    if ( readArticle ) {
     parseArticle();
     return;
    }

    // read the new data from the socket
    QCString s;
    s.resize( commandSocket->bytesAvailable() + 1 );
    commandSocket->readBlock( s.data(), commandSocket->bytesAvailable() );

    if ( !url() )
     return;

    // of the code of the server response was 200, we know that the
    // server is ready to get commands from us now
    if ( s.left( 3 ) == "200" )
     connectionReady = TRUE;
}

void Nntp::parseGroups()
{
    if ( !commandSocket->canReadLine() )
     return;

    // read one line after the other
    while ( commandSocket->canReadLine() ) {
     QString s = commandSocket->readLine();

     // if the  line starts with a dot, all groups or articles have been listed,
     // so we finished processing the listChildren() command
     if ( s[ 0 ] == '.' ) {
         readGroups = FALSE;
         operationInProgress()->setState( StDone );
         emit finished( operationInProgress() );
         return;
     }

     // if the code of the server response is 215 or 211
     // the next line will be the first group or article (depending on what we read).
     // So let others know that we start reading now...
     if ( s.left( 3 ) == "215" || s.left( 3 ) == "211" ) {
         operationInProgress()->setState( StInProgress );
         emit start( operationInProgress() );
         continue;
     }

     // parse the line and create a QUrlInfo object
     // which describes the child (group or article)
     bool tab = s.find( '\t' ) != -1;
     QString group = s.mid( 0, s.find( tab ? '\t' : ' ' ) );
     QUrlInfo inf;
     inf.setName( group );
     QString path = url()->path();
     inf.setDir( path.isEmpty() || path == "/" );
     inf.setSymLink( FALSE );
     inf.setFile( !inf.isDir() );
     inf.setWritable( FALSE );
     inf.setReadable( TRUE );

     // let others know about our new child
     emit newChild( inf, operationInProgress() );
    }

}

void Nntp::parseArticle()
{
    if ( !commandSocket->canReadLine() )
     return;

    // read an article one line after the other
    while ( commandSocket->canReadLine() ) {
     QString s = commandSocket->readLine();

     // if the  line starts with a dot, we finished reading something
     if ( s[ 0 ] == '.' ) {
         readArticle = FALSE;
         operationInProgress()->setState( StDone );
         emit finished( operationInProgress() );
         return;
     }

     if ( s.right( 1 ) == "\n" )
         s.remove( s.length() - 1, 1 );

     // emit the new data of the article which we read
     emit data( QCString( s.ascii() ), operationInProgress() );
    }
}

void Nntp::error( int code )
{
    if ( code == QSocket::ErrHostNotFound ||
      code == QSocket::ErrConnectionRefused ) {
     // this signal is called if connecting to the server failed
     if ( operationInProgress() ) {
         QString msg = tr( "Host not found or couldn't connect to: \n" + url()->host() );
         operationInProgress()->setState( StFailed );
         operationInProgress()->setProtocolDetail( msg );
         operationInProgress()->setErrorCode( (int)ErrHostNotFound );
         clearOperationQueue();
         emit finished( operationInProgress() );
     }
    }
}
```

view.cpp

```
#include "view.h"
#include <qlabel.h>
#include <qpushbutton.h>
#include <qmultilineedit.h>
#include <qfiledialog.h>

View::View()
   : QVBox()
{
   // setup the GUI
   setSpacing( 5 );
   setMargin( 5 );

   QLabel *l = new QLabel( this );
   l->setAlignment( Qt::WordBreak ),
   l->setText( tr( "The button below opens the QFileDialog and you "
            "can choose a file then which is downloaded and "
            "opened below then. You can use for that the <b>local "
            "filesystem</b> using the file protocol, you can download "
            "files from an <b>FTP</b> server using the ftp protocol and "
            "you can download and open <b>USENET</b> articles using the "
            "demo implementation of the nntp protocol of this "
            "example (<i>This implementation of the nntp protocol is a very "
            "basic and incomplete one, so you need to connect to a news server "
            "which allows reading without authentification</i>)\n"
            "To open a file from the local filesystem, enter in the "
            "path combobox of the file dialog a url starting with file "
            "(like <u>file:/usr/bin</u>), to download something from an FTP "
            "server, use something like <u>ftp://ftp.trolltech.com</u> as url, and "
            "for downloading a news article start with an url like "
            "<u>nntp://news.tu-graz.ac.at</u> " ) );
   QPushButton *b = new QPushButton( tr( "Open a file..." ), this );
   connect( b, SIGNAL( clicked() ),
        this, SLOT( downloadFile() ) );

   fileView = new QMultiLineEdit( this );
   fileView->setReadOnly( TRUE );

   // if new data comes in, display it
   connect( &op, SIGNAL( data( const QByteArray &, QNetworkOperation * ) ),
        this, SLOT( newData( const QByteArray & ) ) );
}

void View::downloadFile()
{
    // QString file = QFileDialog::getOpenFileName();
    // under Windows you must not use the native file dialog
    QString file = getOpenFileName();
    if ( !file.isEmpty() ) {
     // clear the view
     fileView->clear();

     // download the data
     op = file;
     op.get();
    }
}

QString View::getOpenFileName()
{
    static QString workingDirectory = QDir::currentDirPath();

    QFileDialog *dlg = new QFileDialog( workingDirectory,  QString::null, 0, 0, TRUE );
    dlg->setCaption( QFileDialog::tr( "Open" ) );
    dlg->setMode( QFileDialog::ExistingFile );
    QString result;
    if ( dlg->exec() == QDialog::Accepted ) {
     result = dlg->selectedFile();
     workingDirectory = dlg->url();
    }
    delete dlg;
    return result;
}

void View::newData( const QByteArray &ba )
{
    // append new data
    fileView->append( ba );
}
```

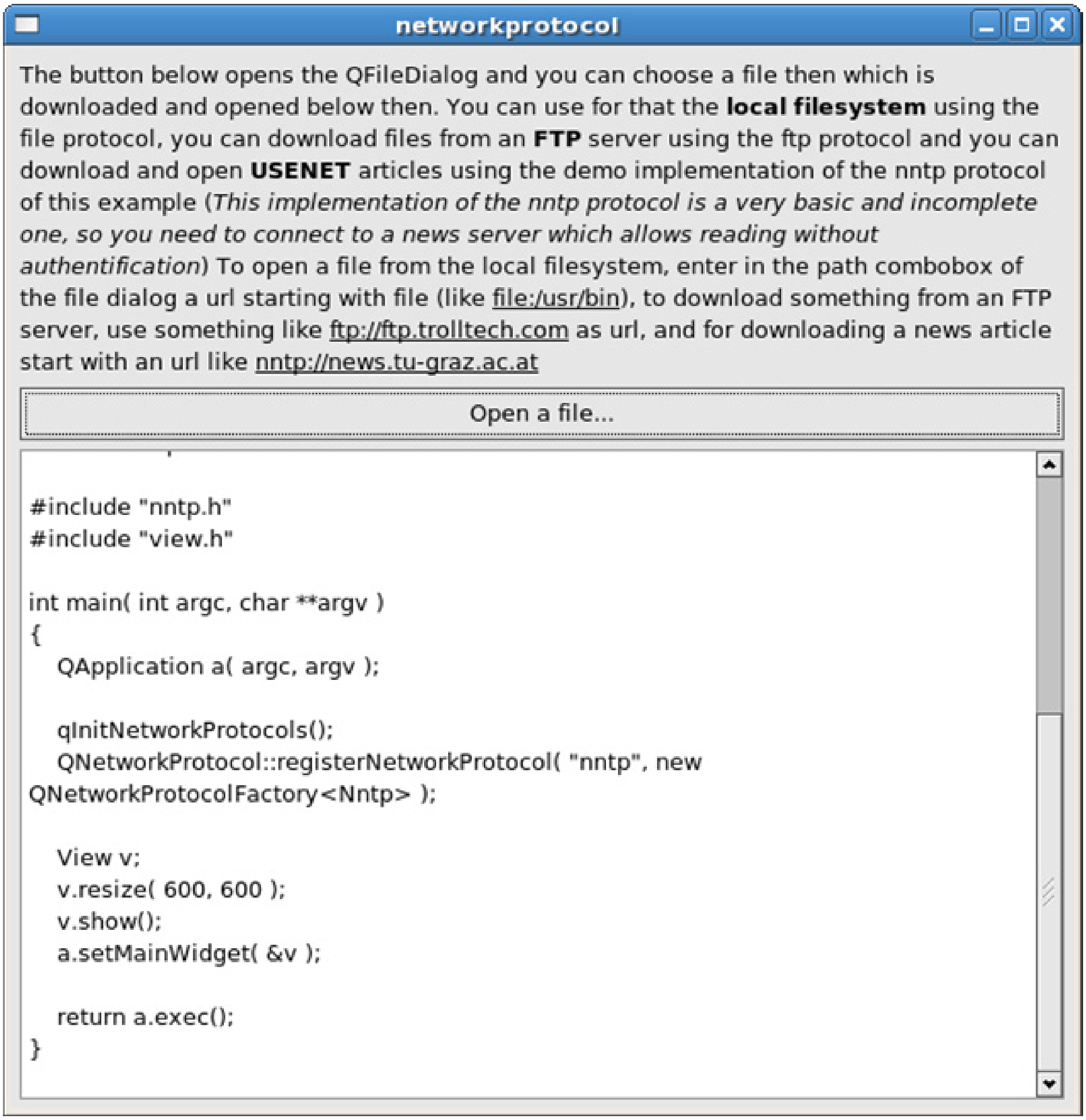
view.h

```
#ifndef VIEW_H
#define VIEW_H

#include <qvbox.h>
#include <qcstring.h>
#include <qurloperator.h>

class QMultiLineEdit;

class View : public QVBox
{
    Q_OBJECT
public:
    View();

private slots:
    void downloadFile();
    void newData( const QByteArray &ba );

private:
    QMultiLineEdit *fileView;
    QUrlOperator op;

    QString getOpenFileName();
};

#endif
```

main.cpp

```
#include <qapplication.h>
#include <qnetwork.h>
#include "nntp.h"
#include "view.h"

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    qInitNetworkProtocols();
    QNetworkProtocol::registerNetworkProtocol( "nntp", new QNetworkProtocolFactory<Nntp> );

    View v;
    v.resize( 600, 600 );
    v.show();
    a.setMainWidget( &v );

    return a.exec();
}
```

실행



The button below opens the QFileDialog and you can choose a file then which is downloaded and opened below then. You can use for that the **local filesystem** using the file protocol, you can download files from an **FTP** server using the ftp protocol and you can download and open **USENET** articles using the demo implementation of the nntp protocol of this example (*This implementation of the nntp protocol is a very basic and incomplete one, so you need to connect to a news server which allows reading without authentification*) To open a file from the local filesystem, enter in the path combobox of the file dialog a url starting with file (like file:/usr/bin), to download something from an FTP server, use something like ftp://ftp.trolltech.com as url, and for downloading a news article start with an url like nntp://news.tu-graz.ac.at

```
#include "nntp.h"
#include "view.h"

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    qInitNetworkProtocols();
    QNetworkProtocol::registerNetworkProtocol( "nntp", new
QNetworkProtocolFactory<Nntp> );

    View v;
    v.resize( 600, 600 );
    v.show();
    a.setMainWidget( &v );

    return a.exec();
}
```

8) 원격조종

remotecontrol.pro

```
TEMPLATE   = app
TARGET     = remotecontrol
CONFIG     += qt warn_on release
HEADERS        = startup.h \
           remotectrlimpl.h \
           ipcserver.h
SOURCES        = main.cpp \
           startup.cpp \
           remotectrlimpl.cpp \
           ipcserver.cpp
INTERFACES     = remotectrl.ui \
```

maindialog.ui

ipcserver.cpp

```
#include "ipcserver.h"
#include <qsocket.h>
#include <qvariant.h>
#include <qimage.h>
#include <qpalette.h>
#include <qapplication.h>

class IpcSocket : public QSocket
{
   Q_OBJECT

public:
   IpcSocket( QObject *parent) : QSocket( parent )
   {
    packetSize = 0;
    connect( this, SIGNAL(readyRead()), SLOT(read()) );
   }

signals:
   void receivedText( const QString& );
   void receivedPixmap( const QPixmap& );

private slots:
   void read()
   {
    Q_ULONG bytesAvail = bytesAvailable();
    for ( ;; ) {
        if ( packetSize == 0 ) {
         QDataStream ds( this );
         if ( bytesAvail < 4 )
             return;
         ds >> packetSize;
         bytesAvail -= 4;
        } else {
         if ( bytesAvail < packetSize )
             return;
         // read the packet in a byte array to be sure that you don't
         // read too much or too less
         QByteArray ba( packetSize );
         readBlock( ba.data(), packetSize );
         bytesAvail -= packetSize;
         packetSize = 0;

         QVariant variant;
         QDataStream ds( ba, IO_ReadOnly );
         ds >> variant;
         switch ( variant.type() ) {
             case QVariant::String:
              emit receivedText( variant.toString() );
              break;
             case QVariant::Image:
                emit receivedPixmap( QPixmap(variant.toImage()) );
                break;
              case QVariant::Palette:
                QApplication::setPalette( variant.toPalette(), TRUE );
                break;
              default:
                break;
          }
        }
      }
    }

private:
    Q_UINT32 packetSize;
};

IpcServer::IpcServer( Q_UINT16 port, QObject *parent ) :
    QServerSocket( 0x7f000001, port, 1, parent )
{
}

void IpcServer::newConnection( int socket )
{
    IpcSocket *s = new IpcSocket( this );
    s->setSocket( socket );
    connect( s, SIGNAL(receivedText(const QString&)),
          SIGNAL(receivedText(const QString&)) );
    connect( s, SIGNAL(receivedPixmap(const QPixmap&)),
          SIGNAL(receivedPixmap(const QPixmap&)) );
}
#include "ipcserver.moc"
```

ipcserver.h

```
#ifndef IPCSERVER_H
#define IPCSERVER_H
#include <qserversocket.h>

class IpcServer : public QServerSocket
{
    Q_OBJECT

public:
    IpcServer( Q_UINT16 port, QObject *parent );

    void newConnection( int socket );

signals:
    void receivedText( const QString& );
    void receivedPixmap( const QPixmap& );
};

#endif // IPCSERVER_H
```

remotectrlimpl.cpp

```
#include "remotectrlimpl.h"
#include <qpushbutton.h>
#include <qlineedit.h>
#include <qsocket.h>
#include <qfiledialog.h>
#include <qcolordialog.h>
#include <qimage.h>

RemoteCtrlImpl::RemoteCtrlImpl( QSocket *s )
{
  socket = s;
  connect( sImage, SIGNAL(clicked()), SLOT(sendImage()) );
  connect( sText, SIGNAL(clicked()), SLOT(sendText()) );
  connect( sPalette, SIGNAL(clicked()), SLOT(sendPalette()) );
}

void RemoteCtrlImpl::sendPacket( const QVariant &v )
{
  QByteArray ba;
  QDataStream varDs( ba, IO_WriteOnly );
  varDs << v;

  QDataStream ds( socket );
  ds << (Q_UINT32) ba.size();
  socket->writeBlock( ba.data(), ba.size() );
}

void RemoteCtrlImpl::sendImage()
{
  QString imageName = QFileDialog::getOpenFileName( QString::null,
       "Images (*.png *.xpm *.jpg)", this );
  QImage image( imageName );
  if ( !image.isNull() ) {
   sendPacket( image );
  }
}

void RemoteCtrlImpl::sendText()
{
  sendPacket( textToSend->text() );
}

void RemoteCtrlImpl::sendPalette()
{
  QColor col = QColorDialog::getColor( white, this );
  if ( col.isValid() ) {
   sendPacket( QPalette(col,col) );
  }
}
```

remotectrlimpl.h

```
#ifndef REMOTECTRLIMPL_H
#define REMOTECTRLIMPL_H
```

```
#include "remotectrl.h"
class QSocket;

class RemoteCtrlImpl : public RemoteCtrl
{
   Q_OBJECT

public:
   RemoteCtrlImpl( QSocket * );

private slots:
   void sendImage();
   void sendText();
   void sendPalette();

private:
   void sendPacket( const QVariant & );

   QSocket *socket;
};

#endif // REMOTECTRLIMPL_H
```

startup.cpp

```
#include "startup.h"
#include "remotectrlimpl.h"
#include "maindialog.h"
#include "ipcserver.h"

#include <qsocket.h>
#include <qlabel.h>

static const Q_UINT16 ipcPort = 54923;

StartUp::StartUp()
{
   remoteCtrl = 0;
   mainDialog = 0;

   socket = new QSocket( this );
   connect( socket, SIGNAL(connected()), SLOT(startRemoteCtrl()) );
   connect( socket, SIGNAL(error(int)), SLOT(startMainDialog()) );
   socket->connectToHost( "localhost", ipcPort );
}

StartUp::~StartUp()
{
   delete remoteCtrl;
   delete mainDialog;
}

void StartUp::startRemoteCtrl()
{
   remoteCtrl = new RemoteCtrlImpl( socket );
```

```
    remoteCtrl->show();
}

void StartUp::startMainDialog()
{
    mainDialog = new MainDialog();
    mainDialog->show();

    IpcServer *server = new IpcServer( ipcPort, this );

    connect( server, SIGNAL(receivedText(const QString&)),
          mainDialog->description, SLOT(setText(const QString&)) );
    connect( server, SIGNAL(receivedPixmap(const QPixmap&)),
          mainDialog->image, SLOT(setPixmap(const QPixmap&)) );
}
```

startup.h

```
#ifndef STARTUP_H
#define STARTUP_H
#include <qobject.h>

class QSocket;
class RemoteCtrlImpl;
class MainDialog;

class StartUp : public QObject
{
    Q_OBJECT

public:
    StartUp();
    ~StartUp();

private slots:
    void startRemoteCtrl();
    void startMainDialog();

private:
    QSocket *socket;
    RemoteCtrlImpl *remoteCtrl;
    MainDialog *mainDialog;
};

#endif // STARTUP_H
```

main.cpp

```
#include <qapplication.h>

#include "startup.h"

int main( int argc, char **argv )
{
    QApplication a( argc, argv );
    StartUp s;
```

```
   QObject::connect( &a, SIGNAL(lastWindowClosed()), &a, SLOT(quit()) );
   return a.exec();
}
```

maindialog.ui

```
<!DOCTYPE UI><UI version="3.0" stdsetdef="1">
<class>MainDialog</class>
<widget class="QDialog">
   <property name="name">
      <cstring>MainDialog</cstring>
…
      <signal>clicked()</signal>
      <receiver>MainDialog</receiver>
      <slot>close()</slot>
   </connection>
</connections>
<layoutdefaults spacing="6" margin="11"/>
</UI>
```

remotectrl.ui

```
<!DOCTYPE UI><UI version="3.0" stdsetdef="1">
<class>RemoteCtrl</class>
<widget class="QDialog">
   <property name="name">
…
      <receiver>RemoteCtrl</receiver>
      <slot>close()</slot>
   </connection>
</connections>
<layoutdefaults spacing="6" margin="11"/>
</UI>
```

실 행

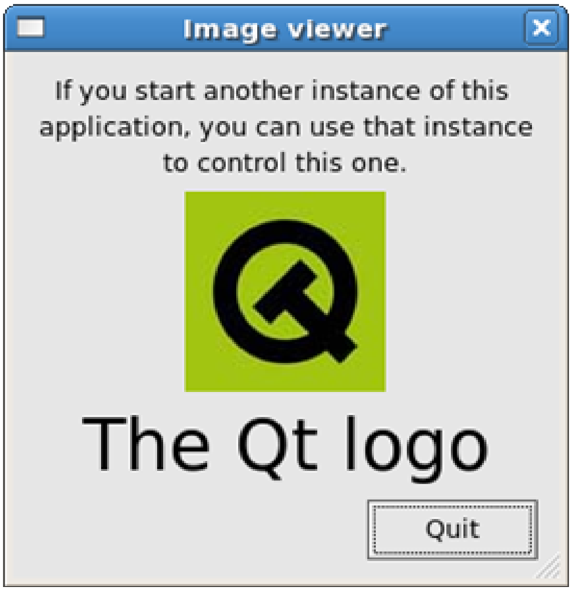


Image viewer

If you start another instance of this application, you can use that instance to control this one.

The Qt logo

39. OpenGL

다음의 실례프로그람들은 Qt OpenGL모듈의 사용법을 보여준다.

1) OpenGL 칸실례

이 실례는 Qt에서 OpenGL의 사용법을 보여준다. 반드시 처리할 문제는 QGLWidget로부터 계승된 클라스에 자기의 OpenGL코드를 넣는것이다. 이 클라스는 그다음 임의의 다른 Qt창문부품처럼 사용될수 있다. 즉 신호와 처리부 및 기하학적관리기능을 가진다.

box.pro

```
TEMPLATE  = app
TARGET      = box
CONFIG      += qt opengl warn_on release
CONFIG -= dlopen_opengl
HEADERS         = glbox.h \
            globjwin.h
SOURCES         = glbox.cpp \
            globjwin.cpp \
            main.cpp
```

glbox.cpp

```
/****************************************************************************
** This is a simple QGLWidget displaying an openGL wireframe box
** The OpenGL code is mostly borrowed from Brian Pauls "spin" example
** in the Mesa distribution
****************************************************************************/
#include "glbox.h"
#if defined(Q_CC_MSVC)
#pragma warning(disable:4305) // init: truncation from const double to float
#endif

/*!
  Create a GLBox widget
*/

GLBox::GLBox( QWidget* parent, const char* name )
    : QGLWidget( parent, name )
{
    xRot = yRot = zRot = 0.0;          // default object rotation
    scale = 1.25;              // default object scale
    object = 0;
}

/*!
  Release allocated resources
*/

GLBox::~GLBox()
{
    makeCurrent();
    glDeleteLists( object, 1 );
}

/*!
```

```
  Paint the box. The actual openGL commands for drawing the box are
  performed here.
*/

void GLBox::paintGL()
{
    glClear( GL_COLOR_BUFFER_BIT );

    glLoadIdentity();
    glTranslatef( 0.0, 0.0, -10.0 );
    glScalef( scale, scale, scale );

    glRotatef( xRot, 1.0, 0.0, 0.0 );
    glRotatef( yRot, 0.0, 1.0, 0.0 );
    glRotatef( zRot, 0.0, 0.0, 1.0 );

    glCallList( object );
}

/*!
  Set up the OpenGL rendering state, and define display list
*/

void GLBox::initializeGL()
{
    qglClearColor( black );          // Let OpenGL clear to black
    object = makeObject();           // Generate an OpenGL display list
    glShadeModel( GL_FLAT );
}

/*!
  Set up the OpenGL view port, matrix mode, etc.
*/

void GLBox::resizeGL( int w, int h )
{
    glViewport( 0, 0, (GLint)w, (GLint)h );
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    glFrustum( -1.0, 1.0, -1.0, 1.0, 5.0, 15.0 );
    glMatrixMode( GL_MODELVIEW );
}

/*!
  Generate an OpenGL display list for the object to be shown, i.e. the box
*/

GLuint GLBox::makeObject()
{
    GLuint list;

    list = glGenLists( 1 );

    glNewList( list, GL_COMPILE );
    qglColor( white );              // Shorthand for glColor3f or glIndex

    glLineWidth( 2.0 );

    glBegin( GL_LINE_LOOP );
    glVertex3f(  1.0,  0.5, -0.4 );
    glVertex3f(  1.0, -0.5, -0.4 );
    glVertex3f( -1.0, -0.5, -0.4 );
    glVertex3f( -1.0,  0.5, -0.4 );
    glEnd();

    glBegin( GL_LINE_LOOP );
    glVertex3f(  1.0,  0.5, 0.4 );
    glVertex3f(  1.0, -0.5, 0.4 );
    glVertex3f( -1.0, -0.5, 0.4 );
    glVertex3f( -1.0,  0.5, 0.4 );
    glEnd();

    glBegin( GL_LINES );
    glVertex3f(  1.0,  0.5, -0.4 );   glVertex3f(  1.0,  0.5, 0.4 );
    glVertex3f(  1.0, -0.5, -0.4 );   glVertex3f(  1.0, -0.5, 0.4 );
    glVertex3f( -1.0, -0.5, -0.4 );   glVertex3f( -1.0, -0.5, 0.4 );
    glVertex3f( -1.0,  0.5, -0.4 );   glVertex3f( -1.0,  0.5, 0.4 );
    glEnd();

    glEndList();

    return list;
}

/*!
  Set the rotation angle of the object to \e degrees around the X axis.
*/

void GLBox::setXRotation( int degrees )
{
    xRot = (GLfloat)(degrees % 360);
    updateGL();
}

/*!
  Set the rotation angle of the object to \e degrees around the Y axis.
*/

void GLBox::setYRotation( int degrees )
{
    yRot = (GLfloat)(degrees % 360);
    updateGL();
}

/*!
  Set the rotation angle of the object to \e degrees around the Z axis.
*/
```

```
void GLBox::setZRotation( int degrees )
{
   zRot = (GLfloat)(degrees % 360);
   updateGL();
}
```

glbox.h

```
/**********************************************************************
** This is a simple QGLWidget displaying an openGL wireframe box
**********************************************************************/
#ifndef GLBOX_H
#define GLBOX_H
#include <qgl.h>

class GLBox : public QGLWidget
{
   Q_OBJECT

public:

   GLBox( QWidget* parent, const char* name );
   ~GLBox();

public slots:

   void     setXRotation( int degrees );
   void     setYRotation( int degrees );
   void     setZRotation( int degrees );

protected:

   void     initializeGL();
   void     paintGL();
   void     resizeGL( int w, int h );

   virtual GLuint  makeObject();

private:

   GLuint object;
   GLfloat xRot, yRot, zRot, scale;

};

#endif // GLBOX_H
```

globjwin.cpp

```
#include <qpushbutton.h>
#include <qslider.h>
#include <qlayout.h>
#include <qframe.h>
#include <qmenubar.h>
#include <qpopupmenu.h>
```

```
#include <qapplication.h>
#include <qkeycode.h>
#include "globjwin.h"
#include "glbox.h"

GLObjectWindow::GLObjectWindow( QWidget* parent, const char* name )
   : QWidget( parent, name )
{

   // Create a menu
   QPopupMenu *file = new QPopupMenu( this );
   file->insertItem( "Exit",  qApp, SLOT(quit()), CTRL+Key_Q );

   // Create a menu bar
   QMenuBar *m = new QMenuBar( this );
   m->setSeparator( QMenuBar::InWindowsStyle );
   m->insertItem("&File", file );

   // Create a nice frame to put around the OpenGL widget
   QFrame* f = new QFrame( this, "frame" );
   f->setFrameStyle( QFrame::Sunken | QFrame::Panel );
   f->setLineWidth( 2 );

   // Create our OpenGL widget
   GLBox* c = new GLBox( f, "glbox");

   // Create the three sliders; one for each rotation axis
   QSlider* x = new QSlider ( 0, 360, 60, 0, QSlider::Vertical, this, "xsl" );
   x->setTickmarks( QSlider::Left );
   QObject::connect( x, SIGNAL(valueChanged(int)),c,SLOT(setXRotation(int)) );

   QSlider* y = new QSlider ( 0, 360, 60, 0, QSlider::Vertical, this, "ysl" );
   y->setTickmarks( QSlider::Left );
   QObject::connect( y, SIGNAL(valueChanged(int)),c,SLOT(setYRotation(int)) );

   QSlider* z = new QSlider ( 0, 360, 60, 0, QSlider::Vertical, this, "zsl" );
   z->setTickmarks( QSlider::Left );
   QObject::connect( z, SIGNAL(valueChanged(int)),c,SLOT(setZRotation(int)) );

   // Now that we have all the widgets, put them into a nice layout

   // Put the sliders on top of each other
   QVBoxLayout* vlayout = new QVBoxLayout( 20, "vlayout");
   vlayout->addWidget( x );
   vlayout->addWidget( y );
   vlayout->addWidget( z );

   // Put the GL widget inside the frame
   QHBoxLayout* flayout = new QHBoxLayout( f, 2, 2, "flayout");
   flayout->addWidget( c, 1 );

   // Top level layout, puts the sliders to the left of the frame/GL widget
   QHBoxLayout* hlayout = new QHBoxLayout( this, 20, 20, "hlayout");
   hlayout->setMenuBar( m );
```

```
    hlayout->addLayout( vlayout );
    hlayout->addWidget( f, 1 );
}
```

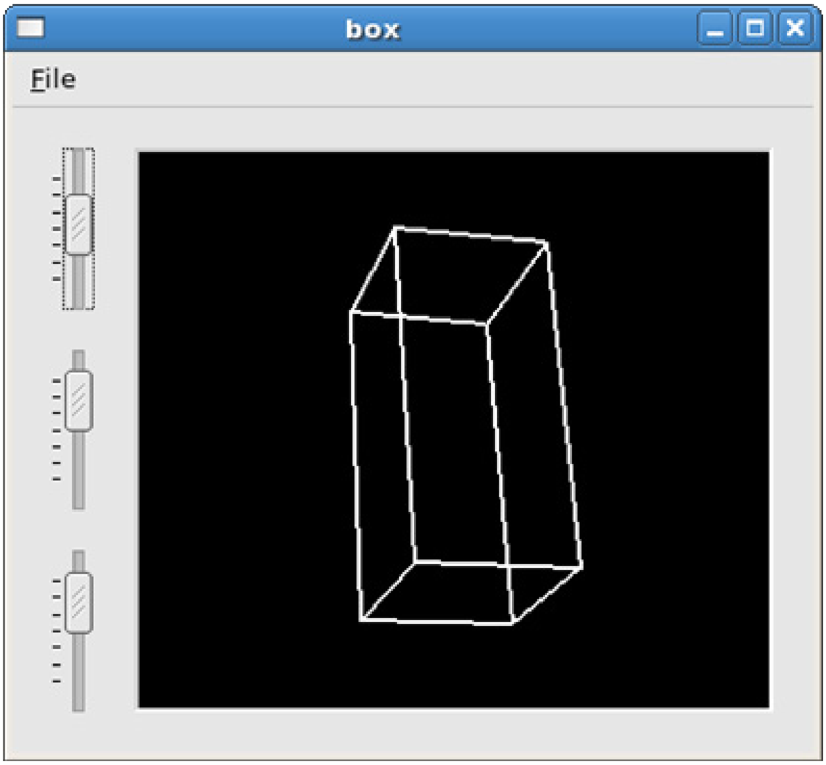
globjwin.h

```
/************************************************************************** The
GLObjectWindow contains a GLBox and three sliders connected to
** the GLBox's rotation slots.
***************************************************************************/

#ifndef GLOBJWIN_H
#define GLOBJWIN_H

#include <qwidget.h>

class GLObjectWindow : public QWidget
{
    Q_OBJECT
public:
    GLObjectWindow( QWidget* parent = 0, const char* name = 0 );

};

#endif // GLOBJWIN_H
```

main.cpp

```
// Qt OpenGL example: Box
// A small example showing how a GLWidget can be used just as any Qt widget
// File: main.cpp
// The main() function
#include "globjwin.h"
#include <qapplication.h>
#include <qgl.h>

/*
  The main program is here.
*/

int main( int argc, char **argv )
{
    QApplication::setColorSpec( QApplication::CustomColor );
    QApplication a(argc,argv);

    if ( !QGLFormat::hasOpenGL() ) {
      qWarning( "This system has no OpenGL support. Exiting." );
      return -1;
    }

    GLObjectWindow w;
    w.resize( 400, 350 );
    a.setMainWidget( &w );
    w.show();
    return a.exec();
}
```

실행



2) OpenGL 이바퀴실례

이 실례는 OpenGL현시목록을 사용하는 법을 보여준다.

gear.pro

```
TEMPLATE  = app
TARGET      = gear
CONFIG       += qt opengl warn_on release
CONFIG -= dlopen_opengl
!mac:unix:LIBS  += -lm
HEADERS        =
SOURCES        = gear.cpp
```

gear.cpp

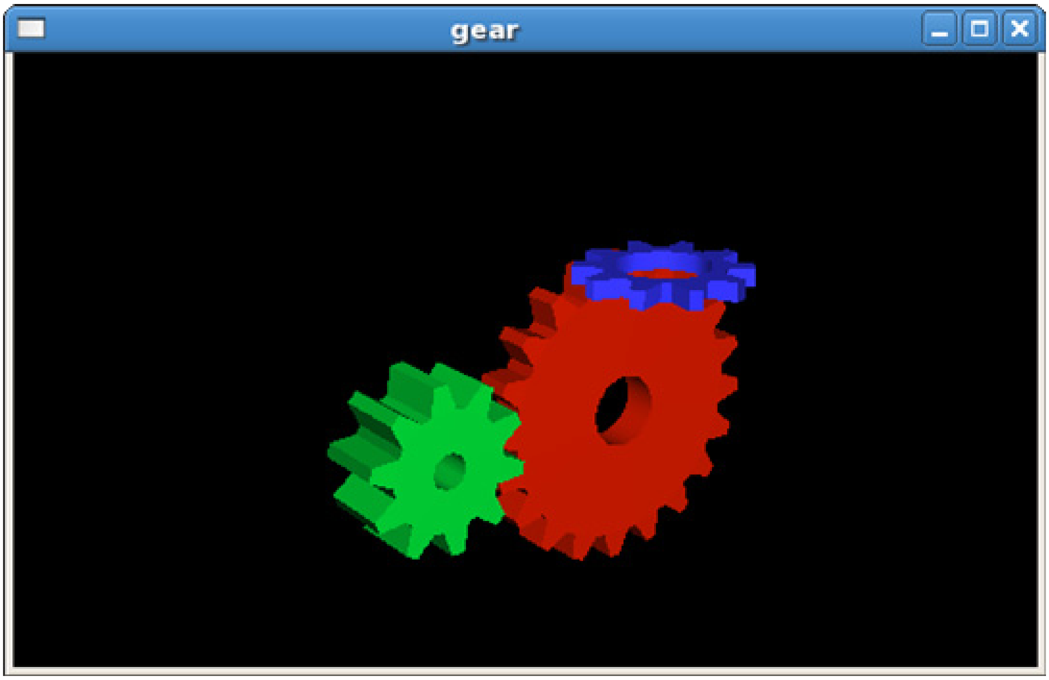
```
// Draws a gear.
// Portions of this code have been borrowed from Brian Paul's Mesa
// distribution.
#include <qgl.h>
#include <qapplication.h>
#include <math.h>

#if defined(Q_CC_MSVC)
#pragma warning(disable:4305) // init: truncation from const double to float
#endif

/*
 * Draw a gear wheel.  You'll probably want to call this function when
 * building a display list since we do a lot of trig here.
* Input:  inner_radius - radius of hole at center
 *     outer_radius - radius at center of teeth
 *     width - width of gear
 *     teeth - number of teeth
 *     tooth_depth - depth of tooth
 */
static void gear( GLfloat inner_radius, GLfloat outer_radius, GLfloat width,
          GLint teeth, GLfloat tooth_depth )
{
   GLint i;
   GLfloat r0, r1, r2;
   GLfloat angle, da;
   GLfloat u, v, len;

   r0 = inner_radius;
   r1 = outer_radius - tooth_depth/2.0;
   r2 = outer_radius + tooth_depth/2.0;

   const double pi = 3.14159264;
   da = 2.0*pi / teeth / 4.0;

   glShadeModel( GL_FLAT );

   glNormal3f( 0.0, 0.0, 1.0 );

   /* draw front face */
   glBegin( GL_QUAD_STRIP );
   for (i=0;i<=teeth;i++) {
    angle = i * 2.0*pi / teeth;
    glVertex3f( r0*cos(angle), r0*sin(angle), width*0.5 );
    glVertex3f( r1*cos(angle), r1*sin(angle), width*0.5 );
    glVertex3f( r0*cos(angle), r0*sin(angle), width*0.5 );
    glVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da), width*0.5 );
   }
   glEnd();

   /* draw front sides of teeth */
   glBegin( GL_QUADS );
   da = 2.0*pi / teeth / 4.0;
   for (i=0;i<teeth;i++) {
    angle = i * 2.0*pi / teeth;

    glVertex3f( r1*cos(angle),      r1*sin(angle),     width*0.5 );
    glVertex3f( r2*cos(angle+da),   r2*sin(angle+da),       width*0.5 );
    glVertex3f( r2*cos(angle+2*da), r2*sin(angle+2*da), width*0.5 );
    glVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da), width*0.5 );
   }
   glEnd();


   glNormal3f( 0.0, 0.0, -1.0 );

   /* draw back face */
```

```
glBegin( GL_QUAD_STRIP );
for (i=0;i<=teeth;i++) {
 angle = i * 2.0*pi / teeth;
 glVertex3f( r1*cos(angle), r1*sin(angle), -width*0.5 );
 glVertex3f( r0*cos(angle), r0*sin(angle), -width*0.5 );
 glVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da), -width*0.5 );
 glVertex3f( r0*cos(angle), r0*sin(angle), -width*0.5 );
}
glEnd();

/* draw back sides of teeth */
glBegin( GL_QUADS );
da = 2.0*pi / teeth / 4.0;
for (i=0;i<teeth;i++) {
 angle = i * 2.0*pi / teeth;

 glVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da), -width*0.5 );
 glVertex3f( r2*cos(angle+2*da), r2*sin(angle+2*da), -width*0.5 );
 glVertex3f( r2*cos(angle+da),   r2*sin(angle+da),       -width*0.5 );
 glVertex3f( r1*cos(angle),      r1*sin(angle),     -width*0.5 );
}
glEnd();

/* draw outward faces of teeth */
glBegin( GL_QUAD_STRIP );
for (i=0;i<teeth;i++) {
 angle = i * 2.0*pi / teeth;

 glVertex3f( r1*cos(angle),      r1*sin(angle),      width*0.5 );
 glVertex3f( r1*cos(angle),      r1*sin(angle),     -width*0.5 );
 u = r2*cos(angle+da) - r1*cos(angle);
 v = r2*sin(angle+da) - r1*sin(angle);
 len = sqrt( u*u + v*v );
 u /= len;
 v /= len;
 glNormal3f( v, -u, 0.0 );
 glVertex3f( r2*cos(angle+da),   r2*sin(angle+da),        width*0.5 );
 glVertex3f( r2*cos(angle+da),   r2*sin(angle+da),       -width*0.5 );
 glNormal3f( cos(angle), sin(angle), 0.0 );
 glVertex3f( r2*cos(angle+2*da), r2*sin(angle+2*da),  width*0.5 );
 glVertex3f( r2*cos(angle+2*da), r2*sin(angle+2*da), -width*0.5 );
 u = r1*cos(angle+3*da) - r2*cos(angle+2*da);
 v = r1*sin(angle+3*da) - r2*sin(angle+2*da);
 glNormal3f( v, -u, 0.0 );
 glVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da),  width*0.5 );
 glVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da), -width*0.5 );
 glNormal3f( cos(angle), sin(angle), 0.0 );
}

glVertex3f( r1*cos(0.0), r1*sin(0.0), width*0.5 );
glVertex3f( r1*cos(0.0), r1*sin(0.0), -width*0.5 );

glEnd();
```

```
   glShadeModel( GL_SMOOTH );

   /* draw inside radius cylinder */
   glBegin( GL_QUAD_STRIP );
   for (i=0;i<=teeth;i++) {
     angle = i * 2.0*pi / teeth;
     glNormal3f( -cos(angle), -sin(angle), 0.0 );
     glVertex3f( r0*cos(angle), r0*sin(angle), -width*0.5 );
     glVertex3f( r0*cos(angle), r0*sin(angle), width*0.5 );
   }
   glEnd();

}

static GLfloat view_rotx=20.0, view_roty=30.0, view_rotz=0.0;
static GLint gear1, gear2, gear3;
static GLfloat angle = 0.0;

static void draw()
{
   angle += 2.0;
   view_roty += 1.0;

   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

   glPushMatrix();
   glRotatef( view_rotx, 1.0, 0.0, 0.0 );
   glRotatef( view_roty, 0.0, 1.0, 0.0 );
   glRotatef( view_rotz, 0.0, 0.0, 1.0 );

   glPushMatrix();
   glTranslatef( -3.0, -2.0, 0.0 );
   glRotatef( angle, 0.0, 0.0, 1.0 );
   glCallList(gear1);
   glPopMatrix();

   glPushMatrix();
   glTranslatef( 3.1, -2.0, 0.0 );
   glRotatef( -2.0*angle-9.0, 0.0, 0.0, 1.0 );
   glCallList(gear2);
   glPopMatrix();

   glPushMatrix();
   glTranslatef( -3.1, 2.2, -1.8 );
   glRotatef( 90.0, 1.0, 0.0, 0.0 );
   glRotatef( 2.0*angle-2.0, 0.0, 0.0, 1.0 );
   glCallList(gear3);
   glPopMatrix();

   glPopMatrix();
}

static int timer_interval = 10;              // timer interval (millisec)
```

```
class GearWidget : public QGLWidget
{
public:
   GearWidget( QWidget *parent=0, const char *name=0 );
protected:
   void initializeGL();
   void resizeGL( int, int );
   void paintGL();
   void timerEvent( QTimerEvent * );
};

GearWidget::GearWidget( QWidget *parent, const char *name )
    : QGLWidget( parent, name )
{
   startTimer( timer_interval );
}

void GearWidget::initializeGL()
{
   static GLfloat pos[4] = {5.0, 5.0, 10.0, 1.0 };
   static GLfloat ared[4] = {0.8, 0.1, 0.0, 1.0 };
   static GLfloat agreen[4] = {0.0, 0.8, 0.2, 1.0 };
   static GLfloat ablue[4] = {0.2, 0.2, 1.0, 1.0 };

   glLightfv( GL_LIGHT0, GL_POSITION, pos );
   glEnable( GL_CULL_FACE );
   glEnable( GL_LIGHTING );
   glEnable( GL_LIGHT0 );
   glEnable( GL_DEPTH_TEST );

   /* make the gears */
   gear1 = glGenLists(1);
   glNewList(gear1, GL_COMPILE);
   glMaterialfv( GL_FRONT, GL_AMBIENT_AND_DIFFUSE, ared );
   gear( 1.0, 4.0, 1.0, 20, 0.7 );
   glEndList();

   gear2 = glGenLists(1);
   glNewList(gear2, GL_COMPILE);
   glMaterialfv( GL_FRONT, GL_AMBIENT_AND_DIFFUSE, agreen );
   gear( 0.5, 2.0, 2.0, 10, 0.7 );
   glEndList();

   gear3 = glGenLists(1);
   glNewList(gear3, GL_COMPILE);
   glMaterialfv( GL_FRONT, GL_AMBIENT_AND_DIFFUSE, ablue );
   gear( 1.3, 2.0, 0.5, 10, 0.7 );
   glEndList();

   glEnable( GL_NORMALIZE );
}

void GearWidget::resizeGL( int width, int height )
{
   GLfloat w = (float) width / (float) height;
   GLfloat h = 1.0;

   glViewport( 0, 0, width, height );
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glFrustum( -w, w, -h, h, 5.0, 60.0 );
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   glTranslatef( 0.0, 0.0, -40.0 );
}

void GearWidget::paintGL()
{
   draw();
}

void GearWidget::timerEvent(QTimerEvent*)
{
   updateGL();
}

int main( int argc, char **argv )
{
   QApplication::setColorSpec( QApplication::CustomColor );
   QApplication a( argc, argv );

   if ( !QGLFormat::hasOpenGL() ) {
    qWarning( "This system has no OpenGL support. Exiting." );
    return -1;
   }

   if ( argc >= 2 ) {
    bool ok = TRUE;
    timer_interval = QString::fromLatin1( argv[1] ).toInt( &ok );
    if ( !ok )
       timer_interval = 10;
   }

   GearWidget w;
   a.setMainWidget( &w );
   w.show();
   return a.exec();
}
```

실행



3) OpenGL 픽스매프실례

이 실례프로그람은 OpenGL칸실례를 확장한것이다. 이것은 OpenGL을 QPixmap로서 그리는 방법을 보여준다.

glpixmap.pro

```
TEMPLATE  = app
TARGET      = glpixmap
CONFIG      += qt opengl warn_on release
CONFIG -= dlopen_opengl
!mac:unix:LIBS  += -lm
HEADERS          = glbox.h \
          globjwin.h
SOURCES          = glbox.cpp \
          globjwin.cpp \
          main.cpp
```

glbox.cpp

```
/****************************************************************************
** This is a simple QGLWidget displaying a box
** The OpenGL code is mostly borrowed from Brian Pauls "spin" example
** in the Mesa distribution
****************************************************************************/
#include <math.h>
#include "glbox.h"

#if defined(Q_CC_MSVC)
#pragma warning(disable:4305) // init: truncation from const double to float
#endif
```

```
/*!
 Create a GLBox widget
*/

GLBox::GLBox( QWidget* parent, const char* name, const QGLWidget* shareWidget )
   : QGLWidget( parent, name, shareWidget )
{
   xRot = yRot = zRot = 0.0;          // default object rotation
   scale = 1.5;          // default object scale
}

/*!
 Create a GLBox widget
*/

GLBox::GLBox( const QGLFormat& format, QWidget* parent, const char* name,
         const QGLWidget* shareWidget )
   : QGLWidget( format, parent, name, shareWidget )
{
   xRot = yRot = zRot = 0.0;          // default object rotation
   scale = 1.5;          // default object scale
}

/*!
 Release allocated resources
*/

GLBox::~GLBox()
{
}

/*!
 Set up the OpenGL rendering state, and define display list
*/

void GLBox::initializeGL()
{
   qglClearColor( green );         // Let OpenGL clear to green
   object = makeObject();          // Make display list
   glEnable( GL_DEPTH_TEST );
}

/*!
 Set up the OpenGL view port, matrix mode, etc.
*/

void GLBox::resizeGL( int w, int h )
{
   glViewport( 0, 0, (GLint)w, (GLint)h );
   glMatrixMode( GL_PROJECTION );
   glLoadIdentity();
   glFrustum(-1.0, 1.0, -1.0, 1.0, 1.0, 200.0);
}
```

```
/*!
  Paint the box. The actual openGL commands for drawing the box are
  performed here.
*/

void GLBox::paintGL()
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
    glTranslatef( 0.0, 0.0, -3.0 );
    glScalef( scale, scale, scale );

    glRotatef( xRot, 1.0, 0.0, 0.0 );
    glRotatef( yRot, 0.0, 1.0, 0.0 );
    glRotatef( zRot, 0.0, 0.0, 1.0 );

    glCallList( object );
}

/*!
  Generate an OpenGL display list for the object to be shown, i.e. the box
*/

GLuint GLBox::makeObject()
{
    GLuint list;

    list = glGenLists( 1 );

    glNewList( list, GL_COMPILE );

    GLint i, j, rings, sides;
    float theta1, phi1, theta2, phi2;
    float v0[03], v1[3], v2[3], v3[3];
    float t0[03], t1[3], t2[3], t3[3];
    float n0[3], n1[3], n2[3], n3[3];
    float innerRadius=0.4;
    float outerRadius=0.8;
    float scalFac;
    double pi = 3.14159265358979323846;

    rings = 8;
    sides = 10;
    scalFac=1/(outerRadius*2);

    for (i = 0; i < rings; i++) {
        theta1 = (float)i * 2.0 * pi / rings;
        theta2 = (float)(i + 1) * 2.0 * pi / rings;
        for (j = 0; j < sides; j++) {
            phi1 = (float)j * 2.0 * pi / sides;
            phi2 = (float)(j + 1) * 2.0 * pi / sides;
            v0[0] = cos(theta1) * (outerRadius + innerRadius * cos(phi1));
            v0[1] = -sin(theta1) * (outerRadius + innerRadius * cos(phi1));
            v0[2] = innerRadius * sin(phi1);

            v1[0] = cos(theta2) * (outerRadius + innerRadius * cos(phi1));
            v1[1] = -sin(theta2) * (outerRadius + innerRadius * cos(phi1));
            v1[2] = innerRadius * sin(phi1);
            v2[0] = cos(theta2) * (outerRadius + innerRadius * cos(phi2));
            v2[1] = -sin(theta2) * (outerRadius + innerRadius * cos(phi2));
            v2[2] = innerRadius * sin(phi2);

            v3[0] = cos(theta1) * (outerRadius + innerRadius * cos(phi2));
            v3[1] = -sin(theta1) * (outerRadius + innerRadius * cos(phi2));
            v3[2] = innerRadius * sin(phi2);

            n0[0] = cos(theta1) * (cos(phi1));
            n0[1] = -sin(theta1) * (cos(phi1));
            n0[2] = sin(phi1);

            n1[0] = cos(theta2) * (cos(phi1));
            n1[1] = -sin(theta2) * (cos(phi1));
            n1[2] = sin(phi1);

            n2[0] = cos(theta2) * (cos(phi2));
            n2[1] = -sin(theta2) * (cos(phi2));
            n2[2] = sin(phi2);

            n3[0] = cos(theta1) * (cos(phi2));
            n3[1] = -sin(theta1) * (cos(phi2));
            n3[2] = sin(phi2);

            t0[0] = v0[0]*scalFac + 0.5;
            t0[1] = v0[1]*scalFac + 0.5;
            t0[2] = v0[2]*scalFac + 0.5;

            t1[0] = v1[0]*scalFac + 0.5;
            t1[1] = v1[1]*scalFac + 0.5;
            t1[2] = v1[2]*scalFac + 0.5;

            t2[0] = v2[0]*scalFac + 0.5;
            t2[1] = v2[1]*scalFac + 0.5;
            t2[2] = v2[2]*scalFac + 0.5;

            t3[0] = v3[0]*scalFac + 0.5;
            t3[1] = v3[1]*scalFac + 0.5;
            t3[2] = v3[2]*scalFac + 0.5;

          // Create blue-black checkered coloring
          if ( (i+j) % 2 )
           qglColor( black );
          else
           qglColor( QColor( "steelblue" ) );

            glBegin(GL_POLYGON);
```

```
                glNormal3fv(n3); glTexCoord3fv(t3); glVertex3fv(v3);
                glNormal3fv(n2); glTexCoord3fv(t2); glVertex3fv(v2);
                glNormal3fv(n1); glTexCoord3fv(t1); glVertex3fv(v1);
                glNormal3fv(n0); glTexCoord3fv(t0); glVertex3fv(v0);
            glEnd();
        }
    }
    glEndList();

    return list;
}

/*!
  Set the rotation angle of the object to \e degrees around the X axis.
*/

void GLBox::setXRotation( int degrees )
{
    xRot = (GLfloat)(degrees % 360);
    updateGL();
}

/*!
  Set the rotation angle of the object to \e degrees around the Y axis.
*/

void GLBox::setYRotation( int degrees )
{
    yRot = (GLfloat)(degrees % 360);
    updateGL();
}

/*!
  Set the rotation angle of the object to \e degrees around the Z axis.
*/

void GLBox::setZRotation( int degrees )
{
    zRot = (GLfloat)(degrees % 360);
    updateGL();
}

/*!
  Sets the rotation angles of this object to that of \a fromBox
*/

void GLBox::copyRotation( const GLBox& fromBox )
{
    xRot = fromBox.xRot;
    yRot = fromBox.yRot;
    zRot = fromBox.zRot;
}
```

glbox.h

```
/*************************************************************************
** This is a simple QGLWidget displaying a box
*************************************************************************/

#ifndef GLBOX_H
#define GLBOX_H

#include <qgl.h>

class GLBox : public QGLWidget
{
   Q_OBJECT

public:

   GLBox( QWidget* parent, const char* name, const QGLWidget* shareWidget=0 );
   GLBox( const QGLFormat& format, QWidget* parent, const char* name,
      const QGLWidget* shareWidget=0 );
   ~GLBox();

   void     copyRotation( const GLBox& fromBox );

public slots:

   void     setXRotation( int degrees );
   void     setYRotation( int degrees );
   void     setZRotation( int degrees );

protected:

   void     initializeGL();
   void     paintGL();
   void     resizeGL( int w, int h );

   virtual GLuint makeObject();

private:

   GLuint object;

   GLfloatxRot, yRot, zRot, scale;

};

#endif // GLBOX_H
```

globjwin.cpp

```
/*************************************************************************
Implementation of GLObjectWindow widget class
*************************************************************************/
#include <qpushbutton.h>
#include <qslider.h>
#include <qlayout.h>
```

```
#include <qframe.h>
#include <qlabel.h>
#include <qmenubar.h>
#include <qpopupmenu.h>
#include <qapplication.h>
#include <qkeycode.h>
#include <qpixmap.h>
#include <qimage.h>
#include <qpainter.h>
#include "globjwin.h"
#include "glbox.h"

GLObjectWindow::GLObjectWindow( QWidget* parent, const char* name )
   : QWidget( parent, name )
{
   // Create a menu
   file = new QPopupMenu( this );
   file->setCheckable( TRUE );
   file->insertItem( "Grab Frame Buffer", this, SLOT(grabFrameBuffer()) );
   file->insertItem( "Render Pixmap", this, SLOT(makePixmap()) );
   file->insertItem( "Render Pixmap Hidden", this, SLOT(makePixmapHidden()) );
   file->insertSeparator();
   fixMenuItemId = file->insertItem( "Use Fixed Pixmap Size", this,
                        SLOT(useFixedPixmapSize()) );
   file->insertSeparator();
   insertMenuItemId = file->insertItem( "Insert Pixmap Here", this,
                        SLOT(makePixmapForMenu()) );
   file->insertSeparator();
   file->insertItem( "Exit",  qApp, SLOT(quit()), CTRL+Key_Q );

   // Create a menu bar
   QMenuBar *m = new QMenuBar( this );
   m->setSeparator( QMenuBar::InWindowsStyle );
   m->insertItem("&File", file );

   // Create nice frames to put around the OpenGL widgets
   QFrame* f1 = new QFrame( this, "frame1" );
   f1->setFrameStyle( QFrame::Sunken | QFrame::Panel );
   f1->setLineWidth( 2 );

   // Create an OpenGL widget
   c1 = new GLBox( f1, "glbox1");

   // Create a label that can display the pixmap
   lb = new QLabel( this, "pixlabel" );
   lb->setFrameStyle( QFrame::Sunken | QFrame::Panel );
   lb->setLineWidth( 2 );
   lb->setAlignment( AlignCenter );
   lb->setMargin( 0 );
   lb->setIndent( 0 );

   // Create the three sliders; one for each rotation axis
   QSlider* x = new QSlider ( 0, 360, 60, 0, QSlider::Vertical, this, "xsl" );
   x->setTickmarks( QSlider::Left );
   connect( x, SIGNAL(valueChanged(int)), c1, SLOT(setXRotation(int)) );

   QSlider* y = new QSlider ( 0, 360, 60, 0, QSlider::Vertical, this, "ysl" );
   y->setTickmarks( QSlider::Left );
   connect( y, SIGNAL(valueChanged(int)), c1, SLOT(setYRotation(int)) );

   QSlider* z = new QSlider ( 0, 360, 60, 0, QSlider::Vertical, this, "zsl" );
   z->setTickmarks( QSlider::Left );
   connect( z, SIGNAL(valueChanged(int)), c1, SLOT(setZRotation(int)) );

   // Now that we have all the widgets, put them into a nice layout
   // Put the sliders on top of each other
   QVBoxLayout* vlayout = new QVBoxLayout( 20, "vlayout");
   vlayout->addWidget( x );
   vlayout->addWidget( y );
   vlayout->addWidget( z );

   // Put the GL widget inside the frame
   QHBoxLayout* flayout1 = new QHBoxLayout( f1, 2, 2, "flayout1");
   flayout1->addWidget( c1, 1 );

   // Top level layout, puts the sliders to the left of the frame/GL widget
   QHBoxLayout* hlayout = new QHBoxLayout( this, 20, 20, "hlayout");
   hlayout->setMenuBar( m );
   hlayout->addLayout( vlayout );
   hlayout->addWidget( f1, 1 );
   hlayout->addWidget( lb, 1 );
}

void GLObjectWindow::grabFrameBuffer()
{
   QImage img = c1->grabFrameBuffer();

   // Convert image to pixmap so we can show it
   QPixmap pm;
   pm.convertFromImage( img, AvoidDither );
   drawOnPixmap( &pm );
   lb->setPixmap( pm );
}

void GLObjectWindow::makePixmap()
{
   // Make a pixmap to to be rendered by the gl widget
   QPixmap pm;

   // Render the pixmap, with either c1's size or the fixed size pmSz
   if ( pmSz.isValid() )
    pm = c1->renderPixmap( pmSz.width(), pmSz.height() );
   else
    pm = c1->renderPixmap();

   if ( !pm.isNull() ) {
    // Present the pixmap to the user
    drawOnPixmap( &pm );
    lb->setPixmap( pm );
  }
  else {
    lb->setText( "Failed to render Pixmap." );
  }
}

void GLObjectWindow::makePixmapHidden()
{
  // Make a QGLWidget to draw the pixmap. This widget will not be shown.
  GLBox* w = new GLBox( this, "temporary glwidget", c1 );

  bool success = FALSE;
  QPixmap pm;

  if ( w->isValid() ) {
    // Set the current rotation
    w->copyRotation( *c1 );

    // Determine wanted pixmap size
    QSize sz = pmSz.isValid() ? pmSz : c1->size();

    // Make our hidden glwidget render the pixmap
    pm = w->renderPixmap( sz.width(), sz.height() );

    if ( !pm.isNull() )
        success = TRUE;
  }

  if ( success ) {
    // Present the pixmap to the user
    drawOnPixmap( &pm );
    lb->setPixmap( pm );
  }
  else {
    lb->setText( "Failed to render Pixmap." );
  }
  delete w;
}

void GLObjectWindow::drawOnPixmap( QPixmap* pm )
{
  // Draw some text on the pixmap to differentiate it from the GL window

  if ( pm->isNull() ) {
    qWarning("Cannot draw on null pixmap");
    return;
  }
  else {
    QPainter p( pm );
     p.setFont( QFont( "Helvetica", 18 ) );
    p.setPen( white );
    p.drawText( pm->rect(), AlignCenter, "This is a Pixmap" );
  }
```

```
}

void GLObjectWindow::useFixedPixmapSize()
{
   if ( pmSz.isValid() ) {
    pmSz = QSize();
    file->setItemChecked( fixMenuItemId, FALSE );
   }
   else {
    pmSz = QSize( 200, 200 );
    file->setItemChecked( fixMenuItemId, TRUE );
   }
}

void GLObjectWindow::makePixmapForMenu()
{
   QPixmap pm = c1->renderPixmap( 32, 32 );
   if ( !pm.isNull() )
    file->changeItem( pm, "Insert Pixmap Here", insertMenuItemId );
}
```

globjwin.h

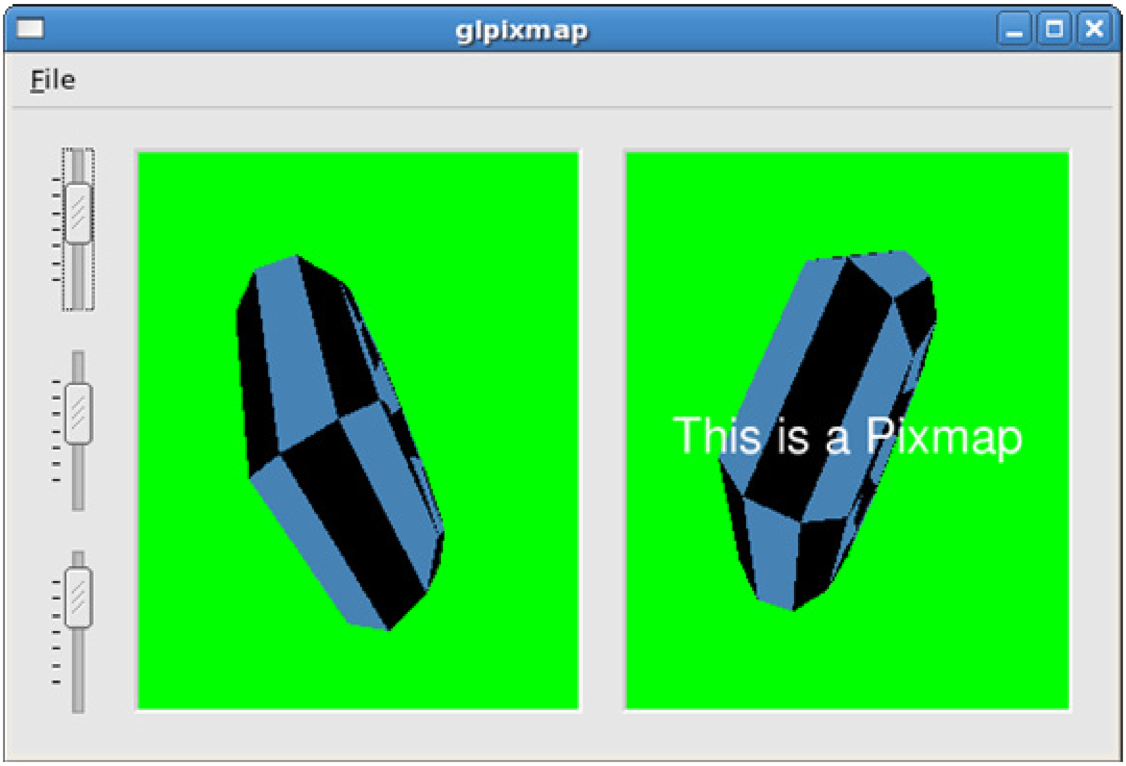
```
/****************************************************************************
** The GLObjectWindow contains a GLBox and three sliders connected to
** the GLBox's rotation slots.
****************************************************************************/
#ifndef GLOBJWIN_H
#define GLOBJWIN_H

#include <qwidget.h>

class GLBox;
class QLabel;
class QPopupMenu;

class GLObjectWindow : public QWidget
{
   Q_OBJECT
public:
   GLObjectWindow( QWidget* parent = 0, const char* name = 0 );

protected slots:

   void     grabFrameBuffer();
   void     makePixmap();
   void     makePixmapHidden();
   void     makePixmapForMenu();
   void     useFixedPixmapSize();

private:
   void     drawOnPixmap( QPixmap* pm );
   GLBox* c1;
   QLabel* lb;
   int fixMenuItemId;
```

```
    int insertMenuItemId;
    QSize pmSz;
    QPopupMenu* file;
};

#endif // GLOBJWIN_H
```

main.cpp

```
// Qt OpenGL example: Shared Box
// A small example showing how to use OpenGL display list sharing
// File: main.cpp
// The main() function
#include "globjwin.h"
#include <qapplication.h>
#include <qgl.h>

/*
  The main program is here.
*/

int main( int argc, char **argv )
{
    QApplication::setColorSpec( QApplication::CustomColor );
    QApplication a(argc,argv);

    if ( !QGLFormat::hasOpenGL() ) {
      qWarning( "This system has no OpenGL support. Exiting." );
      return -1;
    }

    GLObjectWindow w;
    w.resize( 550, 350 );
    a.setMainWidget( &w );
    w.show();
    return a.exec();
}
```

실행



4) OpenGL 공유칸실례

이 실례프로그람은 OpenGL픽스매프실례를 확장한것이다. 이것은 QGLWidget들을 공유하는 OpenGL현시목록을 사용하는 방법을 보여준다.

sharedbox.pro

```
TEMPLATE  = app
TARGET      = sharedbox
CONFIG      += qt opengl warn_on release
CONFIG -= dlopen_opengl
HEADERS        = glbox.h \
           globjwin.h
SOURCES        = glbox.cpp \
           globjwin.cpp \
           main.cpp
```

glbox.cpp

```
/*************************************************************************
** This is a simple QGLWidget displaying a box
** The OpenGL code is mostly borrowed from Brian Pauls "spin" example
** in the Mesa distribution
*************************************************************************/
#include "glbox.h"

// Initialize static class variables:
// Shared display list id:
```

```
GLuint GLBox::sharedDisplayList = 0;

// Counter keeping track of number of GLBox instances sharing
// the display list, so that the last instance can delete it:
int GLBox::sharedListUsers = 0;

/*!
  Create a GLBox widget
*/

GLBox::GLBox( QWidget* parent, const char* name, const QGLWidget* shareWidget )
    : QGLWidget( parent, name, shareWidget )
{
    xRot = yRot = zRot = 0.0;          // default object rotation
    scale = 1.0;           // default object scale
    object = 0;
    localDisplayList = 0;
}

/*!
  Set up the OpenGL rendering state. Robustly access shared display list.
*/

void GLBox::initializeGL()
{
    // Let OpenGL clear to black
    qglClearColor( black );

    glEnable(GL_DEPTH_TEST);

    if ( sharedListUsers == 0 ) {  // No shared list has been made yet
      sharedDisplayList = makeObject();// Make one
      object = sharedDisplayList;       // Use it
      sharedListUsers++;                  // Keep reference count
      qDebug( "GLBox %s created shared display list.", name() );
    }
    else {                   // There is a shared diplay list
      if ( isSharing() ) {           // Can we access it?
          object = sharedDisplayList;        // Yes, use it
          sharedListUsers++;               // Keep reference count
          qDebug( "GLBox %s uses shared display list.", name() );
      }
      else {
          localDisplayList = makeObject();    // No, roll our own
          object = localDisplayList;           // and use that
          qDebug( "GLBox %s uses private display list.", name() );
      }
    }
}

/*!
  Release allocated resources
*/
```

```
GLBox::~GLBox()
{
   makeCurrent();                    // We're going to do gl calls
   if ( localDisplayList != 0 ) {       // Did we make our own?
    glDeleteLists( localDisplayList, 1 );     // Yes, delete it
    qDebug( "GLBox %s deleted private display list.", name() );
   }
   else {
    sharedListUsers--;    // No, we used the shared one; keep refcount
    if ( sharedListUsers == 0 ) {              // Any sharers left?
        glDeleteLists( sharedDisplayList, 1 );    // No, delete it
        sharedDisplayList = 0;
        qDebug( "GLBox %s deleted shared display list.", name() );
    }
   }
}


/*!
  Paint the box. The actual openGL commands for drawing the box are
  performed here.
*/

void GLBox::paintGL()
{
   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

   glMatrixMode( GL_MODELVIEW );
   glLoadIdentity();
   glTranslatef( 0.0, 0.0, -3.0 );
   glScalef( scale, scale, scale );

   glRotatef( xRot, 1.0, 0.0, 0.0 );
   glRotatef( yRot, 0.0, 1.0, 0.0 );
   glRotatef( zRot, 0.0, 0.0, 1.0 );

   glCallList( object );
}

/*!
  Set up the OpenGL view port, matrix mode, etc.
*/

void GLBox::resizeGL( int w, int h )
{
   glViewport( 0, 0, (GLint)w, (GLint)h );
   glMatrixMode( GL_PROJECTION );
   glLoadIdentity();
   glFrustum(-1.0, 1.0, -1.0, 1.0, 1.0, 10.0);
}

/*!
  Generate an OpenGL display list for the object to be shown, i.e. the box
*/
```

```
GLuint GLBox::makeObject()
{
   GLuint list;

   list = glGenLists( 1 );

   glNewList( list, GL_COMPILE );

   glBegin(GL_QUADS);
   /* Front face */
   qglColor( green );
   glVertex3f(-1.0, 1.0, 1.0);
   glVertex3f(1.0, 1.0, 1.0);
   glVertex3f(1.0, -1.0, 1.0);
   glVertex3f(-1.0, -1.0, 1.0);
   /* Back face */
   qglColor( yellow );
   glVertex3f(-1.0, 1.0, -1.0);
   glVertex3f(1.0, 1.0, -1.0);
   glVertex3f(1.0, -1.0, -1.0);
   glVertex3f(-1.0, -1.0, -1.0);
   /* Top side face */
   qglColor( blue );
   glVertex3f(-1.0, 1.0, 1.0);
   glVertex3f(1.0, 1.0, 1.0);
   glVertex3f(1.0, 1.0, -1.0);
   glVertex3f(-1.0, 1.0, -1.0);
   /* Bottom side face */
   qglColor( red );
   glVertex3f(-1.0, -1.0, 1.0);
   glVertex3f(1.0, -1.0, 1.0);
   glVertex3f(1.0, -1.0, -1.0);
   glVertex3f(-1.0, -1.0, -1.0);
   glEnd();
   glEndList();

   return list;
}

/*!
  Set the rotation angle of the object to \e degrees around the X axis.
*/

void GLBox::setXRotation( int degrees )
{
   xRot = (GLfloat)(degrees % 360);
   updateGL();
}


/*!
  Set the rotation angle of the object to \e degrees around the Y axis.
*/
```

```
void GLBox::setYRotation( int degrees )
{
   yRot = (GLfloat)(degrees % 360);
   updateGL();
}

/*!
  Set the rotation angle of the object to \e degrees around the Z axis.
*/

void GLBox::setZRotation( int degrees )
{
   zRot = (GLfloat)(degrees % 360);
   updateGL();
}
```

glbox.h

```
/****************************************************************************
** This is a simple QGLWidget displaying a box
****************************************************************************/
#ifndef GLBOX_H
#define GLBOX_H
#include <qgl.h>
class GLBox : public QGLWidget
{
   Q_OBJECT

public:

   GLBox( QWidget* parent, const char* name, const QGLWidget* shareWidget=0 );
   ~GLBox();

public slots:

   void     setXRotation( int degrees );
   void     setYRotation( int degrees );
   void     setZRotation( int degrees );

protected:

   void     initializeGL();
   void     paintGL();
   void     resizeGL( int w, int h );

   virtual GLuint  makeObject();

private:

   GLuint      object;
   GLuint      localDisplayList;

   static GLuint   sharedDisplayList;
   static int      sharedListUsers;
```

```
    GLfloat xRot, yRot, zRot, scale;

};

#endif // GLBOX_H
```

globjwin.cpp

```
#include <qpushbutton.h>
#include <qslider.h>
#include <qlayout.h>
#include <qframe.h>
#include <qmenubar.h>
#include <qpopupmenu.h>
#include <qapplication.h>
#include <qkeycode.h>
#include "globjwin.h"
#include "glbox.h"


GLObjectWindow::GLObjectWindow( QWidget* parent, const char* name )
    : QWidget( parent, name )
{
    // Create a menu
    QPopupMenu *file = new QPopupMenu( this );
    file->insertItem( "Delete Left QGLWidget", this,
                SLOT(deleteFirstWidget()) );
    file->insertItem( "Exit",  qApp, SLOT(quit()), CTRL+Key_Q );

    // Create a menu bar
    QMenuBar *m = new QMenuBar( this );
    m->setSeparator( QMenuBar::InWindowsStyle );
    m->insertItem("&File", file );

    // Create nice frames to put around the OpenGL widgets
    QFrame* f1 = new QFrame( this, "frame1" );
    f1->setFrameStyle( QFrame::Sunken | QFrame::Panel );
    f1->setLineWidth( 2 );
    QFrame* f2 = new QFrame( this, "frame2" );
    f2->setFrameStyle( QFrame::Sunken | QFrame::Panel );
    f2->setLineWidth( 2 );

    // Create an OpenGL widget
    c1 = new GLBox( f1, "glbox1" );

    // Create another OpenGL widget that shares display lists with the first
    c2 = new GLBox( f2, "glbox2", c1 );

    // Create the three sliders; one for each rotation axis
    // Make them spin the boxes, but not in synch
    QSlider* x = new QSlider ( 0, 360, 60, 0, QSlider::Vertical, this, "xsl" );
    x->setTickmarks( QSlider::Left );
    connect( x, SIGNAL(valueChanged(int)), c1, SLOT(setXRotation(int)) );
    connect( x, SIGNAL(valueChanged(int)), c2, SLOT(setZRotation(int)) );
```

```
   QSlider* y = new QSlider ( 0, 360, 60, 0, QSlider::Vertical, this, "ysl" );
   y->setTickmarks( QSlider::Left );
   connect( y, SIGNAL(valueChanged(int)), c1, SLOT(setYRotation(int)) );
   connect( y, SIGNAL(valueChanged(int)), c2, SLOT(setXRotation(int)) );

   QSlider* z = new QSlider ( 0, 360, 60, 0, QSlider::Vertical, this, "zsl" );
   z->setTickmarks( QSlider::Left );
   connect( z, SIGNAL(valueChanged(int)), c1, SLOT(setZRotation(int)) );
   connect( z, SIGNAL(valueChanged(int)), c2, SLOT(setYRotation(int)) );

   // Now that we have all the widgets, put them into a nice layout

   // Put the sliders on top of each other
   QVBoxLayout* vlayout = new QVBoxLayout( 20, "vlayout");
   vlayout->addWidget( x );
   vlayout->addWidget( y );
   vlayout->addWidget( z );

   // Put the GL widgets inside the frames
   QHBoxLayout* flayout1 = new QHBoxLayout( f1, 2, 2, "flayout1");
   flayout1->addWidget( c1, 1 );
   QHBoxLayout* flayout2 = new QHBoxLayout( f2, 2, 2, "flayout2");
   flayout2->addWidget( c2, 1 );

   // Top level layout, puts the sliders to the left of the frame/GL widget
   QHBoxLayout* hlayout = new QHBoxLayout( this, 20, 20, "hlayout");
   hlayout->setMenuBar( m );
   hlayout->addLayout( vlayout );
   hlayout->addWidget( f1, 1 );
   hlayout->addWidget( f2, 1 );

}

void GLObjectWindow::deleteFirstWidget()
{
   // Delete only c1; c2 will keep working and use the shared display list
   if ( c1 ) {
    delete c1;
    c1 = 0;
   }
}
```

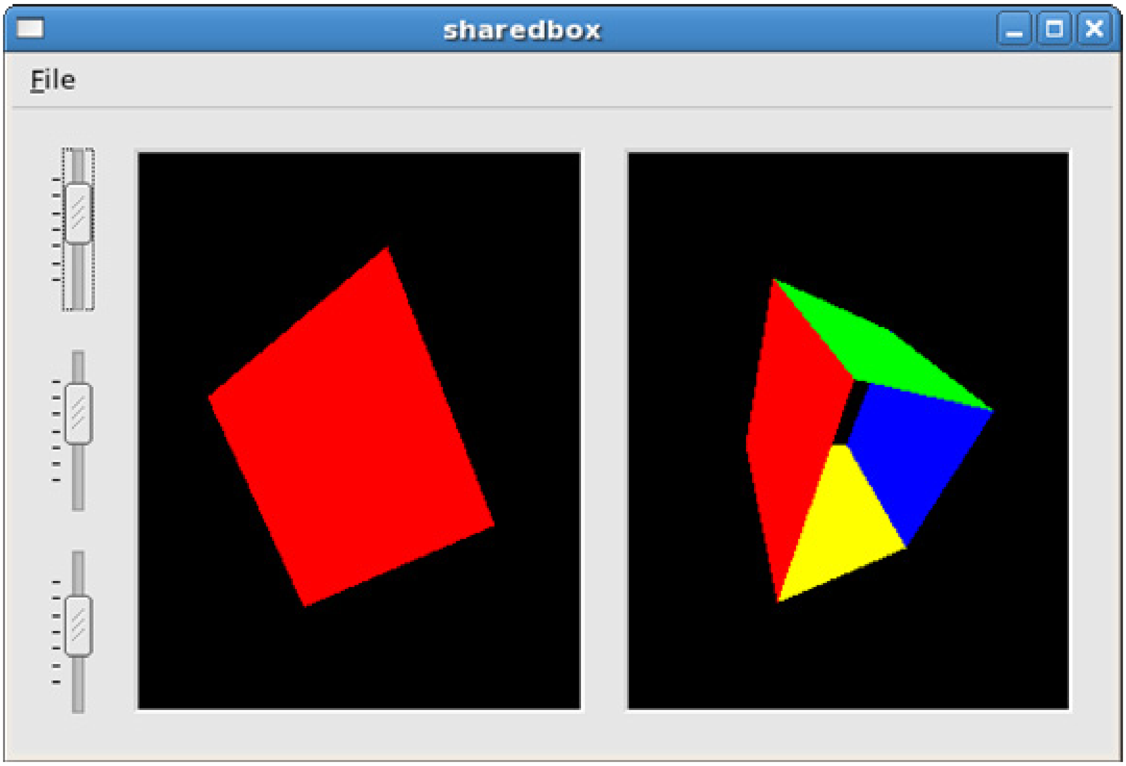
globjwin.h

```
/************************************************************************ The
GLObjectWindow contains a GLBox and three sliders connected to
** the GLBox's rotation slots.
************************************************************************/
#ifndef GLOBJWIN_H
#define GLOBJWIN_H

#include <qwidget.h>
class GLBox;
class GLObjectWindow : public QWidget
{
   Q_OBJECT
public:
   GLObjectWindow( QWidget* parent = 0, const char* name = 0 );

protected slots:

   void      deleteFirstWidget();

private:
   GLBox* c1;
   GLBox* c2;
};

#endif // GLOBJWIN_H
```

main.cpp

```
// Qt OpenGL example: Shared Box
// A small example showing how to use OpenGL display list sharing
// File: main.cpp
// The main() function
#include "globjwin.h"
#include <qapplication.h>
#include <qgl.h>

/*
  The main program is here.
*/

int main( int argc, char **argv )
{
   QApplication::setColorSpec( QApplication::CustomColor );
   QApplication a(argc,argv);

   if ( !QGLFormat::hasOpenGL() ) {
     qWarning( "This system has no OpenGL support. Exiting." );
     return -1;
   }

   GLObjectWindow w;
   w.resize( 550, 350 );
   a.setMainWidget( &w );
   w.show();
   return a.exec();
}
```

실행



5) OpenGL 본문실례

이 실례프로그람은 OpenGL 2D본문의 사용법을 보여준다.

texture.pro

```
TEMPLATE  = app
TARGET     = texture
CONFIG      += qt opengl warn_on release
CONFIG -= dlopen_opengl
HEADERS          = gltexobj.h \
            globjwin.h
SOURCES          = gltexobj.cpp \
            globjwin.cpp \
            main.cpp
```

globjwin.cpp

```
#include <qpushbutton.h>
#include <qslider.h>
#include <qlayout.h>
#include <qframe.h>
#include <qmenubar.h>
#include <qpopupmenu.h>
#include <qapplication.h>
#include <qkeycode.h>
#include "globjwin.h"
#include "gltexobj.h"

GLObjectWindow::GLObjectWindow( QWidget* parent, const char* name )
```

```
    : QWidget( parent, name )
{

    // Create nice frames to put around the OpenGL widgets
    QFrame* f1 = new QFrame( this, "frame1" );
    f1->setFrameStyle( QFrame::Sunken | QFrame::Panel );
    f1->setLineWidth( 2 );

    // Create an OpenGL widget
    GLTexobj* c = new GLTexobj( f1, "glbox1");

    // Create a menu
    QPopupMenu *file = new QPopupMenu( this );
    file->insertItem( "Toggle Animation", c, SLOT(toggleAnimation()),
              CTRL+Key_A );
    file->insertSeparator();
    file->insertItem( "Exit",  qApp, SLOT(quit()), CTRL+Key_Q );

    // Create a menu bar
    QMenuBar *m = new QMenuBar( this );
    m->setSeparator( QMenuBar::InWindowsStyle );
    m->insertItem("&File", file );

    // Create the three sliders; one for each rotation axis
    QSlider* x = new QSlider ( 0, 360, 60, 0, QSlider::Vertical, this, "xsl" );
    x->setTickmarks( QSlider::Left );
    connect( x, SIGNAL(valueChanged(int)), c, SLOT(setXRotation(int)) );

    QSlider* y = new QSlider ( 0, 360, 60, 0, QSlider::Vertical, this, "ysl" );
    y->setTickmarks( QSlider::Left );
    connect( y, SIGNAL(valueChanged(int)), c, SLOT(setYRotation(int)) );

    QSlider* z = new QSlider ( 0, 360, 60, 0, QSlider::Vertical, this, "zsl" );
    z->setTickmarks( QSlider::Left );
    connect( z, SIGNAL(valueChanged(int)), c, SLOT(setZRotation(int)) );


    // Now that we have all the widgets, put them into a nice layout

    // Put the sliders on top of each other
    QVBoxLayout* vlayout = new QVBoxLayout( 20, "vlayout");
    vlayout->addWidget( x );
    vlayout->addWidget( y );
    vlayout->addWidget( z );

    // Put the GL widget inside the frame
    QHBoxLayout* flayout1 = new QHBoxLayout( f1, 2, 2, "flayout1");
    flayout1->addWidget( c, 1 );

    // Top level layout, puts the sliders to the left of the frame/GL widget
    QHBoxLayout* hlayout = new QHBoxLayout( this, 20, 20, "hlayout");
    hlayout->setMenuBar( m );
    hlayout->addLayout( vlayout );
    hlayout->addWidget( f1, 1 );
}
```

globjwin.h

```
/****************************************************************
** The GLObjectWindow contains a GLBox and three sliders connected to
** the GLBox's rotation slots.
****************************************************************/
#ifndef GLOBJWIN_H
#define GLOBJWIN_H

#include <qwidget.h>


class GLObjectWindow : public QWidget
{
    Q_OBJECT
public:
    GLObjectWindow( QWidget* parent = 0, const char* name = 0 );

};

#endif // GLOBJWIN_H
```

gltexobj.cpp

```
/****************************************************************
** This is a simple QGLWidget demonstrating the use of QImages for textures.
** Much of the GL code is inspired by the 'spectex' and 'texcyl'
** public domain demo programs by Brian Paul.
****************************************************************/
#include "gltexobj.h"
#include <qimage.h>
#include <qtimer.h>

const int redrawWait = 50;

/*!
  Create a GLTexobj widget
*/

GLTexobj::GLTexobj( QWidget* parent, const char* name )
    : QGLWidget( parent, name )
{
    xRot = yRot = zRot = 0.0;           // default object rotation
    scale = 5.0;            // default object scale
    object = 0;
    animation = TRUE;
    timer = new QTimer( this );
    connect( timer, SIGNAL(timeout()), SLOT(update()) );
    timer->start( redrawWait, TRUE );
}

/*!
  Release allocated resources
```

```
*/

GLTexobj::~GLTexobj()
{
    makeCurrent();
    glDeleteLists( object, 1 );
}

/*!
  Paint the texobj. The actual openGL commands for drawing the texobj are
  performed here.
*/

void GLTexobj::paintGL()
{
    if ( animation ) {
      xRot += 1.0;
      yRot += 2.5;
      zRot -= 5.0;
    }
    glClear( GL_COLOR_BUFFER_BIT );
    glPushMatrix();
    glRotatef( xRot, 1.0, 0.0, 0.0 );
    glRotatef( yRot, 0.0, 1.0, 0.0 );
    glRotatef( zRot, 0.0, 0.0, 1.0 );
    glScalef( scale, scale, scale );
    glCallList( object );
    glPopMatrix();

    if ( animation ) {
      glFlush(); // Make sure everything is drawn before restarting timer
      timer->start( redrawWait, TRUE ); // Wait this many msecs before redraw
    }
}

/*!
  Set up the OpenGL rendering state, and define display list
*/

void GLTexobj::initializeGL()
{
    // Set up the lights

    GLfloat whiteDir[4] = {2.0, 2.0, 2.0, 1.0};
    GLfloat whiteAmb[4] = {1.0, 1.0, 1.0, 1.0};
    GLfloat lightPos[4] = {30.0, 30.0, 30.0, 1.0};

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_FALSE);
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, whiteAmb);

    glMaterialfv(GL_FRONT, GL_DIFFUSE, whiteDir);
    glMaterialfv(GL_FRONT, GL_SPECULAR, whiteDir);
    glMaterialf(GL_FRONT, GL_SHININESS, 20.0);

    glLightfv(GL_LIGHT0, GL_DIFFUSE, whiteDir);          // enable diffuse
    glLightfv(GL_LIGHT0, GL_SPECULAR, whiteDir); // enable specular
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);

    // Set up the textures

    QImage tex1, tex2, buf;

    if ( !buf.load( "gllogo.bmp" ) ) { // Load first image from file
      qWarning( "Could not read image file, using single-color instead." );
      QImage dummy( 128, 128, 32 );
      dummy.fill( Qt::green.rgb() );
      buf = dummy;
    }
    tex1 = QGLWidget::convertToGLFormat( buf );  // flipped 32bit RGBA

    if ( !buf.load( "qtlogo.bmp" ) ) { // Load first image from file
      qWarning( "Could not read image file, using single-color instead." );
      QImage dummy( 128, 128, 32 );
      dummy.fill( Qt::red.rgb() );
      buf = dummy;
    }
    tex2 = QGLWidget::convertToGLFormat( buf );  // flipped 32bit RGBA

    glTexParameterf( GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR );
    glTexParameterf( GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR );
    glEnable( GL_TEXTURE_2D );

    // Set up various other stuff

    glClearColor( 0.0, 0.0, 0.0, 0.0 ); // Let OpenGL clear to black
    glEnable( GL_CULL_FACE );   // don't need Z testing for convex objects
    glHint( GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST );

    // Make the object display list

    object = makeObject( tex1, tex2 );    // Generate an OpenGL display list
}

/*!
  Set up the OpenGL view port, matrix mode, etc.
*/

void GLTexobj::resizeGL( int w, int h )
{
    glViewport( 0, 0, w, h );
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    glFrustum( -1.0, 1.0, -1.0, 1.0, 10.0, 100.0 );
    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
    glTranslatef( 0.0, 0.0, -70.0 );
```

```
}

/*!
  Generate an OpenGL display list for the object to be shown, i.e. the texobj
*/

GLuint GLTexobj::makeObject( const QImage& tex1, const QImage& tex2 )
{
    GLUquadricObj* q = gluNewQuadric();
    GLuint cylinderObj = glGenLists(1);
    glNewList( cylinderObj, GL_COMPILE );

    glTranslatef( 0.0, 0.0, -1.0 );

    // cylinder
    glTexImage2D( GL_TEXTURE_2D, 0, 3, tex1.width(), tex1.height(), 0,
            GL_RGBA, GL_UNSIGNED_BYTE, tex1.bits() );
    gluQuadricTexture( q, GL_TRUE );
    gluCylinder(q, 0.6, 0.6, 2.0, 24, 1);

    // end cap
    glTexImage2D( GL_TEXTURE_2D, 0, 3, tex2.width(), tex2.height(), 0,
            GL_RGBA, GL_UNSIGNED_BYTE, tex2.bits() );
    glTranslatef( 0.0, 0.0, 2.0 );
    gluDisk( q, 0.0, 0.6, 24, 1 );

    // other end cap
    glTranslatef( 0.0, 0.0, -2.0 );
    gluQuadricOrientation( q, (GLenum)GLU_INSIDE );
    gluDisk( q, 0.0, 0.6, 24, 1 );

    glEndList();
    gluDeleteQuadric( q );

    return cylinderObj;
}

/*!
  Set the rotation angle of the object to \e degrees around the X axis.
*/

void GLTexobj::setXRotation( int degrees )
{
    xRot = (GLfloat)(degrees % 360);
    updateGL();
}

/*!
  Set the rotation angle of the object to \e degrees around the Y axis.
*/

void GLTexobj::setYRotation( int degrees )
{
    yRot = (GLfloat)(degrees % 360);
```

```
   updateGL();
}

/*!
  Set the rotation angle of the object to \e degrees around the Z axis.
*/

void GLTexobj::setZRotation( int degrees )
{
   zRot = (GLfloat)(degrees % 360);
   updateGL();
}

/*!
  Turns animation on or off
*/

void GLTexobj::toggleAnimation()
{
   animation = !animation;
   if ( animation )
     updateGL();
   else
     timer->stop();
}
```

gltexobj.h

```
/********************************************************************
** This is a simple QGLWidget displaying an openGL wireframe box
********************************************************************/
#ifndef GLTEXOBJ_H
#define GLTEXOBJ_H
#include <qgl.h>

class GLTexobj : public QGLWidget
{
   Q_OBJECT

public:

   GLTexobj( QWidget* parent, const char* name );
   ~GLTexobj();

public slots:

   void     setXRotation( int degrees );
   void     setYRotation( int degrees );
   void     setZRotation( int degrees );
   void     toggleAnimation();

protected:

   void     initializeGL();
   void     paintGL();
```

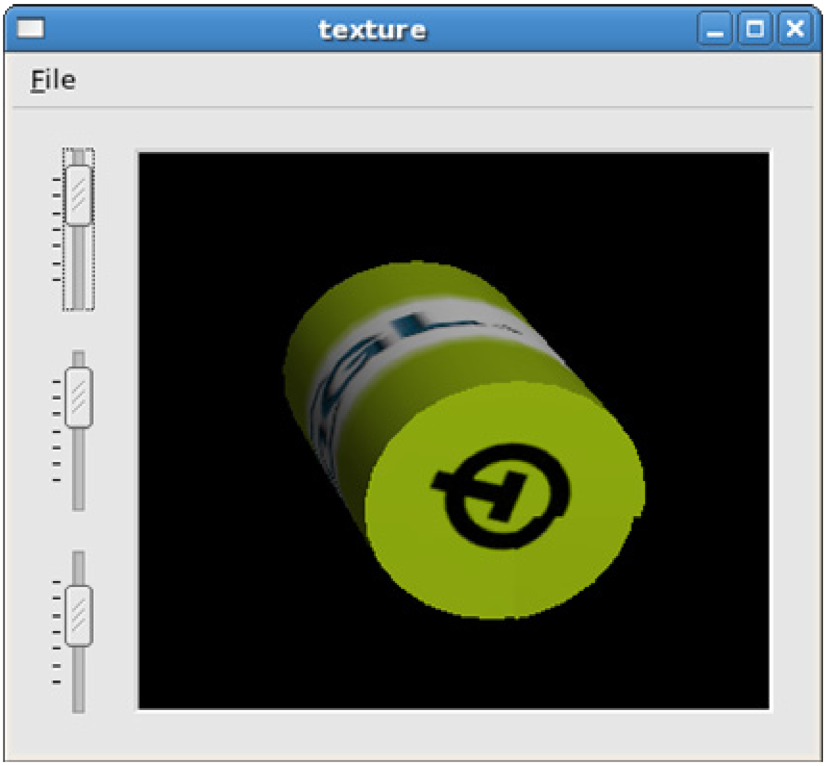
```
    void      resizeGL( int w, int h );

    virtual GLuint  makeObject( const QImage& tex1, const QImage& tex2 );

private:
    bool animation;
    GLuint object;
    GLfloat xRot, yRot, zRot, scale;
    QTimer* timer;
};

#endif // GLTEXOBJ_H
```

main.cpp

```
// Qt OpenGL example: Texture
// File: main.cpp
// The main() function
#include "globjwin.h"
#include <qapplication.h>
#include <qgl.h>

/*
  The main program is here.
*/

int main( int argc, char **argv )
{
    QApplication::setColorSpec( QApplication::CustomColor );
    QApplication a(argc,argv);

    if ( !QGLFormat::hasOpenGL() ) {
      qWarning( "This system has no OpenGL support. Exiting." );
      return -1;
    }

    GLObjectWindow* w = new GLObjectWindow;
    w->resize( 400, 350 );
    a.setMainWidget( w );
    w->show();
    int result = a.exec();
    delete w;
    return result;
}
```

gllogo.png

실행



6) overlay_x11

opengl/overlay_x11/overlayrubber.pro

```
TEMPLATE  = app
TARGET      = overlayrubber
CONFIG      += qt opengl warn_on release
CONFIG -= dlopen_opengl
HEADERS         = gearwidget.h \
          rubberbandwidget.h
SOURCES         = gearwidget.cpp \
          main.cpp \
          rubberbandwidget.cpp
```

opengl/overlay_x11/gearwidget.cpp

```
// A Qt OpenGL widget that draws a gear.
// Portions of this code has been borrowed from Brian Paul's Mesa distribution.

#include "gearwidget.h"
#include <math.h>
#if defined(Q_WS_X11)
#include <X11/Xlib.h>
#endif

#if defined(Q_CC_MSVC)
#pragma warning(disable:4305) // init: truncation from const double to float
#endif
```

```
/*
 * Draw a gear wheel.  You'll probably want to call this function when
 * building a display list since we do a lot of trig here.
 *
 * Input:  inner_radius - radius of hole at center
 *     outer_radius - radius at center of teeth
 *     width - width of gear
 *     teeth - number of teeth
 *     tooth_depth - depth of tooth
 */
static void gear( GLfloat inner_radius, GLfloat outer_radius, GLfloat width,
          GLint teeth, GLfloat tooth_depth )
{
   GLint i;
   GLfloat r0, r1, r2;
   GLfloat angle, da;
   GLfloat u, v, len;

   r0 = inner_radius;
   r1 = outer_radius - tooth_depth/2.0;
   r2 = outer_radius + tooth_depth/2.0;

   const double pi = 3.14159264;
   da = 2.0*pi / teeth / 4.0;

   glShadeModel( GL_FLAT );

   glNormal3f( 0.0, 0.0, 1.0 );

   /* draw front face */
   glBegin( GL_QUAD_STRIP );
   for (i=0;i<=teeth;i++) {
    angle = i * 2.0*pi / teeth;
    glVertex3f( r0*cos(angle), r0*sin(angle), width*0.5 );
    glVertex3f( r1*cos(angle), r1*sin(angle), width*0.5 );
    glVertex3f( r0*cos(angle), r0*sin(angle), width*0.5 );
    glVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da), width*0.5 );
   }
   glEnd();

   /* draw front sides of teeth */
   glBegin( GL_QUADS );
   da = 2.0*pi / teeth / 4.0;
   for (i=0;i<teeth;i++) {
    angle = i * 2.0*pi / teeth;

    glVertex3f( r1*cos(angle),      r1*sin(angle),     width*0.5 );
    glVertex3f( r2*cos(angle+da),   r2*sin(angle+da),       width*0.5 );
    glVertex3f( r2*cos(angle+2*da), r2*sin(angle+2*da), width*0.5 );
    glVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da), width*0.5 );
   }
   glEnd();

   glNormal3f( 0.0, 0.0, -1.0 );
```

```
/* draw back face */
glBegin( GL_QUAD_STRIP );
for (i=0;i<=teeth;i++) {
  angle = i * 2.0*pi / teeth;
  glVertex3f( r1*cos(angle), r1*sin(angle), -width*0.5 );
  glVertex3f( r0*cos(angle), r0*sin(angle), -width*0.5 );
  glVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da), -width*0.5 );
  glVertex3f( r0*cos(angle), r0*sin(angle), -width*0.5 );
}
glEnd();

/* draw back sides of teeth */
glBegin( GL_QUADS );
da = 2.0*pi / teeth / 4.0;
for (i=0;i<teeth;i++) {
  angle = i * 2.0*pi / teeth;

  glVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da), -width*0.5 );
  glVertex3f( r2*cos(angle+2*da), r2*sin(angle+2*da), -width*0.5 );
  glVertex3f( r2*cos(angle+da),   r2*sin(angle+da),      -width*0.5 );
  glVertex3f( r1*cos(angle),      r1*sin(angle),     -width*0.5 );
}
glEnd();

/* draw outward faces of teeth */
glBegin( GL_QUAD_STRIP );
for (i=0;i<teeth;i++) {
  angle = i * 2.0*pi / teeth;

  glVertex3f( r1*cos(angle),      r1*sin(angle),      width*0.5 );
  glVertex3f( r1*cos(angle),      r1*sin(angle),     -width*0.5 );
  u = r2*cos(angle+da) - r1*cos(angle);
  v = r2*sin(angle+da) - r1*sin(angle);
  len = sqrt( u*u + v*v );
  u /= len;
  v /= len;
  glNormal3f( v, -u, 0.0 );
  glVertex3f( r2*cos(angle+da),   r2*sin(angle+da),       width*0.5 );
  glVertex3f( r2*cos(angle+da),   r2*sin(angle+da),      -width*0.5 );
  glNormal3f( cos(angle), sin(angle), 0.0 );
  glVertex3f( r2*cos(angle+2*da), r2*sin(angle+2*da),  width*0.5 );
  glVertex3f( r2*cos(angle+2*da), r2*sin(angle+2*da), -width*0.5 );
  u = r1*cos(angle+3*da) - r2*cos(angle+2*da);
  v = r1*sin(angle+3*da) - r2*sin(angle+2*da);
  glNormal3f( v, -u, 0.0 );
  glVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da),  width*0.5 );
  glVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da), -width*0.5 );
  glNormal3f( cos(angle), sin(angle), 0.0 );
}

glVertex3f( r1*cos(0.0), r1*sin(0.0), width*0.5 );
glVertex3f( r1*cos(0.0), r1*sin(0.0), -width*0.5 );
```

```
   glEnd();

   glShadeModel( GL_SMOOTH );

   /* draw inside radius cylinder */
   glBegin( GL_QUAD_STRIP );
   for (i=0;i<=teeth;i++) {
    angle = i * 2.0*pi / teeth;
    glNormal3f( -cos(angle), -sin(angle), 0.0 );
    glVertex3f( r0*cos(angle), r0*sin(angle), -width*0.5 );
    glVertex3f( r0*cos(angle), r0*sin(angle), width*0.5 );
   }
   glEnd();

}

static GLfloat view_rotx=20.0, view_roty=30.0, view_rotz=0.0;
static GLint gear1, gear2, gear3;
static GLfloat angle = 0.0;

static void draw()
{
   angle += 2.0;
   view_roty += 1.0;

   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

   glPushMatrix();
   glRotatef( view_rotx, 1.0, 0.0, 0.0 );
   glRotatef( view_roty, 0.0, 1.0, 0.0 );
   glRotatef( view_rotz, 0.0, 0.0, 1.0 );

   glPushMatrix();
   glTranslatef( -3.0, -2.0, 0.0 );
   glRotatef( angle, 0.0, 0.0, 1.0 );
   glCallList(gear1);
   glPopMatrix();

   glPushMatrix();
   glTranslatef( 3.1, -2.0, 0.0 );
   glRotatef( -2.0*angle-9.0, 0.0, 0.0, 1.0 );
   glCallList(gear2);
   glPopMatrix();

   glPushMatrix();
   glTranslatef( -3.1, 2.2, -1.8 );
   glRotatef( 90.0, 1.0, 0.0, 0.0 );
   glRotatef( 2.0*angle-2.0, 0.0, 0.0, 1.0 );
   glCallList(gear3);
   glPopMatrix();

   glPopMatrix();
}
```

```
GearWidget::GearWidget( QWidget *parent, const char *name )   : QGLWidget( parent, name )
{
}

void GearWidget::initializeGL()
{
   static GLfloat pos[4] = {5.0, 5.0, 10.0, 1.0 };
   static GLfloat redgear[4] = {0.8, 0.1, 0.0, 1.0 };
   static GLfloat greengear[4] = {0.0, 0.8, 0.2, 1.0 };
   static GLfloat bluegear[4] = {0.2, 0.2, 1.0, 1.0 };

   glLightfv( GL_LIGHT0, GL_POSITION, pos );
   glEnable( GL_CULL_FACE );
   glEnable( GL_LIGHTING );
   glEnable( GL_LIGHT0 );
   glEnable( GL_DEPTH_TEST );

   /* make the gears */
   gear1 = glGenLists(1);
   glNewList(gear1, GL_COMPILE);
   glMaterialfv( GL_FRONT, GL_AMBIENT_AND_DIFFUSE, redgear );
   gear( 1.0, 4.0, 1.0, 20, 0.7 );
   glEndList();

   gear2 = glGenLists(1);
   glNewList(gear2, GL_COMPILE);
   glMaterialfv( GL_FRONT, GL_AMBIENT_AND_DIFFUSE, greengear );
   gear( 0.5, 2.0, 2.0, 10, 0.7 );
   glEndList();

   gear3 = glGenLists(1);
   glNewList(gear3, GL_COMPILE);
   glMaterialfv( GL_FRONT, GL_AMBIENT_AND_DIFFUSE, bluegear );
   gear( 1.3, 2.0, 0.5, 10, 0.7 );
   glEndList();

   glEnable( GL_NORMALIZE );
}

void GearWidget::resizeGL( int width, int height )
{
   GLfloat w = (float) width / (float) height;
   GLfloat h = 1.0;

   glViewport( 0, 0, width, height );
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glFrustum( -w, w, -h, h, 5.0, 60.0 );
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   glTranslatef( 0.0, 0.0, -40.0 );
}

void GearWidget::paintGL()
```

```
{
    qDebug( "GearWidget: Doing GL rendering." );
#if defined (Q_GLX)
    static bool doneIt = FALSE;
    if ( !doneIt ) {
        doneIt = TRUE;
        // Print out the Visual ID. Access to this will be made
        // simpler in future versions of Qt!
        XWindowAttributes a;
        XGetWindowAttributes( x11Display(), winId(), &a );
        qDebug( "QGLWidget: using Visual ID: 0x%x.",
                (int)XVisualIDFromVisual( a.visual ) );
    }
#endif
    draw();
}
```

opengl/overlay_x11/gearwidget.h

```
#ifndef GEAR_H
#define GEAR_H

#include <qgl.h>

class GearWidget : public QGLWidget
{
public:
    GearWidget( QWidget *parent=0, const char *name=0 );

protected:
    void initializeGL();
    void resizeGL( int, int );
    void paintGL();
};

#endif
```

opengl/overlay_x11/main.cpp

```
#include <qapplication.h>
#include "gearwidget.h"
#include "rubberbandwidget.h"

#if defined(Q_WS_X11)
#include <X11/Xlib.h>
#endif

QColor findOverlayTransparentColor()
{
    QColor invalidColor;

#if defined(Q_WS_X11)

    Display* appDisplay;
    Visual* appVisual;
```

```
  // The static methods are called 'App' in Qt 2.x
#if QT_VERSION < 200
  appDisplay = QPaintDevice::x__Display();
  appVisual = (Visual*)QPaintDevice::x11Visual();
#else
  appDisplay = QPaintDevice::x11AppDisplay();
  appVisual = (Visual*)QPaintDevice::x11AppVisual();
#endif

  qDebug( "Default Visual ID: 0x%x", (int)XVisualIDFromVisual(appVisual) );

  typedef struct OverlayProp {
   long  visual;
   long  type;
   long  value;
   long  layer;
  } OverlayProp;

  QWidget* rootWin = QApplication::desktop();
  if ( !rootWin )
   return invalidColor; // Should not happen

  Atom overlayVisualsAtom = XInternAtom( appDisplay,
                          "SERVER_OVERLAY_VISUALS", True );
  if ( overlayVisualsAtom == None ) {
   warning( "Server has no overlay visuals" );
   return invalidColor;
  }

  Atom actualType;
  int actualFormat;
  ulong nItems;
  ulong bytesAfter;
  OverlayProp* overlayProp;
  int res = XGetWindowProperty( appDisplay, QApplication::desktop()->winId(),
                  overlayVisualsAtom, 0, 10000, False,
                  overlayVisualsAtom, &actualType,
                  &actualFormat, &nItems, &bytesAfter,
                  (uchar**)&overlayProp );

  if ( res != Success || actualType != overlayVisualsAtom
    || actualFormat != 32 || nItems < 4 ) {
   warning( "Failed to get overlay visual property from server" );
   return invalidColor;
  }


  for ( uint i = 0; i < nItems/4; i++ ) {
   if ( (VisualID)overlayProp[i].visual == XVisualIDFromVisual(appVisual)
      && overlayProp[i].type == 1 )
     return QColor( qRgb( 1, 2, 3 ), overlayProp[i].value );
  }

  qWarning( "Default visual is not in overlay plane" );
```

```
    return invalidColor;

#else // defined(Q_WS_X11)
    qWarning( "Wrong window system - Only X11 has overlay support." );
    return invalidColor;
#endif
}

int main( int argc, char **argv )
{
    QApplication::setColorSpec( QApplication::CustomColor );
    QApplication a( argc, argv );

    if ( !QGLFormat::hasOpenGL() ) {
      qWarning( "This system has no OpenGL support. Exiting." );
      return -1;
    }

    QColor transparentColor = findOverlayTransparentColor();
    if ( !transparentColor.isValid() ) {
      qWarning( "Failed to get transparent color for overlay. Exiting." );
      return -1;
    }

    QWidget top;
    a.setMainWidget( &top );
    top.setGeometry( 50, 50, 600, 400 );

    // Make an OpenGL widget. It will use the deepest visual available
    // (typically a TrueColor visual), which typically is in the normal layer.
    GearWidget g( &top );
    g.setGeometry( 20, 20, 560, 360 );

    // Put the rubberband widget (which uses the default, i.e. overlay visual)
    // on top of the OpenGL widget:
    RubberbandWidget r( transparentColor, &top );
    r.setGeometry( 20, 20, 560, 360 );

    top.show();
    return a.exec();
}
```

opengl/overlay_x11/rubberbandwidget.cpp

```
#include "rubberbandwidget.h"
#include <qpainter.h>

RubberbandWidget::RubberbandWidget( QColor transparentColor, QWidget *parent,
                        const char *name, WFlags f )  : QWidget( parent, name, f )
{
    setBackgroundColor( transparentColor );
    on = FALSE;
}

void RubberbandWidget::mousePressEvent( QMouseEvent* e )
```

```
{
  p1 = e->pos();
  p2 = p1;
  p3 = p1;
  on = TRUE;
  setMouseTracking( TRUE );
}

void RubberbandWidget::mouseMoveEvent( QMouseEvent* e )
{
  if ( on ) {
   p2 = e->pos();
   QPainter p( this );
   // Erase last drawn rubberband:
   p.setPen( QPen( backgroundColor(), 3 ) );
   p.drawRect( QRect( p1, p3 ) );
   // Draw the new one:
   p.setPen( QPen( white, 3 ) );
   p.drawRect( QRect(p1, p2) );
   p3 = p2;
  }
}

void RubberbandWidget::mouseReleaseEvent( QMouseEvent* )
{
  if ( on ) {
   QPainter p ( this );
   p.eraseRect( rect() );
  }
  on = FALSE;
  setMouseTracking( FALSE );
}
```

opengl/overlay_x11/rubberbandwidget.h

```
#ifndef RUBBERBANDWIDGET_H
#define RUBBERBANDWIDGET_H

#include <qwidget.h>

class RubberbandWidget : public QWidget
{
public:
  RubberbandWidget( QColor transparentColor, QWidget *parent=0,
          const char *name=0, WFlags f=0 );

protected:
  void mousePressEvent( QMouseEvent* e );
  void mouseMoveEvent( QMouseEvent* e );
  void mouseReleaseEvent( QMouseEvent* e );

  QColor c;
  QPoint p1;
  QPoint p2;
  QPoint p3;
  bool on;
};

#endif
```

opengl/overlay_x11/utilities/glxvisuals/glxvisuals.c

```
#include <stdlib.h>
#include <stdio.h>
#include <X11/Xlib.h>
#include <GL/glx.h>

static char *ClassOf(int c);
static char *Format(int n, int w);

void
main(int argc, char *argv[])
{
 Display *dpy;
 XVisualInfo match, *visualList, *vi, *visualToTry;
 int errorBase, eventBase, major, minor, found;
 int glxCapable, bufferSize, level, renderType, doubleBuffer, stereo,
   auxBuffers, redSize, greenSize, blueSize, alphaSize, depthSize,
   stencilSize, acRedSize, acGreenSize, acBlueSize, acAlphaSize;

 dpy = XOpenDisplay(NULL);
 if (!dpy) {
   fprintf(stderr, "Could not connect to %s.\n", XDisplayName(NULL));
   exit(1);
 }
 if (glXQueryExtension(dpy, &errorBase, &eventBase) == False) {
   fprintf(stderr, "OpenGL not supported by X server.\n");
   exit(1);
 }

 glXQueryVersion(dpy, &major, &minor);
 printf("display: %s\n", XDisplayName(NULL));
 printf("using GLX version: %d.%d\n\n", major, minor);

 match.screen = DefaultScreen(dpy);
 visualList = XGetVisualInfo(dpy, VisualScreenMask, &match, &found);

 printf("   visual     bf lv rg d st  r  g  b a   ax dp st accum buffs\n");
 printf(" id dep cl    sz l  ci b ro sz sz sz sz  bf th cl  r  g  b  a\n");
 printf("-------------------------------------------------------------\n");

 visualToTry = NULL;
 for(vi = visualList; found > 0; found--, vi++) {
   glXGetConfig(dpy, vi, GLX_USE_GL, &glxCapable);
   if (glxCapable) {
     printf("0x%x %2d %s", vi->visualid, vi->depth, ClassOf(vi->class));
     glXGetConfig(dpy, vi, GLX_BUFFER_SIZE, &bufferSize);
     glXGetConfig(dpy, vi, GLX_LEVEL, &level);
     glXGetConfig(dpy, vi, GLX_RGBA, &renderType);
     glXGetConfig(dpy, vi, GLX_DOUBLEBUFFER, &doubleBuffer);
```

```
      glXGetConfig(dpy, vi, GLX_STEREO, &stereo);
      glXGetConfig(dpy, vi, GLX_AUX_BUFFERS, &auxBuffers);
      glXGetConfig(dpy, vi, GLX_RED_SIZE, &redSize);
      glXGetConfig(dpy, vi, GLX_GREEN_SIZE, &greenSize);
      glXGetConfig(dpy, vi, GLX_BLUE_SIZE, &blueSize);
      glXGetConfig(dpy, vi, GLX_ALPHA_SIZE, &alphaSize);
      glXGetConfig(dpy, vi, GLX_DEPTH_SIZE, &depthSize);
      glXGetConfig(dpy, vi, GLX_STENCIL_SIZE, &stencilSize);
      glXGetConfig(dpy, vi, GLX_ACCUM_RED_SIZE, &acRedSize);
      glXGetConfig(dpy, vi, GLX_ACCUM_GREEN_SIZE, &acGreenSize);
      glXGetConfig(dpy, vi, GLX_ACCUM_BLUE_SIZE, &acBlueSize);
      glXGetConfig(dpy, vi, GLX_ACCUM_ALPHA_SIZE, &acAlphaSize);
      printf("    %2s %2s %1s  %1s  %1s ",
        Format(bufferSize, 2), Format(level, 2),
        renderType ? "r" : "c",
      doubleBuffer ? "y" : ".",
      stereo ? "y" : ".");
      printf("%2s %2s %2s %2s ",
        Format(redSize, 2), Format(greenSize, 2),
      Format(blueSize, 2), Format(alphaSize, 2));
      printf("%2s %2s %2s %2s %2s %2s %2s",
        Format(auxBuffers, 2), Format(depthSize, 2), Format(stencilSize, 2),
        Format(acRedSize, 2), Format(acGreenSize, 2),
        Format(acBlueSize, 2), Format(acAlphaSize, 2));
      printf("\n");
      visualToTry = vi;
    }
  }

  if (visualToTry) {
    GLXContext context;
    Window window;
    Colormap colormap;
    XSetWindowAttributes swa;

    context = glXCreateContext(dpy, visualToTry, 0, GL_TRUE);
    colormap = XCreateColormap(dpy,
      RootWindow(dpy, visualToTry->screen),
      visualToTry->visual, AllocNone);
    swa.colormap = colormap;
    swa.border_pixel = 0;
    window = XCreateWindow(dpy, RootWindow(dpy, visualToTry->screen), 0, 0, 100, 100,
      0, visualToTry->depth, InputOutput, visualToTry->visual,
      CWBorderPixel | CWColormap, &swa);
    glXMakeCurrent(dpy, window, context);
    printf("\n");
    printf("OpenGL vendor string: %s\n", glGetString(GL_VENDOR));
    printf("OpenGL renderer string: %s\n", glGetString(GL_RENDERER));
    printf("OpenGL version string: %s\n", glGetString(GL_VERSION));
    if (glXIsDirect(dpy, context))
      printf("direct rendering: supported\n");
    printf( "GL extensions: '%s'\n\n", glGetString(GL_EXTENSIONS) );
#if defined(GLX_VERSION_1_1)
    printf( "GLX extensions: '%s'\n\n", glXQueryExtensionsString( dpy, visualToTry->screen ) );
```

```
#endif

  } else
    printf("No GLX-capable visuals!\n");
  XFree(visualList);
}

static char *
ClassOf(int c)
{
  switch (c) {
  case StaticGray:   return "sg";
  case GrayScale:    return "gs";
  case StaticColor:  return "sc";
  case PseudoColor:  return "pc";
  case TrueColor:    return "tc";
  case DirectColor:  return "dc";
  default:           return "??";
  }
}

static char *
Format(int n, int w)
{
  static char buffer[256];
  static int bufptr;
  char *buf;

  if (bufptr >= sizeof(buffer) - w)
    bufptr = 0;
  buf = buffer + bufptr;
  if (n == 0)
    sprintf(buf, "%*s", w, ".");
  else
    sprintf(buf, "%*d", w, n);
  bufptr += w + 1;
  return buf;
}
```

opengl/overlay_x11/utilities/sovinfo/sovinfo.c

```
/* compile: cc -o sovinfo sovinfo.c sovLayerUtil.c -lX11 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "sovLayerUtil.h"

int main(int argc, char *argv[])
{
  Display *dpy;
  char *display_name, *arg, *class;
  sovVisualInfo template, *lvinfo;
  int nVisuals, i, overlaysOnly = 0;

  display_name = NULL;
```

```
for (i = 1; i < argc; i++) {
  arg = argv[i];
  if (!strcmp(arg, "-display")) {
    if (++i >= argc) {
      fprintf(stderr, "sovinfo: missing argument to -display\n");
      exit(1);
    }
    display_name = argv[i];
  } else if (!strcmp(arg, "-overlays_only")) {
    overlaysOnly = 1;
  } else {
    fprintf(stderr,
      "usage: sovinfo [-display dpy] [-overlays_only]\n");
    exit(1);
  }
}
dpy = XOpenDisplay(display_name);
if (dpy == NULL) {
  fprintf(stderr, "sovinfo: cannot open display %s\n",
    XDisplayName(NULL));
  exit(1);
}
lvinfo = sovGetVisualInfo(dpy, 0L, &template, &nVisuals);
for (i = 0; i < nVisuals; i++) {
  if (!overlaysOnly || lvinfo[i].layer > 0) {
    printf("  Visual ID: 0x%x\n", lvinfo[i].vinfo.visualid);
    printf("    screen: %d\n", lvinfo[i].vinfo.screen);
    printf("    depth: %d\n", lvinfo[i].vinfo.depth);
    switch (lvinfo[i].vinfo.class) {
    case StaticGray:
      class = "StaticGray";
      break;
    case GrayScale:
      class = "GrayScale";
      break;
    case StaticColor:
      class = "StaticColor";
      break;
    case PseudoColor:
      class = "PseudoColor";
      break;
    case TrueColor:
      class = "TrueColor";
      break;
    case DirectColor:
      class = "DirectColor";
      break;
    default:
      class = "Unknown";
      break;
    }
    printf("    class: %s\n", class);
    switch (lvinfo[i].type) {
    case None:
        printf("    transparent type: None\n");
        break;
      case TransparentPixel:
        printf("    transparent type: TransparentPixel\n");
        printf("    pixel value: %d\n", lvinfo[i].value);
        break;
      case TransparentMask:
        printf("    transparent type: TransparentMask\n");
        printf("    transparency mask: %0x%x\n", lvinfo[i].value);
        break;
      default:
        printf("    transparent type: Unknown or invalid\n");
        break;
      }
      printf("    layer: %d\n", lvinfo[i].layer);
    }
  }
  return 0;
}
```

opengl/overlay_x11/utilities/sovinfo/sovlayerutil.c

```
#include <stdlib.h>
#include "sovLayerUtil.h"

static Bool layersRead;
static Atom overlayVisualsAtom;
static sovOverlayInfo **overlayInfoPerScreen;
static int *numOverlaysPerScreen;

sovVisualInfo * sovGetVisualInfo(Display *display, long lvinfo_mask,
  sovVisualInfo *lvinfo_template, int *nitems_return)
{
  XVisualInfo *vinfo;
  sovVisualInfo *layerInfo;
  Window root;
  Status status;
  Atom actualType;
  unsigned long sizeData, bytesLeft;
  int actualFormat, numVisuals, numScreens, count, i, j;

  vinfo = XGetVisualInfo(display, lvinfo_mask & VisualAllMask,
    &lvinfo_template->vinfo, nitems_return);
  if (vinfo == NULL)
    return NULL;
  numVisuals = *nitems_return;
  if (layersRead == False) {
    overlayVisualsAtom = XInternAtom(display,
        "SERVER_OVERLAY_VISUALS", True);
    if (overlayVisualsAtom != None) {
      numScreens = ScreenCount(display);
      overlayInfoPerScreen = (sovOverlayInfo **)
        malloc(numScreens * sizeof(sovOverlayInfo *));
      numOverlaysPerScreen = (int *) malloc(numScreens * sizeof(int));
      if (overlayInfoPerScreen != NULL &&
      numOverlaysPerScreen != NULL) {
      for (i = 0; i < numScreens; i++) {
        root = RootWindow(display, i);
        status = XGetWindowProperty(display, root, overlayVisualsAtom,
          0L, (long) 10000, False, overlayVisualsAtom,
        &actualType, &actualFormat,
          &sizeData, &bytesLeft,
        (unsigned char **) &overlayInfoPerScreen[i]);
        if (status != Success ||
        actualType != overlayVisualsAtom ||
          actualFormat != 32 || sizeData < 4)
          numOverlaysPerScreen[i] = 0;
        else
          numOverlaysPerScreen[i] = sizeData / 4;
      }
      layersRead = True;
    } else {
      if (overlayInfoPerScreen != NULL)
        free(overlayInfoPerScreen);
      if (numOverlaysPerScreen != NULL)
        free(numOverlaysPerScreen);
    }
  }
}
layerInfo = (sovVisualInfo *)
  malloc(numVisuals * sizeof(sovVisualInfo));
if (layerInfo == NULL) {
  XFree(vinfo);
  return NULL;
}
count = 0;
for (i = 0; i < numVisuals; i++) {
  XVisualInfo *pVinfo;
  int screen;
  sovOverlayInfo *overlayInfo;

  pVinfo = &vinfo[i];
  screen = pVinfo->screen;
  overlayInfo = NULL;
  if (layersRead) {
    for (j = 0; j < numOverlaysPerScreen[screen]; j++)
      if (pVinfo->visualid ==
      overlayInfoPerScreen[screen][j].overlay_visual) {
        overlayInfo = &overlayInfoPerScreen[screen][j];
        break;
      }
  }
  if (lvinfo_mask & VisualLayerMask)
    if (overlayInfo == NULL) {
      if (lvinfo_template->layer != 0)
        continue;
    } else if (lvinfo_template->layer != overlayInfo->layer)
      continue;
  if (lvinfo_mask & VisualTransparentType)
```

```
      if (overlayInfo == NULL) {
        if (lvinfo_template->type != None)
          continue;
      } else if (lvinfo_template->type !=
        overlayInfo->transparent_type)
        continue;
    if (lvinfo_mask & VisualTransparentValue)
      if (overlayInfo == NULL)
        /* non-overlay visuals have no sense of
           TransparentValue */
        continue;
      else if (lvinfo_template->value != overlayInfo->value)
        continue;
    layerInfo[count].vinfo = *pVinfo;
    if (overlayInfo == NULL) {
      layerInfo[count].layer = 0;
      layerInfo[count].type = None;
      layerInfo[count].value = 0;  /* meaningless */
    } else {
      layerInfo[count].layer = overlayInfo->layer;
      layerInfo[count].type = overlayInfo->transparent_type;
      layerInfo[count].value = overlayInfo->value;
    }
    count++;
  }
  XFree(vinfo);
  *nitems_return = count;
  if (count == 0) {
    XFree(layerInfo);
    return NULL;
  } else
    return layerInfo;
}

Status
sovMatchVisualInfo(Display *display, int screen,
  int depth, int class, int layer, sovVisualInfo *lvinfo_return)
{
  sovVisualInfo *lvinfo;
  sovVisualInfo lvinfoTemplate;
  int nitems;

  lvinfoTemplate.vinfo.screen = screen;
  lvinfoTemplate.vinfo.depth = depth;
  lvinfoTemplate.vinfo.class = class;
  lvinfoTemplate.layer = layer;
  lvinfo = sovGetVisualInfo(display,
    VisualScreenMask|VisualDepthMask|VisualClassMask|VisualLayerMask,
    &lvinfoTemplate, &nitems);
  if (lvinfo != NULL && nitems > 0) {
    *lvinfo_return = *lvinfo;
    return 1;
  } else
    return 0;
```

```
}
```

opengl/overlay_x11/utilities/sovinfo/sovlayerutil.h

```
#ifndef __sovLayerUtil_h__
#define __sovLayerUtil_h__

#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xmd.h>

/* Transparent type values */
/*      None                0 */
#define TransparentPixel      1
#define TransparentMask       2

/* layered visual info template flags */
#define VisualLayerMask         0x200
#define VisualTransparentType 0x400
#define VisualTransparentValue    0x800
#define VisualAllLayerMask    0xFFF

/* layered visual info structure */
typedef struct _sovVisualInfo {
   XVisualInfo vinfo;
   int layer;
   int type;
   unsigned long value;
} sovVisualInfo;

/* SERVER_OVERLAY_VISUALS property element */
typedef struct _sovOverlayInfo {
   long  overlay_visual;
   long  transparent_type;
   long  value;
   long  layer;
} sovOverlayInfo;

extern sovVisualInfo *sovGetVisualInfo(
  Display *display,
  long lvinfo_mask,
  sovVisualInfo *lvinfo_template,
  int *nitems_return);
extern Status sovMatchVisualInfo(
  Display *display,
  int screen,
  int depth,
  int class,
  int layer,
  sovVisualInfo *lvinfo_return);

#endif /* __sovLayerUtil_h__ */
```

40. 그림

이 실례는 그림을 만들고 파일에 보관하고 그것을 그리기지령들의 모임으로 읽어들이는 방법을 보여준다.

picture.pro

```
TEMPLATE  = app
TARGET    = picture
CONFIG    += qt warn_on release
HEADERS       =
SOURCES       = picture.cpp
```

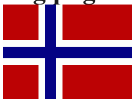
picture.cpp

```
#include <qapplication.h>
#include <qpainter.h>
#include <qpicture.h>
#include <qpixmap.h>
#include <qwidget.h>
#include <qmessagebox.h>
#include <qfile.h>
#include <ctype.h>

void paintCar( QPainter *p )              // paint a car
{
   QPointArray a;
   QBrush brush( Qt::yellow, Qt::SolidPattern );
   p->setBrush( brush );           // use solid, yellow brush

   a.setPoints( 5, 50,50, 350,50, 450,120, 450,250, 50,250 );
   p->drawPolygon( a );           // draw car body

   QFont f( "courier", 12, QFont::Bold );
   p->setFont( f );

   QColor windowColor( 120, 120, 255 ); // a light blue color
   brush.setColor( windowColor );       // set this brush color
   p->setBrush( brush );           // set brush
   p->drawRect( 80, 80, 250, 70 );       // car window
   p->drawText( 180, 80, 150, 70, Qt::AlignCenter, "--  Qt  --\nTrolltech AS" );

   QPixmap pixmap;
   if ( pixmap.load("flag.bmp") )         // load and draw image
     p->drawPixmap( 100, 85, pixmap );

   p->setBackgroundMode( Qt::OpaqueMode );   // set opaque mode
   p->setBrush( Qt::DiagCrossPattern );    // black diagonal cross pattern
   p->drawEllipse( 90, 210, 80, 80 );         // back wheel
   p->setBrush( Qt::CrossPattern );      // black cross fill pattern
   p->drawEllipse( 310, 210, 80, 80 );       // front wheel
}

class PictureDisplay : public QWidget       // picture display widget
{
public:
   PictureDisplay( const char *fileName );
```

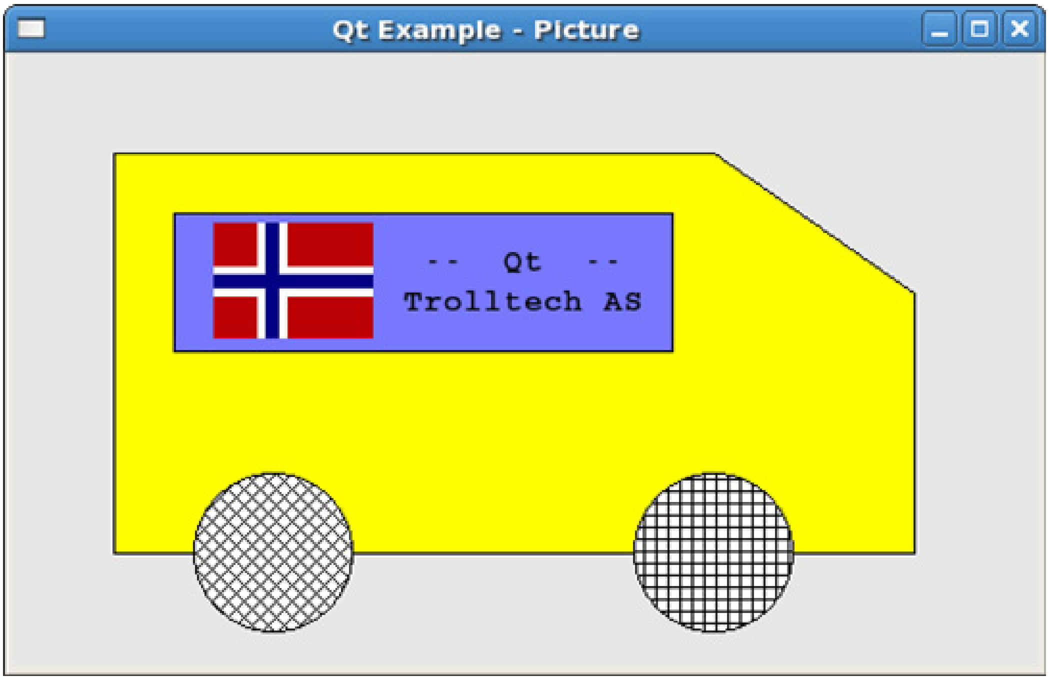
```
  ~PictureDisplay();
protected:
   voidpaintEvent( QPaintEvent * );
   voidkeyPressEvent( QKeyEvent * );
private:
   QPicture   *pict;
   QStringname;
};

PictureDisplay::PictureDisplay( const char *fileName )
{
   pict = new QPicture;
   name = fileName;
   if ( !pict->load(fileName) ) {      // cannot load picture
     delete pict;
     pict = 0;
     name.sprintf( "Not able to load picture: %s", fileName );
   }
}

PictureDisplay::~PictureDisplay()
{
   delete pict;
}

void PictureDisplay::paintEvent( QPaintEvent * )
{
   QPainter paint( this );           // paint widget
   if ( pict )
     paint.drawPicture( *pict );        // draw picture
   else
     paint.drawText( rect(), AlignCenter, name );
}

void PictureDisplay::keyPressEvent( QKeyEvent *k )
{
   switch ( tolower(k->ascii()) ) {
     case 'r':                    // reload
        pict->load( name );
        update();
        break;
     case 'q':                    // quit
        QApplication::exit();
        break;
   }
}


int main( int argc, char **argv )
{
   QApplication a( argc, argv );      // QApplication required!

   const char *fileName = "car.pic";          // default picture file name
```

```
    if ( argc == 2 )              // use argument as file name
      fileName = argv[1];

    if ( !QFile::exists(fileName) ) {
      QPicture pict;              // our picture
      QPainter paint;                 // our painter

      paint.begin( &pict );           // begin painting onto picture
      paintCar( &paint );             // paint!
      paint.end();                // painting done

      pict.save( fileName );              // save picture
      QMessageBox::information(0, "Qt Example - Picture", "Saved.  Run me again!");
      return 0;
    } else {
      PictureDisplay test( fileName );// create picture display
      a.setMainWidget( &test);            // set main widget
      test.setCaption("Qt Example - Picture");
      test.show();                // show it

      return a.exec();            // start event loop
    }
}
```

flag.png



실행



41. 튀여나오기창문부품

이 실례는 튀여나오기창문부품들을 실현하는 방법을 보여준다.

popup.pro

```
TEMPLATE  = app
TARGET      = popup
CONFIG      += qt warn_on release
HEADERS       = popup.h
SOURCES       = popup.cpp
```

popup.cpp

```
#include "popup.h"
#include <qapplication.h>
#include <qlayout.h>

FancyPopup::FancyPopup( QWidget* parent, const char*  name ):
   QLabel( parent, name, WType_Popup ){
      setFrameStyle( WinPanel|Raised );
      setAlignment( AlignCenter );
      resize(150,100);
      moves = 0;
      setMouseTracking( TRUE );
}

void FancyPopup::mouseMoveEvent( QMouseEvent * e){
   moves++;
   QString s;
   s.sprintf("%d/%d", e->pos().x(), e->pos().y());
   if (e->state() & QMouseEvent::LeftButton)
      s += " (down)";
   setText(s);
}

void FancyPopup::mouseReleaseEvent( QMouseEvent * e){
   if  (rect().contains( e->pos() ) || moves > 5)
      close();
}

void FancyPopup::closeEvent( QCloseEvent *e ){
   e->accept();
   moves = 0;
   if (!popupParent)
      return;

   // remember that we (as a popup) might recieve the mouse release
   // event instead of the popupParent. This is due to the fact that
   // the popupParent popped us up in its mousePressEvent handler. To
   // avoid the button remaining in pressed state we simply send a
   // faked mouse button release event to it.
   QMouseEvent me( QEvent::MouseButtonRelease, QPoint(0,0), QPoint(0,0),
QMouseEvent::LeftButton, QMouseEvent::NoButton);
   QApplication::sendEvent( popupParent, &me );
}
```

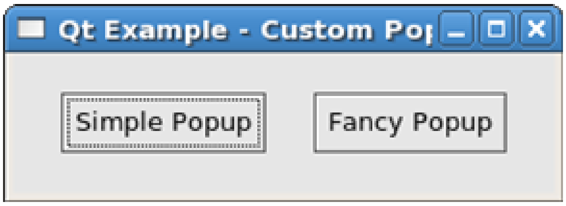
```
void FancyPopup::popup( QWidget* parent) {
   popupParent = parent;
   setText("Move the mouse!");
   if (popupParent)
      move( popupParent->mapToGlobal( popupParent->rect().bottomLeft() ) );
   show();
}

Frame::Frame(QWidget* parent, const char* name): QFrame(parent, name){
   button1 = new QPushButton("Simple Popup", this);
   connect ( button1, SIGNAL( clicked() ), SLOT( button1Clicked() ) );
   button2 = new QPushButton("Fancy Popup", this);
   connect ( button2, SIGNAL( pressed() ), SLOT( button2Pressed() ) );

   QBoxLayout * l = new QHBoxLayout( this );
   button1->setMaximumSize(button1->sizeHint());
   button2->setMaximumSize(button2->sizeHint());
   l->addWidget( button1 );
   l->addWidget( button2 );
   l->activate();

//    button1->setGeometry(20,20,100,30);
//    button2->setGeometry(140,20,100,30);
   resize(270, 70);

   //create a very simple popup: it is just composed with other
   //widget and will be shown after clicking on button1

   popup1 = new QFrame( this ,0, WType_Popup);
   popup1->setFrameStyle( WinPanel|Raised );
   popup1->resize(150,100);
   QLineEdit *tmpE = new QLineEdit( popup1 );
   connect( tmpE, SIGNAL( returnPressed() ), popup1, SLOT( hide() ) );
   tmpE->setGeometry(10,10, 130, 30);
   tmpE->setFocus();
   QPushButton *tmpB = new QPushButton("Click me!", popup1);
   connect( tmpB, SIGNAL( clicked() ), popup1, SLOT( close() ) );
   tmpB->setGeometry(10, 50, 130, 30);

   // the fancier version uses its own class. It will be shown when
   // pressing button2, so they behavior is more like a modern menu
   // or toolbar.

   popup2 = new FancyPopup( this );

   // you might also add new widgets to the popup, just like you do
   // it with any other widget.  The next four lines (if not
   // commented out) will for instance add a line edit widget.

//    tmpE = new QLineEdit( popup2 );
//    tmpE->setFocus();
//    connect( tmpE, SIGNAL( returnPressed() ), popup2, SLOT( close() ) );
//    tmpE->setGeometry(10, 10, 130, 30);
}


void Frame::button1Clicked(){
   popup1->move( mapToGlobal( button1->geometry().bottomLeft() ) );
   popup1->show();
}

void Frame::button2Pressed(){
   popup2->popup(button2);
}

int main( int argc, char **argv )
{
   QApplication a(argc,argv);

   Frame frame;
   frame.setCaption("Qt Example - Custom Popups");
   a.setMainWidget(&frame);
   frame.show();
   return a.exec();
}
```

popup.h

```
#ifndef POPUP_H
#define POPUP_H
#include <qlabel.h>
#include <qpushbutton.h>
#include <qlineedit.h>

class FancyPopup : public QLabel
{
   Q_OBJECT
public:
   FancyPopup( QWidget* parent = 0, const char*  name=0);

   void popup( QWidget* parent = 0);
protected:
   virtual void mouseMoveEvent( QMouseEvent * );
   virtual void mouseReleaseEvent( QMouseEvent * );
   virtual void closeEvent( QCloseEvent * );

private:
   QWidget* popupParent;
   int moves;
};

 class Frame : public QFrame
 {
    Q_OBJECT
 public:
    Frame( QWidget *parent=0, const char*  name=0);

 protected:
```

```
 private slots:
    void button1Clicked();
    void button2Pressed();

 private:
    QPushButton *button1;
    QPushButton *button2;

    QFrame* popup1;
    FancyPopup* popup2;
};

#endif
```

실행



42. 입출력방향을 지정한 프로세스기동

이 실례는 Qt에서 다른 프로세스들을 기동하는 방법과 입출력방향을 지정하는 방법을 보여준다. 이 실례는 일정한 ui파일에 대하여 uic를 기동하고 지령의 출력을 현시한다.

process.pro

```
TEMPLATE  = app
TARGET      = process
CONFIG      += qt warn_on release
HEADERS         =
SOURCES         = process.cpp
INTERFACES    =
```

process.cpp

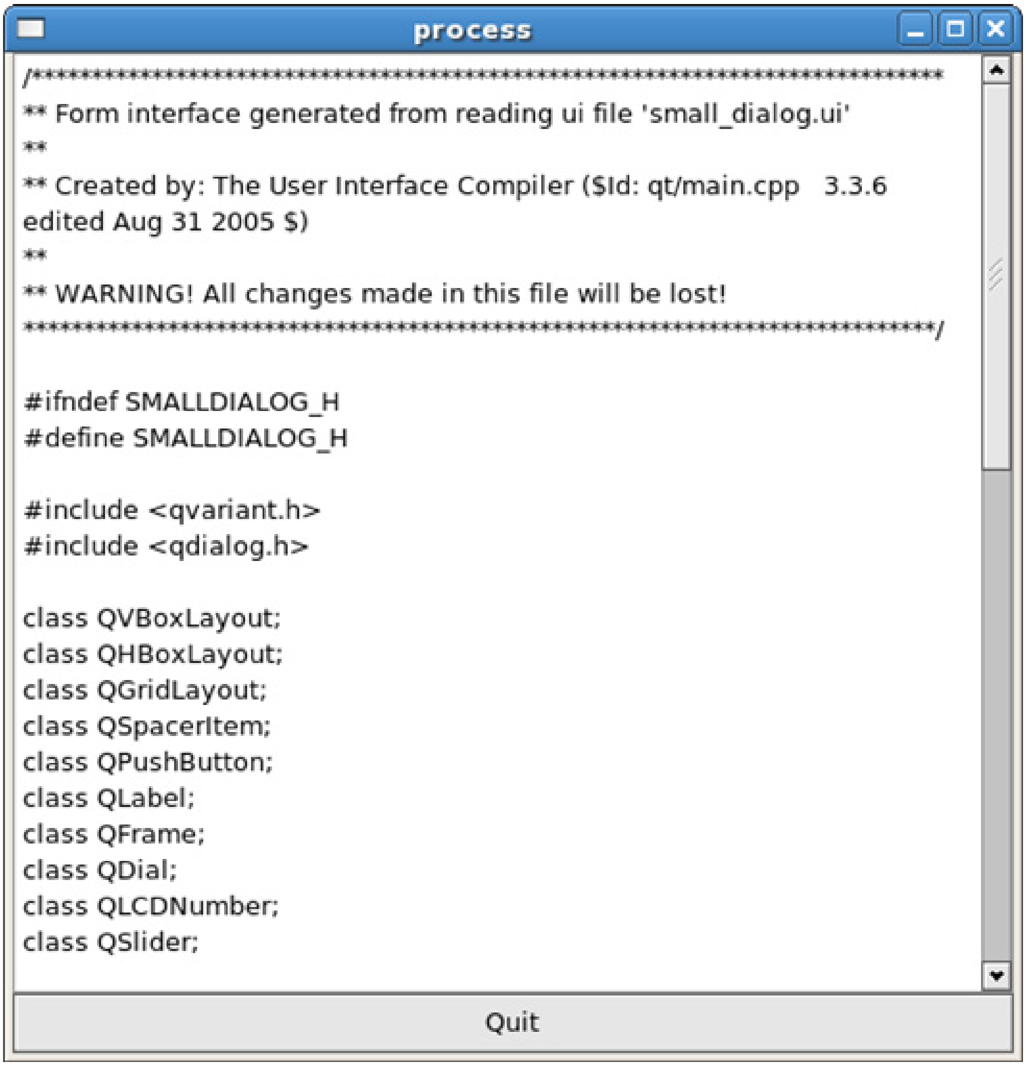
```
#include <qobject.h>
#include <qprocess.h>
#include <qvbox.h>
#include <qtextview.h>
#include <qpushbutton.h>
#include <qapplication.h>
#include <qmessagebox.h>

#include <stdlib.h>

class UicManager : public QVBox
{
   Q_OBJECT

public:
   UicManager();
   ~UicManager() {}
public slots:
   void readFromStdout();
   void scrollToTop();

private:
   QProcess *proc;
   QTextView *output;
   QPushButton *quitButton;
};

UicManager::UicManager()
{
   // Layout
   output = new QTextView( this );
   quitButton = new QPushButton( tr("Quit"), this );
   connect( quitButton, SIGNAL(clicked()),
       qApp, SLOT(quit()) );
   resize( 500, 500 );

   // QProcess related code
   proc = new QProcess( this );

   // Set up the command and arguments.
   // On the command line you would do:
   //   uic -tr i18n "small_dialog.ui"
   proc->addArgument( "uic" );
   proc->addArgument( "-tr" );
   proc->addArgument( "i18n" );
   proc->addArgument( "small_dialog.ui" );

   connect( proc, SIGNAL(readyReadStdout()),
       this, SLOT(readFromStdout()) );
   connect( proc, SIGNAL(processExited()),
       this, SLOT(scrollToTop()) );

   if ( !proc->start() ) {
    // error handling
    QMessageBox::critical( 0,
        tr("Fatal error"),
        tr("Could not start the uic command."),
        tr("Quit") );
    exit( -1 );
   }
}

void UicManager::readFromStdout()
{
   // Read and process the data.
   // Bear in mind that the data might be output in chunks.
   output->append( proc->readStdout() );
}

void UicManager::scrollToTop()
```

```
{
   output->setContentsPos( 0, 0 );
}

int main( int argc, char **argv )
{
   QApplication a( argc, argv );
   UicManager manager;
   a.setMainWidget( &manager );
   manager.show();
   return a.exec();
}

#include "process.moc"
```

small_dialog.ui

```
<!DOCTYPE UI><UI version="3.0" stdsetdef="1">
<class>SmallDialog</class>
<widget class="QDialog">
   <property name="name">
…
      <receiver>Slider1</receiver>
      <slot>setValue(int)</slot>
   </connection>
</connections>
</UI>
```

실행



```
/****************************************************************************
** Form interface generated from reading ui file 'small_dialog.ui'
**
** Created by: The User Interface Compiler ($Id: qt/main.cpp   3.3.6
edited Aug 31 2005 $)
**
** WARNING! All changes made in this file will be lost!
****************************************************************************/

#ifndef SMALLDIALOG_H
#define SMALLDIALOG_H

#include <qvariant.h>
#include <qdialog.h>

class QVBoxLayout;
class QHBoxLayout;
class QGridLayout;
class QSpacerItem;
class QPushButton;
class QLabel;
class QFrame;
class QDial;
class QLCDNumber;
class QSlider;
```

43. 진행띠와 대화칸실례

이 실례는 단순한(본문전용) 혹은 전용표식형(사용자제공창문부품)진행대화칸을 현시한다. 또한 차림표의 간단한 사용법을 보여준다.

progress.pro

```
TEMPLATE  = app
TARGET    = progress
CONFIG    += qt warn_on release
HEADERS      =
SOURCES      = progress.cpp
```

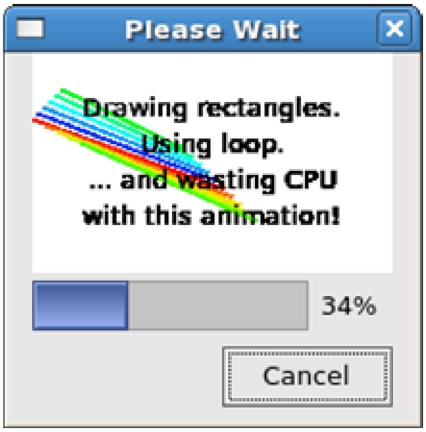
progress.cpp

```
#include <qprogressdialog.h>
#include <qapplication.h>
#include <qmenubar.h>
#include <qpopupmenu.h>
```

```
#include <qpainter.h>
#include <stdlib.h>

class AnimatedThingy : public QLabel {
public:
   AnimatedThingy( QWidget* parent, const QString& s ) :
    QLabel(parent),
    label(s),
    step(0)
   {
      setBackgroundColor(white);
    label+="\n... and wasting CPU\nwith this animation!\n";

    for (int i=0; i<nqix; i++)
       ox[0][i] = oy[0][i] = ox[1][i] = oy[1][i] = 0;
    x0 = y0 = x1 = y1 = 0;
    dx0 = rand()%8+2;
    dy0 = rand()%8+2;
    dx1 = rand()%8+2;
    dy1 = rand()%8+2;
   }

   void show()
   {
    if (!isVisible()) startTimer(100);
    QWidget::show();
   }

   void hide()
   {
    QWidget::hide();
    killTimers();
   }

   QSize sizeHint() const
   {
    return QSize(120,100);
   }

protected:
   void timerEvent(QTimerEvent*)
   {
    QPainter p(this);
    QPen pn=p.pen();
    pn.setWidth(2);
    pn.setColor(backgroundColor());
    p.setPen(pn);

    step = (step + 1) % nqix;

    p.drawLine(ox[0][step], oy[0][step], ox[1][step], oy[1][step]);

    inc(x0, dx0, width());
    inc(y0, dy0, height());
    inc(x1, dx1, width());
    inc(y1, dy1, height());
    ox[0][step] = x0;
    oy[0][step] = y0;
    ox[1][step] = x1;
    oy[1][step] = y1;

    QColor c;
    c.setHsv( (step*255)/nqix, 255, 255 ); // rainbow effect
    pn.setColor(c);
    p.setPen(pn);
    p.drawLine(ox[0][step], oy[0][step], ox[1][step], oy[1][step]);
    p.setPen(colorGroup().text());
    p.drawText(rect(), AlignCenter, label);
   }

   void paintEvent(QPaintEvent* event)
   {
    QPainter p(this);
    QPen pn=p.pen();
    pn.setWidth(2);
    p.setPen(pn);
    p.setClipRect(event->rect());
    for (int i=0; i<nqix; i++) {
       QColor c;
       c.setHsv( (i*255)/nqix, 255, 255 ); // rainbow effect
       pn.setColor(c);
       p.setPen(pn);
       p.drawLine(ox[0][i], oy[0][i], ox[1][i], oy[1][i]);
    }
    p.setPen(colorGroup().text());
    p.drawText(rect(), AlignCenter, label);
   }

private:
   void inc(int& x, int& dx, int b)
   {
    x+=dx;
    if (x<0) { x=0; dx=rand()%8+2; }
    else if (x>=b) { x=b-1; dx=-(rand()%8+2); }
   }

   enum {nqix=10};
   int ox[2][nqix];
   int oy[2][nqix];
   int x0,y0,x1,y1;
   int dx0,dy0,dx1,dy1;
   QString label;
   int step;
};

class CPUWaster : public QWidget
{
   Q_OBJECT
   enum { first_draw_item = 1000, last_draw_item = 1006 };

   int drawItemRects(int id)
   {
    int n = id - first_draw_item;
    int r = 100;
    while (n--) r*=(n%3 ? 5 : 4);
    return r;
   }
   QString drawItemText(int id)
   {
    QString str;
    str.sprintf("%d Rectangles", drawItemRects(id));
    return str;
   }

public:
   CPUWaster() :
    pb(0)
   {
    menubar = new QMenuBar( this, "menu" );
    Q_CHECK_PTR( menubar );

    QPopupMenu* file = new QPopupMenu();
    Q_CHECK_PTR( file );
    menubar->insertItem( "&File", file );
    for (int i=first_draw_item; i<=last_draw_item; i++)
       file->insertItem( drawItemText(i), i );
    connect( menubar, SIGNAL(activated(int)), this, SLOT(doMenuItem(int)) );
    file->insertSeparator();
    file->insertItem( "Quit", qApp,  SLOT(quit()) );

    options = new QPopupMenu();
    Q_CHECK_PTR( options );
    menubar->insertItem( "&Options", options );
    td_id = options->insertItem( "Timer driven", this, SLOT(timerDriven()) );
    ld_id = options->insertItem( "Loop driven", this, SLOT(loopDriven()) );
    options->insertSeparator();
    dl_id = options->insertItem( "Default label", this, SLOT(defaultLabel()) );
    cl_id = options->insertItem( "Custom label", this, SLOT(customLabel()) );
    options->insertSeparator();
    md_id = options->insertItem( "No minimum duration", this, SLOT(toggleMinimumDuration()) );
    options->setCheckable( TRUE );
    loopDriven();
    defaultLabel();

    setFixedSize( 400, 300 );

    setBackgroundColor( black );
   }

public slots:
   void doMenuItem(int id)
   {
    if (id >= first_draw_item && id <= last_draw_item)
       draw(drawItemRects(id));
   }

   void stopDrawing() { got_stop = TRUE; }

   void timerDriven()
   {
    timer_driven = TRUE;
    options->setItemChecked( td_id, TRUE );
    options->setItemChecked( ld_id, FALSE );
   }

   void loopDriven()
   {
    timer_driven = FALSE;
    options->setItemChecked( ld_id, TRUE );
    options->setItemChecked( td_id, FALSE );
   }

   void defaultLabel()
   {
    default_label = TRUE;
    options->setItemChecked( dl_id, TRUE );
    options->setItemChecked( cl_id, FALSE );
   }

   void customLabel()
   {
    default_label = FALSE;
    options->setItemChecked( dl_id, FALSE );
    options->setItemChecked( cl_id, TRUE );
   }

   void toggleMinimumDuration()
   {
    options->setItemChecked( md_id,
      !options->isItemChecked( md_id ) );
   }

private:
   void timerEvent( QTimerEvent* )
   {
    if (!got_stop)
       pb->setProgress( pb->totalSteps() - rects );
    rects--;

    {
       QPainter p(this);

       int ww = width();
       int wh = height();
       if ( ww > 8 && wh > 8 ) {
        QColor c(rand()%255, rand()%255, rand()%255);
        int x = rand() % (ww-8);
        int y = rand() % (wh-8);
        int w = rand() % (ww-x);
        int h = rand() % (wh-y);
        p.fillRect( x, y, w, h, c );
       }
    }

    if (!rects || got_stop) {
       if (!got_stop)
        pb->setProgress( pb->totalSteps() );
       QPainter p(this);
       p.fillRect(0, 0, width(), height(), backgroundColor());
       enableDrawingItems(TRUE);
       killTimers();
       delete pb;
       pb = 0;
    }
   }

   QProgressDialog* newProgressDialog( const char* label, int steps, bool modal )
   {
    QProgressDialog *d = new QProgressDialog(label, "Cancel", steps, this,
                     "progress", modal);
      if ( options->isItemChecked( md_id ) )
       d->setMinimumDuration(0);
    if ( !default_label )
       d->setLabel( new AnimatedThingy(d,label) );
    return d;
   }

   void enableDrawingItems(bool yes)
   {
    for (int i=first_draw_item; i<=last_draw_item; i++) {
       menubar->setItemEnabled(i, yes);
    }
   }

   void draw(int n)
   {
    if ( timer_driven ) {
       if ( pb ) {
        qWarning("This cannot happen!");
        return;
       }
       rects = n;
       pb = newProgressDialog("Drawing rectangles.\n"
                "Using timer event.", n, FALSE);
       pb->setCaption("Please Wait");
       connect(pb, SIGNAL(cancelled()), this, SLOT(stopDrawing()));
       enableDrawingItems(FALSE);
       startTimer(0);
```

```
     got_stop = FALSE;
   } else {
     QProgressDialog* lpb = newProgressDialog(
         "Drawing rectangles.\nUsing loop.", n, TRUE);
     lpb->setCaption("Please Wait");

     QPainter p(this);
     for (int i=0; i<n; i++) {
      lpb->setProgress(i);
      if ( lpb->wasCancelled() )
        break;

      QColor c(rand()%255, rand()%255, rand()%255);
      int x = rand()%(width()-8);
      int y = rand()%(height()-8);
      int w = rand()%(width()-x);
      int h = rand()%(height()-y);
      p.fillRect(x,y,w,h,c);
     }

     p.fillRect(0, 0, width(), height(), backgroundColor());

     delete lpb;
   }
  }

  QMenuBar* menubar;
  QProgressDialog* pb;
  QPopupMenu* options;
  int td_id, ld_id;
  int dl_id, cl_id;
  int md_id;
  int rects;
  bool timer_driven;
  bool default_label;
  bool got_stop;
};

int main( int argc, char **argv )
{
  QApplication a( argc, argv );

  int wincount = argc > 1 ? atoi(argv[1]) : 1;

  for ( int i=0; i<wincount; i++ ) {
   CPUWaster* cpuw = new CPUWaster;
   if ( i == 0 ) a.setMainWidget(cpuw);
   cpuw->show();
  }
  return a.exec();
}

#include "progress.moc"
```

실행



Drawing rectangles.
Using loop.
... and wasting CPU
with this animation!

44. 진행띠

이 실례는 진행띠의 사용법을 보여준다.

progressbar.pro

```
TEMPLATE  = app
TARGET      = progressbar
CONFIG      += qt warn_on release
HEADERS        = progressbar.h
SOURCES        = main.cpp \
            progressbar.cpp
```

progressbar.h

```
#ifndef PROGRESSBAR_H
#define PROGRESSBAR_H

#include <qbuttongroup.h>
#include <qtimer.h>

class QRadioButton;
class QPushButton;
class QProgressBar;

class ProgressBar : public QButtonGroup
{
   Q_OBJECT

public:
   ProgressBar( QWidget *parent = 0, const char *name = 0 );

protected:
   QRadioButton *slow, *normal, *fast;
   QPushButton *start, *pause, *reset;
   QProgressBar *progress;
   QTimer timer;

protected slots:
   void slotStart();
```

```
    void slotReset();
    void slotTimeout();

};

#endif
```

progressbar.cpp

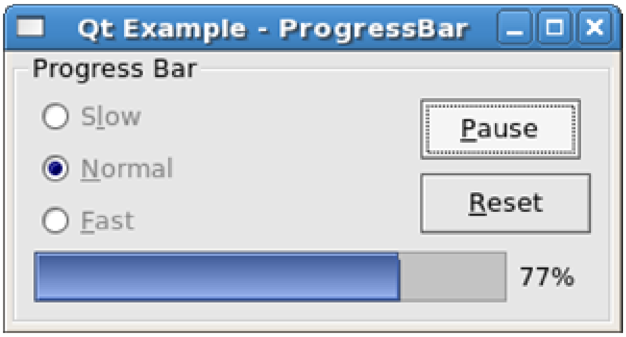
```
#include "progressbar.h"
#include <qradiobutton.h>
#include <qpushbutton.h>
#include <qprogressbar.h>
#include <qlayout.h>
#include <qmotifstyle.h>

/*
 * Constructor
 *
 * Creates child widgets of the ProgressBar widget
 */

ProgressBar::ProgressBar( QWidget *parent, const char *name )
    : QButtonGroup( 0, Horizontal, "Progress Bar", parent, name ), timer()
{
    setMargin( 10 );

    QGridLayout* toplayout = new QGridLayout( layout(), 2, 2, 5);

    setRadioButtonExclusive( TRUE );

    // insert three radiobuttons which the user can use
    // to set the speed of the progress and two pushbuttons
    // to start/pause/continue and reset the progress
    slow = new QRadioButton( "S&low", this );
    normal = new QRadioButton( "&Normal", this );
    fast = new QRadioButton( "&Fast", this );
    QVBoxLayout* vb1 = new QVBoxLayout;
    toplayout->addLayout( vb1, 0, 0 );
    vb1->addWidget( slow );
    vb1->addWidget( normal );
    vb1->addWidget( fast );

    // two push buttons, one for start, for for reset.
    start = new QPushButton( "&Start", this );
    reset = new QPushButton( "&Reset", this );
    QVBoxLayout* vb2 = new QVBoxLayout;
    toplayout->addLayout( vb2, 0, 1 );
    vb2->addWidget( start );
    vb2->addWidget( reset );

    // Create the progressbar
    progress = new QProgressBar( 100, this );
    //    progress->setStyle( new QMotifStyle() );
    toplayout->addMultiCellWidget( progress, 1, 1, 0, 1 );
    // connect the clicked() SIGNALs of the pushbuttons to SLOTs
    connect( start, SIGNAL( clicked() ), this, SLOT( slotStart() ) );
    connect( reset, SIGNAL( clicked() ), this, SLOT( slotReset() ) );

    // connect the timeout() SIGNAL of the progress-timer to a SLOT
    connect( &timer, SIGNAL( timeout() ), this, SLOT( slotTimeout() ) );

    // Let's start with normal speed...
    normal->setChecked( TRUE );


    // some contraints
    start->setFixedWidth( 80 );
    setMinimumWidth( 300 );
}

/*
 * SLOT slotStart
* This SLOT is called if the user clicks start/pause/continue
 * button
 */

void ProgressBar::slotStart()
{
    // If the progress bar is at the beginning...
    if ( progress->progress() == -1 ) {
        // ...set according to the checked speed-radiobutton
        // the number of steps which are needed to complete the process
        if ( slow->isChecked() )
            progress->setTotalSteps( 10000 );
        else if ( normal->isChecked() )
            progress->setTotalSteps( 1000 );
        else
            progress->setTotalSteps( 50 );

        // disable the speed-radiobuttons
        slow->setEnabled( FALSE );
        normal->setEnabled( FALSE );
        fast->setEnabled( FALSE );
    }

    // If the progress is not running...
    if ( !timer.isActive() ) {
        // ...start the timer (and so the progress) with a interval of 1 ms...
        timer.start( 1 );
        // ...and rename the start/pause/continue button to Pause
        start->setText( "&Pause" );
    } else { // if the prgress is running...
        // ...stop the timer (and so the prgress)...
        timer.stop();
        // ...and rename the start/pause/continue button to Continue
        start->setText( "&Continue" );
    }
```

```
}

/*
 * SLOT slotReset
 *
 * This SLOT is called when the user clicks the reset button
 */

void ProgressBar::slotReset()
{
   // stop the timer and progress
   timer.stop();

   // rename the start/pause/continue button to Start...
   start->setText( "&Start" );
   // ...and enable this button
   start->setEnabled( TRUE );

   // enable the speed-radiobuttons
   slow->setEnabled( TRUE );
   normal->setEnabled( TRUE );
   fast->setEnabled( TRUE );

   // reset the progressbar
   progress->reset();
}

/*
 * SLOT slotTimeout
* This SLOT is called each ms when the timer is
 * active (== progress is running)
 */

void ProgressBar::slotTimeout()
{
   int p = progress->progress();

#if 1
   // If the progress is complete...
   if ( p == progress->totalSteps() )  {
      // ...rename the start/pause/continue button to Start...
      start->setText( "&Start" );
      // ...and disable it...
      start->setEnabled( FALSE );
      // ...and return
      return;
   }
#endif

   // If the process is not complete increase it
   progress->setProgress( ++p );
}
```

main.cpp

```
#include "progressbar.h"
#include <qapplication.h>

int main(int argc,char **argv)
{
   QApplication a(argc,argv);

   ProgressBar progressbar;
   progressbar.setCaption("Qt Example - ProgressBar");
   a.setMainWidget(&progressbar);
   progressbar.show();

   return a.exec();
}
```

실행



45. QDir

qdir.pro

```
TEMPLATE  = app
TARGET     = qdir
CONFIG     += qt warn_on release
HEADERS        = qdir.h ../dirview/dirview.h
SOURCES        = qdir.cpp ../dirview/dirview.cpp
```

qdir.h

```
#ifndef QDIREXAMPLE_H
#define QDIREXAMPLE_H

#include <qscrollview.h>
#include <qfiledialog.h>
#include <qwidgetstack.h>
#include <qvbox.h>
#include <qurl.h>
#include <qpixmap.h>
#include <qstringlist.h>

class QMultiLineEdit;
class QTextView;
class DirectoryView;
class QSpinBox;
class QShowEvent;
class QPopupMenu;

class PixmapView : public QScrollView
{
   Q_OBJECT

public:
   PixmapView( QWidget *parent );
   void setPixmap( const QPixmap &pix );
   void drawContents( QPainter *p, int, int, int, int );

private:
   QPixmap pixmap;

};

class Preview : public QWidgetStack
{
   Q_OBJECT

public:
   Preview( QWidget *parent );
   void showPreview( const QUrl &u, int size );

private:
   QMultiLineEdit *normalText;
   QTextView *html;
   PixmapView *pixmap;

};

class PreviewWidget : public QVBox,
              public QFilePreview
{
   Q_OBJECT

public:
   PreviewWidget( QWidget *parent );
   void previewUrl( const QUrl &u );

private:
   QSpinBox *sizeSpinBox;
   Preview *preview;

};

class CustomFileDialog : public QFileDialog
{
   Q_OBJECT

public:
   CustomFileDialog();
   ~CustomFileDialog();
protected:
    void showEvent( QShowEvent *e );

public slots:
    void setDir2( const QString & );

private slots:
    void bookmarkChosen( int i );
    void goHome();

private:
    DirectoryView *dirView;
    QPopupMenu *bookmarkMenu;
    QStringList bookmarkList;
    int addId;

};

#endif
```

qdir.cpp

```
#include "../dirview/dirview.h"
#include "qdir.h"
#include <qapplication.h>
#include <qtextview.h>
#include <qfileinfo.h>
#include <qfile.h>
#include <qtextstream.h>
#include <qhbox.h>
#include <qspinbox.h>
#include <qlabel.h>
#include <qmultilineedit.h>
#include <qheader.h>
#include <qevent.h>
#include <qpainter.h>
#include <qpopupmenu.h>
#include <qpushbutton.h>
#include <qtoolbutton.h>
#include <qfile.h>
#include <qtextstream.h>
#include <qtooltip.h>

#include <stdlib.h>

/* XPM */
static const char *bookmarks[]={
    "22 14 8 1",
    "# c #000080",
    "a c #585858",
    "b c #000000",
    "c c #ffffff",
    "d c #ffffff",
    "e c #ffffff",
    "f c #000000",
    ". c None",
    "...bb................",
    "..bacb....bbb.........",
    "..badcb.bbccbab.......",
    "..bacccbadccbab.......",
    "..baecdbcccdbab.......",
    "..bacccbacccbab.......",
    "..badcdbcecdfab.......",
    "..bacecbacccbab.......",
    "..baccdbcccdbab.......",
    "...badcbacdbbab.......",
    "....bacbcbbccab.......",
    ".....babbaaaaab.......",
    ".....bbabbbbbbb.......",
    "......bb............."
};

/* XPM */
static const char *home[]={
    "16 15 4 1",
    "# c #000000",
    "a c #ffffff",
    "b c #c0c0c0",
    ". c None",
    ".......##.......",
    "..#...####......",
    "..#..#aabb#.....",
    "..#.#aaaabb#....",
    "..##aaaaaabb#...",
    "..#aaaaaaaabb#..",
    ".#aaaaaaaaabbb#.",
    "###aaaaaaaabb###",
    "..#aaaaaaaabb#..",
    "..#aaa###aabb#..",
    "..#aaa#.#aabb#..",
    "..#aaa#.#aabb#..",
    "..#aaa#.#aabb#..",
    "..#aaa#.#aabb#..",
    "..#####.######.."
};

//
PixmapView::PixmapView( QWidget *parent )
    : QScrollView( parent )
{
    viewport()->setBackgroundMode( PaletteBase );
}

void PixmapView::setPixmap( const QPixmap &pix )
{
    pixmap = pix;
    resizeContents( pixmap.size().width(), pixmap.size().height() );
    viewport()->repaint( FALSE );
```

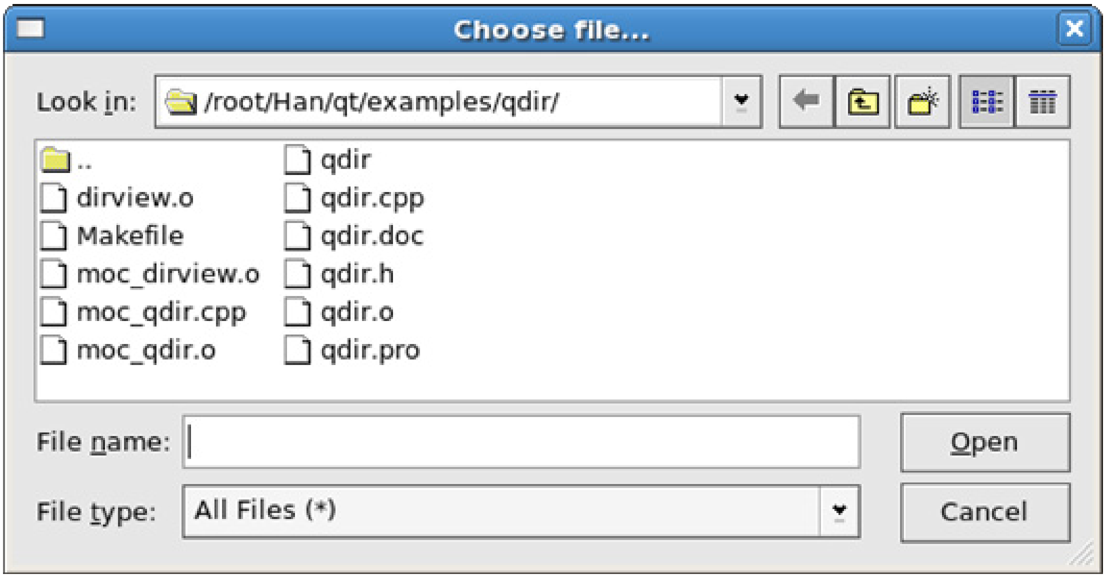
```
}

void PixmapView::drawContents( QPainter *p, int cx, int cy, int cw, int ch )
{
   p->fillRect( cx, cy, cw, ch, colorGroup().brush( QColorGroup::Base ) );
   p->drawPixmap( 0, 0, pixmap );
}

//
Preview::Preview( QWidget *parent )
   : QWidgetStack( parent )
{
   normalText = new QMultiLineEdit( this );
   normalText->setReadOnly( TRUE );
   html = new QTextView( this );
   pixmap = new PixmapView( this );
   raiseWidget( normalText );
}

void Preview::showPreview( const QUrl &u, int size )
{
   if ( u.isLocalFile() ) {
    QString path = u.path();
    QFileInfo fi( path );
    if ( fi.isFile() && (int)fi.size() > size * 1000 ) {
       normalText->setText( tr( "The File\n%1\nis too large, so I don't show it!" ).arg( path ) );
       raiseWidget( normalText );
       return;
    }

    QPixmap pix( path );
    if ( pix.isNull() ) {
       if ( fi.isFile() ) {
        QFile f( path );
        if ( f.open( IO_ReadOnly ) ) {
           QTextStream ts( &f );
           QString text = ts.read();
           f.close();
           if ( fi.extension().lower().contains( "htm" ) ) {
            QString url = html->mimeSourceFactory()->makeAbsolute( path, html->context() );
            html->setText( text, url );
            raiseWidget( html );
            return;
           } else {
            normalText->setText( text );
            raiseWidget( normalText );
            return;
           }
        }
       }
       normalText->setText( QString::null );
       raiseWidget( normalText );
    } else {
       pixmap->setPixmap( pix );
```

```
        raiseWidget( pixmap );
      }
    } else {
      normalText->setText( "I only show local files!" );
      raiseWidget( normalText );
    }
}

//
PreviewWidget::PreviewWidget( QWidget *parent )
    : QVBox( parent ), QFilePreview()
{
    setSpacing( 5 );
    setMargin( 5 );
    QHBox *row = new QHBox( this );
    row->setSpacing( 5 );
    (void)new QLabel( tr( "Only show files smaller than: " ), row );
    sizeSpinBox = new QSpinBox( 1, 10000, 1, row );
    sizeSpinBox->setSuffix( " KB" );
    sizeSpinBox->setValue( 64 );
    row->setFixedHeight( 10 + sizeSpinBox->sizeHint().height() );
    preview = new Preview( this );
}

void PreviewWidget::previewUrl( const QUrl &u )
{
    preview->showPreview( u, sizeSpinBox->value() );
}

//
CustomFileDialog::CustomFileDialog()
    :  QFileDialog( 0, 0, TRUE )
{
    setDir( "/" );

    dirView = new DirectoryView( this, 0, TRUE );
    dirView->addColumn( "" );
    dirView->header()->hide();
    ::Directory *root = new ::Directory( dirView, "/" );
    root->setOpen( TRUE );
    dirView->setFixedWidth( 150 );

    addLeftWidget( dirView );

    QPushButton *p = new QPushButton( this );
    p->setPixmap( QPixmap( bookmarks ) );
    QToolTip::add( p, tr( "Bookmarks" ) );

    bookmarkMenu = new QPopupMenu( this );
    connect( bookmarkMenu, SIGNAL( activated( int ) ),
          this, SLOT( bookmarkChosen( int ) ) );
    addId = bookmarkMenu->insertItem( tr( "Add bookmark" ) );
    bookmarkMenu->insertSeparator();
    QFile f( ".bookmarks" );
    if ( f.open( IO_ReadOnly ) ) {
     QDataStream ds( &f );
     ds >> bookmarkList;
     f.close();

     QStringList::Iterator it = bookmarkList.begin();
     for ( ; it != bookmarkList.end(); ++it ) {
        bookmarkMenu->insertItem( *it );
     }
    }

    p->setPopup( bookmarkMenu );

    addToolButton( p, TRUE );

    connect( dirView, SIGNAL( folderSelected( const QString & ) ),
          this, SLOT( setDir2( const QString & ) ) );
    connect( this, SIGNAL( dirEntered( const QString & ) ),
          dirView, SLOT( setDir( const QString & ) ) );

    QToolButton *b = new QToolButton( this );
    QToolTip::add( b, tr( "Go Home!" ) );
    b->setPixmap( QPixmap( home ) );
    connect( b, SIGNAL( clicked() ),
          this, SLOT( goHome() ) );

    addToolButton( b );

    resize( width() + width() / 3, height() );
}

CustomFileDialog::~CustomFileDialog()
{
    if ( !bookmarkList.isEmpty() ) {
     QFile f( ".bookmarks" );
     if ( f.open( IO_WriteOnly ) ) {
        QDataStream ds( &f );
        ds << bookmarkList;
        f.close();
     }
    }
}

void CustomFileDialog::setDir2( const QString &s )
{
    blockSignals( TRUE );
    setDir( s );
    blockSignals( FALSE );
}

void CustomFileDialog::showEvent( QShowEvent *e )
{
    QFileDialog::showEvent( e );
```

```
  dirView->setDir( dirPath() );
}

void CustomFileDialog::bookmarkChosen( int i )
{
  if ( i == addId ) {
   bookmarkList << dirPath();
   bookmarkMenu->insertItem( dirPath() );
  } else {
   setDir( bookmarkMenu->text( i ) );
  }
}

void CustomFileDialog::goHome()
{
  if ( getenv( "HOME" ) )
   setDir( getenv( "HOME" ) );
  else
   setDir( "/" );
}

//
int main( int argc, char ** argv )
{
  QFileDialog::Mode mode = QFileDialog::ExistingFile;
  QString start;
  QString filter;
  QString caption;
  bool preview = FALSE;
  bool custom = FALSE;
  QApplication a( argc, argv );
  for (int i=1; i<argc; i++) {
   QString arg = argv[i];
   if ( arg == "-any" )
     mode = QFileDialog::AnyFile;
   else if ( arg == "-dir" )
     mode = QFileDialog::Directory;
   else if ( arg == "-default" )
     start = argv[++i];
   else if ( arg == "-filter" )
     filter = argv[++i];
   else if ( arg == "-preview" )
     preview = TRUE;
   else if ( arg == "-custom" )
     custom = TRUE;
   else if ( arg[0] == '-' ) {
     qDebug("Usage: qdir [-any | -dir | -custom] [-preview] [-default f] {-filter f} [caption ...]\n"
        "    -any        Get any filename, need not exist.\n"
        "    -dir        Return a directory rather than a file.\n"
        "    -custom     Opens a customized QFileDialog with \n"
        "                dir browser, bookmark menu, etc.\n"
        "    -preview    Show a preview widget.\n"
        "    -default f  Start from directory/file f.\n"
        "    -filter f   eg. '*.gif' '*.bmp'\n"
```

```
            "     caption ...  Caption for dialog.\n"
            );
        return 1;
      } else {
        if ( !caption.isNull() )
          caption += ' ';
        caption += arg;
      }
    }

    if ( !start )
     start = QDir::currentDirPath();

    if ( !caption )
     caption = mode == QFileDialog::Directory
            ? "Choose directory..." : "Choose file...";

    if ( !custom ) {
     QFileDialog fd( QString::null, filter, 0, 0, TRUE );
     fd.setMode( mode );
     if ( preview ) {
        fd.setContentsPreviewEnabled( TRUE );
        PreviewWidget *pw = new PreviewWidget( &fd );
        fd.setContentsPreview( pw, pw );
        fd.setViewMode( QFileDialog::List );
        fd.setPreviewMode( QFileDialog::Contents );
     }
     fd.setCaption( caption );
     fd.setSelection( start );
     if ( fd.exec() == QDialog::Accepted ) {
        QString result = fd.selectedFile();
        printf("%s\n", (const char*)result);
        return 0;
     } else {
        return 1;
     }
    } else {
     CustomFileDialog fd;
     fd.exec();
     return 1;
    }
}
```

실행



Choose file...

Look in: /root/Han/qt/examples/qdir/

..
dirview.o
Makefile
moc_dirview.o
moc_qdir.cpp
moc_qdir.o
qdir
qdir.cpp
qdir.doc
qdir.h
qdir.o
qdir.pro

File name:

File type: All Files (*)

46. 서체현시기

이 실례프로그람은 모든 서체의 문자들을 현시한다.

qfd.pro

```
TEMPLATE  = app
TARGET      = qfd
CONFIG      += qt warn_on release
HEADERS         = fontdisplayer.h
SOURCES         = fontdisplayer.cpp \
          qfd.cpp
```

fontdisplayer.cpp

```
#include "fontdisplayer.h"
#include <qapplication.h>
#include <qslider.h>
#include <qspinbox.h>
#include <qpainter.h>
#include <qtoolbar.h>
#include <qstatusbar.h>
#include <qlabel.h>
#include <qpushbutton.h>
#include <qfontdialog.h>
#include <stdlib.h>

FontRowTable::FontRowTable( QWidget* parent, const char* name ) :
   QFrame(parent,name)
{
   setBackgroundMode(PaletteBase);
   setFrameStyle(Panel|Sunken);
   setMargin(8);
   setRow(0);
   tablefont = QApplication::font();
}

QSize FontRowTable::sizeHint() const
{
   return 24*cellSize()+QSize(2,2)*(margin()+frameWidth());
}

QSize FontRowTable::cellSize() const
{
   QFontMetrics fm = fontMetrics();
   return QSize( fm.maxWidth(), fm.lineSpacing()+1 );
}

void FontRowTable::paintEvent( QPaintEvent* e )
{
   QFrame::paintEvent(e);
   QPainter p(this);
   p.setClipRegion(e->region());
   QRect r = e->rect();
   QFontMetrics fm = fontMetrics();
```

```
int ml = frameWidth()+margin() + 1 + QMAX(0,-fm.minLeftBearing());
int mt = frameWidth()+margin();
QSize cell((width()-15-ml)/16,(height()-15-mt)/16);

if ( !cell.width() || !cell.height() )
 return;

int mini = r.left() / cell.width();
int maxi = (r.right()+cell.width()-1) / cell.width();
int minj = r.top() / cell.height();
int maxj = (r.bottom()+cell.height()-1) / cell.height();

int h = fm.height();

QColor body(255,255,192);
QColor negative(255,192,192);
QColor positive(192,192,255);
QColor rnegative(255,128,128);
QColor rpositive(128,128,255);

for (int j = minj; j<=maxj; j++) {
 for (int i = mini; i<=maxi; i++) {
    if ( i < 16 && j < 16 ) {
     int x = i*cell.width();
     int y = j*cell.height();

     QChar ch = QChar(j*16+i,row);

     if ( fm.inFont(ch) ) {
        int w = fm.width(ch);
        int l = fm.leftBearing(ch);
        int r = fm.rightBearing(ch);

        x += ml;
        y += mt+h;

        p.fillRect(x,y,w,-h,body);
        if ( w ) {
         if ( l ) {
            p.fillRect(x+(l>0?0:l), y-h/2, abs(l),-h/2,
                  l < 0 ? negative : positive);
         }
         if ( r ) {
            p.fillRect(x+w-(r>0?r:0),y+2, abs(r),-h/2,
                  r < 0 ? rnegative : rpositive);
         }
        }
        QString s;
        s += ch;
        p.setPen(QPen(Qt::black));
        p.drawText(x,y,s);
     }
    }
 }
```

```
   }
}

void FontRowTable::setRow(int r)
{
   row = r;

   QFontMetrics fm = fontMetrics();
   QFontInfo fi = fontInfo();
   QString str = QString("%1 %2pt%3%4 mLB=%5 mRB=%6 mW=%7")
            .arg(fi.family())
            .arg(fi.pointSize())
            .arg(fi.bold() ? " bold" : "")
            .arg(fi.italic() ? " italic" : "")
            .arg(fm.minLeftBearing())
            .arg(fm.minRightBearing())
            .arg(fm.maxWidth());

   emit fontInformation(str);
   update();
}

void FontRowTable::chooseFont()
{
   bool ok;
   QFont oldfont = tablefont;
   tablefont = QFontDialog::getFont(&ok, oldfont, this);

   if (ok)
    setFont(tablefont);
   else
    tablefont = oldfont;
}

FontDisplayer::FontDisplayer( QWidget* parent, const char* name ) :
   QMainWindow(parent,name)
{
   FontRowTable* table = new FontRowTable(this);
   QToolBar* controls = new QToolBar(this);
   (void) new QLabel(tr("Row:"), controls);
   QSpinBox *row = new QSpinBox(0,255,1,controls);
   controls->addSeparator();
   QPushButton *fontbutton = new QPushButton(tr("Font..."), controls);

   connect(row,SIGNAL(valueChanged(int)),table,SLOT(setRow(int)));
   connect(fontbutton, SIGNAL(clicked()), table, SLOT(chooseFont()));
   connect(table,SIGNAL(fontInformation(const QString&)),
       statusBar(),SLOT(message(const QString&)));
   table->setRow(0);
   setCentralWidget(table);
}
```

fontdisplayer.h

```
#ifndef FontDisplayer_H
```

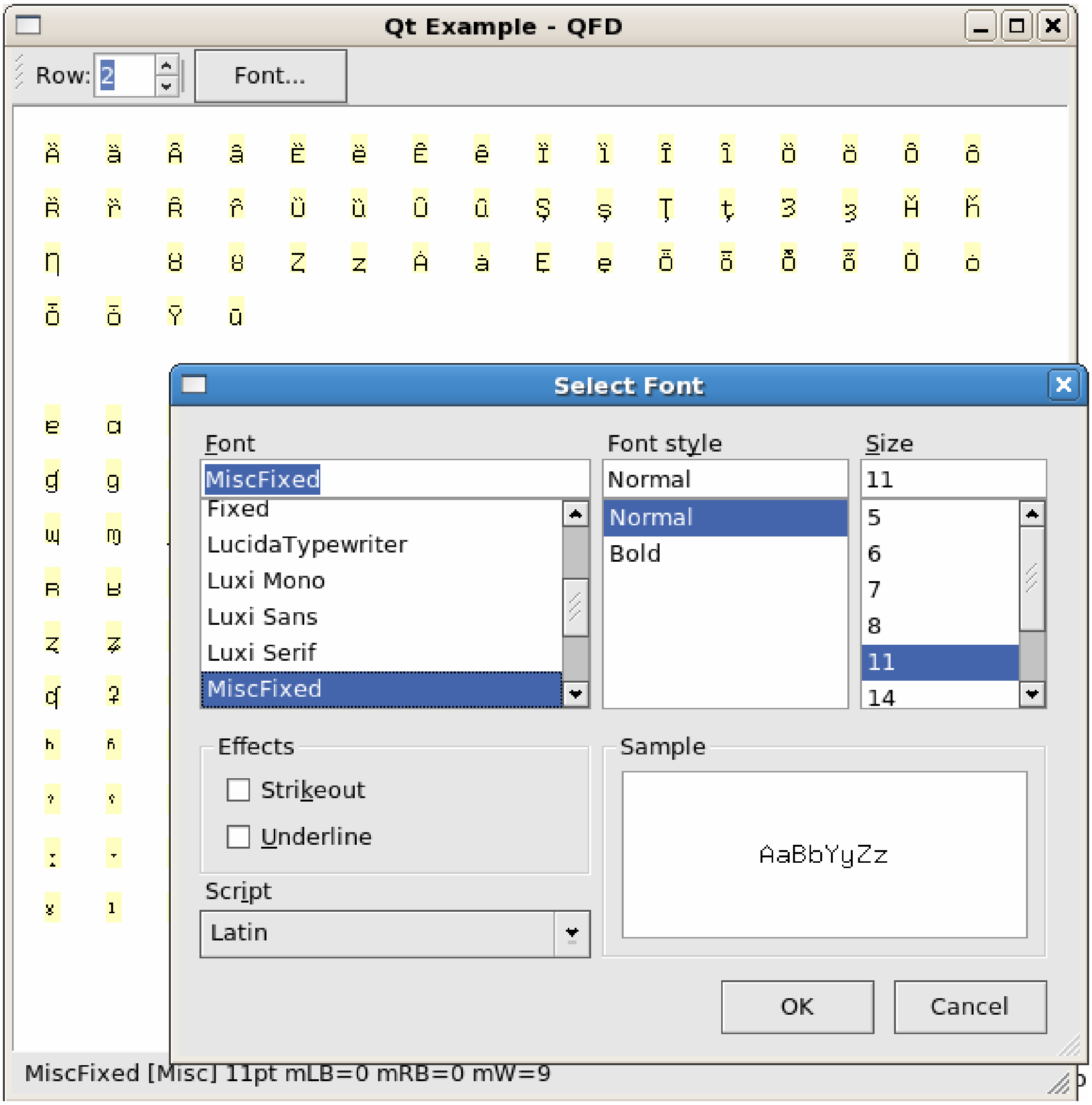
```
#define FontDisplayer_H

#include <qframe.h>
#include <qmainwindow.h>

class QSlider;

class FontRowTable : public QFrame {
   Q_OBJECT
public:
   FontRowTable( QWidget* parent=0, const char* name=0 );

   QSize sizeHint() const;

signals:
   void fontInformation(const QString&);

public slots:
   void setRow(int);
   void chooseFont();


protected:
   QSize cellSize() const;
   void paintEvent( QPaintEvent* );
private:
   QFont tablefont;
   int row;
};

class FontDisplayer : public QMainWindow {
   Q_OBJECT
public:
   FontDisplayer( QWidget* parent=0, const char* name=0 );
};

#endif
```

qfd.cpp

```
#include "fontdisplayer.h"
#include <qapplication.h>
#include <qslider.h>
#include <qpainter.h>
#include <qstatusbar.h>

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

   FontDisplayer m;
   QSize sh = m.centralWidget()->sizeHint();
   m.resize(sh.width(),
         sh.height()+3*m.statusBar()->height());
   app.setMainWidget(&m);
```

```
    m.setCaption("Qt Example - QFD");
    m.show();

    return app.exec();
}
```

실행



MiscFixed [Misc] 11pt mLB=0 mRB=0 mW=9

47. QMag

이것은 간단한 확대경형프로그람이다. 이것은 Qt에 의하여 이식가능한 방법으로 아주 저준위조작들을 수행하는 방법을 보여준다.

그것을 실행한 다음 확대경창문에서 찰칵하고 확대하거나 직4각형밖으로 끌고가려는곳을 찰칵한다. 두개의 복합칸에서 확대률과 재생빈도수를 선택할수 있으며 본문표식자는 유표가 설정되여있는 화소의 색을 알려주며 단추는 확대구역을 .bmp파일에 보관하게 한다.

qmag.pro

```
TEMPLATE  = app
TARGET    = qmag
CONFIG    += qt warn_on release
HEADERS       =
SOURCES       = qmag.cpp
```

qmag.cpp

```
#include <qcombobox.h>
#include <qpushbutton.h>
#include <qpixmap.h>
#include <qimage.h>
#include <qlabel.h>
#include <qfiledialog.h>
#include <qregexp.h>

#include <qapplication.h>
#include <qpainter.h>
#include <qwmatrix.h>

class MagWidget : public QWidget
{
   Q_OBJECT
public:
   MagWidget( QWidget *parent=0, const char *name=0 );

public slots:
   voidsetZoom( int );
   voidsetRefresh( int );
   voidsave();
   voidmultiSave();

protected:
   voidpaintEvent( QPaintEvent * );
   voidmousePressEvent( QMouseEvent * );
   voidmouseReleaseEvent( QMouseEvent * );
   voidmouseMoveEvent( QMouseEvent * );
   voidfocusOutEvent( QFocusEvent * );
   voidtimerEvent( QTimerEvent * );
   voidresizeEvent( QResizeEvent * );

private:
   voidgrabAround(QPoint pos);
   voidgrab();

   QComboBox   *zoom;
   QComboBox   *refresh;
   QPushButton *saveButton;
   QPushButton *multiSaveButton;
   QPushButton *quitButton;
   QPixmap   pm;         // pixmap, magnified
   QPixmap   p;       // pixmap
   QImageimage;     // image of pixmap (for RGB)
   QLabel     *rgb;
```

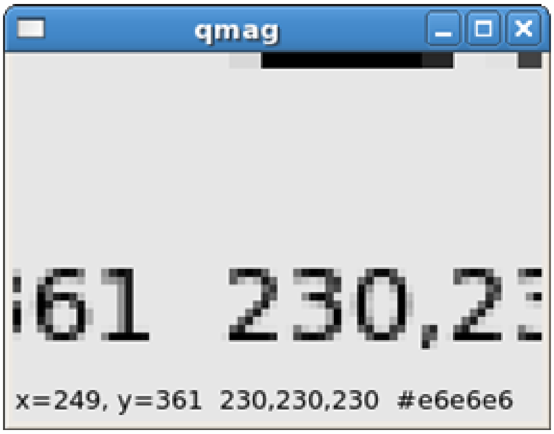
```
    int         yoffset;    // pixels in addition to the actual picture
    int         z;          // magnification factor
    int         r;          // autorefresh rate (index into refreshrates)
    bool grabbing;  // TRUE if qmag is currently grabbing
    int         grabx, graby;
    QString multifn;    // filename for multisave
};

#ifdef COMPLEX_GUI
static const char *zoomfactors[] = {
    "100%", "200%", "300%", "400%", "500%",
    "600%", "700%", "800%", "1600%", 0 };

static const char *refreshrates[] = {
    "No autorefresh", "50 per second", "4 per second", "3 per second", "2 per second",
    "Every second", "Every two seconds", "Every three seconds",
    "Every five seconds", "Every ten seconds", 0 };
#endif

static const int timer[] = {
    0, 20, 250, 333, 500, 1000, 2000, 3000, 5000, 10000 };

MagWidget::MagWidget( QWidget *parent, const char *name )
    : QWidget( parent, name)
{
    z = 1;              // default zoom (100%)
    r = 0;              // default refresh (none)

#ifdef COMPLEX_GUI
    int w=0, x=0, n;

    zoom = new QComboBox( FALSE, this );
    Q_CHECK_PTR(zoom);
    zoom->insertStrList( zoomfactors, 9 );
    connect( zoom, SIGNAL(activated(int)), SLOT(setZoom(int)) );

    refresh = new QComboBox( FALSE, this );
    Q_CHECK_PTR(refresh);
    refresh->insertStrList( refreshrates, 9 );
    connect( refresh, SIGNAL(activated(int)), SLOT(setRefresh(int)) );

    for( n=0; n<9; n++) {
     int w2 = zoom->fontMetrics().width( zoomfactors[n] );
     w = QMAX(w2, w);
    }
    zoom->setGeometry( 2, 2, w+30, 20 );

    x = w+34;
    w = 0;
    for( n=0; n<9; n++) {
     int w2 = refresh->fontMetrics().width( refreshrates[n] );
     w = QMAX(w2, w);
    }
    refresh->setGeometry( x, 2, w+30, 20 );
    saveButton = new QPushButton( this );
    Q_CHECK_PTR(saveButton);
    connect( saveButton, SIGNAL(clicked()), this, SLOT(save()) );
    saveButton->setText( "Save" );
    saveButton->setGeometry( x+w+30+2, 2,
                  10+saveButton->fontMetrics().width("Save"), 20 );

    multiSaveButton = new QPushButton( this );
    multiSaveButton->setToggleButton(TRUE);
    Q_CHECK_PTR(multiSaveButton);
    connect( multiSaveButton, SIGNAL(clicked()), this, SLOT(multiSave()) );
    multiSaveButton->setText( "MultiSave" );
    multiSaveButton->setGeometry( saveButton->geometry().right() + 2, 2,
                  10+multiSaveButton->fontMetrics().width("MultiSave"), 20 );

    quitButton = new QPushButton( this );
    Q_CHECK_PTR(quitButton);
    connect( quitButton, SIGNAL(clicked()), qApp, SLOT(quit()) );
    quitButton->setText( "Quit" );
    quitButton->setGeometry( multiSaveButton->geometry().right() + 2, 2,
                  10+quitButton->fontMetrics().width("Quit"), 20 );
#else
    zoom = 0;
    multiSaveButton = 0;
#endif

    setRefresh(1);
    setZoom(5);

    rgb = new QLabel( this );
    Q_CHECK_PTR( rgb );
    rgb->setText( "" );
    rgb->setAlignment( AlignVCenter );
    rgb->resize( width(), rgb->fontMetrics().height() + 4 );

#ifdef COMPLEX_GUI
    yoffset = zoom->height() // top buttons
      + 4            // space around top buttons
      + rgb->height();  // color-value text height
    setMinimumSize( quitButton->pos().x(), yoffset+20 );
    resize( quitButton->geometry().topRight().x() + 2, yoffset+60 );
#else
    yoffset = 0;
    resize(350,350);
#endif

    grabx = graby = -1;
    grabbing = FALSE;

    setMouseTracking( TRUE ); // and do let me know what pixel I'm at, eh?

    grabAround( QPoint(grabx=qApp->desktop()->width()/2, graby=qApp->desktop()->height()/2) );
}
```

```
void MagWidget::setZoom( int index )
{
   if (index == 8)
    z = 16;
   else
    z = index+1;
   grab();
}

void MagWidget::setRefresh( int index )
{
   r = index;
   killTimers();
   if (index && !grabbing)
    startTimer( timer[r] );
}

void MagWidget::save()
{
   if ( !p.isNull() ) {
    killTimers();
    QString fn = QFileDialog::getSaveFileName();
    if ( !fn.isEmpty() )
        p.save( fn, "BMP" );
    if ( r )
        startTimer( timer[r] );
   }
}

void MagWidget::multiSave()
{
   if ( !p.isNull() ) {
    multifn = ""; // stops saving
    multifn = QFileDialog::getSaveFileName();
    if ( multifn.isEmpty() )
        multiSaveButton->setOn(FALSE);
    if ( !r )
        p.save( multifn, "BMP" );
   } else {
    multiSaveButton->setOn(FALSE);
   }
}

void MagWidget::grab()
{
   if ( !isVisible() )
    return;            // don't eat resources when iconified

   if ( grabx < 0 || graby < 0 )
    return;            // don't grab until the user has said to

   int x,y, w,h;
```

```
    w = (width()+z-1)/z;
    h = (height()+z-1-yoffset)/z;
    if ( w<1 || h<1 )
      return;             // don't ask too much from the window system :)

    x = grabx-w/2;       // find a suitable position to grab from
    y = graby-h/2;
    if ( x + w > QApplication::desktop()->width() )
      x = QApplication::desktop()->width()-w;
    else if ( x < 0 )
      x = 0;
    if ( y + h > QApplication::desktop()->height() )
      y = QApplication::desktop()->height()-h;
    else if ( y < 0 )
      y = 0;

    p = QPixmap::grabWindow( QApplication::desktop()->winId(),  x, y, w, h );
    image = p.convertToImage();
    QWMatrix m;              // after getting it, scale it
    m.scale( (double)z, (double)z );
    pm = p.xForm( m );

    if ( !multiSaveButton || !multiSaveButton->isOn() )
      repaint( FALSE );           // and finally repaint, flicker-free
}

void MagWidget::paintEvent( QPaintEvent * )
{
    if ( !pm.isNull() ) {
      QPainter paint( this );
      paint.drawPixmap( 0, zoom ? zoom->height()+4 : 0, pm,
                  0,0, width(), height()-yoffset );
    }
}

void MagWidget::mousePressEvent( QMouseEvent *e )
{
    if ( !grabbing ) {         // prepare to grab...
      grabbing = TRUE;
      killTimers();
      grabMouse( crossCursor );
      grabx = -1;
      graby = -1;
    } else {            // REALLY prepare to grab
      grabx = mapToGlobal(e->pos()).x();
      graby = mapToGlobal(e->pos()).y();
    }
}

void MagWidget::mouseReleaseEvent( QMouseEvent * e )
{
    if ( grabbing && grabx >= 0 && graby >= 0 ) {
      grabbing = FALSE;
```

```
     grabAround(e->pos());
     releaseMouse();
   }
}

void MagWidget::grabAround(QPoint pos)
{
   int rx, ry;
   rx = mapToGlobal(pos).x();
   ry = mapToGlobal(pos).y();
   int w = QABS(rx-grabx);
   int h = QABS(ry-graby);
   if ( w > 10 && h > 10 ) {
     int pz;
     pz = 1;
     while ( w*pz*h*pz < width()*(height()-yoffset) &&
           w*pz < QApplication::desktop()->width() &&
           h*pz < QApplication::desktop()->height() )
         pz++;
     if ( (w*pz*h*pz - width()*(height()-yoffset)) >
          (width()*(height()-yoffset) - w*(pz-1)*h*(pz-1)) )
         pz--;
     if ( pz < 1 )
         pz = 1;
     if ( pz > 8 )
         pz = 8;
     if ( zoom )
         zoom->setCurrentItem( pz-1 );

     z = pz;
     grabx = QMIN(rx, grabx) + w/2;
     graby = QMIN(ry, graby) + h/2;
     resize( w*z, h*z+yoffset );
   }
   grab();
   if ( r )
     startTimer( timer[r] );
}

void MagWidget::mouseMoveEvent( QMouseEvent *e )
{
   if ( grabbing || pm.isNull() ||
       e->pos().y() > height() - (zoom ? zoom->fontMetrics().height() - 4 : 0) ||
       e->pos().y() < (zoom ? zoom->height()+4 : 4) ) {
     rgb->setText( "" );
   } else {
     int x,y;
     x = e->pos().x() / z;
     y = (e->pos().y() - ( zoom ? zoom->height() : 0 ) - 4) / z;
     QString pixelinfo;
     if ( image.valid(x,y) )
     {
         QRgb px = image.pixel(x,y);
         pixelinfo.sprintf(" %3d,%3d,%3d  #%02x%02x%02x",
      qRed(px), qGreen(px), qBlue(px),
      qRed(px), qGreen(px), qBlue(px));
   }
   QString label;
   label.sprintf( "x=%d, y=%d %s",
     x+grabx, y+graby, (const char*)pixelinfo );
   rgb->setText( label );
  }
}

void MagWidget::focusOutEvent( QFocusEvent * )
{
  rgb->setText( "" );
}

void MagWidget::timerEvent( QTimerEvent * )
{
  grab();
/*
  if ( multiSaveButton->isOn() && !multifn.isEmpty() ) {
   QRegExp num("[0-9][0-9]*");
   int start;
   int len;
   if ((start=num.match(multifn,0,&len))>=0)
     multifn.replace(num,
      QString().setNum(multifn.mid(start,len).toInt()+1)
     );
   p.save( multifn, "BMP" );
  }
*/
}

void MagWidget::resizeEvent( QResizeEvent * )
{
  rgb->setGeometry( 0, height() - rgb->height(), width(), rgb->height() );
  grab();
}

#include "qmag.moc"

int main( int argc, char **argv )
{
  QApplication a( argc, argv );
  MagWidget m;
  a.setMainWidget( &m );
  m.show();
  return a.exec();
}
```

실행



x=249, y=361 230,230,230 #e6e6e6

48. 아주 작은 QTL 실례

이 작은 실례는 QValueListIterator 을 보여준다.

qtl.pro

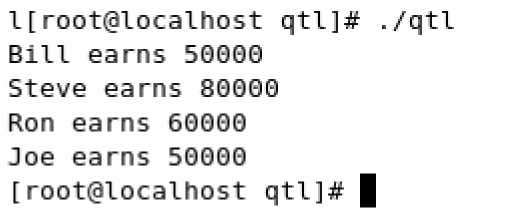
```
TEMPLATE  = app
TARGET        = qtl
CONFIG      += qt console warn_on release
SOURCES        = qvaluelistiterator.cpp
INTERFACES    =
```

qvaluelistiterator.cpp

```
#include <qvaluelist.h>
#include <qstring.h>
#include <qwindowdefs.h>
#include <stdio.h>

class Employee
{
public:
    Employee(): s(0) {}
    Employee( const QString& name, int salary )
      : n(name), s(salary) {}

    QString name() const { return n; }

    int salary() const { return s; }
    void setSalary( int salary ) { s = salary; }

    // this is here to support very old compilers
    Q_DUMMY_COMPARISON_OPERATOR( Employee )

private:
    QString n;
    int s;
};

int main( int, char** )
```

```
{
   typedef QValueList<Employee> EmployeeList;
   EmployeeList list;

   list.append( Employee("Bill", 50000) );
   list.append( Employee("Steve",80000) );
   list.append( Employee("Ron",  60000) );

   Employee joe( "Joe", 50000 );
   list.append( joe );
   joe.setSalary( 4000 );

   EmployeeList::ConstIterator it = list.begin();
   while( it != list.end() ) {
    printf( "%s earns %d\n", (*it).name().latin1(), (*it).salary() );
    ++it;
   }

   return 0;
}
```

실행



```
l[root@localhost qtl]# ./qtl
Bill earns 50000
Steve earns 80000
Ron earns 60000
Joe earns 50000
[root@localhost qtl]# 
```

49. 부호화를 적재할수 있는 간단한 편집기

qwerty.pro

```
TEMPLATE  = app
TARGET      = qwerty
CONFIG      += qt warn_on release
HEADERS        = qwerty.h
SOURCES        = main.cpp \
          qwerty.cpp
```

qwerty.h

```
#ifndef QWERTY_H
#define QWERTY_H

#include <qwidget.h>
#include <qmenubar.h>
#include <qmultilineedit.h>
#include <qprinter.h>

class Editor : public QWidget
{
   Q_OBJECT
public:
   Editor( QWidget *parent=0, const char *name="qwerty" );
  ~Editor();

   void load( const QString& fileName, int code=-1 );

public slots:
   void newDoc();
   void load();
   bool save();
   void print();
   void addEncoding();
   void toUpper();
   void toLower();
   void font();
protected:
   void resizeEvent( QResizeEvent * );
   void closeEvent( QCloseEvent * );

private slots:
   void saveAsEncoding( int );
   void openAsEncoding( int );
   void textChanged();

private:
   bool saveAs( const QString& fileName, int code=-1 );
   void rebuildCodecList();
   QMenuBar       *m;
   QMultiLineEdit *e;
#ifndef QT_NO_PRINTER
   QPrinter        printer;
#endif
   QPopupMenu     *save_as;
   QPopupMenu     *open_as;
   bool changed;
};

#endif // QWERTY_H
```

qwerty.cpp

```
#include "qwerty.h"
#include <qapplication.h>
#include <qfile.h>
#include <qfiledialog.h>
#include <qpopupmenu.h>
#include <qtextstream.h>
#include <qpainter.h>
#include <qmessagebox.h>
#include <qpaintdevicemetrics.h>
#include <qptrlist.h>
#include <qfontdialog.h>

#include <qtextcodec.h>

const bool no_writing = FALSE;
```

```
static QPtrList<QTextCodec> *codecList = 0;

enum { Uni = 0, MBug = 1, Lat1 = 2, Local = 3, Guess = 4, Codec = 5 };


Editor::Editor( QWidget * parent , const char * name )
    : QWidget( parent, name, WDestructiveClose )
{
    m = new QMenuBar( this, "menu" );

    QPopupMenu * file = new QPopupMenu();
    Q_CHECK_PTR( file );
    m->insertItem( "&File", file );

    file->insertItem( "&New",   this, SLOT(newDoc()),   ALT+Key_N );
    file->insertItem( "&Open...",  this, SLOT(load()),     ALT+Key_O );
    file->insertItem( "&Save...",  this, SLOT(save()),     ALT+Key_S );
    file->insertSeparator();
    open_as = new QPopupMenu();
    file->insertItem( "Open &As",  open_as );
    save_as = new QPopupMenu();
    file->insertItem( "Sa&ve As",  save_as );
    file->insertItem( "Add &Encoding", this, SLOT(addEncoding()) );
#ifndef QT_NO_PRINTER
    file->insertSeparator();
    file->insertItem( "&Print...", this, SLOT(print()),    ALT+Key_P );
#endif
    file->insertSeparator();
    file->insertItem( "&Close", this, SLOT(close()),ALT+Key_W );
    file->insertItem( "&Quit",  qApp, SLOT(closeAllWindows()),     ALT+Key_Q );

    connect( save_as, SIGNAL(activated(int)), this, SLOT(saveAsEncoding(int)) );
    connect( open_as, SIGNAL(activated(int)), this, SLOT(openAsEncoding(int)) );
    rebuildCodecList();

    QPopupMenu * edit = new QPopupMenu();
    Q_CHECK_PTR( edit );
    m->insertItem( "&Edit", edit );

    edit->insertItem( "To &Uppercase",   this, SLOT(toUpper()),   ALT+Key_U );
    edit->insertItem( "To &Lowercase",   this, SLOT(toLower()),   ALT+Key_L );
#ifndef QT_NO_FONTDIALOG
    edit->insertSeparator();
    edit->insertItem( "&Select Font" ,     this, SLOT(font()),     ALT+Key_T );
#endif
    changed = FALSE;
    e = new QMultiLineEdit( this, "editor" );
    connect( e, SIGNAL( textChanged() ), this, SLOT( textChanged() ) );

    // We use Unifont - if you have it installed you'll see all
    // Unicode character glyphs.
    //
    // Unifont only comes in one pixel size, so we cannot let
    // it change pixel size as the display DPI changes.
    //
    QFont unifont("unifont",16,50); unifont.setPixelSize(16);
    e->setFont( unifont );

    e->setFocus();
}

Editor::~Editor()
{
}

void Editor::font()
{
#ifndef QT_NO_FONTDIALOG
    bool ok;
    QFont f = QFontDialog::getFont( &ok, e->font() );
    if ( ok ) {
        e->setFont( f );
    }
#endif
}

void Editor::rebuildCodecList()
{
    delete codecList;
    codecList = new QPtrList<QTextCodec>;
    QTextCodec *codec;
    int i;
    for (i = 0; (codec = QTextCodec::codecForIndex(i)); i++)
      codecList->append( codec );
    int n = codecList->count();
    for (int pm=0; pm<2; pm++) {
      QPopupMenu* menu = pm ? open_as : save_as;
      menu->clear();
      QString local = "Local (";
      local += QTextCodec::codecForLocale()->name();
      local += ")";
      menu->insertItem( local, Local );
      menu->insertItem( "Unicode", Uni );
      menu->insertItem( "Latin1", Lat1 );
      menu->insertItem( "Microsoft Unicode", MBug );
      if ( pm )
          menu->insertItem( "[guess]", Guess );
      for ( i = 0; i < n; i++ )
          menu->insertItem( codecList->at(i)->name(), Codec + i );
    }
}

void Editor::newDoc()
{
    Editor *ed = new Editor;
    if ( qApp->desktop()->size().width() < 450
      || qApp->desktop()->size().height() < 450 ) {
```

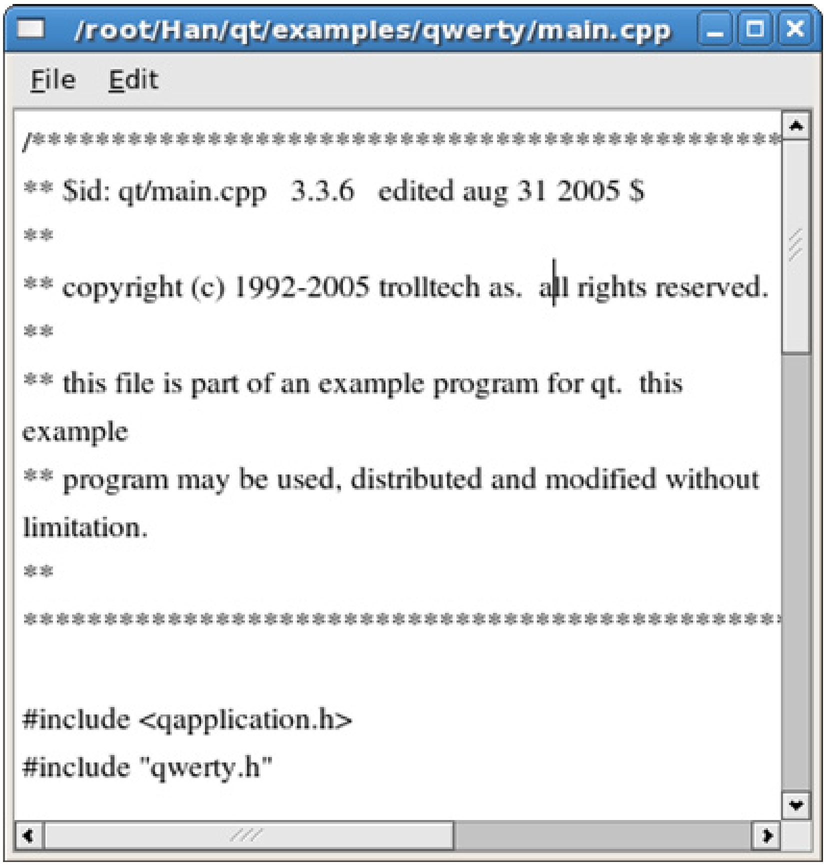
```
     ed->showMaximized();
   } else {
     ed->resize( 400, 400 );
     ed->show();
   }
}

void Editor::load()
{
#ifndef QT_NO_FILEDIALOG
   QString fn = QFileDialog::getOpenFileName( QString::null, QString::null, this );
   if ( !fn.isEmpty() )
     load( fn, -1 );
#endif
}

void Editor::load( const QString& fileName, int code )
{
   QFile f( fileName );
   if ( !f.open( IO_ReadOnly ) )
     return;

   e->setAutoUpdate( FALSE );

   QTextStream t(&f);
   if ( code >= Codec )
     t.setCodec( codecList->at(code-Codec) );
   else if ( code == Uni )
     t.setEncoding( QTextStream::Unicode );
   else if ( code == MBug )
     t.setEncoding( QTextStream::UnicodeReverse );
   else if ( code == Lat1 )
     t.setEncoding( QTextStream::Latin1 );
   else if ( code == Guess ) {
     QFile f(fileName);
     f.open(IO_ReadOnly);
     char buffer[256];
     int l = 256;
     l=f.readBlock(buffer,l);
     QTextCodec* codec = QTextCodec::codecForContent(buffer, l);
     if ( codec ) {
         QMessageBox::information(this,"Encoding",QString("Codec: ")+codec->name());
         t.setCodec( codec );
     }
   }
   e->setText( t.read() );
   f.close();

   e->setAutoUpdate( TRUE );
   e->repaint();
   setCaption( fileName );

   changed = FALSE;
}
```

```
void Editor::openAsEncoding( int code )
{
#ifndef QT_NO_FILEDIALOG
   //storing filename (proper save) is left as an exercise...
   QString fn = QFileDialog::getOpenFileName( QString::null, QString::null, this );
   if ( !fn.isEmpty() )
    (void) load( fn, code );
#endif
}

bool Editor::save()
{
#ifndef QT_NO_FILEDIALOG
   //storing filename (proper save) is left as an exercise...
   QString fn = QFileDialog::getSaveFileName( QString::null, QString::null, this );
   if ( !fn.isEmpty() )
    return saveAs( fn );
   return FALSE;
#endif
}

void Editor::saveAsEncoding( int code )
{
#ifndef QT_NO_FILEDIALOG
   //storing filename (proper save) is left as an exercise...
   QString fn = QFileDialog::getSaveFileName( QString::null, QString::null, this );
   if ( !fn.isEmpty() )
    (void) saveAs( fn, code );
#endif
}

void Editor::addEncoding()
{
#ifndef QT_NO_FILEDIALOG
   QString fn = QFileDialog::getOpenFileName( QString::null, "*.map", this );
   if ( !fn.isEmpty() ) {
    QFile f(fn);
    if (f.open(IO_ReadOnly)) {
       if (QTextCodec::loadCharmap(&f)) {
        rebuildCodecList();
       } else {
        QMessageBox::warning(0,"Charmap error",
           "The file did not contain a valid charmap.\n\n"
           "A charmap file should look like this:\n"
             "  <code_set_name> thename\n"
             "  <escape_char> /\n"
             "  % alias thealias\n"
             "  CHARMAP\n"
             "  <tokenname> /x12 <U3456>\n"
             "  <tokenname> /xAB/x12 <U0023>\n"
             "  ...\n"
             "  END CHARMAP\n"
        );
        }
      }
    }
#endif
}

bool Editor::saveAs( const QString& fileName, int code )
{
    QFile f( fileName );
    if ( no_writing || !f.open( IO_WriteOnly ) ) {
      QMessageBox::warning(this,"I/O Error",
              QString("The file could not be opened.\n\n")
               +fileName);
      return FALSE;
    }
    QTextStream t(&f);
    if ( code >= Codec )
      t.setCodec( codecList->at(code-Codec) );
    else if ( code == Uni )
      t.setEncoding( QTextStream::Unicode );
    else if ( code == MBug )
      t.setEncoding( QTextStream::UnicodeReverse );
    else if ( code == Lat1 )
      t.setEncoding( QTextStream::Latin1 );
    t << e->text();
    f.close();
    setCaption( fileName );
    changed = FALSE;
    return TRUE;
}

void Editor::print()
{
#ifndef QT_NO_PRINTER
    if ( printer.setup(this) ) {       // opens printer dialog
      printer.setFullPage(TRUE);       // we'll set our own margins
      QPainter p;
      p.begin( &printer );             // paint on printer
      p.setFont( e->font() );
      QFontMetrics fm = p.fontMetrics();
      QPaintDeviceMetrics metrics( &printer ); // need width/height
                                // of printer surface
      const int MARGIN = metrics.logicalDpiX() / 2; // half-inch margin
      int yPos        = MARGIN;        // y position for each line

      for( int i = 0 ; i < e->numLines() ; i++ ) {
          if ( printer.aborted() )
            break;
          if ( yPos + fm.lineSpacing() > metrics.height() - MARGIN ) {
            // no more room on this page
            if ( !printer.newPage() )         // start new page
                break;                        // some error
            yPos = MARGIN;                    // back to top of page
          }
        p.drawText( MARGIN, yPos, metrics.width() - 2*MARGIN,
                fm.lineSpacing(), ExpandTabs, e->textLine( i ) );
        yPos += fm.lineSpacing();
     }
     p.end();                    // send job to printer
   }
#endif
}

void Editor::resizeEvent( QResizeEvent * )
{
   if ( e && m )
     e->setGeometry( 0, m->height(), width(), height() - m->height() );
}

void Editor::closeEvent( QCloseEvent *event )
{
   event->accept();

   if ( changed ) { // the text has been changed
     switch ( QMessageBox::warning( this, "Qwerty",
                         "Save changes to Document?",
                         tr("&Yes"),
                         tr("&No"),
                         tr("Cancel"),
                         0, 2) ) {
     case 0: // yes
        if ( save() )
         event->accept();
        else
         event->ignore();
        break;
     case 1: // no
        event->accept();
        break;
     default: // cancel
        event->ignore();
        break;
     }
   }
}

void Editor::toUpper()
{
   e->setText(e->text().upper());
}

void Editor::toLower()
{
   e->setText(e->text().lower());
}

void Editor::textChanged()
{
    changed = TRUE;
}
```

main.cpp

```
#include <qapplication.h>
#include "qwerty.h"

int main( int argc, char **argv )
{
    QApplication a( argc, argv );


    bool isSmall =  qApp->desktop()->size().width() < 450
            || qApp->desktop()->size().height() < 450;

    int i;
    for ( i= argc <= 1 ? 0 : 1; i<argc; i++ ) {
     Editor *e = new Editor;
     e->setCaption("Qt Example - QWERTY");
     if ( i > 0 )
         e->load( argv[i] );
     if ( isSmall ) {
         e->showMaximized();
     } else {
         e->resize( 400, 400 );
         e->show();
     }
    }
    a.connect( &a, SIGNAL(lastWindowClosed()), &a, SLOT(quit()) );
    return a.exec();
}
```

실행



```
/**************************************************************
** $id: qt/main.cpp   3.3.6   edited aug 31 2005 $
**
** copyright (c) 1992-2005 trolltech as.  all rights reserved.
**
** this file is part of an example program for qt.  this
example
** program may be used, distributed and modified without
limitation.
**
***************************************************************


#include <qapplication.h>
#include "qwerty.h"
```

50. 범위조종

이 실례는 Qt 가 제공하는 각종 범위조종 즉 다이얄, 스핀칸, 미끄럼띠(slider)를 보여준다.

rangecontrols.pro

```
TEMPLATE  = app
TARGET    = rangecontrols
CONFIG    += qt warn_on release
HEADERS      = rangecontrols.h
SOURCES      = main.cpp \
          rangecontrols.cpp
```

rangecontrols.cpp

```
#include "rangecontrols.h"
#include <qhbox.h>
#include <qlcdnumber.h>
#include <qspinbox.h>
#include <qlabel.h>
#include <qstring.h>
#include <qslider.h>
#include <qcheckbox.h>

#include <limits.h>
```

```
RangeControls::RangeControls( QWidget *parent, const char *name )
    : QVBox( parent, name )
{
    QHBox *row1 = new QHBox( this );

    QVBox *cell2 = new QVBox( row1 );
    cell2->setMargin( 10 );
    cell2->setFrameStyle( QFrame::WinPanel | QFrame::Sunken );

    (void)new QWidget( cell2 );

    QLabel *label1 = new QLabel( QString( "Enter a value between\n%1 and %2:" ).arg( -
INT_MAX ).arg( INT_MAX ), cell2 );
    label1->setMaximumHeight( label1->sizeHint().height() );
    QSpinBox *sb1 = new QSpinBox( -INT_MAX, INT_MAX, 1, cell2 );
    sb1->setValue( 0 );

    QLabel *label2 = new QLabel( "Enter a zoom value:", cell2 );
    label2->setMaximumHeight( label2->sizeHint().height() );
    QSpinBox *sb2 = new QSpinBox( 0, 1000, 10, cell2 );
    sb2->setSuffix( " %" );
    sb2->setSpecialValueText( "Automatic" );

    QLabel *label3 = new QLabel( "Enter a price:", cell2 );
    label3->setMaximumHeight( label3->sizeHint().height() );
    QSpinBox *sb3 = new QSpinBox( 0, INT_MAX, 1, cell2 );
    sb3->setPrefix( "$" );
    sb3->setValue( 355 );

    (void)new QWidget( cell2 );

    QHBox *row2 = new QHBox( this );

    QVBox *cell3 = new QVBox( row2 );
    cell3->setMargin( 10 );
    cell3->setFrameStyle( QFrame::WinPanel | QFrame::Sunken );
    QSlider *hslider = new QSlider( 0, 64, 1, 33, Qt::Horizontal, cell3, "horizontal_s" );
    QLCDNumber *lcd2 = new QLCDNumber( 2, cell3 );
    lcd2->display( 33 );
    lcd2->setSegmentStyle( QLCDNumber::Filled );
    connect( hslider, SIGNAL( valueChanged( int ) ), lcd2, SLOT( display( int ) ) );

    QHBox *cell4 = new QHBox( row2 );
    cell4->setFrameStyle( QFrame::WinPanel | QFrame::Sunken );
    cell4->setMargin( 10 );
    QSlider *vslider = new QSlider( 0, 64, 1, 8, Qt::Vertical, cell4 );
    QLCDNumber *lcd3 = new QLCDNumber( 3, cell4 );
    lcd3->display( 8 );
    connect( vslider, SIGNAL( valueChanged( int ) ), lcd3, SLOT( display( int ) ) );
}
```

rangecontrols.h

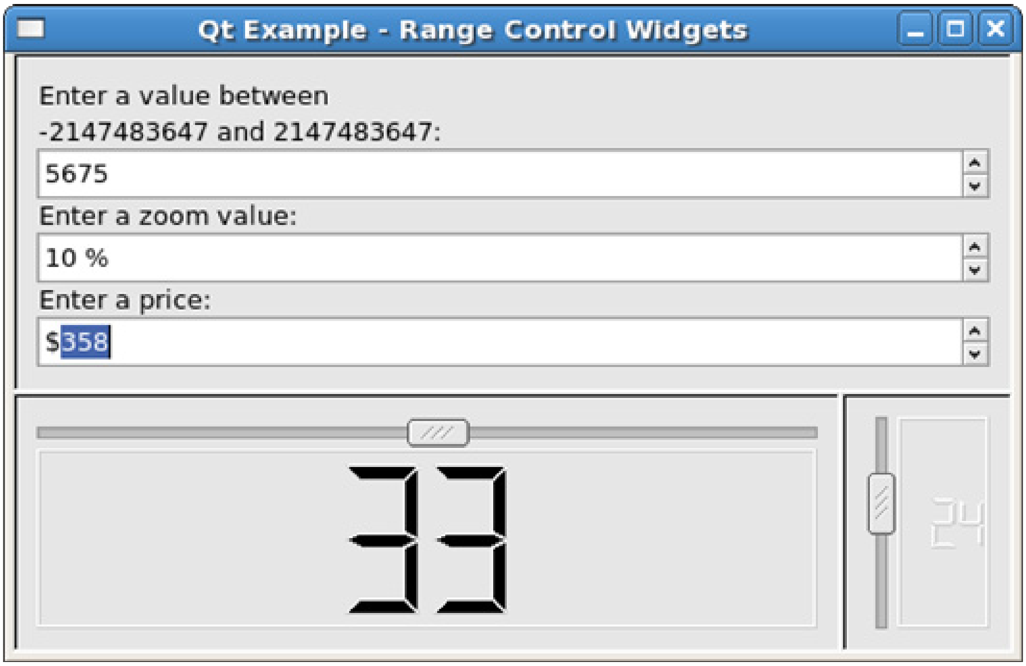
```
#ifndef RANGECONTROLS_H
#define RANGECONTROLS_H
```

```
#include <qvbox.h>

class QCheckBox;

class RangeControls : public QVBox
{
   Q_OBJECT

public:
   RangeControls( QWidget *parent = 0, const char *name = 0 );

private:
   QCheckBox *notches, *wrapping;
};

#endif
```

main.cpp

```
#include "rangecontrols.h"
#include <qapplication.h>

int main( int argc, char **argv )
{
   QApplication a( argc, argv );

   RangeControls rangecontrols;
   rangecontrols.resize( 500, 300 );
   rangecontrols.setCaption( "Qt Example - Range Control Widgets" );
   a.setMainWidget( &rangecontrols );
   rangecontrols.show();

   return a.exec();
}
```

실행



Qt Example - Range Control Widgets

Enter a value between
-2147483647 and 2147483647:

5675

Enter a zoom value:

10 %

Enter a price:

$358

51. 정규식을 시험하는 작은 응용프로그람

정규식은 흔히 정확히 얻기 힘들고 *기호를 사용할 때 특히 더하다. 이 응용프로그람은 정규식(regexp, 2중역사선이 없다)과 검사본문에 입력하게 하고 정규식을 실행하고 결과를 본다. Copy단추를 찰칵하면 정규식이 오려둠판에 복사된다. (2중역사선이 있고 자기 프로그람에 붙이기할 준비가 된다.) 이전의 정규식들과 검사본문들은 쎄숀을 통하여 기억되고 복합칸들을 아래로 펼치여 호출할수 있다.

regexptester.pro

```
SOURCES     += main.cpp
HEADERS     += regexptester.h
SOURCES     += regexptester.cpp
```

regexptester.cpp

```
#include <qapplication.h>
#include <qcheckbox.h>
#include <qclipboard.h>
#include <qcombobox.h>
#include <qlabel.h>
#include <qlayout.h>
#include <qpushbutton.h>
#include <qregexp.h>
#include <qstatusbar.h>
#include <qtable.h>

#include "regexptester.h"

RegexpTester::RegexpTester(QWidget* parent, const char* name, bool modal,
                WFlags f)
    : QDialog(parent, name, modal, f)
```

```
{
    regexLabel = new QLabel(this);
    regexComboBox = new QComboBox(this);
    regexComboBox->setEditable(true);
    regexComboBox->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Preferred);
    regexLabel->setBuddy(regexComboBox);
    textLabel = new QLabel(this);
    textComboBox = new QComboBox(this);
    textComboBox->setEditable(true);
    textComboBox->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Preferred);
    textLabel->setBuddy(textComboBox);
    caseSensitiveCheckBox = new QCheckBox(this);
    caseSensitiveCheckBox->setChecked(true);
    minimalCheckBox = new QCheckBox(this);
    wildcardCheckBox = new QCheckBox(this);
    resultTable = new QTable(3, 3, this);
    resultTable->verticalHeader()->hide();
    resultTable->setLeftMargin(0);
    resultTable->horizontalHeader()->hide();
    resultTable->setTopMargin(0);
    resultTable->setReadOnly(true);
    executePushButton = new QPushButton(this);
    executePushButton->setDefault(true);
    copyPushButton = new QPushButton(this);
    quitPushButton = new QPushButton(this);
    statusBar = new QStatusBar(this);

    QGridLayout *gridLayout = new QGridLayout(2, 2, 6);
    gridLayout->addWidget(regexLabel, 0, 0);
    gridLayout->addWidget(regexComboBox, 0, 1);
    gridLayout->addWidget(textLabel, 1, 0);
    gridLayout->addWidget(textComboBox, 1, 1);
    QHBoxLayout *checkboxLayout = new QHBoxLayout(0, 6, 6);
    checkboxLayout->addWidget(caseSensitiveCheckBox);
    checkboxLayout->addWidget(minimalCheckBox);
    checkboxLayout->addWidget(wildcardCheckBox);
    checkboxLayout->addStretch(1);
    QVBoxLayout *buttonLayout = new QVBoxLayout(0, 6, 6);
    buttonLayout->addWidget(executePushButton);
    buttonLayout->addWidget(copyPushButton);
    buttonLayout->addWidget(quitPushButton);
    buttonLayout->addStretch(1);
    QHBoxLayout *middleLayout = new QHBoxLayout(0, 6, 6);
    middleLayout->addWidget(resultTable);
    middleLayout->addLayout(buttonLayout);
    QVBoxLayout *mainLayout = new QVBoxLayout(this, 6, 6);
    mainLayout->addLayout(gridLayout);
    mainLayout->addLayout(checkboxLayout);
    mainLayout->addLayout(middleLayout);
    mainLayout->addWidget(statusBar);

    resize(QSize(500, 350).expandedTo(minimumSizeHint()));

    languageChange();
    connect(copyPushButton, SIGNAL(clicked()), this, SLOT(copy()));
    connect(executePushButton, SIGNAL(clicked()), this, SLOT(execute()));
    connect(quitPushButton, SIGNAL(clicked()), this, SLOT(accept()));

    execute();
}

void RegexpTester::execute()
{
    QString regex = regexComboBox->currentText();
    QString text = textComboBox->currentText();
    if (!regex.isEmpty() && !text.isEmpty()) {
        QRegExp re(regex);
        re.setCaseSensitive(caseSensitiveCheckBox->isChecked());
        re.setMinimal(minimalCheckBox->isChecked());
        bool wildcard = wildcardCheckBox->isChecked();
        re.setWildcard(wildcard);
        if (!re.isValid()) {
            statusBar->message(tr("Invalid regular expression: %1")
                               .arg(re.errorString()));
            return;
        }
        int offset = re.search(text);
        int captures = re.numCaptures();
        int row = 0;
        const int OFFSET = 5;
        resultTable->setNumRows(0);
        resultTable->setNumRows(captures + OFFSET);
        resultTable->setText(row, 0, tr("Regex"));
        QString escaped = regex;
        escaped = escaped.replace("\\", "\\\\");
        resultTable->setText(row, 1, escaped);
        resultTable->item(row, 1)->setSpan(1, 2);
        if (offset != -1) {
            ++row;
            resultTable->setText(row, 0, tr("Offset"));
            resultTable->setText(row, 1, QString::number(offset));
            resultTable->item(row, 1)->setSpan(1, 2);
            if (!wildcard) {
                ++row;
                resultTable->setText(row, 0, tr("Captures"));
                resultTable->setText(row, 1, QString::number(captures));
                resultTable->item(row, 1)->setSpan(1, 2);
                ++row;
                resultTable->setText(row, 1, tr("Text"));
                resultTable->setText(row, 2, tr("Characters"));
            }
            ++row;
            resultTable->setText(row, 0, tr("Match"));
            resultTable->setText(row, 1, re.cap(0));
            resultTable->setText(row, 2, QString::number(re.matchedLength()));
            if (!wildcard) {
                for (int i = 1; i <= captures; ++i) {
```

```
         resultTable->setText(row + i, 0, tr("Capture #%1").arg(i));
         resultTable->setText(row + i, 1, re.cap(i));
         resultTable->setText(row + i, 2,
                 QString::number(re.cap(i).length()));
       }
      }
      else
       resultTable->setNumRows(3);
    }
    else {
      resultTable->setNumRows(2);
      ++row;
      resultTable->setText(row, 0, tr("No matches"));
      resultTable->item(row, 0)->setSpan(1, 3);
    }
    resultTable->adjustColumn(0);
    resultTable->adjustColumn(1);
    resultTable->adjustColumn(2);
    statusBar->message(tr("Executed \"%1\" on \"%2\"")
             .arg(escaped).arg(text));
   }
   else
    statusBar->message(tr("A regular expression and a text must be given"));
}

void RegexpTester::copy()
{
   QString escaped = regexComboBox->currentText();
   if (!escaped.isEmpty()) {
    escaped = escaped.replace("\\", "\\\\");
    QClipboard *cb = QApplication::clipboard();
    cb->setText(escaped, QClipboard::Clipboard);
    if (cb->supportsSelection())
      cb->setText(escaped, QClipboard::Selection);
    statusBar->message(tr("Copied \"%1\" to the clipboard")
             .arg(escaped));
   }
}

void RegexpTester::languageChange()
{
   setCaption(tr("Regex Tester"));
   regexLabel->setText(tr("&Regex:"));
   regexComboBox->insertItem(tr("[A-Z]+=(\\d+):(\\d*)"));
   textLabel->setText(tr("&Text:"));
   textComboBox->insertItem(tr("ABC=12:3456"));
   caseSensitiveCheckBox->setText(tr("Case &Sensitive"));
   minimalCheckBox->setText(tr("&Minimal"));
   wildcardCheckBox->setText(tr("&Wildcard"));
   copyPushButton->setText(tr("&Copy"));
   executePushButton->setText(tr("&Execute"));
   quitPushButton->setText(tr("&Quit"));
}
```

regexptester.h

```
#ifndef REGEXPTESTER_H
#define REGEXPTESTER_H

#include <qdialog.h>

class QCheckBox;
class QComboBox;
class QLabel;
class QPushButton;
class QStatusBar;
class QTable;

class RegexpTester : public QDialog
{
   Q_OBJECT

public:
   RegexpTester(QWidget* parent=0, const char* name=0, bool modal=false,
           WFlags f=0);

   QLabel *regexLabel;
   QComboBox *regexComboBox;
   QLabel *textLabel;
   QComboBox *textComboBox;
   QCheckBox *caseSensitiveCheckBox;
   QCheckBox *minimalCheckBox;
   QCheckBox *wildcardCheckBox;
   QTable *resultTable;
   QPushButton *executePushButton;
   QPushButton *copyPushButton;
   QPushButton *quitPushButton;
   QStatusBar *statusBar;

public slots:
   void copy();
   void execute();

private:
   void languageChange();
};

#endif // REGEXPTESTER_H
```

main.cpp

```
#include <qapplication.h>
#include "regexptester.h"

int main(int argc, char **argv)
{
   QApplication app(argc, argv);
   RegexpTester form;
   form.show();
   app.connect(&app, SIGNAL(lastWindowClosed()), &app, SLOT(quit()));
```

```
    "<big>Evil is that which one believes of others.  It is a sin to believe evil "
    "of others, but it is seldom a mistake.</big><br><br>"
    "<center><i>-- H.L. Mencken</i></center>",

    "<b>Saying 2:</b><br>"
    "<hr><br><br>"
    "<big>A well-used door needs no oil on its hinges.<br>"
    "A swift-flowing steam does not grow stagnant.<br>"
    "Neither sound nor thoughts can travel through a vacuum.<br>"
    "Software rots if not used.<br><br>"
    "These are great mysteries.</big><br><br>"
    "<center><i>-- Geoffrey James, \"The Tao of Programming\"</i></center>",

    "<b>Saying 3:</b><br>"
    "<hr><br><br>"
    "<big>Show business is just like high school, except you get paid.</big><br><br>"
    "<center><i>-- Martin Mull</i></center>",

    "<b>Saying 4:</b><br>"
    "<hr><br><br>"
    "<big><b>The Least Successful Executions</b><br>"
    "<twocolumn><p>      History has furnished us with two executioners worthy of attention. "
    "The first performed in Sydney in Australia.  In 1803 three attempts were "
    "made to hang a Mr. Joseph Samuels.  On the first two of these the rope "
    "snapped, while on the third Mr. Samuels just hung there peacefully until he "
    "and everyone else got bored.  Since he had proved unsusceptible to capital "
    "punishment, he was reprieved.</p>"
    "<p>      The most important British executioner was Mr. James Berry who "
    "tried three times in 1885 to hang Mr. John Lee at Exeter Jail, but on each "
    "occasion failed to get the trap door open.<!p>"
    "<p>      In recognition of this achievement, the Home Secretary commuted "
    "Lee's sentence to \"life\" imprisonment.  He was released in 1917, emigrated "
    "to America and lived until 1933.</p></twocolumn></big><br><br>"
    "<center><i>-- Stephen Pile, \"The Book of Heroic Failures\"</i></center>",

    "<b>Saying 5:</b><br>"
    "<hr><br><br>"
    "<big>If you can, help others.  If you can't, at least don't hurt others.</big><br><br>"
    "<center><i>-- the Dalai Lama</i></center>",

    "<b>Saying 6:</b><br>"
    "<hr><br><br>"
    "<big>Television has brought back murder into the home -- where it belongs.</big><br><br>"
    "<center><i>-- Alfred Hitchcock</i></center>",

    "<b>Saying 7:</b><br>"
    "<hr><br><br>"
    "<big>I don't know who my grandfather was; I am much more concerned to know "
    "what his grandson will be.</big><br><br>"
    "<center><i>-- Abraham Lincoln</i></center>",

    0
};
```

```
MyRichText::MyRichText( QWidget *parent, const char *name )
    : QVBox( parent, name )
{
    setMargin( 5 );

    view = new QTextView( this );
    view->setText( "This is a <b>Test</b> with <i>italic</i> <u>stuff</u>" );
    QBrush paper;
    paper.setPixmap( QPixmap( "../richtext/marble.png" ) );
    if ( paper.pixmap() != 0 )
      view->setPaper( paper );
    else
      view->setPaper( white );

    view->setText( sayings[0] );
    view->setMinimumSize( 450, 250 );

    QHBox *buttons = new QHBox( this );
    buttons->setMargin( 5 );

    bClose = new QPushButton( "&Close", buttons );
    bPrev = new QPushButton( "<< &Prev", buttons );
    bNext = new QPushButton( "&Next >>", buttons );

    bPrev->setEnabled( FALSE );

    connect( bClose, SIGNAL( clicked() ), qApp, SLOT( quit() ) );
    connect( bPrev, SIGNAL( clicked() ), this, SLOT( prev() ) );
    connect( bNext, SIGNAL( clicked() ), this, SLOT( next() ) );

    num = 0;
}

void MyRichText::prev()
{
    if ( num <= 0 )
        return;

    num--;

    view->setText( sayings[num] );

    if ( num == 0 )
        bPrev->setEnabled( FALSE );

    bNext->setEnabled( TRUE );
}

void MyRichText::next()
{
    if ( !sayings[++num] )
        return;

    view->setText( sayings[num] );
```

```
   if ( !sayings[num + 1] )
      bNext->setEnabled( FALSE );

   bPrev->setEnabled( TRUE );
}
```

richtext.h

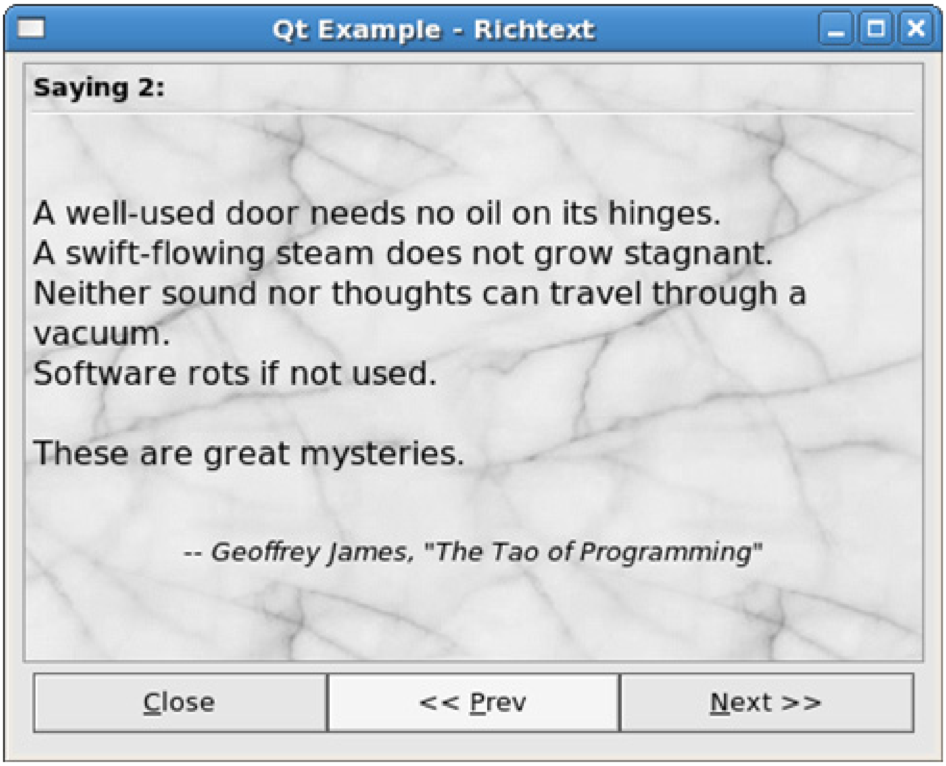
```
#ifndef RICHTEXT_H
#define RICHTEXT_H

#include <qvbox.h>

class QTextView;
class QPushButton;

class MyRichText : public QVBox
{
   Q_OBJECT

public:
   MyRichText( QWidget *parent = 0, const char *name = 0 );

protected:
   QTextView *view;
   QPushButton *bClose, *bNext, *bPrev;
   int num;

protected slots:
   void prev();
   void next();

};

#endif
```

main.cpp

```
#include "richtext.h"
#include <qapplication.h>

int main( int argc, char **argv )
{
   QApplication a( argc, argv );

   MyRichText richtext;
   richtext.resize( 450, 350 );
   richtext.setCaption( "Qt Example - Richtext" );
   a.setMainWidget( &richtext );
   richtext.show();

   return a.exec();
}
```

marble.png

실행



Saying 2:

A well-used door needs no oil on its hinges.
A swift-flowing steam does not grow stagnant.
Neither sound nor thoughts can travel through a vacuum.
Software rots if not used.

These are great mysteries.

-- Geoffrey James, "The Tao of Programming"

53. Rot13

이 실례는 여러행편집창문부품에 본문을 입력하게 한다. 이것은 rot13알고리듬에 의해 정확히 변환되여 편집창문부품에 현시된다.

rot13.pro

```
TEMPLATE  = app
TARGET     = rot13
CONFIG     += qt warn_on release
HEADERS        = rot13.h
```

```
SOURCES         = rot13.cpp
```

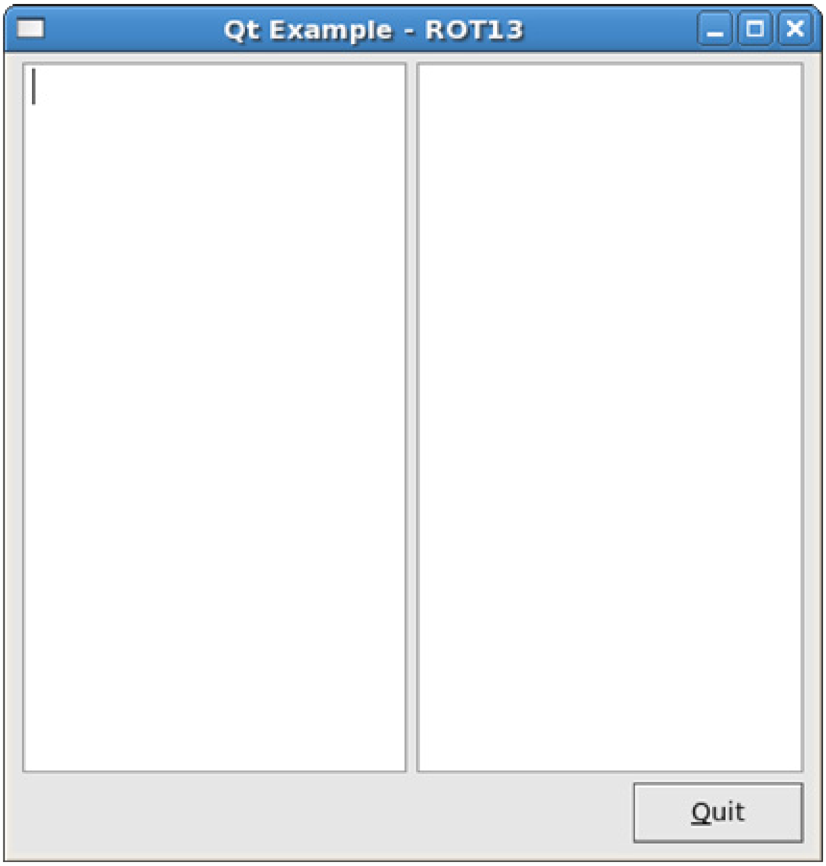
rot13.cpp

```
#include "rot13.h"
#include <qmultilineedit.h>
#include <qpushbutton.h>
#include <qapplication.h>
#include <qlayout.h>

Rot13::Rot13()
{
   left = new QMultiLineEdit( this, "left" );
   right = new QMultiLineEdit( this, "right" );
   connect( left, SIGNAL(textChanged()), this, SLOT(changeRight()) );
   connect( right, SIGNAL(textChanged()), this, SLOT(changeLeft()) );

   QPushButton * quit = new QPushButton( "&Quit", this );
   quit->setFocusPolicy( NoFocus );
   connect( quit, SIGNAL(clicked()), qApp, SLOT(quit()) );

   QGridLayout * l = new QGridLayout( this, 2, 2, 5 );
   l->addWidget( left, 0, 0 );
   l->addWidget( right, 0, 1 );
   l->addWidget( quit, 1, 1, AlignRight );

   left->setFocus();
}


void Rot13::changeLeft()
{
   left->blockSignals( TRUE );
   left->setText( rot13( right->text() ) );
   left->blockSignals( FALSE );
}


void Rot13::changeRight()
{
   right->blockSignals( TRUE );
   right->setText( rot13( left->text() ) );
   right->blockSignals( FALSE );
}


QString Rot13::rot13( const QString & input ) const
{
   QString r = input;
   int i = r.length();
   while( i-- ) {
    if ( r[i] >= QChar('A') && r[i] <= QChar('M') ||
         r[i] >= QChar('a') && r[i] <= QChar('m') )
        r[i] = (char)((int)QChar(r[i]) + 13);
    else if  ( r[i] >= QChar('N') && r[i] <= QChar('Z') ||
           r[i] >= QChar('n') && r[i] <= QChar('z') )
        r[i] = (char)((int)QChar(r[i]) - 13);
   }
   return r;
}

int main( int argc, char ** argv )
{
   QApplication a( argc, argv );
   Rot13 r;
   r.resize( 400, 400 );
   a.setMainWidget( &r );
   r.setCaption("Qt Example - ROT13");
   r.show();
   return a.exec();
}
```

rot13.h

```
#ifndef ROT13_H
#define ROT13_H
#include <qwidget.h>

class QMultiLineEdit;

class Rot13: public QWidget {
    Q_OBJECT
public:
    Rot13();

    QString rot13( const QString & ) const;

private slots:
    void changeLeft();
    void changeRight();

private:
    QMultiLineEdit * left, * right;
};

#endif
```

실행



54. 간단한 그리기응용프로그람

이 실례는 유명한 란필실례를 실현한다. 각이한 펜으로 캔버스에 그리고 결과를 그림으로 보관할수 있다.

scribble.pro

```
TEMPLATE  = app
TARGET      = scribble
CONFIG      += qt warn_on release
HEADERS       = scribble.h
SOURCES       = main.cpp \
          scribble.cpp
```

scribble.cpp

```
#include "scribble.h"
#include <qapplication.h>
#include <qevent.h>
#include <qpainter.h>
#include <qtoolbar.h>
#include <qtoolbutton.h>
#include <qspinbox.h>
#include <qtooltip.h>
#include <qrect.h>
#include <qpoint.h>
```

```
#include <qcolordialog.h>
#include <qfiledialog.h>
#include <qcursor.h>
#include <qimage.h>
#include <qstrlist.h>
#include <qpopupmenu.h>
#include <qintdict.h>

const bool no_writing = FALSE;

Canvas::Canvas( QWidget *parent, const char *name )
    : QWidget( parent, name, WStaticContents ), pen( Qt::red, 3 ), polyline(3),
      mousePressed( FALSE ), buffer( width(), height() )
{

    if ((qApp->argc() > 0) && !buffer.load(qApp->argv()[1]))
      buffer.fill( colorGroup().base() );
    setBackgroundMode( QWidget::PaletteBase );
#ifndef QT_NO_CURSOR
    setCursor( Qt::crossCursor );
#endif
}

void Canvas::save( const QString &filename, const QString &format )
{
    if ( !no_writing )
      buffer.save( filename, format.upper() );
}

void Canvas::clearScreen()
{
    buffer.fill( colorGroup().base() );
    repaint( FALSE );
}

void Canvas::mousePressEvent( QMouseEvent *e )
{
    mousePressed = TRUE;
    polyline[2] = polyline[1] = polyline[0] = e->pos();
}

void Canvas::mouseReleaseEvent( QMouseEvent * )
{
    mousePressed = FALSE;
}

void Canvas::mouseMoveEvent( QMouseEvent *e )
{
    if ( mousePressed ) {
      QPainter painter;
      painter.begin( &buffer );
      painter.setPen( pen );
      polyline[2] = polyline[1];
      polyline[1] = polyline[0];
```

```
     polyline[0] = e->pos();
     painter.drawPolyline( polyline );
     painter.end();

     QRect r = polyline.boundingRect();
     r = r.normalize();
     r.setLeft( r.left() - penWidth() );
     r.setTop( r.top() - penWidth() );
     r.setRight( r.right() + penWidth() );
     r.setBottom( r.bottom() + penWidth() );

     bitBlt( this, r.x(), r.y(), &buffer, r.x(), r.y(), r.width(), r.height() );
   }
}

void Canvas::resizeEvent( QResizeEvent *e )
{
   QWidget::resizeEvent( e );

   int w = width() > buffer.width() ?
        width() : buffer.width();
   int h = height() > buffer.height() ?
        height() : buffer.height();

   QPixmap tmp( buffer );
   buffer.resize( w, h );
   buffer.fill( colorGroup().base() );
   bitBlt( &buffer, 0, 0, &tmp, 0, 0, tmp.width(), tmp.height() );
}

void Canvas::paintEvent( QPaintEvent *e )
{
   QWidget::paintEvent( e );

   QMemArray<QRect> rects = e->region().rects();
   for ( uint i = 0; i < rects.count(); i++ ) {
     QRect r = rects[(int)i];
     bitBlt( this, r.x(), r.y(), &buffer, r.x(), r.y(), r.width(), r.height() );
   }
}

//------------------------------------------------------

Scribble::Scribble( QWidget *parent, const char *name )
   : QMainWindow( parent, name )
{
   canvas = new Canvas( this );
   setCentralWidget( canvas );

   QToolBar *tools = new QToolBar( this );

   bSave = new QToolButton( QPixmap(), "Save", "Save as PNG image", this, SLOT( slotSave() ),
tools );
   bSave->setText( "Save as..." );
   tools->addSeparator();

   bPColor = new QToolButton( QPixmap(), "Choose Pen Color", "Choose Pen Color", this,
SLOT( slotColor() ), tools );
   bPColor->setText( "Choose Pen Color..." );

   tools->addSeparator();

   bPWidth = new QSpinBox( 1, 20, 1, tools );
   QToolTip::add( bPWidth, "Choose Pen Width" );
   connect( bPWidth, SIGNAL( valueChanged( int ) ), this, SLOT( slotWidth( int ) ) );
   bPWidth->setValue( 3 );

   tools->addSeparator();

   bClear = new QToolButton( QPixmap(), "Clear Screen", "Clear Screen", this, SLOT( slotClear() ),
tools );
   bClear->setText( "Clear Screen" );
}

void Scribble::slotSave()
{
   QPopupMenu *menu = new QPopupMenu( 0 );
   QIntDict<QString> formats;
   formats.setAutoDelete( TRUE );

   for ( unsigned int i = 0; i < QImageIO::outputFormats().count(); i++ ) {
    QString str = QString( QImageIO::outputFormats().at( i ) );
    formats.insert( menu->insertItem( QString( "%1..." ).arg( str ) ), new QString( str ) );
   }

   menu->setMouseTracking( TRUE );
   int id = menu->exec( bSave->mapToGlobal( QPoint( 0, bSave->height() + 1 ) ) );

   if ( id != -1 ) {
    QString format = *formats[ id ];

    QString filename = QFileDialog::getSaveFileName( QString::null,
QString( "*.%1" ).arg( format.lower() ), this );
    if ( !filename.isEmpty() )
       canvas->save( filename, format );
   }

   delete menu;
}

void Scribble::slotColor()
{
   QColor c = QColorDialog::getColor( canvas->penColor(), this );
   if ( c.isValid() )
    canvas->setPenColor( c );
}
```

```
void Scribble::slotWidth( int w )
{
   canvas->setPenWidth( w );
}

void Scribble::slotClear()
{
   canvas->clearScreen();
}
```

scribble.h

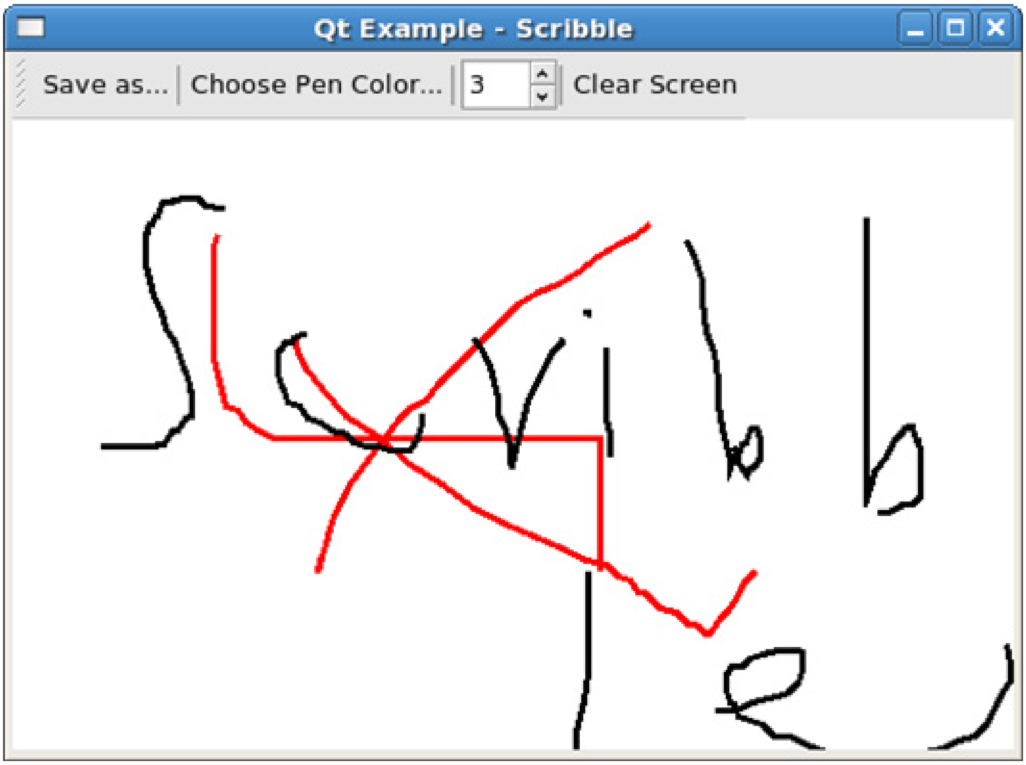
```
#ifndef SCRIBBLE_H
#define SCRIBBLE_H
#include <qmainwindow.h>
#include <qpen.h>
#include <qpoint.h>
#include <qpixmap.h>
#include <qwidget.h>
#include <qstring.h>
#include <qpointarray.h>

class QMouseEvent;
class QResizeEvent;
class QPaintEvent;
class QToolButton;
class QSpinBox;

class Canvas : public QWidget
{
   Q_OBJECT

public:
   Canvas( QWidget *parent = 0, const char *name = 0 );

   void setPenColor( const QColor &c )
   { pen.setColor( c ); }

   void setPenWidth( int w )
   { pen.setWidth( w ); }

   QColor penColor()
   { return pen.color(); }

   int penWidth()
   { return pen.width(); }

   void save( const QString &filename, const QString &format );

   void clearScreen();

protected:
   void mousePressEvent( QMouseEvent *e );
   void mouseReleaseEvent( QMouseEvent *e );
   void mouseMoveEvent( QMouseEvent *e );
   void resizeEvent( QResizeEvent *e );
   void paintEvent( QPaintEvent *e );

   QPen pen;
   QPointArray polyline;

   bool mousePressed;

   QPixmap buffer;

};

class Scribble : public QMainWindow
{
   Q_OBJECT

public:
   Scribble( QWidget *parent = 0, const char *name = 0 );

protected:
   Canvas* canvas;

   QSpinBox *bPWidth;
   QToolButton *bPColor, *bSave, *bClear;

protected slots:
   void slotSave();
   void slotColor();
   void slotWidth( int );
   void slotClear();

};

#endif
```

main.cpp

```
#include "scribble.h"
#include <qapplication.h>

int main( int argc, char **argv )
{
   QApplication a( argc, argv );

   Scribble scribble;

   scribble.resize( 500, 350 );
   scribble.setCaption("Qt Example - Scribble");
   a.setMainWidget( &scribble );
   if ( QApplication::desktop()->width() > 550
     && QApplication::desktop()->height() > 366 )
    scribble.show();
   else
    scribble.showMaximized();
   return a.exec();
```

}

실행



55. 흘림보기

이 실례는 Qt의 흘림보기사용법을 보여준다. 이것은 매우 큰 내용들을 표시할 때 가장 합리적인 창문부품이다.

scrollview.pro

```
TEMPLATE  = app
TARGET    = scrollview
CONFIG    += qt warn_on release
HEADERS      =
SOURCES      = scrollview.cpp
```

scrollview.cpp

```
#include <qscrollview.h>
#include <qapplication.h>
#include <qmenubar.h>
#include <qpopupmenu.h>
#include <qpushbutton.h>
#include <qpainter.h>
#include <qpixmap.h>
#include <qmessagebox.h>
#include <qlayout.h>
#include <qlabel.h>
#include <qmultilineedit.h>
```

```
#include <qsizegrip.h>
#include <stdlib.h>

static const int style_id   = 0x1000;
static const int lw_id      = 0x2000;
static const int mlw_id        = 0x4000;
static const int mw_id         = 0x8000;
static const int max_lw        = 16;
static const int max_mlw     = 5;
static const int max_mw      = 10;


class BigShrinker : public QFrame {
  Q_OBJECT
public:
  BigShrinker(QWidget* parent) :
   QFrame(parent)
  {
   setFrameStyle(QFrame::Box|QFrame::Sunken);
   int h=35;
   int b=0;
   for (int y=0; y<2000-h; y+=h+10) {
      if (y == 0) {
       QButton* q=new QPushButton("Quit", this);
       connect(q, SIGNAL(clicked()), qApp, SLOT(quit()));
      } else {
       QString str;
       if ( b > 0 ) {
          str.sprintf("Button %d", b++);
       } else {
          str = "I'm shrinking!";
          ++b;
       }
       (new QPushButton(str, this))->move(y/2,y);
      }
   }
   resize(1000,2000);

   startTimer(250);
  }

  void timerEvent(QTimerEvent*)
  {
   int w=width();
   int h=height();
   if ( w > 50 ) w -= 1;
   if ( h > 50 ) h -= 2;
   resize(w,h);
  }

  void mouseReleaseEvent(QMouseEvent* e)
  {
   emit clicked(e->x(), e->y());
  }
```

```
signals:
   void clicked(int,int);
};

class BigMatrix : public QScrollView {
   QMultiLineEdit *dragging;
public:
   BigMatrix(QWidget* parent) :
    QScrollView(parent,"matrix", WStaticContents),
    bg("bg.ppm")
   {
    bg.load("bg.ppm");
    resizeContents(400000,300000);

    dragging = 0;
   }

   void viewportMousePressEvent(QMouseEvent* e)
   {
    int x, y;
    viewportToContents( e->x(),  e->y(), x, y );
    dragging = new QMultiLineEdit(viewport(),"Another");
    dragging->setText("Thanks!");
    dragging->resize(100,100);
    addChild(dragging, x, y);
    showChild(dragging);
   }

   void viewportMouseReleaseEvent(QMouseEvent*)
   {
    dragging = 0;
   }

   void viewportMouseMoveEvent(QMouseEvent* e)
   {
    if ( dragging ) {
       int mx, my;
       viewportToContents( e->x(),  e->y(), mx, my );
       int cx = childX(dragging);
       int cy = childY(dragging);
       int w = mx - cx + 1;
       int h = my - cy + 1;
       QString msg;
       msg.sprintf("at (%d,%d) %d by %d",cx,cy,w,h);
       dragging->setText(msg);
       dragging->resize(w,h);
    }
   }

protected:
   void drawContents(QPainter* p, int cx, int cy, int cw, int ch)
   {
    // The Background
```

```
if ( !bg.isNull() ) {
    int rowheight=bg.height();
    int toprow=cy/rowheight;
    int bottomrow=(cy+ch+rowheight-1)/rowheight;
    int colwidth=bg.width();
    int leftcol=cx/colwidth;
    int rightcol=(cx+cw+colwidth-1)/colwidth;
    for (int r=toprow; r<=bottomrow; r++) {
     int py=r*rowheight;
     for (int c=leftcol; c<=rightcol; c++) {
         int px=c*colwidth;
         p->drawPixmap(px, py, bg);
     }
    }
} else {
    p->fillRect(cx, cy, cw, ch, QColor(240,222,208));
}

// The Numbers
{
    QFontMetrics fm=p->fontMetrics();
    int rowheight=fm.lineSpacing();
    int toprow=cy/rowheight;
    int bottomrow=(cy+ch+rowheight-1)/rowheight;
    int colwidth=fm.width("00000,000000 ")+3;
    int leftcol=cx/colwidth;
    int rightcol=(cx+cw+colwidth-1)/colwidth;
    QString str;
    for (int r=toprow; r<=bottomrow; r++) {
     int py=r*rowheight;
     for (int c=leftcol; c<=rightcol; c++) {
         int px=c*colwidth;
         str.sprintf("%d,%d",c,r);
         p->drawText(px+3, py+fm.ascent(), str);
     }
    }

    // The Big Hint
    if (leftcol<10 && toprow<5) {
     p->setFont(QFont("Charter",30));
     p->setPen(red);
     QString text;
     text.sprintf("HINT:  Look at %d,%d",215000/colwidth,115000/rowheight);
     p->drawText(100,50,text);
    }
}

// The Big X
{
    if (cx+cw>200000 && cy+ch>100000 && cx<230000 && cy<130000) {
     // Note that some X server cannot even handle co-ordinates
     // beyond about 4000, so you might not see this.
     p->drawLine(200000,100000,229999,129999);
     p->drawLine(229999,100000,200000,129999);
```

```
         // X marks the spot!
         p->setFont(QFont("Charter",100));
         p->setPen(blue);
         p->drawText(215000-500,115000-100,1000,200,AlignCenter,"YOU WIN!!!!!");
        }
     }
   }

private:
   QPixmap bg;
};

class ScrollViewExample : public QWidget {
   Q_OBJECT

public:
   ScrollViewExample(int technique, QWidget* parent=0, const char* name=0) :
    QWidget(parent,name)
   {
    QMenuBar* menubar = new QMenuBar(this);
    Q_CHECK_PTR( menubar );

    QPopupMenu* file = new QPopupMenu( menubar );
    Q_CHECK_PTR( file );
    menubar->insertItem( "&File", file );
    file->insertItem( "Quit", qApp,  SLOT(quit()) );

    vp_options = new QPopupMenu( menubar );
    Q_CHECK_PTR( vp_options );
    vp_options->setCheckable( TRUE );
    menubar->insertItem( "&ScrollView", vp_options );
    connect( vp_options, SIGNAL(activated(int)),
       this, SLOT(doVPMenuItem(int)) );

    vauto_id = vp_options->insertItem( "Vertical Auto" );
    vaoff_id = vp_options->insertItem( "Vertical AlwaysOff" );
    vaon_id = vp_options->insertItem( "Vertical AlwaysOn" );
    vp_options->insertSeparator();
    hauto_id = vp_options->insertItem( "Horizontal Auto" );
    haoff_id = vp_options->insertItem( "Horizontal AlwaysOff" );
    haon_id = vp_options->insertItem( "Horizontal AlwaysOn" );
    vp_options->insertSeparator();
    corn_id = vp_options->insertItem( "cornerWidget" );

    if (technique == 1) {
       vp = new QScrollView(this);
       BigShrinker *bs = new BigShrinker(0);//(vp->viewport());
       vp->addChild(bs);
       bs->setAcceptDrops(TRUE);
       QObject::connect(bs, SIGNAL(clicked(int,int)),
                vp, SLOT(center(int,int)));
    } else {
       vp = new BigMatrix(this);
```

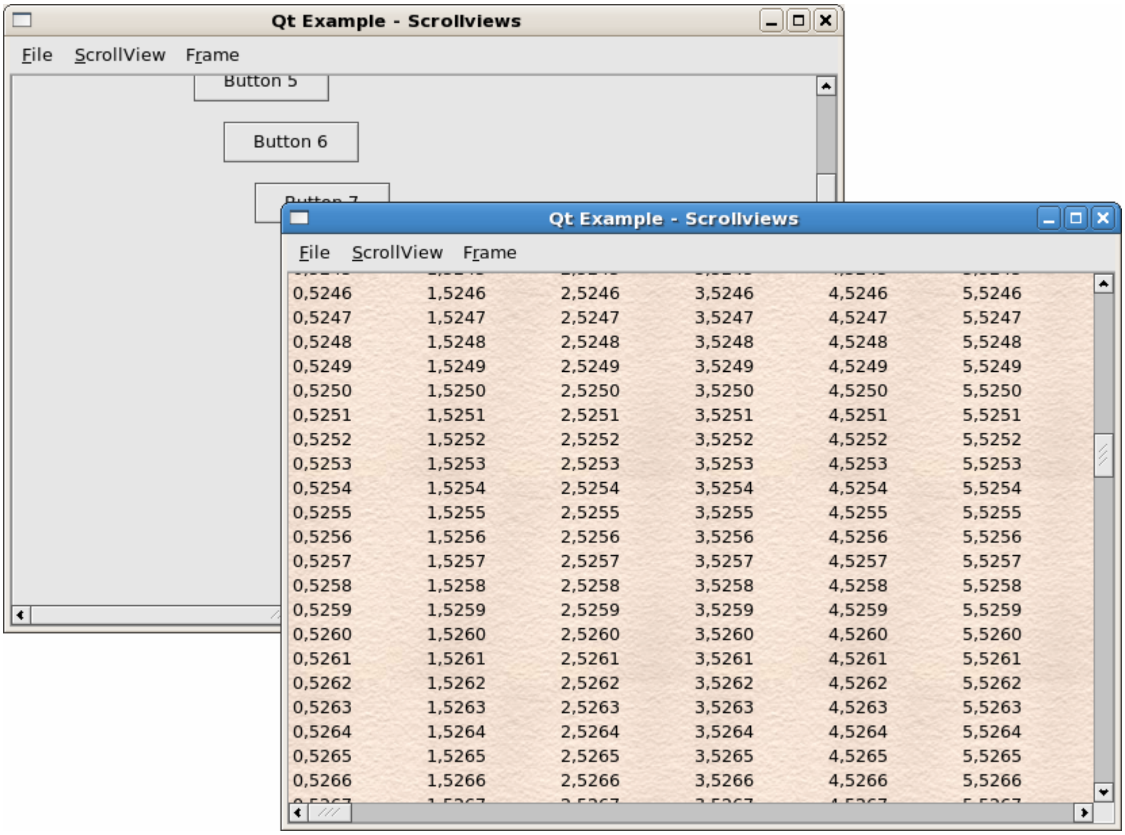
```
  if ( technique == 3 )
   vp->enableClipper(TRUE);
  srand(1);
  for (int i=0; i<30; i++) {
   QMultiLineEdit *l = new QMultiLineEdit(vp->viewport(),"First");
   l->setText("Drag out more of these.");
   l->resize(100,100);
   vp->addChild(l, rand()%800, rand()%10000);
  }
  vp->viewport()->setBackgroundMode(NoBackground);
}

f_options = new QPopupMenu( menubar );
Q_CHECK_PTR( f_options );
f_options->setCheckable( TRUE );
menubar->insertItem( "F&rame", f_options );
connect( f_options, SIGNAL(activated(int)),
  this, SLOT(doFMenuItem(int)) );

f_options->insertItem( "No Frame", style_id );
f_options->insertItem( "Box", style_id|QFrame::Box );
f_options->insertItem( "Panel", style_id|QFrame::Panel );
f_options->insertItem( "WinPanel", style_id|QFrame::WinPanel );
f_options->insertSeparator();
f_options->insertItem( "Plain", style_id|QFrame::Plain );
f_options->insertItem( "Raised", style_id|QFrame::Raised );
f_laststyle = f_options->indexOf(
  f_options->insertItem( "Sunken", style_id|QFrame::Sunken ));
f_options->insertSeparator();
lw_options = new QPopupMenu( menubar );
Q_CHECK_PTR( lw_options );
lw_options->setCheckable( TRUE );
for (int lw = 1; lw <= max_lw; lw++) {
  QString str;
  str.sprintf("%d Pixels", lw);
  lw_options->insertItem( str, lw_id | lw );
}
f_options->insertItem( "Line Width", lw_options );
connect( lw_options, SIGNAL(activated(int)),
  this, SLOT(doFMenuItem(int)) );
mlw_options = new QPopupMenu( menubar );
Q_CHECK_PTR( mlw_options );
mlw_options->setCheckable( TRUE );
for (int mlw = 0; mlw <= max_mlw; mlw++) {
  QString str;
  str.sprintf("%d Pixels", mlw);
  mlw_options->insertItem( str, mlw_id | mlw );
}
f_options->insertItem( "Midline Width", mlw_options );
connect( mlw_options, SIGNAL(activated(int)),
  this, SLOT(doFMenuItem(int)) );
mw_options = new QPopupMenu( menubar );
Q_CHECK_PTR( mw_options );
mw_options->setCheckable( TRUE );
```

```
   for (int mw = 0; mw <= max_mw; mw++) {
     QString str;
     str.sprintf("%d Pixels", mw);
     mw_options->insertItem( str, mw_id | mw );
   }
   f_options->insertItem( "Margin Width", mw_options );
   connect( mw_options, SIGNAL(activated(int)),
     this, SLOT(doFMenuItem(int)) );

   setVPMenuItems();
   setFMenuItems();

   QVBoxLayout* vbox = new QVBoxLayout(this);
   vbox->setMenuBar(menubar);
   menubar->setSeparator(QMenuBar::InWindowsStyle);
   vbox->addWidget(vp);
   vbox->activate();

   corner = new QSizeGrip(this);
   corner->hide();
  }

private slots:
  void doVPMenuItem(int id)
  {
   if (id == vauto_id ) {
     vp->setVScrollBarMode(QScrollView::Auto);
   } else if (id == vaoff_id) {
     vp->setVScrollBarMode(QScrollView::AlwaysOff);
   } else if (id == vaon_id) {
     vp->setVScrollBarMode(QScrollView::AlwaysOn);
   } else if (id == hauto_id) {
     vp->setHScrollBarMode(QScrollView::Auto);
   } else if (id == haoff_id) {
     vp->setHScrollBarMode(QScrollView::AlwaysOff);
   } else if (id == haon_id) {
     vp->setHScrollBarMode(QScrollView::AlwaysOn);
   } else if (id == corn_id) {
     bool corn = !vp->cornerWidget();
     vp->setCornerWidget(corn ? corner : 0);
   } else {
     return; // Not for us to process.
   }
   setVPMenuItems();
  }

  void setVPMenuItems()
  {
   QScrollView::ScrollBarMode vm = vp->vScrollBarMode();
   vp_options->setItemChecked( vauto_id, vm == QScrollView::Auto );
   vp_options->setItemChecked( vaoff_id, vm == QScrollView::AlwaysOff );
   vp_options->setItemChecked( vaon_id, vm == QScrollView::AlwaysOn );

   QScrollView::ScrollBarMode hm = vp->hScrollBarMode();
```

```
 vp_options->setItemChecked( hauto_id, hm == QScrollView::Auto );
 vp_options->setItemChecked( haoff_id, hm == QScrollView::AlwaysOff );
 vp_options->setItemChecked( haon_id, hm == QScrollView::AlwaysOn );

 vp_options->setItemChecked( corn_id, !!vp->cornerWidget() );
}

void doFMenuItem(int id)
{
 if (id & style_id) {
    int sty;

    if (id == style_id) {
     sty = 0;
    } else if (id & QFrame::MShape) {
     sty = vp->frameStyle()&QFrame::MShadow;
     sty = (sty ? sty : QFrame::Plain) | (id&QFrame::MShape);
    } else {
     sty = vp->frameStyle()&QFrame::MShape;
     sty = (sty ? sty : QFrame::Box) | (id&QFrame::MShadow);
    }
    vp->setFrameStyle(sty);
 } else if (id & lw_id) {
    vp->setLineWidth(id&~lw_id);
 } else if (id & mlw_id) {
    vp->setMidLineWidth(id&~mlw_id);
 } else {
    vp->setMargin(id&~mw_id);
 }

 vp->update();
 setFMenuItems();
}

void setFMenuItems()
{
 int sty = vp->frameStyle();

 f_options->setItemChecked( style_id, !sty );

 for (int i=1; i <= f_laststyle; i++) {
    int id = f_options->idAt(i);
    if (id & QFrame::MShape)
     f_options->setItemChecked( id,
        ((id&QFrame::MShape) == (sty&QFrame::MShape)) );
    else
     f_options->setItemChecked( id,
        ((id&QFrame::MShadow) == (sty&QFrame::MShadow)) );
 }

 for (int lw=1; lw<=max_lw; lw++)
    lw_options->setItemChecked( lw_id|lw, vp->lineWidth() == lw );

 for (int mlw=0; mlw<=max_mlw; mlw++)
```

```
          mlw_options->setItemChecked( mlw_id|mlw, vp->midLineWidth() == mlw );

      for (int mw=0; mw<=max_mw; mw++)
          mw_options->setItemChecked( mw_id|mw, vp->margin() == mw );
    }

private:
    QScrollView* vp;
    QPopupMenu* vp_options;
    QPopupMenu* f_options;
    QPopupMenu* lw_options;
    QPopupMenu* mlw_options;
    QPopupMenu* mw_options;
    QSizeGrip* corner;

    int vauto_id, vaoff_id, vaon_id,
      hauto_id, haoff_id, haon_id,
      corn_id;

    int f_laststyle;
};

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    ScrollViewExample ve1(1,0,"ve1");
    ScrollViewExample ve2(2,0,"ve2");
    ScrollViewExample ve3(3,0,"ve3");
    ve1.setCaption("Qt Example - Scrollviews");
    ve1.show();
    ve2.setCaption("Qt Example - Scrollviews");
    ve2.show();
    ve3.setCaption("Qt Example - Scrollviews");
    ve3.show();

    QObject::connect(qApp, SIGNAL(lastWindowClosed()), qApp, SLOT(quit()));

    return a.exec();
}

#include "scrollview.moc"
```

실행



```
0,5246    1,5246    2,5246    3,5246    4,5246    5,5246
0,5247    1,5247    2,5247    3,5247    4,5247    5,5247
0,5248    1,5248    2,5248    3,5248    4,5248    5,5248
0,5249    1,5249    2,5249    3,5249    4,5249    5,5249
0,5250    1,5250    2,5250    3,5250    4,5250    5,5250
0,5251    1,5251    2,5251    3,5251    4,5251    5,5251
0,5252    1,5252    2,5252    3,5252    4,5252    5,5252
0,5253    1,5253    2,5253    3,5253    4,5253    5,5253
0,5254    1,5254    2,5254    3,5254    4,5254    5,5254
0,5255    1,5255    2,5255    3,5255    4,5255    5,5255
0,5256    1,5256    2,5256    3,5256    4,5256    5,5256
0,5257    1,5257    2,5257    3,5257    4,5257    5,5257
0,5258    1,5258    2,5258    3,5258    4,5258    5,5258
0,5259    1,5259    2,5259    3,5259    4,5259    5,5259
0,5260    1,5260    2,5260    3,5260    4,5260    5,5260
0,5261    1,5261    2,5261    3,5261    4,5261    5,5261
0,5262    1,5262    2,5262    3,5262    4,5262    5,5262
0,5263    1,5263    2,5263    3,5263    4,5263    5,5263
0,5264    1,5264    2,5264    3,5264    4,5264    5,5264
0,5265    1,5265    2,5265    3,5265    4,5265    5,5265
0,5266    1,5266    2,5266    3,5266    4,5266    5,5266
```

56. 화상표시

이 실례는 제공하는 화상형식(GIF, BMP, PPM, XMP 등)으로 화상을 읽어들이고 보관한다.

showimg.pro

```
TEMPLATE  = app
TARGET    = showimg
CONFIG    += qt warn_on release
HEADERS       = showimg.h imagetexteditor.h \
          imagefip.h
SOURCES       = main.cpp \
          imagetexteditor.cpp \
          showimg.cpp \
          imagefip.cpp
```

imagefip.cpp

```
#include "imagefip.h"
#include <qimage.h>

/* XPM */
static const char *image_xpm[] = {
"17 15 9 1",
"   c #7F7F7F",
".  c #FFFFFF",
"X c #00B6FF",
"o  c #BFBFBF",
"O c #FF6C00",
"+  c #000000",
"@c #0000FF",
"#  c #6CFF00",
"$  c #FFB691",
"           ..XX",
" ........o   .XXX",
" .OOOOOOOo.  XXX+",
" .O@@@@@@+++XXX++",
" .O@@@@@@O.XXX+++",
" .O@@@@@@OXXX+++.",
" .O######XXX++...",
" .O#####XXX++....",
" .O##$#$XX+o+....",
" .O#$$$$$+.o+....",
" .O##$$##O.o+....",
" .OOOOOOOO.o+....",
" ..........o+....",
" ooooooooooo+....",
"+++++++++++++...."
};

ImageIconProvider::ImageIconProvider( QWidget *parent, const char *name ) :
   QFileIconProvider( parent, name ),
   imagepm(image_xpm)
{
   fmts = QImage::inputFormats();
}

ImageIconProvider::~ImageIconProvider()
{
}

const QPixmap * ImageIconProvider::pixmap( const QFileInfo &fi )
{
   QString ext = fi.extension().upper();
   if ( fmts.contains(ext) ) {
    return &imagepm;
   } else {
    return QFileIconProvider::pixmap(fi);
   }
}
```

imagefip.h

```
#ifndef IMAGEFIP_H
#define IMAGEFIP_H

#include <qfiledialog.h>
#include <qstrlist.h>
#include <qpixmap.h>

class ImageIconProvider : public QFileIconProvider
```

```
{
   Q_OBJECT
   QStrList fmts;
   QPixmap imagepm;

public:
   ImageIconProvider( QWidget *parent=0, const char *name=0 );
   ~ImageIconProvider();

   const QPixmap * pixmap( const QFileInfo &fi );
};

#endif // IMAGEFIP_H
```

imagetexteditor.cpp

```
#include "imagetexteditor.h"
#include <qimage.h>
#include <qlayout.h>
#include <qgrid.h>
#include <qvbox.h>
#include <qhbox.h>
#include <qcombobox.h>
#include <qmultilineedit.h>
#include <qlabel.h>
#include <qlineedit.h>
#include <qlistbox.h>
#include <qpushbutton.h>

ImageTextEditor::ImageTextEditor( QImage& i, QWidget *parent, const char *name, WFlags f ) :
   QDialog(parent,name,TRUE,f),
   image(i)
{
   QVBoxLayout* vbox = new QVBoxLayout(this,8);
   vbox->setAutoAdd(TRUE);

   QGrid* controls = new QGrid(3,QGrid::Horizontal,this);
   controls->setSpacing(8);
   QLabel* l;
   l=new QLabel("Language",controls); l->setAlignment(AlignCenter);
   l=new QLabel("Key",controls); l->setAlignment(AlignCenter);
   (void)new QLabel("",controls); // dummy
   languages = new QComboBox(controls);
   keys = new QComboBox(controls);
   QPushButton* remove = new QPushButton("Remove",controls);

   newlang = new QLineEdit(controls);
   newkey = new QLineEdit(controls);
   QPushButton* add = new QPushButton("Add",controls);

   text = new QMultiLineEdit(this);

   QHBox* hbox = new QHBox(this);
   QPushButton* cancel = new QPushButton("Cancel",hbox);
   QPushButton* ok = new QPushButton("OK",hbox);
```

```
  connect(add,SIGNAL(clicked()),
   this,SLOT(addText()));

  connect(remove,SIGNAL(clicked()),
   this,SLOT(removeText()));

  connect(ok,SIGNAL(clicked()),
   this,SLOT(accept()));

  connect(cancel,SIGNAL(clicked()),
   this,SLOT(reject()));

  connect(languages,SIGNAL(activated(int)),
   this,SLOT(updateText()));

  connect(keys,SIGNAL(activated(int)),
   this,SLOT(updateText()));

  imageChanged();
}

ImageTextEditor::~ImageTextEditor()
{
}

void ImageTextEditor::imageChanged()
{
  languages->clear();
  keys->clear();
  text->clear();
  languages->insertItem("<any>");

  languages->insertStringList(image.textLanguages());
  keys->insertStringList(image.textKeys());

  updateText();
}

void ImageTextEditor::accept()
{
  storeText();
  QDialog::accept();
}

void ImageTextEditor::updateText()
{
  storeText();
  newlang->setText(languages->currentText());
  newkey->setText(keys->currentText());
  QString t = image.text(currKey(),currLang());

  text->setText(t);
}
```

```
QString ImageTextEditor::currKey()
{
   return newkey->text();
}

QString ImageTextEditor::currLang()
{
   QString l = newlang->text();
   if ( l=="<any>" )
    l = QString::null;
   return l;
}

QString ImageTextEditor::currText()
{
   QString t = text->text();
   if ( t.isNull() ) t = "";
   return t;
}


void ImageTextEditor::removeText()
{
   image.setText(currKey(),currLang(),QString::null);
}

void ImageTextEditor::addText()
{
   storeText();
}

void ImageTextEditor::storeText()
{
   if ( currKey().length() > 0 ) {
    image.setText(currKey(),currLang(),currText());
   }
}
```

imagetexteditor.h

```
#ifndef IMAGETEXTEDITOR_H
#define IMAGETEXTEDITOR_H

#include <qdialog.h>

class QImage;
class QComboBox;
class QListBox;
class QLineEdit;
class QMultiLineEdit;

class ImageTextEditor : public QDialog
{
   Q_OBJECT
public:
   ImageTextEditor( QImage& i, QWidget *parent=0, const char *name=0, WFlags f=0 );
   ~ImageTextEditor();
   void accept();
public slots:
   void imageChanged();
   void updateText();
   void addText();
   void removeText();
private:
   void storeText();
   QImage& image;
   QComboBox* languages;
   QComboBox* keys;
   QMultiLineEdit* text;
   QLineEdit* newlang;
   QLineEdit* newkey;
   QString currKey();
   QString currLang();
   QString currText();
};

#endif // IMAGETEXTEDITOR_H
```

showimg.cpp

```
#include "showimg.h"
#include "imagetexteditor.h"
#include <qmenubar.h>
#include <qfiledialog.h>
#include <qmessagebox.h>
#include <qpopupmenu.h>
#include <qlabel.h>
#include <qpainter.h>
#include <qapplication.h>
#include <qclipboard.h>

/*
  In the constructor, we just pass the standard parameters on to
  QWidget.

  The menu uses a single slot to simplify the process of adding
  more items to the options menu.
*/
ImageViewer::ImageViewer( QWidget *parent, const char *name, int wFlags )
    : QWidget( parent, name, wFlags ),
      conversion_flags( PreferDither ),
      helpmsg( 0 )
{
    pickx = -1;
    picky = -1;
    clickx = -1;
    clicky = -1;
    alloc_context = 0;
```

```
menubar = new QMenuBar(this);
menubar->setSeparator( QMenuBar::InWindowsStyle );

QStrList fmt = QImage::outputFormats();
saveimage = new QPopupMenu( menubar );
savepixmap = new QPopupMenu( menubar );
for (const char* f = fmt.first(); f; f = fmt.next()) {
  saveimage->insertItem( f );
  savepixmap->insertItem( f );
}
connect( saveimage, SIGNAL(activated(int)), this, SLOT(saveImage(int)) );
connect( savepixmap, SIGNAL(activated(int)), this, SLOT(savePixmap(int)) );

file = new QPopupMenu( menubar );
menubar->insertItem( "&File", file );
file->insertItem( "&New window", this,  SLOT(newWindow()), CTRL+Key_N );
file->insertItem( "&Open...", this,  SLOT(openFile()), CTRL+Key_O );
si = file->insertItem( "Save image", saveimage );
sp = file->insertItem( "Save pixmap", savepixmap );
file->insertSeparator();
file->insertItem( "E&xit", qApp,  SLOT(quit()), CTRL+Key_Q );

edit =  new QPopupMenu( menubar );
menubar->insertItem( "&Edit", edit );
edit->insertItem("&Copy", this, SLOT(copy()), CTRL+Key_C);
edit->insertItem("&Paste", this, SLOT(paste()), CTRL+Key_V);
edit->insertSeparator();
edit->insertItem("&Horizontal flip", this, SLOT(hFlip()), ALT+Key_H);
edit->insertItem("&Vertical flip", this, SLOT(vFlip()), ALT+Key_V);
edit->insertItem("&Rotate 180", this, SLOT(rot180()), ALT+Key_R);
edit->insertSeparator();
edit->insertItem("&Text...", this, SLOT(editText()));
edit->insertSeparator();
t1 = edit->insertItem( "Convert to &1 bit", this, SLOT(to1Bit()) );
t8 = edit->insertItem( "Convert to &8 bit", this, SLOT(to8Bit()) );
t32 = edit->insertItem( "Convert to &32 bit", this, SLOT(to32Bit()) );

options =  new QPopupMenu( menubar );
menubar->insertItem( "&Options", options );
ac = options->insertItem( "AutoColor" );
co = options->insertItem( "ColorOnly" );
mo = options->insertItem( "MonoOnly" );
options->insertSeparator();
fd = options->insertItem( "DiffuseDither" );
bd = options->insertItem( "OrderedDither" );
td = options->insertItem( "ThresholdDither" );
options->insertSeparator();
ta = options->insertItem( "ThresholdAlphaDither" );
ba = options->insertItem( "OrderedAlphaDither" );
fa = options->insertItem( "DiffuseAlphaDither" );
options->insertSeparator();
ad = options->insertItem( "PreferDither" );
dd = options->insertItem( "AvoidDither" );
options->insertSeparator();
```

```
    ss = options->insertItem( "Smooth scaling" );
    cc = options->insertItem( "Use color context" );
    if ( QApplication::colorSpec() == QApplication::ManyColor )
      options->setItemEnabled( cc, FALSE );
    options->setCheckable( TRUE );
    setMenuItemFlags();

    menubar->insertSeparator();

    QPopupMenu* help = new QPopupMenu( menubar );
    menubar->insertItem( "&Help", help );
    help->insertItem( "Help!", this, SLOT(giveHelp()), CTRL+Key_H );

    connect( options, SIGNAL(activated(int)), this, SLOT(doOption(int)) );

    status = new QLabel(this);
    status->setFrameStyle( QFrame::WinPanel | QFrame::Sunken );
    status->setFixedHeight( fontMetrics().height() + 4 );

    setMouseTracking( TRUE );
}

ImageViewer::~ImageViewer()
{
    if ( alloc_context )
      QColor::destroyAllocContext( alloc_context );
    if ( other == this )
      other = 0;
}

/*
  This function modifies the conversion_flags when an options menu item
  is selected, then ensures all menu items are up to date, and reconverts
  the image if possibly necessary.
*/
void ImageViewer::doOption(int item)
{
    if ( item == ss || item == cc ) {
      // Toggle
      bool newbool = !options->isItemChecked(item);
      options->setItemChecked(item, newbool);
      // And reconvert...
      reconvertImage();
      repaint(image.hasAlphaBuffer());   // show image in widget
      return;
    }

    if ( options->isItemChecked( item ) ) return; // They are all radio buttons

    int ocf = conversion_flags;

    if ( item == ac ) {
      conversion_flags = ( conversion_flags & ~ColorMode_Mask ) | AutoColor;
    } else if ( item == co ) {
      conversion_flags = ( conversion_flags & ~ColorMode_Mask ) | ColorOnly;
    } else if ( item == mo ) {
      conversion_flags = ( conversion_flags & ~ColorMode_Mask ) | MonoOnly;
    } else if ( item == fd ) {
      conversion_flags = ( conversion_flags & ~Dither_Mask ) | DiffuseDither;
    } else if ( item == bd ) {
      conversion_flags = ( conversion_flags & ~Dither_Mask ) | OrderedDither;
    } else if ( item == td ) {
      conversion_flags = ( conversion_flags & ~Dither_Mask ) | ThresholdDither;
    } else if ( item == ta ) {
      conversion_flags = ( conversion_flags & ~AlphaDither_Mask ) | ThresholdAlphaDither;
    } else if ( item == fa ) {
      conversion_flags = ( conversion_flags & ~AlphaDither_Mask ) | DiffuseAlphaDither;
    } else if ( item == ba ) {
      conversion_flags = ( conversion_flags & ~AlphaDither_Mask ) | OrderedAlphaDither;
    } else if ( item == ad ) {
      conversion_flags = ( conversion_flags & ~DitherMode_Mask ) | PreferDither;
    } else if ( item == dd ) {
      conversion_flags = ( conversion_flags & ~DitherMode_Mask ) | AvoidDither;
    }

    if ( ocf != conversion_flags ) {
      setMenuItemFlags();
      // And reconvert...
      reconvertImage();
      repaint(image.hasAlphaBuffer());   // show image in widget
    }
}

/*
  Set the options menu to reflect the conversion_flags value.
*/
void ImageViewer::setMenuItemFlags()
{
    // File
    bool valid_image = pm.size() != QSize( 0, 0 );
    file->setItemEnabled( si, valid_image );
    file->setItemEnabled( sp, valid_image );

    // Edit
    edit->setItemEnabled( t1, image.depth() != 1 );
    edit->setItemEnabled( t8, image.depth() != 8 );
    edit->setItemEnabled( t32, image.depth() != 32 );

    // Options
    bool may_need_color_dithering =
         !valid_image
      || image.depth() == 32 && QPixmap::defaultDepth() < 24;
    bool may_need_dithering = may_need_color_dithering
       || image.depth() > 1 && options->isItemChecked(mo)
       || image.depth() > 1 && QPixmap::defaultDepth() == 1;
    bool has_alpha_mask = !valid_image || image.hasAlphaBuffer();

    options->setItemEnabled( fd, may_need_dithering );
    options->setItemEnabled( bd, may_need_dithering );
    options->setItemEnabled( td, may_need_dithering );

    options->setItemEnabled( ta, has_alpha_mask );
    options->setItemEnabled( fa, has_alpha_mask );
    options->setItemEnabled( ba, has_alpha_mask );

    options->setItemEnabled( ad, may_need_color_dithering );
    options->setItemEnabled( dd, may_need_color_dithering );

    options->setItemChecked( ac, (conversion_flags & ColorMode_Mask) == AutoColor );
    options->setItemChecked( co, (conversion_flags & ColorMode_Mask) == ColorOnly );
    options->setItemChecked( mo, (conversion_flags & ColorMode_Mask) == MonoOnly );
    options->setItemChecked( fd, (conversion_flags & Dither_Mask) == DiffuseDither );
    options->setItemChecked( bd, (conversion_flags & Dither_Mask) == OrderedDither );
    options->setItemChecked( td, (conversion_flags & Dither_Mask) == ThresholdDither );
    options->setItemChecked( ta, (conversion_flags & AlphaDither_Mask) == ThresholdAlphaDither );
    options->setItemChecked( fa, (conversion_flags & AlphaDither_Mask) == DiffuseAlphaDither );
    options->setItemChecked( ba, (conversion_flags & AlphaDither_Mask) == OrderedAlphaDither );
    options->setItemChecked( ad, (conversion_flags & DitherMode_Mask) == PreferDither );
    options->setItemChecked( dd, (conversion_flags & DitherMode_Mask) == AvoidDither );
}

void ImageViewer::updateStatus()
{
    if ( pm.size() == QSize( 0, 0 ) ) {
      if ( !filename.isEmpty() )
          status->setText("Could not load image");
      else
          status->setText("No image - select Open from File menu.");
    } else {
      QString message, moremsg;
      message.sprintf("%dx%d", image.width(), image.height());
      if ( pm.size() != pmScaled.size() ) {
          moremsg.sprintf(" [%dx%d]", pmScaled.width(),
            pmScaled.height());
          message += moremsg;
      }
      moremsg.sprintf(", %d bits ", image.depth());
      message += moremsg;
      if (image.valid(pickx,picky)) {
          moremsg.sprintf("(%d,%d)=#%0*x ",
                pickx, picky,
                image.hasAlphaBuffer() ? 8 : 6,
                image.pixel(pickx,picky));
          message += moremsg;
      }
      if ( image.numColors() > 0 ) {
          if (image.valid(pickx,picky)) {
            moremsg.sprintf(", %d/%d colors", image.pixelIndex(pickx,picky),
                image.numColors());
          } else {
            moremsg.sprintf(", %d colors", image.numColors());
          }
```

```
     message += moremsg;
   }
   if ( image.hasAlphaBuffer() ) {
     if ( image.depth() == 8 ) {
      int i;
      bool alpha[256];
      int nalpha=0;

      for (i=0; i<256; i++)
        alpha[i] = FALSE;

      for (i=0; i<image.numColors(); i++) {
        int alevel = image.color(i) >> 24;
        if (!alpha[alevel]) {
         alpha[alevel] = TRUE;
         nalpha++;
        }
      }
      moremsg.sprintf(", %d alpha levels", nalpha);
     } else {
      // Too many pixels to bother counting.
      moremsg = ", 8-bit alpha channel";
     }
     message += moremsg;
   }
   status->setText(message);
  }
}

/*
 This function saves the image.
*/
void ImageViewer::saveImage( int item )
{
  const char* fmt = saveimage->text(item);
  QString savefilename = QFileDialog::getSaveFileName(QString::null, QString::null,
               this, filename);
  if ( !savefilename.isEmpty() )
   if ( !image.save( savefilename, fmt ) )
     QMessageBox::warning( this, "Save failed", "Error saving file" );
}

/*
 This function saves the converted image.
*/
void ImageViewer::savePixmap( int item )
{
  const char* fmt = savepixmap->text(item);
  QString savefilename = QFileDialog::getSaveFileName(QString::null,
               QString::null, this, filename);
  if ( !savefilename.isEmpty() )
   if ( !pmScaled.save( savefilename, fmt ) )
     QMessageBox::warning( this, "Save failed", "Error saving file" );
}
```

```
void ImageViewer::newWindow()
{
   ImageViewer* that = new ImageViewer(0, "new window", WDestructiveClose);
   that->options->setItemChecked( that->cc, useColorContext() );
   that->show();
}

/*
  This function is the slot for processing the Open menu item.
*/
void ImageViewer::openFile()
{
   QString newfilename = QFileDialog::getOpenFileName( QString::null,
                               QString::null,
                               this );
   if ( !newfilename.isEmpty() ) {
    loadImage( newfilename ) ;
    repaint();          // show image in widget
   }
}

/*
  This function loads an image from a file and resizes the widget to
  exactly fit the image size. If the file was not found or the image
  format was unknown it will resize the widget to fit the errorText
  message (see above) displayed in the current font.

  Returns TRUE if the image was successfully loaded.
*/

bool ImageViewer::loadImage( const QString& fileName )
{
   filename = fileName;
   bool ok = FALSE;
   if ( !filename.isEmpty() ) {
    QApplication::setOverrideCursor( waitCursor ); // this might take time
    ok = image.load(filename, 0);
    pickx = -1;
    clickx = -1;
    if ( ok )
       ok = reconvertImage();
    if ( ok ) {
       setCaption( filename );           // set window caption
       int w = pm.width();
       int h = pm.height();

       const int reasonable_width = 128;
       if ( w < reasonable_width ) {
        // Integer scale up to something reasonable
        int multiply = ( reasonable_width + w - 1 ) / w;
        w *= multiply;
        h *= multiply;
       }
```

```
        h += menubar->heightForWidth(w) + status->height();
        resize( w, h );               // we resize to fit image
      } else {
        pm.resize(0,0);               // couldn't load image
        update();
      }
      QApplication::restoreOverrideCursor();    // restore original cursor
    }
    updateStatus();
    setMenuItemFlags();
    return ok;
}

bool ImageViewer::reconvertImage()
{
    bool success = FALSE;

    if ( image.isNull() ) return FALSE;

    if ( alloc_context ) {
      QColor::destroyAllocContext( alloc_context );
      alloc_context = 0;
    }
    if ( useColorContext() ) {
      alloc_context = QColor::enterAllocContext();
      // Clear the image to hide flickering palette
      QPainter painter(this);
      painter.eraseRect(0, menubar->heightForWidth( width() ), width(), height());
    }

    QApplication::setOverrideCursor( waitCursor ); // this might take time
    if ( pm.convertFromImage(image, conversion_flags) )
    {
      pmScaled = QPixmap();
      scale();
      resize( width(), height() );
      success = TRUE;               // load successful
    } else {
      pm.resize(0,0);               // couldn't load image
    }
    updateStatus();
    setMenuItemFlags();
    QApplication::restoreOverrideCursor();// restore original cursor

    if ( useColorContext() )
      QColor::leaveAllocContext();

    return success;              // TRUE if loaded OK
}

bool ImageViewer::smooth() const
{
    return options->isItemChecked(ss);
```

```
}

bool ImageViewer::useColorContext() const
{
   return options->isItemChecked(cc);
}

/*
  This functions scales the pixmap in the member variable "pm" to fit the
  widget size and  puts the resulting pixmap in the member variable "pmScaled".
*/

void ImageViewer::scale()
{
   int h = height() - menubar->heightForWidth( width() ) - status->height();

   if ( image.isNull() ) return;

   QApplication::setOverrideCursor( waitCursor ); // this might take time
   if ( width() == pm.width() && h == pm.height() )
   {                              // no need to scale if widget
    pmScaled = pm;                // size equals pixmap size
   } else {
    if (smooth()) {
       pmScaled.convertFromImage(image.smoothScale(width(), h),
        conversion_flags);
    } else {
       QWMatrix m;                // transformation matrix
       m.scale(((double)width())/pm.width(),// define scale factors
           ((double)h)/pm.height());
       pmScaled = pm.xForm( m );       // create scaled pixmap
    }
   }
   QApplication::restoreOverrideCursor();// restore original cursor
}

/*
  The resize event handler, if a valid pixmap was loaded it will call
  scale() to fit the pixmap to the new widget size.
*/

void ImageViewer::resizeEvent( QResizeEvent * )
{
   status->setGeometry(0, height() - status->height(),
            width(), status->height());

   if ( pm.size() == QSize( 0, 0 ) )        // we couldn't load the image
    return;

   int h = height() - menubar->heightForWidth( width() ) - status->height();
   if ( width() != pmScaled.width() || h != pmScaled.height())
   {                              // if new size,
    scale();                      // scale pmScaled to window
    updateStatus();
    }
    if ( image.hasAlphaBuffer() )
      erase();
}

bool ImageViewer::convertEvent( QMouseEvent* e, int& x, int& y)
{
    if ( pm.size() != QSize( 0, 0 ) ) {
      int h = height() - menubar->heightForWidth( width() ) - status->height();
      int nx = e->x() * image.width() / width();
      int ny = (e->y()-menubar->heightForWidth( width() )) * image.height() / h;
      if (nx != x || ny != y ) {
          x = nx;
          y = ny;
          updateStatus();
          return TRUE;
      }
    }
    return FALSE;
}

void ImageViewer::mousePressEvent( QMouseEvent *e )
{
    may_be_other = convertEvent(e, clickx, clicky);
}

void ImageViewer::mouseReleaseEvent( QMouseEvent * )
{
    if ( may_be_other )
      other = this;
}

/*
  Record the pixel position of interest.
*/
void ImageViewer::mouseMoveEvent( QMouseEvent *e )
{
    if (convertEvent(e,pickx,picky)) {
      updateStatus();
      if ((e->state()&LeftButton)) {
          may_be_other = FALSE;
          if ( clickx >= 0 && other) {
            copyFrom(other);
          }
      }
    }
}

/*
  Draws the portion of the scaled pixmap that needs to be updated or prints
  an error message if no legal pixmap has been loaded.
*/

void ImageViewer::paintEvent( QPaintEvent *e )
```

```
{
   if ( pm.size() != QSize( 0, 0 ) ) {      // is an image loaded?
    QPainter painter(this);
    painter.setClipRect(e->rect());
    painter.drawPixmap(0, menubar->heightForWidth( width() ), pmScaled);
   }
}

/*
  Explain anything that might be confusing.
*/
void ImageViewer::giveHelp()
{
   if (!helpmsg) {
    QString helptext =
       "<b>Usage:</b> <tt>showimg [-m] <i>filename ...</i></tt>"
       "<blockquote>"
        "<tt>-m</tt> - use <i>ManyColor</i> color spec"
       "</blockquote>"
       "<p>Supported input formats:"
       "<blockquote>";
    helptext += QImage::inputFormatList().join(", ");
    helptext += "</blockquote>";

    helpmsg = new QMessageBox( "Help", helptext,
       QMessageBox::Information, QMessageBox::Ok, 0, 0, 0, 0, FALSE );
   }
   helpmsg->show();
   helpmsg->raise();
}

void ImageViewer::copyFrom(ImageViewer* s)
{
   if ( clickx >= 0 ) {
    int dx = clickx;
    int dy = clicky;
    int sx = s->clickx;
    int sy = s->clicky;
    int sw = QABS(clickx - pickx)+1;
    int sh = QABS(clicky - picky)+1;
    if ( clickx > pickx ) {
       dx = pickx;
       sx -= sw-1;
    }
    if ( clicky > picky ) {
       dy = picky;
       sy -= sh-1;
    }
    bitBlt( &image, dx, dy, &s->image, sx, sy, sw, sh );
    reconvertImage();
    repaint( image.hasAlphaBuffer() );
   }
}
ImageViewer* ImageViewer::other = 0;
```

```
void ImageViewer::hFlip()
{
   setImage(image.mirror(TRUE,FALSE));
}

void ImageViewer::vFlip()
{
   setImage(image.mirror(FALSE,TRUE));
}

void ImageViewer::rot180()
{
   setImage(image.mirror(TRUE,TRUE));
}

void ImageViewer::copy()
{
#ifndef QT_NO_MIMECLIPBOARD
   QApplication::clipboard()->setImage(image); // Less information loss
#endif
}

void ImageViewer::paste()
{
#ifndef QT_NO_MIMECLIPBOARD
   QImage p = QApplication::clipboard()->image();
   if ( !p.isNull() ) {
    filename = "pasted";
    setImage(p);
   }
#endif
}

void ImageViewer::setImage(const QImage& newimage)
{
   image = newimage;

   pickx = -1;
   clickx = -1;
   setCaption( filename );                // set window caption
   int w = image.width();
   int h = image.height();
   if ( !w )
    return;

   const int reasonable_width = 128;
   if ( w < reasonable_width ) {
    // Integer scale up to something reasonable
    int multiply = ( reasonable_width + w - 1 ) / w;
    w *= multiply;
    h *= multiply;
   }
```

```
    h += menubar->heightForWidth(w) + status->height();
    resize( w, h );                    // we resize to fit image

    reconvertImage();
    repaint( image.hasAlphaBuffer() );

    updateStatus();
    setMenuItemFlags();
}

void ImageViewer::editText()
{
    ImageTextEditor editor(image,this);
    editor.exec();
}

void ImageViewer::to1Bit()
{
    toBitDepth(1);
}

void ImageViewer::to8Bit()
{
    toBitDepth(8);
}

void ImageViewer::to32Bit()
{
    toBitDepth(32);
}

void ImageViewer::toBitDepth(int d)
{
    image = image.convertDepth(d);
    reconvertImage();
    repaint( image.hasAlphaBuffer() );
}
```

showimg.h

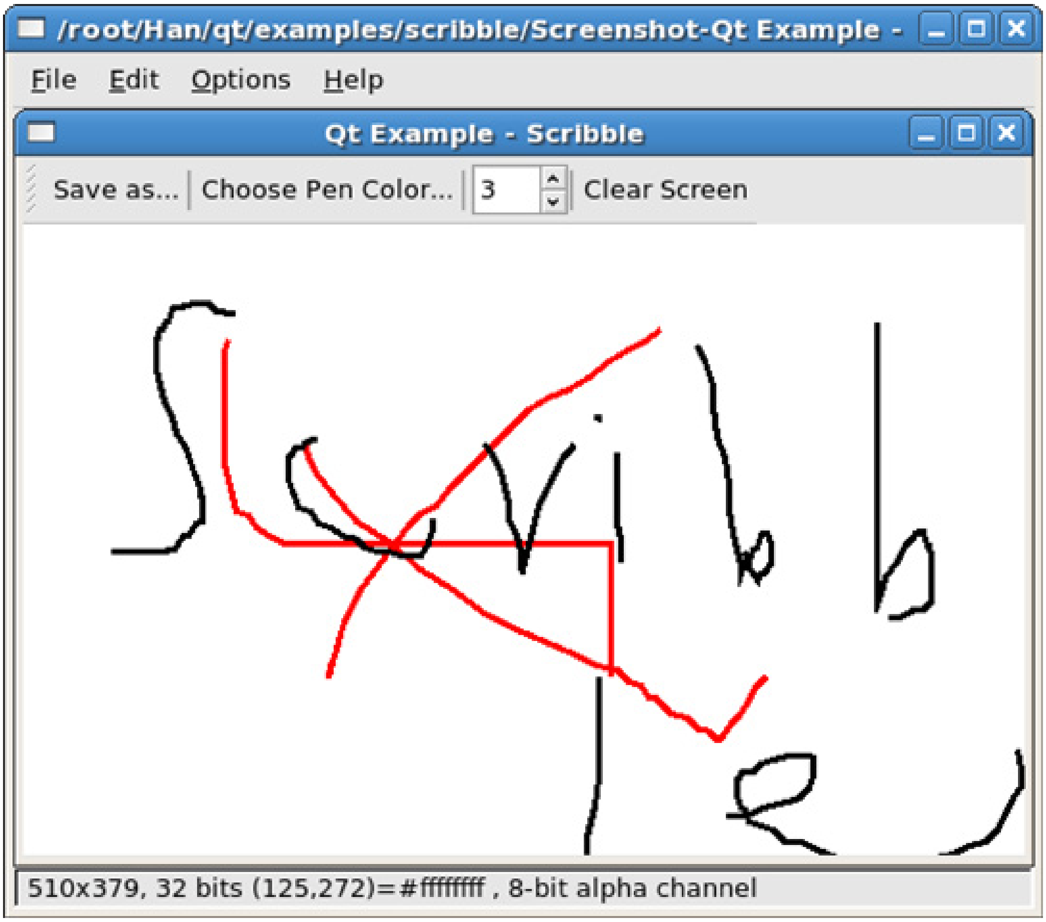
```
#ifndef SHOWIMG_H
#define SHOWIMG_H
#include <qwidget.h>
#include <qimage.h>

class QLabel;
class QMenuBar;
class QPopupMenu;

class ImageViewer : public QWidget
{
    Q_OBJECT
public:
    ImageViewer( QWidget *parent=0, const char *name=0, int wFlags=0 );
    ~ImageViewer();
    boolloadImage( const QString& );
protected:
    voidpaintEvent( QPaintEvent * );
    voidresizeEvent( QResizeEvent * );
    voidmousePressEvent( QMouseEvent * );
    voidmouseReleaseEvent( QMouseEvent * );
    voidmouseMoveEvent( QMouseEvent * );

private:
    voidscale();
    int         conversion_flags;
    boolsmooth() const;
    booluseColorContext() const;
    int         alloc_context;
    boolconvertEvent( QMouseEvent* e, int& x, int& y );
    QString     filename;
    QImageimage;            // the loaded image
    QPixmap    pm;               // the converted pixmap
    QPixmap    pmScaled;      // the scaled pixmap

    QMenuBar   *menubar;
    QPopupMenu  *file;
    QPopupMenu   *saveimage;
    QPopupMenu   *savepixmap;
    QPopupMenu  *edit;
    QPopupMenu  *options;

    QWidget    *helpmsg;
    QLabel     *status;
    int         si, sp, ac, co, mo, fd, bd, // Menu item ids
          td, ta, ba, fa, au, ad, dd,
          ss, cc, t1, t8, t32;
    voidupdateStatus();
    voidsetMenuItemFlags();
    bool      reconvertImage();
    int         pickx, picky;
    int         clickx, clicky;
    boolmay_be_other;
    static ImageViewer* other;
    voidsetImage(const QImage& newimage);

private slots:
    voidto1Bit();
    voidto8Bit();
    voidto32Bit();
    voidtoBitDepth(int);

    voidcopy();
    voidpaste();

    voidhFlip();
    voidvFlip();
    voidrot180();
    voideditText();

    voidnewWindow();
    voidopenFile();
    voidsaveImage(int);
    voidsavePixmap(int);
    voidgiveHelp();
    voiddoOption(int);
    voidcopyFrom(ImageViewer*);
};

#endif // SHOWIMG_H
```

main.cpp

```
#include "showimg.h"
#include "imagefip.h"
#include <qapplication.h>
#include <qimage.h>

int main( int argc, char **argv )
{
    if ( argc > 1 && QString(argv[1]) == "-m" ) {
     QApplication::setColorSpec( QApplication::ManyColor );
     argc--;
     argv++;
    }
    else if ( argc > 1 && QString(argv[1]) == "-n" ) {
     QApplication::setColorSpec( QApplication::NormalColor );
     argc--;
     argv++;
    }
    else {
     QApplication::setColorSpec( QApplication::CustomColor );
    }

    QApplication a( argc, argv );

    ImageIconProvider iip;
    QFileDialog::setIconProvider( &iip );

    if ( argc <= 1 ) {
     // Create a window which looks after its own existence.
     ImageViewer *w =
         new ImageViewer(0, "new window", Qt::WDestructiveClose | Qt::WResizeNoErase );
     w->setCaption("Qt Example - Image Viewer");
     w->show();
    } else {
     for ( int i=1; i<argc; i++ ) {
         // Create a window which looks after its own existence.
         ImageViewer *w =
           new ImageViewer(0, argv[i], Qt::WDestructiveClose | Qt::WResizeNoErase );
         w->setCaption("Qt Example - Image Viewer");
         w->loadImage( argv[i] );
         w->show();
```

```
      }
    }

    QObject::connect(qApp, SIGNAL(lastWindowClosed()), qApp, SLOT(quit()));

    return a.exec();
}
```

실행



510x379, 32 bits (125,272)=#ffffffff , 8-bit alpha channel

57. QFont 성원함수들의 간단한 보여주기

이 실례프로그람은 여러가지 QFont성원함수들의 사용법을 보여준다.

simple-qfont-demo.pro

```
TEMPLATE  = app
TARGET    = fontdemo
CONFIG    += qt warn_on release
HEADERS       = viewer.h
SOURCES       = simple-qfont-demo.cpp \
          viewer.cpp
```

viewer.cpp

```
#include "viewer.h"
```

```
#include <qstring.h>
#include <qstringlist.h>
#include <qtextview.h>
#include <qpushbutton.h>
#include <qlayout.h>

Viewer::Viewer()
      :QWidget()
{
   setFontSubstitutions();

   QString greeting_heb = QString::fromUtf8( "\327\251\327\234\327\225\327\235" );
   QString greeting_ru =
QString::fromUtf8( "\320\227\320\264\321\200\320\260\320\262\321\201\321\202\320\262\321\203\320\271\321\202\320\265" );
   QString greeting_en( "Hello" );

   greetings = new QTextView( this, "textview" );

   greetings->setText( greeting_en + "\n" + greeting_ru + "\n" + greeting_heb );

   fontInfo = new QTextView( this, "fontinfo" );

   setDefault();

   defaultButton = new QPushButton( "Default", this, "pushbutton1" );
   defaultButton->setFont( QFont( "times" ) );
   connect( defaultButton, SIGNAL( clicked() ), this, SLOT( setDefault() ) );

   sansSerifButton = new QPushButton( "Sans Serif", this, "pushbutton2" );
   sansSerifButton->setFont( QFont( "Helvetica", 12 ) );
   connect( sansSerifButton, SIGNAL( clicked() ), this, SLOT( setSansSerif() ) );

   italicsButton = new QPushButton( "Italics", this, "pushbutton3" );
   italicsButton->setFont( QFont( "lucida", 12, QFont::Bold, TRUE ) );
   connect( italicsButton, SIGNAL( clicked() ), this, SLOT( setItalics() ) );

   layout();
}

void Viewer::setDefault()
{
   QFont font( "Bavaria" );
   font.setPointSize( 24 );

   font.setWeight( QFont::Bold );
   font.setUnderline( TRUE );

   greetings->setFont( font );

   showFontInfo( font );
}

void Viewer::setSansSerif()
```

```
{
   QFont font( "Newyork", 18 );
   font.setStyleHint( QFont::SansSerif );
   greetings->setFont( font );

   showFontInfo( font );
}

void Viewer::setItalics()
{
   QFont font( "Tokyo" );
   font.setPointSize( 32 );
   font.setWeight( QFont::Bold );
   font.setItalic( TRUE );

   greetings->setFont( font );

   showFontInfo( font );
}

void Viewer::showFontInfo( QFont & font )
{
   QFontInfo info( font );

   QString messageText;
   messageText = "Font requested: \"" + font.family() + "\" " +
              QString::number( font.pointSize() ) + "pt<BR>" + "Font used: \"" +
              info.family() + "\" " + QString::number( info.pointSize() ) + "pt<P>";

   QStringList substitutions = QFont::substitutes( font.family() );

   if ( ! substitutions.isEmpty() ){
    messageText += "The following substitutions exist for " + \
              font.family() + ":<UL>";

    QStringList::Iterator i = substitutions.begin();
    while ( i != substitutions.end() ){
       messageText += "<LI>\"" + (* i) + "\"";
       i++;
    }
     messageText += "</UL>";
   } else {
    messageText += "No substitutions exist for " +  font.family() + ".";
   }

   fontInfo->setText( messageText );
}

void Viewer::setFontSubstitutions()
{
   QStringList substitutes;
   substitutes.append( "Times" );
   substitutes +=  "Mincho",
   substitutes << "Arabic Newspaper" << "crox";
   QFont::insertSubstitutions( "Bavaria", substitutes );

   QFont::insertSubstitution( "Tokyo", "Lucida" );
}

// For those who prefer to use Qt Designer for creating GUIs
// the following function might not be of particular interest:
// all it does is creating the widget layout.

void Viewer::layout()
{
   QHBoxLayout * textViewContainer = new QHBoxLayout();
   textViewContainer->addWidget( greetings );
   textViewContainer->addWidget( fontInfo );

   QHBoxLayout * buttonContainer = new QHBoxLayout();

   buttonContainer->addWidget( defaultButton );
   buttonContainer->addWidget( sansSerifButton );
   buttonContainer->addWidget( italicsButton );

   int maxButtonHeight = defaultButton->height();

   if ( sansSerifButton->height() > maxButtonHeight )
     maxButtonHeight = sansSerifButton->height();
   if ( italicsButton->height() > maxButtonHeight )
       maxButtonHeight = italicsButton->height();

   defaultButton->setFixedHeight( maxButtonHeight );
   sansSerifButton->setFixedHeight( maxButtonHeight );
   italicsButton->setFixedHeight( maxButtonHeight );

   QVBoxLayout * container = new QVBoxLayout( this );
   container->addLayout( textViewContainer );
   container->addLayout( buttonContainer );

   resize( 700, 250 );
}
```

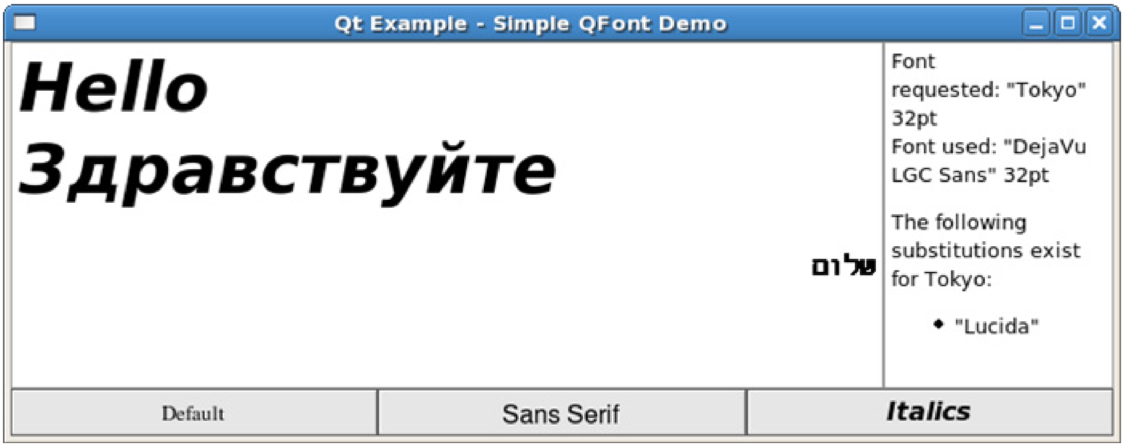
viewer.h

```
#ifndef VIEWER_H
#define VIEWER_H

#include <qwidget.h>
#include <qfont.h>

class QTextView;
class QPushButton;

class Viewer : public QWidget
{
Q_OBJECT
public:
   Viewer();

private slots:
   void setDefault();
   void setSansSerif();
   void setItalics();

private:
   void setFontSubstitutions();
   void layout();
   void showFontInfo( QFont & );

   QTextView * greetings;
   QTextView * fontInfo;

   QPushButton * defaultButton;
   QPushButton * sansSerifButton;
   QPushButton * italicsButton;
};

#endif
```

simple-qfont-demo.cpp

```
#include "viewer.h"
#include <qapplication.h>

int main( int argc, char **argv )
{
   QApplication app( argc, argv );
   Viewer * textViewer = new Viewer();
   textViewer->setCaption( "Qt Example - Simple QFont Demo" );
   app.setMainWidget( textViewer );
   textViewer->show();
   return app.exec();
}
```

실 행



Hello
Здравствуйте
שלום

Font requested: "Tokyo" 32pt
Font used: "DejaVu LGC Sans" 32pt

The following substitutions exist for Tokyo:

- "Lucida"

58. 음성실례

이 실례는 자기의 콤퓨터가 음성을 연주하도록 설정되여있으면 .WAV파일과 같은 음성파일을 재생하는 간단한 방법을 보여준다.

sound.pro

```
TEMPLATE  = app
TARGET    = sound
CONFIG    += qt warn_on release
x11:REQUIRES  = nas
HEADERS       = sound.h
SOURCES       = sound.cpp
```

sound.cpp

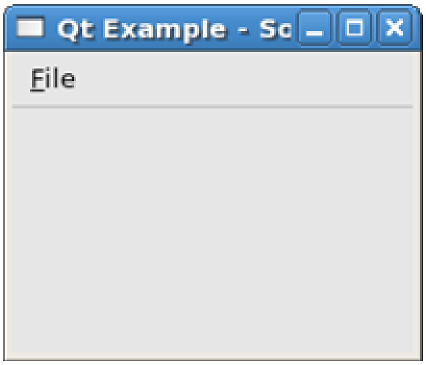
```
// Very simple example of QSound::play(filename)
// 99% of this program is just boilerplate Qt code to put up a nice
// window so you think something special is happening.
#include "sound.h"
#include <qapplication.h>
#include <qmessagebox.h>
#include <qmenubar.h>

SoundPlayer::SoundPlayer() :
   QMainWindow(),
   bucket3("sounds/3.wav"),
   bucket4("sounds/4.wav")
{
   if (!QSound::isAvailable()) {
    // Bail out.  Programs in which sound is not critical
    // could just silently (hehe) ignore the lack of a server.
    QMessageBox::warning(this,"No Sound",
         "<p><b>Sorry, you are not running the Network Audio System.</b>"
         "<p>If you have the `au' command, run it in the background before this program. "
         "The latest release of the Network Audio System can be obtained from:"
         "<pre>\n"
         " &nbsp;\n"
         "   ftp.ncd.com:/pub/ncd/technology/src/nas\n"
         "   ftp.x.org:/contrib/audio/nas\n"
         "</pre>"
         "<p>Release 1.2 of NAS is also included with the X11R6"
         "contrib distribution."
         "<p>After installing NAS, you will then need to reconfigure Qt with NAS sound support");
   }

   QPopupMenu *file = new QPopupMenu;
   file->insertItem("Play &1",  this, SLOT(doPlay1()), CTRL+Key_1);
   file->insertItem("Play &2",  this, SLOT(doPlay2()), CTRL+Key_2);
   file->insertItem("Play from bucket &3",  this, SLOT(doPlay3()), CTRL+Key_3);
   file->insertItem("Play from bucket &4",  this, SLOT(doPlay4()), CTRL+Key_4);
   file->insertSeparator();
   file->insertItem("Play 3 and 4 together",  this, SLOT(doPlay34()));
   file->insertItem("Play all together",  this, SLOT(doPlay1234()));
   file->insertSeparator();
   file->insertItem("E&xit",  qApp, SLOT(quit()));
   menuBar()->insertItem("&File", file);
}

void SoundPlayer::doPlay1()
{
   QSound::play("sounds/1.wav");
}

void SoundPlayer::doPlay2()
{
   QSound::play("sounds/2.wav");
}

void SoundPlayer::doPlay3()
{
   bucket3.play();
}

void SoundPlayer::doPlay4()
{
   bucket4.play();
}

void SoundPlayer::doPlay34()
{
   // Some sound platforms will only play one sound at a time
   bucket3.play();
   bucket4.play();
}

void SoundPlayer::doPlay1234()
{
   // Some sound platforms will only play one sound at a time
   QSound::play("sounds/1.wav");
   QSound::play("sounds/2.wav");
   bucket3.play();
   bucket4.play();
}

int main(int argc, char** argv)
{
   QApplication app(argc,argv);
   SoundPlayer sp;
   app.setMainWidget(&sp);
   sp.setCaption("Qt Example - Sounds");
   sp.show();
   return app.exec();
}
```

sound.h

```
#ifndef PLAY_H
#define PLAY_H

#include "qsound.h"
#include <qmainwindow.h>
```

```
class SoundPlayer : public QMainWindow {
   Q_OBJECT
public:
   SoundPlayer();

public slots:
   void doPlay1();
   void doPlay2();
   void doPlay3();
   void doPlay4();
   void doPlay34();
   void doPlay1234();

private:
   QSound bucket3;
   QSound bucket4;
};

#endif
```

실행



59. 분할기

이 실례는 분할기(splitter)의 사용법을 보여준다.

splitter.pro

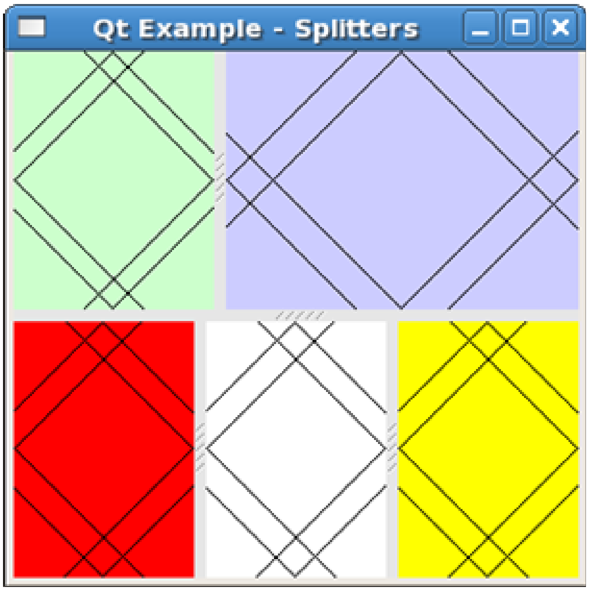
```
TEMPLATE  = app
TARGET     = splitter
CONFIG     += qt warn_on release
HEADERS        =
SOURCES        = splitter.cpp
```

splitter.cpp

```
#include <qapplication.h>
#include <qlabel.h>
#include <qsplitter.h>
#include <qmultilineedit.h>
#include <qpainter.h>

class Test : public QWidget {
public:
   Test(QWidget* parent=0, const char* name=0, int f=0);
   void paintEvent(QPaintEvent* e);
private:
};

Test::Test(QWidget* parent, const char* name, int f) :
   QWidget(parent, name, f)
{
}

void Test::paintEvent(QPaintEvent* e)
{
   QPainter p(this);
   p.setClipRect(e->rect());
   const int d = 1000; //large number
   int x1 = 0;
   int x2 = width()-1;
   int y1 = 0;
   int y2 = height()-1;

   int x = (x1+x2)/2;
   p.drawLine( x, y1, x+d, y1+d   );
   p.drawLine( x, y1, x-d, y1+d   );
   p.drawLine( x, y2, x+d, y2-d   );
   p.drawLine( x, y2, x-d, y2-d   );

   int y = (y1+y2)/2;
   p.drawLine( x1, y, x1+d, y+d   );
   p.drawLine( x1, y, x1+d, y-d   );
   p.drawLine( x2, y, x2-d, y+d   );
   p.drawLine( x2, y, x2-d, y-d   );
}

int main( int argc, char ** argv )
{
   QApplication a( argc, argv );

   QSplitter *s1 = new QSplitter( QSplitter::Vertical, 0 , "main" );

   QSplitter *s2 = new QSplitter( QSplitter::Horizontal, s1, "top" );

   Test *t1 = new Test( s2, "topLeft" );
   t1->setBackgroundColor( Qt::blue.light( 180 ) );
   t1->setMinimumSize( 50, 0 );

   Test *t2 = new Test( s2, "topRight" );
   t2->setBackgroundColor( Qt::green.light( 180 ) );
   s2->setResizeMode( t2, QSplitter::KeepSize );
   s2->moveToFirst( t2 );

   QSplitter *s3 = new QSplitter( QSplitter::Horizontal,  s1, "bottom" );

   Test *t3 = new Test( s3, "bottomLeft" );
   t3->setBackgroundColor( Qt::red );
```

```
    Test *t4 = new Test( s3, "bottomMiddle" );
    t4->setBackgroundColor( Qt::white );

    Test *t5 = new Test( s3, "bottomRight" );
    t5->setMaximumHeight( 250 );
    t5->setMinimumSize( 80, 50 );
    t5->setBackgroundColor( Qt::yellow );

#ifdef Q_WS_QWS
    // Qt/Embedded XOR drawing not yet implemented.
    s1->setOpaqueResize( TRUE );
#endif
    s2->setOpaqueResize( TRUE );
    s3->setOpaqueResize( TRUE );

    a.setMainWidget( s1 );
    s1->setCaption("Qt Example - Splitters");
    s1->show();
    int result = a.exec();
    delete s1;
    return result;
}
```

실행



60. 타브대화칸

이 실례는 여러개의 타브(페지)들을 가지는 대화칸의 사용법을 보여준다. 프로그람을 기동하려면 첫 인수로서 파일이름을 지정해야 한다. 대화칸은 여러개의 타브들로 분리된 파일정보를 보여준다.

tabdialog.pro

```
TEMPLATE  = app
TARGET     = tabdialog
CONFIG      += qt warn_on release
```

```
HEADERS         = tabdialog.h
SOURCES         = main.cpp \
            tabdialog.cpp
```

tabdialog.cpp

```
#include "tabdialog.h"
#include <qvbox.h>
#include <qlabel.h>
#include <qlineedit.h>
#include <qdatetime.h>
#include <qbuttongroup.h>
#include <qcheckbox.h>
#include <qlistbox.h>
#include <qapplication.h>

TabDialog::TabDialog( QWidget *parent, const char *name, const QString &_filename )
   : QTabDialog( parent, name ), filename( _filename ), fileinfo( filename )
{
   setupTab1();
   setupTab2();
   setupTab3();

   connect( this, SIGNAL( applyButtonPressed() ), qApp, SLOT( quit() ) );
}

void TabDialog::setupTab1()
{
   QVBox *tab1 = new QVBox( this );
   tab1->setMargin( 5 );

   (void)new QLabel( "Filename:", tab1 );
   QLineEdit *fname = new QLineEdit( filename, tab1 );
   fname->setFocus();

   (void)new QLabel( "Path:", tab1 );
   QLabel *path = new QLabel( fileinfo.dirPath( TRUE ), tab1 );
   path->setFrameStyle( QFrame::Panel | QFrame::Sunken );

   (void)new QLabel( "Size:", tab1 );
   ulong kb = (ulong)(fileinfo.size()/1024);
   QLabel *size = new QLabel( QString( "%1 KB" ).arg( kb ), tab1 );
   size->setFrameStyle( QFrame::Panel | QFrame::Sunken );

   (void)new QLabel( "Last Read:", tab1 );
   QLabel *lread = new QLabel( fileinfo.lastRead().toString(), tab1 );
   lread->setFrameStyle( QFrame::Panel | QFrame::Sunken );

   (void)new QLabel( "Last Modified:", tab1 );
   QLabel *lmodif = new QLabel( fileinfo.lastModified().toString(), tab1 );
   lmodif->setFrameStyle( QFrame::Panel | QFrame::Sunken );

   addTab( tab1, "General" );
}
```

```
void TabDialog::setupTab2()
{
   QVBox *tab2 = new QVBox( this );
   tab2->setMargin( 5 );

   QButtonGroup *bg = new QButtonGroup( 1, QGroupBox::Horizontal, "Permissions", tab2 );

   QCheckBox *readable = new QCheckBox( "Readable", bg );
   if ( fileinfo.isReadable() )
      readable->setChecked( TRUE );

   QCheckBox *writable = new QCheckBox( "Writeable", bg );
   if ( fileinfo.isWritable() )
      writable->setChecked( TRUE );

   QCheckBox *executable = new QCheckBox( "Executable", bg );
   if ( fileinfo.isExecutable() )
      executable->setChecked( TRUE );

   QButtonGroup *bg2 = new QButtonGroup( 2, QGroupBox::Horizontal, "Owner", tab2 );

   (void)new QLabel( "Owner", bg2 );
   QLabel *owner = new QLabel( fileinfo.owner(), bg2 );
   owner->setFrameStyle( QFrame::Panel | QFrame::Sunken );

   (void)new QLabel( "Group", bg2 );
   QLabel *group = new QLabel( fileinfo.group(), bg2 );
   group->setFrameStyle( QFrame::Panel | QFrame::Sunken );

   addTab( tab2, "Permissions" );
}

void TabDialog::setupTab3()
{
   QVBox *tab3 = new QVBox( this );
   tab3->setMargin( 5 );
   tab3->setSpacing( 5 );

   (void)new QLabel( QString( "Open %1 with:" ).arg( filename ), tab3 );

   QListBox *prgs = new QListBox( tab3 );
   for ( unsigned int i = 0; i < 30; i++ ) {
      QString prg = QString( "Application %1" ).arg( i );
      prgs->insertItem( prg );
   }
   prgs->setCurrentItem( 3 );

   (void)new QCheckBox( QString( "Open files with the extension '%1' always with this
application" ).arg( fileinfo.extension() ), tab3 );

   addTab( tab3, "Applications" );
}
```

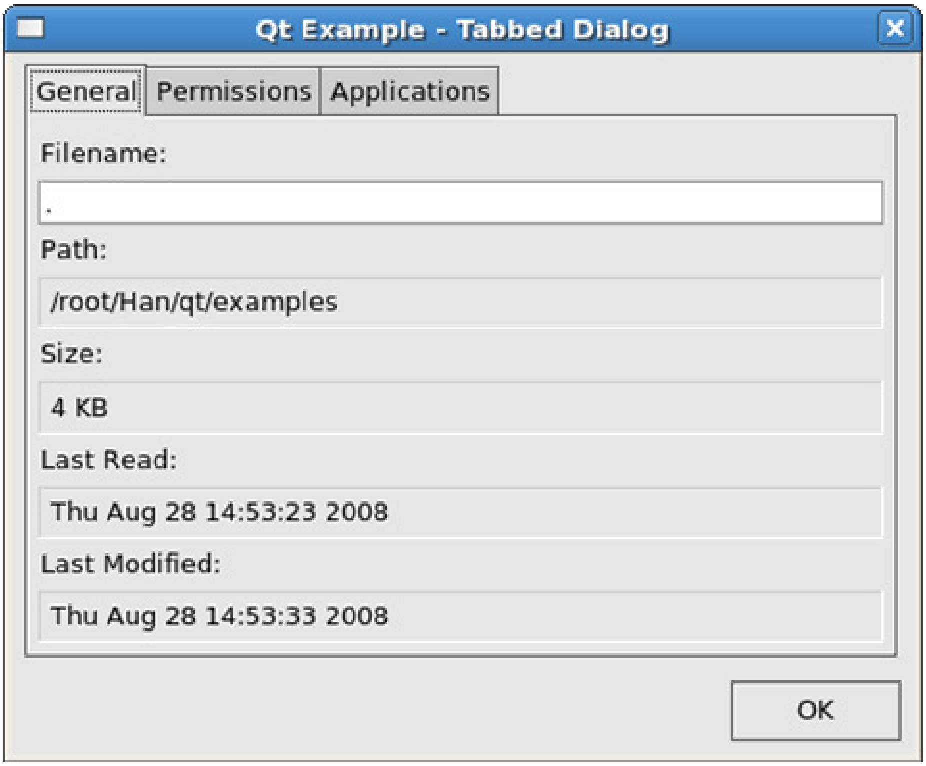
tabdialog.h

```
#ifndef TABDIALOG_H
#define TABDIALOG_H
#include <qtabdialog.h>
#include <qstring.h>
#include <qfileinfo.h>

class TabDialog : public QTabDialog
{
   Q_OBJECT

public:
   TabDialog( QWidget *parent, const char *name, const QString &_filename );

protected:
   QString filename;
   QFileInfo fileinfo;

   void setupTab1();
   void setupTab2();
   void setupTab3();

};

#endif
```

main.cpp

```
#include "tabdialog.h"
#include <qapplication.h>
#include <qstring.h>

int main( int argc, char **argv )
{

   QApplication a( argc, argv );

   TabDialog tabdialog( 0, "tabdialog", QString( argc < 2 ? "." : argv[1] ) );
   tabdialog.resize( 450, 350 );
   tabdialog.setCaption( "Qt Example - Tabbed Dialog" );
   a.setMainWidget( &tabdialog );
   tabdialog.show();

   return a.exec();
}
```

실행



Filename:
.
Path:
/root/Han/qt/examples
Size:
4 KB
Last Read:
Thu Aug 28 14:53:23 2008
Last Modified:
Thu Aug 28 14:53:33 2008

61. 표

다음의 실례프로그람들은 Qt표모듈의 사용법을 보여준다.

1) QTable 의 창조방법

이 실례는 QIntDict를 사용하여 세포들이 드문드문 배치된 표의 실현방법을 보여준다.

bigtable.pro

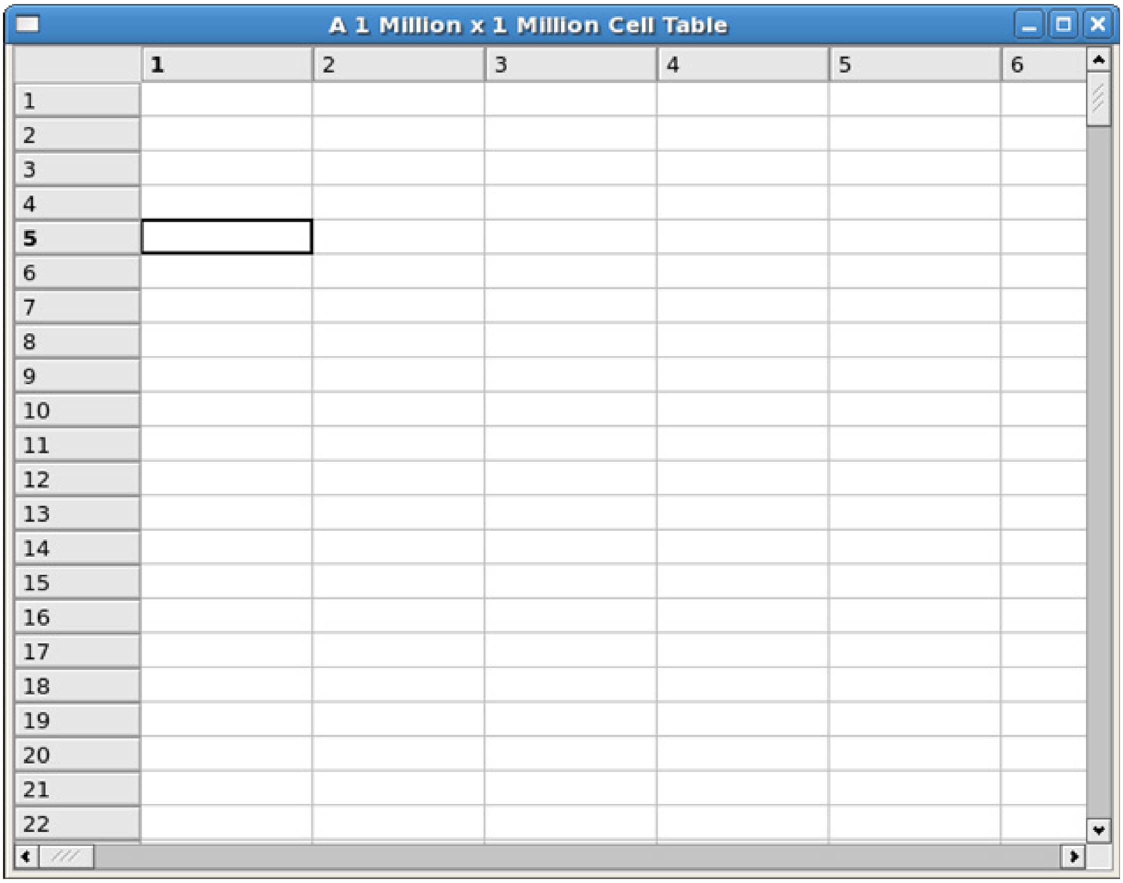
```
TEMPLATE  = app
TARGET    = bigtable
CONFIG    += qt warn_on release
HEADERS      =
SOURCES      = main.cpp
```

main.cpp

```
#include <qapplication.h>
#include <qtable.h>

// Table size
const int numRows = 1000000;
const int numCols = 1000000;

class MyTable : public QTable
{
public:
    MyTable( int r, int c ) : QTable( r, c ) {
      items.setAutoDelete( TRUE );
```

```
     widgets.setAutoDelete( TRUE );
     setCaption( tr( "A 1 Million x 1 Million Cell Table" ) );
     setLeftMargin( fontMetrics().width( "W999999W" ) );
    }

    void resizeData( int ) {}
    QTableItem *item( int r, int c ) const { return items.find( indexOf( r, c ) ); }
    void setItem( int r, int c, QTableItem *i ) { items.replace( indexOf( r, c ), i ); }
    void clearCell( int r, int c ) { items.remove( indexOf( r, c ) ); }
    void takeItem( QTableItem *item )
    {
     items.setAutoDelete( FALSE );
     items.remove( indexOf( item->row(), item->col() ) );
     items.setAutoDelete( TRUE );
    }
    void insertWidget( int r, int c, QWidget *w ) { widgets.replace( indexOf( r, c ), w );  }
    QWidget *cellWidget( int r, int c ) const { return widgets.find( indexOf( r, c ) ); }
    void clearCellWidget( int r, int c )
    {
     QWidget *w = widgets.take( indexOf( r, c ) );
     if ( w )
         w->deleteLater();
    }

private:
    QIntDict<QTableItem> items;
    QIntDict<QWidget> widgets;

};

// The program starts here.
int main( int argc, char **argv )
{
    QApplication app( argc, argv );

    MyTable table( numRows, numCols );
    app.setMainWidget( &table );
    table.show();
    return app.exec();
}
```

실행



2) 작은 표실례

이 실례는 하나의 QTable과 여러개의 QTableItem들을 현시한다.

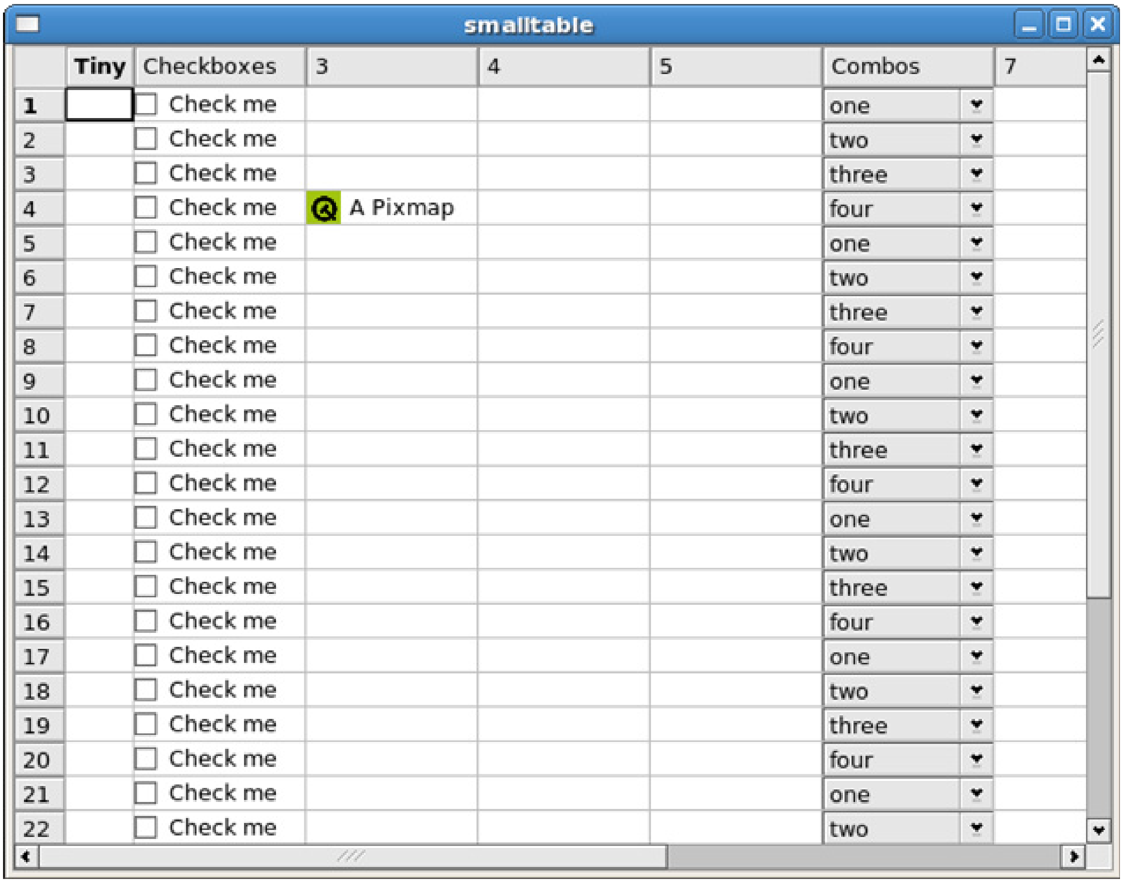
smalltable.pro

```
TEMPLATE  = app
TARGET    = smalltable
CONFIG    += qt warn_on release
HEADERS      =
SOURCES      = main.cpp
```

main.cpp

```
#include <qapplication.h>
#include <qtable.h>
#include <qimage.h>
#include <qpixmap.h>
#include <qstringlist.h>

// Qt logo: static const char *qtlogo_xpm[]
#include "qtlogo.xpm"

// Table size

const int numRows = 30;
const int numCols = 10;

// The program starts here.

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

   QTable table( numRows, numCols );

   QHeader *header = table.horizontalHeader();
   header->setLabel( 0, QObject::tr( "Tiny" ), 40 );
   header->setLabel( 1, QObject::tr( "Checkboxes" ) );
   header->setLabel( 5, QObject::tr( "Combos" ) );
   table.setColumnMovingEnabled(TRUE);

   QImage img( qtlogo_xpm );
   QPixmap pix = img.scaleHeight( table.rowHeight(3) );
   table.setPixmap( 3, 2, pix );
   table.setText( 3, 2, "A Pixmap" );

   QStringList comboEntries;
   comboEntries << "one" << "two" << "three" << "four";

   for ( int i = 0; i < numRows; ++i ){
    QComboTableItem * item = new QComboTableItem( &table, comboEntries, FALSE );
    item->setCurrentItem( i % 4 );
    table.setItem( i, 5, item );
   }
   for ( int j = 0; j < numRows; ++j )
    table.setItem( j, 1, new QCheckTableItem( &table, "Check me" ) );

   app.setMainWidget( &table );
   table.show();
   return app.exec();
}
```

실행



	Tiny	Checkboxes	3	4	5	Combos	7
1		□ Check me				one	
2		□ Check me				two	
3		□ Check me				three	
4		□ Check me	A Pixmap			four	
5		□ Check me				one	
6		□ Check me				two	
7		□ Check me				three	
8		□ Check me				four	
9		□ Check me				one	
10		□ Check me				two	
11		□ Check me				three	
12		□ Check me				four	
13		□ Check me				one	
14		□ Check me				two	
15		□ Check me				three	
16		□ Check me				four	
17		□ Check me				one	
18		□ Check me				two	
19		□ Check me				three	
20		□ Check me				four	
21		□ Check me				one	
22		□ Check me				two	

3) 표실례

statistics.pro

```
TEMPLATE   = app
TARGET     = statistics
CONFIG     += qt warn_on release
HEADERS        = statistics.h
SOURCES        = statistics.cpp main.cpp
```

statistics.cpp

```
#include "statistics.h"
#include <qdir.h>
#include <qstringlist.h>
#include <qheader.h>
#include <qcombobox.h>
#include <stdlib.h>

const char* dirs[] = {
    "kernel",
    "tools",
    "widgets",
    "dialogs",
    "xml",
    "table",
    "network",
    "opengl",
    "canvas",
    0
};

Table::Table()
    : QTable( 10, 100, 0, "table" )
{
    setSorting( TRUE );
    horizontalHeader()->setLabel( 0, tr( "File" ) );
    horizontalHeader()->setLabel( 1, tr( "Size (bytes)" ) );
    horizontalHeader()->setLabel( 2, tr( "Use in Sum" ) );
    initTable();
    adjustColumn( 0 );

    // if the user edited something we might need to recalculate the sum
    connect( this, SIGNAL( valueChanged( int, int ) ),
           this, SLOT( recalcSum( int, int ) ) );
}

void Table::initTable()
{
    // read all the Qt source and header files into a list
    QStringList all;
    int i = 0;
    QString srcdir( "../../../src/" );
    while ( dirs[ i ] ) {
     QDir dir( srcdir + dirs[ i ] );
     QStringList lst = dir.entryList( "*.cpp; *.h" );
     for ( QStringList::Iterator it = lst.begin(); it != lst.end(); ++it ) {
         if ( ( *it ).contains( "moc" ) )
          continue;
         all << (QString( dirs[ i ] ) + "/" + *it);
     }
     ++i;
    }

    // set the number of rows we'll need for the table
    setNumRows( all.count() + 1 );
    i = 0;
    int sum = 0;

    // insert the data into the table
    for ( QStringList::Iterator it = all.begin(); it != all.end(); ++it ) {
     setText( i, 0, *it );
     QFile f( srcdir + *it );
     setText( i, 1, QString::number( (ulong)f.size() ) );
     ComboItem *ci = new ComboItem( this, QTableItem::WhenCurrent );
     setItem( i++, 2, ci );
     sum += f.size();
    }
    // last row should show the sum
    TableItem *i1 = new TableItem( this, QTableItem::Never, tr( "Sum" ) );
    setItem( i, 0, i1 );
    TableItem *i2 = new TableItem( this, QTableItem::Never, QString::number( sum ) );
    setItem( i, 1, i2 );
}

void Table::recalcSum( int, int col )
{
    // only recalc if a value in the second or third column changed
    if ( col < 1 || col > 2 )
     return;

    // recalc sum
    int sum = 0;
    for ( int i = 0; i < numRows() - 1; ++i ) {
     if ( text( i, 2 ) == "No" )
         continue;
     sum += text( i, 1 ).toInt();
    }

    // insert calculated data
    TableItem *i1 = new TableItem( this, QTableItem::Never, tr( "Sum" ) );
    setItem( numRows() - 1, 0, i1 );
    TableItem *i2 = new TableItem( this, QTableItem::Never, QString::number( sum ) );
    setItem( numRows() - 1, 1, i2 );
}

void Table::sortColumn( int col, bool ascending, bool /*wholeRows*/ )
{
    // sum row should not be sorted, so get rid of it for now
    clearCell( numRows() - 1, 0 );
    clearCell( numRows() - 1, 1 );
    // do sort
    QTable::sortColumn( col, ascending, TRUE );
    // re-insert sum row
    recalcSum( 0, 1 );
}

void TableItem::paint( QPainter *p, const QColorGroup &cg, const QRect &cr, bool selected )
{
    QColorGroup g( cg );
    // last row is the sum row - we want to make it more visible by
    // using a red background
    if ( row() == table()->numRows() - 1 )
     g.setColor( QColorGroup::Base, red );
    QTableItem::paint( p, g, cr, selected );
}

ComboItem::ComboItem( QTable *t, EditType et )   : QTableItem( t, et, "Yes" ), cb( 0 )
{
    // we do not want this item to be replaced
    setReplaceable( FALSE );
}
```

```
QWidget *ComboItem::createEditor() const
{
   // create an editor - a combobox in our case
   ( (ComboItem*)this )->cb = new QComboBox( table()->viewport() );
   QObject::connect( cb, SIGNAL( activated( int ) ), table(), SLOT( doValueChanged() ) );
   cb->insertItem( "Yes" );
   cb->insertItem( "No" );
   // and initialize it
   cb->setCurrentItem( text() == "No" ? 1 : 0 );
   return cb;
}

void ComboItem::setContentFromEditor( QWidget *w )
{
   // the user changed the value of the combobox, so synchronize the
   // value of the item (its text), with the value of the combobox
   if ( w->inherits( "QComboBox" ) )
    setText( ( (QComboBox*)w )->currentText() );
   else
    QTableItem::setContentFromEditor( w );
}

void ComboItem::setText( const QString &s )
{
   if ( cb ) {
    // initialize the combobox from the text
    if ( s == "No" )
       cb->setCurrentItem( 1 );
    else
       cb->setCurrentItem( 0 );
   }
   QTableItem::setText( s );
}
```

statistics.h

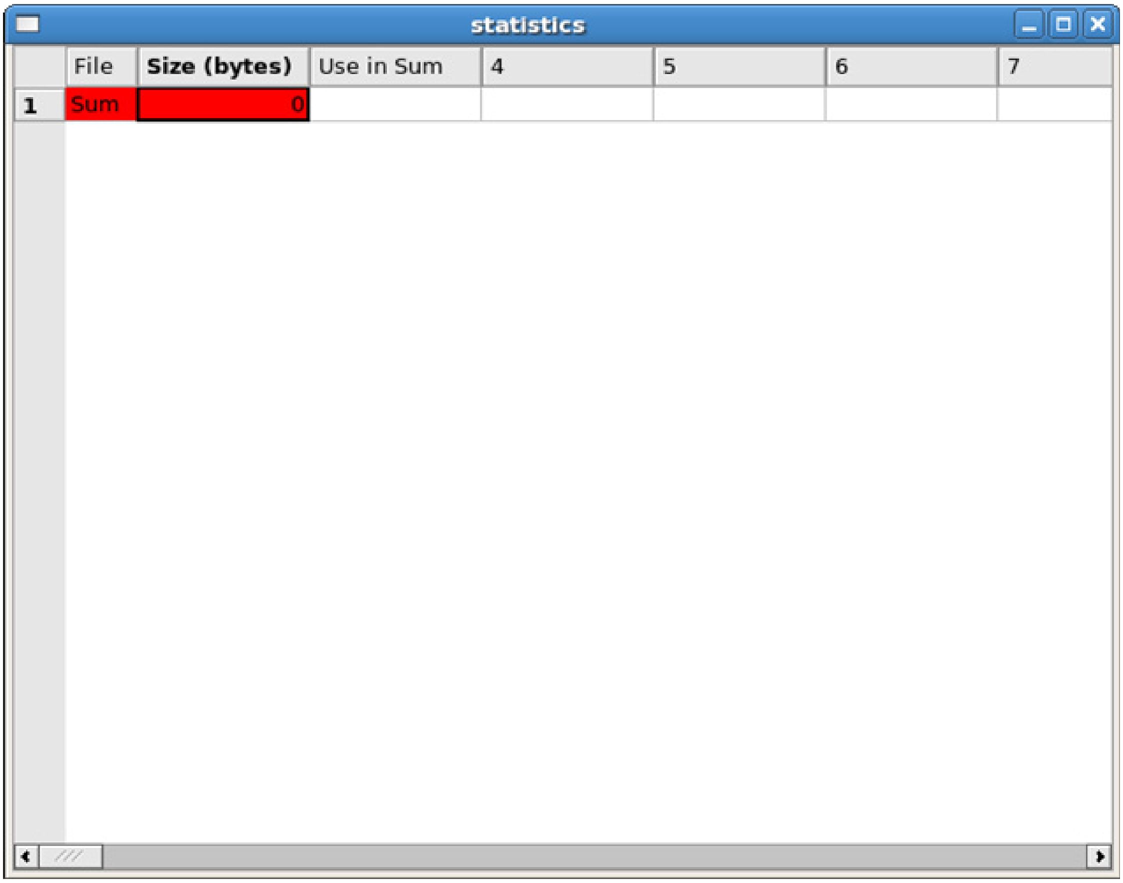
```
#ifndef STATISTICS_H
#define STATISTICS_H

#include <qtable.h>
#include <qcombobox.h>

class TableItem : public QTableItem
{
public:
   TableItem( QTable *t, EditType et, const QString &txt ) : QTableItem( t, et, txt ) {}
   void paint( QPainter *p, const QColorGroup &cg, const QRect &cr, bool selected );
};

class ComboItem : public QTableItem
{
public:
   ComboItem( QTable *t, EditType et );
   QWidget *createEditor() const;
```

```
    void setContentFromEditor( QWidget *w );
    void setText( const QString &s );

private:
    QComboBox *cb;

};

class Table : public QTable
{
    Q_OBJECT

public:
    Table();
    void sortColumn( int col, bool ascending, bool wholeRows );

private slots:
    void recalcSum( int row, int col );

private:
    void initTable();

};

#endif
```

main.cpp

```
#include "statistics.h"
#include <qapplication.h>
int main( int argc, char **argv )
{
    QApplication a(argc,argv);

    Table t;
    a.setMainWidget( &t );
    t.show();
    return a.exec();
}
```

실행



	File	Size (bytes)	Use in Sum	4	5	6	7
1	Sum	0					

62. 타블레트실례

이 실례는 타블레트장치와 교제하는 방법을 보여준다.

tablet.pro

```
###########################################################
# Automatically generated by qmake Thu Jul 26 13:46:03 2001
###########################################################
TEMPLATE  = app
TARGET      = tablet
# Input
HEADERS     += canvas.h scribble.h tabletstats.h
INTERFACES     += tabletstatsbase.ui
SOURCES     += canvas.cpp main.cpp scribble.cpp tabletstats.cpp
```

canvas.cpp

```
#include "canvas.h"
#include <qapplication.h>
#include <qpainter.h>
#include <qevent.h>
#include <qrect.h>

const bool no_writing = FALSE;
```

```
Canvas::Canvas( QWidget *parent, const char *name, WFlags fl )
    : QWidget( parent, name, WStaticContents | fl ),
      pen( Qt::red, 3 ), polyline(3),
      mousePressed( FALSE ), oldPressure( 0 ), saveColor( red ),
      buffer( width(), height() )
{

    if ((qApp->argc() > 0) && !buffer.load(qApp->argv()[1]))
      buffer.fill( colorGroup().base() );
    setBackgroundMode( QWidget::PaletteBase );
#ifndef QT_NO_CURSOR
    setCursor( Qt::crossCursor );
#endif
}

void Canvas::save( const QString &filename, const QString &format )
{
    if ( !no_writing )
      buffer.save( filename, format.upper() );
}

void Canvas::clearScreen()
{
    buffer.fill( colorGroup().base() );
    repaint( FALSE );
}

void Canvas::mousePressEvent( QMouseEvent *e )
{
    mousePressed = TRUE;
    polyline[2] = polyline[1] = polyline[0] = e->pos();
}

void Canvas::mouseReleaseEvent( QMouseEvent * )
{
    mousePressed = FALSE;
}

void Canvas::mouseMoveEvent( QMouseEvent *e )
{
    if ( mousePressed ) {
      QPainter painter;
      painter.begin( &buffer );
      painter.setPen( pen );
      polyline[2] = polyline[1];
      polyline[1] = polyline[0];
      polyline[0] = e->pos();
      painter.drawPolyline( polyline );
      painter.end();

      QRect r = polyline.boundingRect();
      r = r.normalize();
      r.setLeft( r.left() - penWidth() );
      r.setTop( r.top() - penWidth() );
      r.setRight( r.right() + penWidth() );
      r.setBottom( r.bottom() + penWidth() );

      bitBlt( this, r.x(), r.y(), &buffer, r.x(), r.y(), r.width(), r.height() );
    }
}

void Canvas::tabletEvent( QTabletEvent *e )
{
    e->accept();
    // change the width based on range of pressure
    if ( e->device() == QTabletEvent::Stylus )  {
      if ( e->pressure() >= 0 && e->pressure() <= 32 )
          pen.setColor( saveColor.light(175) );
      else if ( e->pressure() > 32 && e->pressure() <= 64 )
          pen.setColor( saveColor.light(150) );
      else if ( e->pressure() > 64  && e->pressure() <= 96 )
          pen.setColor( saveColor.light(125) );
      else if ( e->pressure() > 96 && e->pressure() <= 128 )
          pen.setColor( saveColor );
      else if ( e->pressure() > 128 && e->pressure() <= 160 )
          pen.setColor( saveColor.dark(150) );
      else if ( e->pressure() > 160 && e->pressure() <= 192 )
          pen.setColor( saveColor.dark(200) );
      else if ( e->pressure() > 192 && e->pressure() <= 224 )
          pen.setColor( saveColor.dark(250) );
      else // pressure > 224
          pen.setColor( saveColor.dark(300) );
    } else if ( e->device() == QTabletEvent::Eraser
            && pen.color() != backgroundColor() ) {
      pen.setColor( backgroundColor() );
    }

    int xt = e->xTilt();
    int yt = e->yTilt();
    if ( ( xt > -15 && xt < 15 ) && ( yt > -15 && yt < 15 ) )
      pen.setWidth( 3 );
    else if ( ((xt < -15 && xt > -30) || (xt > 15 && xt < 30)) &&
            ((yt < -15 && yt > -30) || (yt > 15 && yt < 30 )) )
      pen.setWidth( 6 );
    else if ( ((xt < -30 && xt > -45) || (xt > 30 && xt < 45)) &&
            ((yt < -30 && yt > -45) || (yt > 30 && yt < 45)) )
      pen.setWidth( 9 );
    else if (  (xt < -45 || xt > 45 ) && ( yt < -45 || yt > 45 ) )
      pen.setWidth( 12 );

    switch ( e->type() ) {
    case QEvent::TabletPress:
      mousePressed = TRUE;
      polyline[2] = polyline[1] = polyline[0] = e->pos();
      break;
    case QEvent::TabletRelease:
      mousePressed = FALSE;
      break;
```

```
  case QEvent::TabletMove:
   if ( mousePressed ) {
      QPainter painter;
      painter.begin( &buffer );
      painter.setPen( pen );
      polyline[2] = polyline[1];
      polyline[1] = polyline[0];
      polyline[0] = e->pos();
      painter.drawPolyline( polyline );
      painter.end();

      QRect r = polyline.boundingRect();
      r = r.normalize();
      r.setLeft( r.left() - penWidth() );
      r.setTop( r.top() - penWidth() );
      r.setRight( r.right() + penWidth() );
      r.setBottom( r.bottom() + penWidth() );

      bitBlt( this, r.x(), r.y(), &buffer, r.x(), r.y(), r.width(),  r.height() );
   }
   break;
  default:
   break;
  }
}

void Canvas::resizeEvent( QResizeEvent *e )
{
  QWidget::resizeEvent( e );

  int w = width() > buffer.width() ?  width() : buffer.width();
  int h = height() > buffer.height() ?  height() : buffer.height();

  QPixmap tmp( buffer );
  buffer.resize( w, h );
  buffer.fill( colorGroup().base() );
  bitBlt( &buffer, 0, 0, &tmp, 0, 0, tmp.width(), tmp.height() );
}

void Canvas::paintEvent( QPaintEvent *e )
{
  QWidget::paintEvent( e );

  QMemArray<QRect> rects = e->region().rects();
  for ( uint i = 0; i < rects.count(); i++ ) {
   QRect r = rects[(int)i];
   bitBlt( this, r.x(), r.y(), &buffer, r.x(), r.y(), r.width(), r.height() );
  }
}
```

canvas.h

```
#include <qpen.h>
#include <qpixmap.h>
#include <qpoint.h>
```

```
#include <qpointarray.h>
#include <qwidget.h>

#ifndef _MY_CANVAS_
#define _MY_CANVAS_

class Canvas : public QWidget
{
   Q_OBJECT

public:
   Canvas( QWidget *parent = 0, const char *name = 0, WFlags fl = 0 );
   virtual ~Canvas() {};

   void setPenColor( const QColor &c )
   {     saveColor = c;
    pen.setColor( saveColor ); }

   void setPenWidth( int w )
   { pen.setWidth( w ); }

   QColor penColor()
   { return pen.color(); }

   int penWidth()
   { return pen.width(); }

   void save( const QString &filename, const QString &format );

   void clearScreen();

protected:
   virtual void mousePressEvent( QMouseEvent *e );
   virtual void mouseReleaseEvent( QMouseEvent *e );
   virtual void mouseMoveEvent( QMouseEvent *e );
   virtual void resizeEvent( QResizeEvent *e );
   virtual void paintEvent( QPaintEvent *e );
   virtual void tabletEvent( QTabletEvent *e );

   QPen pen;
   QPointArray polyline;

   bool mousePressed;
   int oldPressure;
   QColor saveColor;

   QPixmap buffer;

};

#endif
```

scribble.cpp

```
#include "canvas.h"
```

```
#include "scribble.h"
#include <qapplication.h>
#include <qevent.h>
#include <qpainter.h>
#include <qtoolbar.h>
#include <qtoolbutton.h>
#include <qspinbox.h>
#include <qtooltip.h>
#include <qrect.h>
#include <qpoint.h>
#include <qcolordialog.h>
#include <qfiledialog.h>
#include <qcursor.h>
#include <qimage.h>
#include <qstrlist.h>
#include <qpopupmenu.h>
#include <qintdict.h>

Scribble::Scribble( QWidget *parent, const char *name )
   : QMainWindow( parent, name )
{
   canvas = new Canvas( this );
   setCentralWidget( canvas );

   QToolBar *tools = new QToolBar( this );

   bSave = new QToolButton( QPixmap(), "Save", "Save as PNG image", this, SLOT( slotSave() ),
tools );
   bSave->setText( "Save as..." );

   tools->addSeparator();

   bPColor = new QToolButton( QPixmap(), "Choose Pen Color", "Choose Pen Color", this,
SLOT( slotColor() ), tools );
   bPColor->setText( "Choose Pen Color..." );

   tools->addSeparator();

   bPWidth = new QSpinBox( 1, 20, 1, tools );
   QToolTip::add( bPWidth, "Choose Pen Width" );
   connect( bPWidth, SIGNAL( valueChanged( int ) ), this, SLOT( slotWidth( int ) ) );
   bPWidth->setValue( 3 );

   tools->addSeparator();

   bClear = new QToolButton( QPixmap(), "Clear Screen", "Clear Screen", this, SLOT( slotClear() ),
tools );
   bClear->setText( "Clear Screen" );
}

/*

Scribble::Scribble( QWidget *parent, const char *name )
   : QMainWindow( parent, name )
```

```
{
   canvas = new Canvas( this );
   setCentralWidget( canvas );

   QToolBar *tools = new QToolBar( this );

   bSave = new QPushButton( "Save as...", tools );

   tools->addSeparator();

   bPColor = new QPushButton( "Choose Pen Color...", tools );
   //    bPColor->setText( "Choose Pen Color..." );

   tools->addSeparator();

   bPWidth = new QSpinBox( 1, 20, 1, tools );
   QToolTip::add( bPWidth, "Choose Pen Width" );
   connect( bPWidth, SIGNAL( valueChanged( int ) ), this, SLOT( slotWidth( int ) ) );
   bPWidth->setValue( 3 );

   tools->addSeparator();

   bClear = new QPushButton( "Clear Screen", tools );
   QObject::connect( bSave, SIGNAL( clicked() ), this, SLOT( slotSave() ) );
   QObject::connect( bPColor, SIGNAL( clicked() ), this, SLOT( slotColor() ) );
   QObject::connect( bClear, SIGNAL( clicked() ), this, SLOT( slotClear() ) );

}

  */
void Scribble::slotSave()
{
   QPopupMenu *menu = new QPopupMenu( 0 );
   QIntDict<QString> formats;
   formats.setAutoDelete( TRUE );

   for ( unsigned int i = 0; i < QImageIO::outputFormats().count(); i++ ) {
    QString str = QString( QImageIO::outputFormats().at( i ) );
    formats.insert( menu->insertItem( QString( "%1..." ).arg( str ) ), new QString( str ) );
   }

   menu->setMouseTracking( TRUE );
   int id = menu->exec( bSave->mapToGlobal( QPoint( 0, bSave->height() + 1 ) ) );

   if ( id != -1 ) {
    QString format = *formats[ id ];

    QString filename = QFileDialog::getSaveFileName( QString::null,
QString( "*.%1" ).arg( format.lower() ), this );
    if ( !filename.isEmpty() )
        canvas->save( filename, format );
   }

   delete menu;
}

void Scribble::slotColor()
{
   QColor c = QColorDialog::getColor( canvas->penColor(), this );
   canvas->setPenColor( c );
}

void Scribble::slotWidth( int w )
{
   canvas->setPenWidth( w );
}

void Scribble::slotClear()
{
   canvas->clearScreen();
}
```

scribble.h

```
#ifndef SCRIBBLE_H
#define SCRIBBLE_H

#include <qmainwindow.h>
#include <qpen.h>
#include <qpoint.h>
#include <qpixmap.h>
#include <qwidget.h>
#include <qstring.h>
#include <qpointarray.h>

class QMouseEvent;
class QResizeEvent;
class QPaintEvent;
class QSpinBox;
class QToolButton;
class Canvas;

class Scribble : public QMainWindow
{
   Q_OBJECT

public:
   Scribble( QWidget *parent = 0, const char *name = 0 );

protected:
   Canvas* canvas;

   QSpinBox *bPWidth;
   QToolButton *bPColor, *bSave, *bClear;

protected slots:
   void slotSave();
   void slotColor();
   void slotWidth( int );
```

```
    void slotClear();

};

#endif
```

tabletstats.cpp

```
#include <qlabel.h>
#include <qlayout.h>
#include <qpainter.h>
#include <math.h>
#include "tabletstats.h"

MyOrientation::MyOrientation( QWidget *parent, const char *name )
    : QFrame( parent, name, WRepaintNoErase )
{
//    QSizePolicy mySize( QSizePolicy::Minimum, QSizePolicy::Expanding );
//    setSizePolicy( mySize );
    setFrameStyle( QFrame::Box | QFrame::Sunken );
}

MyOrientation::~MyOrientation()
{
}

void MyOrientation::newOrient( int tiltX, int tiltY )
{
    double PI = 3.14159265359;
    int realWidth,
        realHeight,
        hypot,     // a faux hypoteneus, to mess with calculations
        shaX,
      shaY;
    static int oldX = 0,
            oldY = 0;
    realWidth = width() - 2 * frameWidth();
    realHeight = height() - 2 * frameWidth();

    QRect cr( 0 + frameWidth(), 0 + frameWidth(), realWidth, realHeight );
    QPixmap pix( cr.size() );
    pix.fill( this, cr.topLeft() );
    QPainter p( &pix );

    if ( realWidth > realHeight )
      hypot = realHeight / 2;
    else
      hypot = realWidth / 2;

    // create a shadow...
    shaX = int(hypot * sin( tiltX * (PI / 180) ));
    shaY = int(hypot * sin( tiltY * (PI / 180) ));

    p.translate( realWidth / 2, realHeight / 2 );
    p.setPen( backgroundColor() );
    p.drawLine( 0, 0, oldX, oldY );
    p.setPen( foregroundColor() );
    p.drawLine( 0, 0,shaX, shaY );
    oldX = shaX;
    oldY = shaY;
    p.end();

    QPainter p2( this );
    p2.drawPixmap( cr.topLeft(), pix );
    p2.end();
}

StatsCanvas::StatsCanvas( QWidget *parent, const char* name )
  : Canvas( parent, name, WRepaintNoErase )
{
    QSizePolicy mySize( QSizePolicy::Expanding, QSizePolicy::Minimum );
    setSizePolicy( mySize );
}

StatsCanvas::~StatsCanvas()
{
}

void StatsCanvas::tabletEvent( QTabletEvent *e )
{
    static QRect oldR( -1, -1, -1, -1);
    QPainter p;

    e->accept();
    switch( e->type() ) {
    case QEvent::TabletPress:
      qDebug( "Tablet Press" );
      mousePressed = TRUE;
      break;
    case QEvent::TabletRelease:
      qDebug( "Tablet Release" );
      mousePressed = FALSE;
      clearScreen();
      break;
    default:
      break;
    }

    r.setRect( e->x() - e->pressure() / 2, e->y() - e->pressure() / 2, e->pressure(), e->pressure() );
    QRect tmpR = r | oldR;
    oldR = r;

    update( tmpR );
    emit signalNewTilt( e->xTilt(), e->yTilt() );
    emit signalNewDev( e->device() );
    emit signalNewLoc( e->x(), e->y() );
    emit signalNewPressure( e->pressure() );
}
```

```
void StatsCanvas::mouseMoveEvent( QMouseEvent *e )
{
   qDebug( "Mouse Move" );
   // do nothing
   QWidget::mouseMoveEvent( e );
}

void StatsCanvas::mousePressEvent( QMouseEvent *e )
{
   qDebug( "Mouse Press" );
   QWidget::mousePressEvent( e );
}

void StatsCanvas::mouseReleaseEvent( QMouseEvent *e )
{
   qDebug( "Mouse Release" );
   QWidget::mouseReleaseEvent( e );
}

void StatsCanvas::paintEvent( QPaintEvent *e )
{
   QPainter p;
   p.begin( &buffer );
   p.fillRect( e->rect(), colorGroup().base() );

   // draw a circle if we have the tablet down
   if ( mousePressed ) {
    p.setBrush( red );
    p.drawEllipse( r );
   }
   bitBlt( this, e->rect().x(), e->rect().y(), &buffer, e->rect().x(),
        e->rect().y(), e->rect().width(), e->rect().height() );
   p.end();
}

TabletStats::TabletStats( QWidget *parent, const char *name )
     : TabletStatsBase( parent, name )
{
   lblXPos->setMinimumSize( lblXPos->sizeHint() );
   lblYPos->setMinimumSize( lblYPos->sizeHint() );
   lblPressure->setMinimumSize( lblPressure->sizeHint() );
   lblDev->setMinimumSize( lblDev->sizeHint() );
   lblXTilt->setMinimumSize( lblXTilt->sizeHint() );
   lblYTilt->setMinimumSize( lblYTilt->sizeHint() );

   QObject::connect( statCan, SIGNAL(signalNewTilt(int, int)), orient, SLOT(newOrient(int, int)) );
   QObject::connect( statCan, SIGNAL(signalNewTilt(int, int)), this, SLOT(slotTiltChanged(int, int)) );
   QObject::connect( statCan, SIGNAL(signalNewDev(int)), this, SLOT(slotDevChanged(int)) );
   QObject::connect( statCan, SIGNAL(signalNewLoc(int,int)),
                 this, SLOT( slotLocationChanged(int,int)) );
}

TabletStats::~TabletStats()
{
}

void TabletStats::slotDevChanged( int newDev )
{
   if ( newDev == QTabletEvent::Stylus )
     lblDev->setText( tr("Stylus") );
   else if ( newDev == QTabletEvent::Eraser )
     lblDev->setText( tr("Eraser") );
}

void TabletStats::slotLocationChanged( int newX, int newY )
{
   lblXPos->setNum( newX );
   lblYPos->setNum( newY );
}

void TabletStats::slotTiltChanged( int newTiltX, int newTiltY )
{
   lblXTilt->setNum( newTiltX );
   lblYTilt->setNum( newTiltY );
}
```

tabletstats.h

```
#ifndef _TABLET_STATS_
#define _TABLET_STATS_

#include <qwidget.h>
#include <qframe.h>
#include "canvas.h"
#include "tabletstatsbase.h"

class QLabel;

class MyOrientation : public QFrame
{
   Q_OBJECT
public:
   MyOrientation( QWidget *parent = 0, const char *name = 0 );
   virtual ~MyOrientation();

public slots:
   void newOrient( int tiltX, int tiltY );

};

class StatsCanvas : public Canvas
{
   Q_OBJECT
public:
   StatsCanvas( QWidget *parent = 0, const char* name = 0 );
   ~StatsCanvas();
signals:
   void signalNewPressure( int );
   void signalNewTilt( int, int );
```

```
    void signalNewDev( int );
    void signalNewLoc( int, int );

protected:
    void tabletEvent( QTabletEvent *e );
    void mouseMoveEvent( QMouseEvent *e );
    void paintEvent( QPaintEvent *e );
    void mousePressEvent( QMouseEvent *e );
    void mouseReleaseEvent( QMouseEvent *e );

private:
    QRect r;
};

class TabletStats : public TabletStatsBase
{
    Q_OBJECT
public:
    TabletStats( QWidget *parent, const char* name );
    ~TabletStats();

private slots:
    void slotTiltChanged( int newTiltX, int newTiltY );
    void slotDevChanged( int newDev );
    void slotLocationChanged( int newX, int newY );

protected:
};

#endif
```

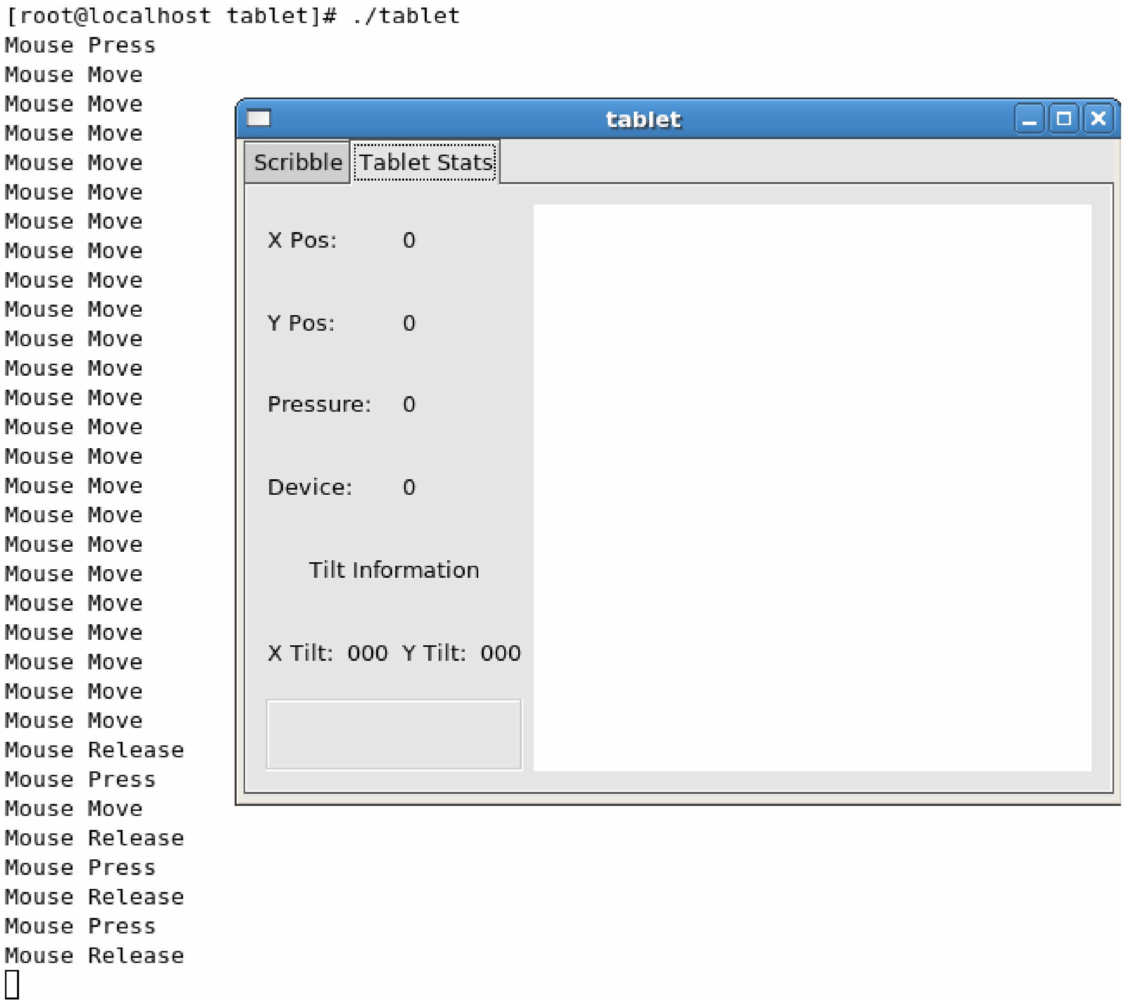
main.cpp

```
#include "scribble.h"
#include "tabletstats.h"
#include <qapplication.h>
#include <qtabwidget.h>

int main( int argc, char **argv )
{
    QApplication a( argc, argv );
    QTabWidget tab;
    Scribble scribble(&tab, "scribble");
    TabletStats tabStats( &tab, "tablet stats" );

    scribble.setMinimumSize( 500, 350 );
    tabStats.setMinimumSize( 500, 350 );
    tab.addTab(&scribble, "Scribble" );
    tab.addTab(&tabStats, "Tablet Stats" );

    a.setMainWidget( &tab );
    if ( QApplication::desktop()->width() > 550
      && QApplication::desktop()->height() > 366 )
     tab.show();
    else
    tab.showMaximized();
  return a.exec();
}
```

tabletstatsbase.ui

```
<!DOCTYPE UI><UI version="3.0" stdsetdef="1">
<class>TabletStatsBase</class>
<layoutdefaults spacing="6" margin="11"/>
<widget class="QWidget">
    <property name="name">
        <cstring>TabletStatsBase</cstring>
…
        <slot>setNum(int)</slot>
    </connection>
</connections>
</UI>
```

실행



```
[root@localhost tablet]# ./tablet
Mouse Press
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Move
Mouse Release
Mouse Press
Mouse Move
Mouse Release
Mouse Press
Mouse Release
Mouse Press
Mouse Release
```

X Pos: 0

Y Pos: 0

Pressure: 0

Device: 0

Tilt Information

X Tilt: 000 Y Tilt: 000

63. Tetrix

이것은 잘 알려진 게임 Tetris 를 Qt 로 실현한것이다.

tetrix.pro

```
TEMPLATE  = app
TARGET      = tetrix
CONFIG      += qt warn_on release
HEADERS         = gtetrix.h \
            qdragapp.h \
            qtetrix.h \
            qtetrixb.h \
            tpiece.h
SOURCES         = gtetrix.cpp \
            qdragapp.cpp \
            qtetrix.cpp \
            qtetrixb.cpp \
            tetrix.cpp \
            tpiece.cpp
```

gtetrix.cpp

```
#include "gtetrix.h"
#include <string.h>
GenericTetrix::GenericTetrix(int boardWidth,int boardHeight)
{
   int i,j;

   width    = boardWidth;
   height   = boardHeight;
   boardPtr = new int[height*width]; // Note the order, this makes it easier
                                  // to remove full lines.
   for(i = 0 ; i < height ; i++)
      for(j = 0 ; j < width ; j++)
         board(j,i) = 0;
   currentLine      = -1;          // -1 if no falling piece.
   currentPos       = 0;
   showNext         = 0;           // FALSE
   nLinesRemoved    = 0;
   nPiecesDropped   = 0;
   score            = 0;
   level            = 1;
   gameID           = 0;
   nClearLines      = height;
}

GenericTetrix::~GenericTetrix()
{
   delete[] boardPtr;
}

void GenericTetrix::clearBoard(int fillRandomLines)
{
   int i,j;

   if (fillRandomLines >= height)
```

```
      fillRandomLines = height - 1;

   erasePiece();
   for(i = height - nClearLines - 1 ; i >= fillRandomLines ; i--)
      for(j = 0 ; j < width ; j++)
         if (board(j,i) != 0) {
            draw(j,i,0);
            board(j,i) = 0;
         }
   if (fillRandomLines != 0)
      for (i = 0 ; i < fillRandomLines ; i++) {
         fillRandom(i);
   }
   nClearLines = height - fillRandomLines;
}

void GenericTetrix::showBoard()
{
   int i,j;

   showPiece();
   for(i = height - nClearLines - 1 ; i >= 0 ; i--)
      for(j = 0 ; j < width ; j++)
         if (board(j,i) != 0)
            draw(j,i,board(j,i));
}

void GenericTetrix::hideBoard()
{
   int i,j;

   erasePiece();
   for(i = height - nClearLines - 1 ; i >= 0 ; i--)
      for(j = 0 ; j < width ; j++)
         if (board(j,i) != 0)
            draw(j,i,0);
}

void GenericTetrix::startGame(int gameType,int fillRandomLines)
{
   gameID          = gameType;
   clearBoard(fillRandomLines);
   nLinesRemoved   = 0;
   updateRemoved(nLinesRemoved);
   nClearLines     = height;
   nPiecesDropped  = 0;
   score           = 0;
   updateScore(score);
   level           = 1;
   updateLevel(level);
   newPiece();
}

void GenericTetrix::revealNextPiece(int revealIt)
```

```
{
   if (showNext == revealIt)
      return;
   showNext = revealIt;
   if (!showNext)
      eraseNextPiece();
   else
      showNextPiece();
}

void GenericTetrix::updateBoard(int x1,int y1,int x2, int y2, int dontUpdateBlanks)
{
   int i,j;
   int tmp;

   if (x1 > x2) {
      tmp = x2;
      x2  = x1;
      x1  = tmp;
   }
   if (y1 > y2) {
      tmp = y2;
      y2  = y1;
      y1  = tmp;
   }
   if (x1 < 0)
      x1 = 0;
   if (x2 >= width)
      x2 = width - 1;
   if (y1 < 0)
      y1 = 0;
   if (y2 >= height)
      y2 = height - 1;
   for(i = y1 ; i <= y2 ; i++)
      for(j = x1 ; j <=  x2 ; j++)
        if (!dontUpdateBlanks || board(j,height - i - 1) != 0)
             draw(j,height - i - 1,board(j,height - i - 1));
   showPiece();        // Remember to update piece correctly!!!!
}

void GenericTetrix::fillRandom(int line)
{
   int i,j;
   int holes;

   for(i = 0 ; i < width ; i++)
      board(i,line) = TetrixPiece::randomValue(7);
   holes = 0;
   for(i = 0 ; i < width ; i++)
      if (board(i,line) == 0)   // Count holes in the line.
         holes++;
   if (holes == 0)               // Full line, make a random hole:
      board(TetrixPiece::randomValue(width),line) = 0;
   if (holes == width)           // Empty line, make a random square:
      board(TetrixPiece::randomValue(width),line) = TetrixPiece::randomValue(6) + 1;
   for(j = 0 ; j < width ; j++)
      draw(j,i,board(j,i));
}

void GenericTetrix::moveLeft(int steps)
{
   while(steps) {
      if (!canMoveTo(currentPos - 1,currentLine))
         return;
      moveTo(currentPos - 1,currentLine);
      steps--;
   }
}

void GenericTetrix::moveRight(int steps)
{
   while(steps) {
      if (!canMoveTo(currentPos + 1,currentLine))
         return;
      moveTo(currentPos + 1,currentLine);
      steps--;
   }
}

void GenericTetrix::rotateLeft()
{
   TetrixPiece tmp(currentPiece);

   tmp.rotateLeft();
   if (!canPosition(tmp))
      return;
   position(tmp);
   currentPiece = tmp;
}

void GenericTetrix::rotateRight()
{
   TetrixPiece tmp(currentPiece);

   tmp.rotateRight();
   if (!canPosition(tmp))
      return;
   position(tmp);
   currentPiece = tmp;
}

void GenericTetrix::dropDown()
{
   if (currentLine == -1)
      return;

   int dropHeight = 0;
   int newLine    = currentLine;
```

```
    while(newLine) {
        if (!canMoveTo(currentPos,newLine - 1))
            break;
        newLine--;
        dropHeight++;
    }
    if (dropHeight != 0)
        moveTo(currentPos,newLine);
    internalPieceDropped(dropHeight);
}

void GenericTetrix::oneLineDown()
{
    if (currentLine == -1)
        return;
    if (canMoveTo(currentPos,currentLine - 1)) {
        moveTo(currentPos,currentLine - 1);
    } else {
      internalPieceDropped(0);
    }
}

void GenericTetrix::newPiece()
{
    currentPiece = nextPiece;
    if (showNext)
        eraseNextPiece();
    nextPiece.setRandomType();
    if (showNext)
        showNextPiece();
    currentLine = height - 1 + currentPiece.getMinY();
    currentPos  = width/2 + 1;
    if (!canMoveTo(currentPos,currentLine)) {
      currentLine = -1;
        gameOver();
    } else {
        showPiece();
    }
}

void GenericTetrix::removePiece()
{
    erasePiece();
    currentLine = -1;
}

void GenericTetrix::drawNextSquare(int,int,int)
{
}

void GenericTetrix::pieceDropped(int)
{
    newPiece();
}
```

```
void GenericTetrix::updateRemoved(int)
{
}

void GenericTetrix::updateScore(int)
{
}

void GenericTetrix::updateLevel(int)
{
}

void GenericTetrix::removeFullLines()
{
   int i,j,k;
   int nFullLines;

   for(i = 0 ; i < height - nClearLines ; i++) {
      for(j = 0 ; j < width ; j++)
         if (board(j,i) == 0)
            break;
      if (j == width) {
       nFullLines = 1;
       for(k = i + 1 ; k < height - nClearLines ; k++) {
            for(j = 0 ; j < width ; j++)
               if (board(j,k) == 0)
              break;
        if (j == width) {
           nFullLines++;
        } else {
               for(j = 0 ; j < width ; j++) {
              if (board(j,k - nFullLines) != board(j,k)) {
                board(j,k - nFullLines) = board(j,k);
                draw(      j,k - nFullLines,
                        board(j,k - nFullLines));
              }
           }
        }
       }
       nClearLines   = nClearLines + nFullLines;
       nLinesRemoved = nLinesRemoved + nFullLines;
       updateRemoved(nLinesRemoved);
       score = score + 10*nFullLines; // updateScore must be called by caller!
       for (i = height - nClearLines              ;
           i < height - nClearLines + nFullLines ;
         i++)
          for(j = 0 ; j < width ; j++)
           if (board(j,i) != 0) {
            draw(j,i,0);
            board(j,i) = 0;
           }
    }
   }
```

```
}

void GenericTetrix::showPiece()
{
   int x,y;

   if (currentLine == -1)
      return;

   for(int i = 0 ; i < 4 ; i++) {
      currentPiece.getCoord(i,x,y);
      draw(currentPos + x,currentLine - y,currentPiece.getType());
   }
}

void GenericTetrix::erasePiece()
{
   int x,y;

   if (currentLine == -1)
      return;

   for(int i = 0 ; i < 4 ; i++) {
      currentPiece.getCoord(i,x,y);
      draw(currentPos + x,currentLine - y,0);
   }
}

void GenericTetrix::internalPieceDropped(int dropHeight)
{
   gluePiece();
   nPiecesDropped++;
   if (nPiecesDropped % 25 == 0) {
      level++;
    updateLevel(level);
   }
   score = score + 7 + dropHeight;
   removeFullLines();
   updateScore(score);
   pieceDropped(dropHeight);
}

void GenericTetrix::gluePiece()
{
   int x,y;
   int min;

   if (currentLine == -1)
      return;

   for(int i = 0 ; i < 4 ; i++) {
      currentPiece.getCoord(i,x,y);
      board(currentPos + x,currentLine - y) = currentPiece.getType();
   }
```

```
    min = currentPiece.getMinY();
    if (currentLine - min >= height - nClearLines)
        nClearLines = height - currentLine + min - 1;
}

void GenericTetrix::showNextPiece(int erase)
{
    int x,y;
    int minX = nextPiece.getMinX();
    int minY = nextPiece.getMinY();
    int maxX = nextPiece.getMaxX();
    int maxY = nextPiece.getMaxY();

    int xOffset = (3 - (maxX - minX))/2;
    int yOffset = (3 - (maxY - minY))/2;

    for(int i = 0 ; i < 4 ; i++) {
        nextPiece.getCoord(i,x,y);
      if (erase)
            drawNextSquare(x + xOffset - minX,
                        y + yOffset - minY,0);
      else
            drawNextSquare(x + xOffset - minX,
                        y + yOffset - minY,nextPiece.getType());
    }
}

int GenericTetrix::canPosition(TetrixPiece &piece)
{
    if (currentLine == -1)
        return 0;

    int x,y;

    for(int i = 0 ; i < 4 ; i++) {
        piece.getCoord(i,x,y);
        x = currentPos + x;
        y = currentLine - y; // Board and pieces have inverted y-coord. systems.
        if (x < 0 || x >= width || y < 0 || y >= height)
            return 0;     // Outside board, cannot put piece here.
        if (board(x,y) != 0)
            return 0;     // Over a non-zero square, cannot put piece here.
    }
    return 1;             // Inside board and no non-zero squares underneath.

}

int GenericTetrix::canMoveTo(int xPosition,int line)
{
    if (currentLine == -1)
        return 0;

    int x,y;
    for(int i = 0 ; i < 4 ; i++) {
        currentPiece.getCoord(i,x,y);
        x = xPosition + x;
        y = line - y;     // Board and pieces have inverted y-coord. systems.
        if (x < 0 || x >= width || y < 0 || y >= height)
            return 0;     // Outside board, cannot put piece here.
        if (board(x,y) != 0)
            return 0;     // Over a non-zero square, cannot put piece here.
    }
    return 1;             // Inside board and no non-zero squares underneath.
}

void GenericTetrix::moveTo(int xPosition,int line)
{
    if (currentLine == -1)
        return;
    optimizedMove(xPosition,line,currentPiece);
    currentPos  = xPosition;
    currentLine = line;
}

void GenericTetrix::position(TetrixPiece &piece)
{
    if (currentLine == -1)
        return;

    optimizedMove(currentPos,currentLine,piece);
}

void GenericTetrix::optimizedMove(int newPos, int newLine, TetrixPiece &newPiece)
{
    int updates [8][3];
    int nUpdates;
    int value;
    int x,y;
    int i,j;

    for(i = 0 ; i < 4 ; i++) { // Put the erasing coords into updates
        currentPiece.getCoord(i,x,y);
      updates[i][0] = currentPos  + x;
      updates[i][1] = currentLine - y;
      updates[i][2] = 0;
    }
    nUpdates = 4;
    for(i = 0 ; i < 4 ; i++) { // Any drawing coord same as an erasing one?
        newPiece.getCoord(i,x,y);
      x = newPos  + x;
      y = newLine - y;
      for (j = 0 ; j < 4 ; j++)
          if (updates[j][0] == x && updates[j][1] == y) { // Same coord, don't have to erase
              if (currentPiece.getType() == newPiece.getType())
                  updates[j][2] = -1; // Correct on screen, no update!
              else
                  updates[j][2] = newPiece.getType();
          break;
        }
      if (j == 4) {         // This coord does not overlap an erasing one
        updates[nUpdates][0] = x;
        updates[nUpdates][1] = y;
        updates[nUpdates][2] = newPiece.getType();
        nUpdates++;
      }
    }
    for (i = 0 ; i < nUpdates ; i++) {  // Do the updating
      x     = updates[i][0];
      y     = updates[i][1];
      value = updates[i][2];
      if (value != -1)              // Only update if new value != current
        draw(x,y,value);
    }
}
```

gtetrix.h

```
#ifndef GTETRIX_H
#define GTETRIX_H
#include "tpiece.h"

class GenericTetrix
{
public:
    GenericTetrix(int boardWidth = 10,int boardHeight = 22);
    virtual ~GenericTetrix();

    void clearBoard(int fillRandomLines = 0);
    void revealNextPiece(int revealIt);
    void updateBoard(int x1,int y1,int x2,int y2,int dontUpdateBlanks = 0);
    void updateNext(){if (showNext) showNextPiece();}
    void hideBoard();
    void showBoard();
    void fillRandom(int line);

    void moveLeft(int steps = 1);
    void moveRight(int steps = 1);
    void rotateLeft();
    void rotateRight();
    void dropDown();
    void oneLineDown();
    void newPiece();
    void removePiece();

    int  noOfClearLines()                {return nClearLines;}
    int  getLinesRemoved()               {return nLinesRemoved;}
    int  getPiecesDropped()              {return nPiecesDropped;}
    int  getScore()                      {return score;}
    int  getLevel()                      {return level;}
    int  boardHeight()                   {return height;}
    int  boardWidth()                    {return width;}
    virtual void drawSquare(int x,int y,int value) = 0;
    virtual void gameOver() = 0;

    virtual void startGame(int gameType = 0,int fillRandomLines = 0);
    virtual void drawNextSquare(int x,int y,int value);
    virtual void pieceDropped(int dropHeight);
    virtual void updateRemoved(int noOfLines);
    virtual void updateScore(int newScore);
    virtual void updateLevel(int newLevel);

private:
    void  draw(int x, int y, int value){drawSquare(x,height - y,value);}
    void  removeFullLines();
    void  removeLine(int line);
    void  showPiece();
    void  erasePiece();
    void  internalPieceDropped(int dropHeight);
    void  gluePiece();
    void  showNextPiece(int erase = 0);
    void  eraseNextPiece(){showNextPiece(1);};
    int   canPosition(TetrixPiece &piece);    // Returns a boolean value.
    int   canMoveTo(int xPosition, int line); // Returns a boolean value.
    void  moveTo(int xPosition,int line);
    void  position(TetrixPiece &piece);
    void  optimizedMove(int newPos, int newLine,TetrixPiece &newPiece);

    int  &board(int x,int y){return boardPtr[width*y + x];}

    TetrixPiece currentPiece;
    TetrixPiece nextPiece;
    int         currentLine;
    int         currentPos;
    int         showNext;                 // Boolean variable.
    int         nLinesRemoved;
    int         nPiecesDropped;
    int         score;
    int         level;
    int         gameID;
    int         nClearLines;
    int         width;
    int         height;
    int         *boardPtr;
};

#endif
```

qdragapp.cpp

```
#include "qdragapp.h"
#include "qptrlist.h"
#include "qintdict.h"
#include "qpopupmenu.h"
#include "qguardedptr.h"
#include "qcolor.h"
#include "qwidget.h"
```

```
#include "qfontmetrics.h"
#include "qcursor.h"
#include "qobjectlist.h"

QWidget *cursorWidget( QPoint * = 0 );

class QDragger;

class DropWindow : public QWidget
{
   Q_OBJECT
public:
   void paintEvent( QPaintEvent * );
   void closeEvent( QCloseEvent * );

   QDragger *master;
};

struct DropInfo {
   DropInfo() { w=0; }
  ~DropInfo()  { delete w; }
   DropWindow *w;
   bool userOpened;
};

struct DraggedInfo {
   QWidget *w;
   QWidget *mother;
   QPoint   pos;
};


class QDragger : public QObject
{
   Q_OBJECT
public:
   QDragger();
   ~QDragger();

   bool notify( QObject *, QEvent * ); // event filter
   void closeDropWindow( DropWindow * );
public slots:
   void openDropWindow();
   void killDropWindow();
   void killAllDropWindows();
   void sendChildHome();
   void sendAllChildrenHome();
private:
   bool isParentToDragged( QWidget * );
   bool noWidgets( QWidget * );
   void killDropWindow( DropInfo * );
   void killAllDropWindows( bool );
   void sendChildHome( DraggedInfo * );
   void sendAllChildrenHome( QWidget * );
```

```
    QWidget *openDropWindow( const QRect&, bool );

    bool startGrab();
    void grabFinished();
    bool dragEvent( QWidget *, QMouseEvent * );
    bool killDropEvent( QMouseEvent * );
    bool sendChildEvent( QMouseEvent * );

    bool         killingDrop;
    bool         sendingChild;
    QWidget            *clickedWidget;
    QGuardedPtr<QWidget>   hostWidget;
    QCursor              cursor;

    QPopupMenu*              menu;
    QPoint          clickOffset;
    QColor          dragBackground;
    QColor          dragForeground;
    DraggedInfo            dragInfo;
    QIntDict<DraggedInfo>  draggedDict;
    QIntDict<DropInfo>     dropDict;
};

QDragApplication::QDragApplication( int &argc, char **argv )
    : QApplication( argc, argv ), dragger( 0 )
{
    dragger = new QDragger;
}

QDragApplication::~QDragApplication()
{
    delete dragger;
}

bool QDragApplication::notify( QObject *o, QEvent *e )
{
    if ( dragger && !dragger->notify( o, e ) )
     return QApplication::notify( o, e );
    else
     return FALSE;
}

void DropWindow::paintEvent( QPaintEvent * )
{
    const char *msg    = "Drag widgets and drop them here or anywhere!";
    int         startX = ( width() - fontMetrics().width( msg ) )/2;
    startX          = startX < 0 ? 0 : startX;

    drawText( startX, height()/2, msg );
}

void DropWindow::closeEvent( QCloseEvent *e )
{
    master->closeDropWindow( this );
    e->ignore();
}

QDragger::QDragger()
{
    dragInfo.w  = 0;
    killingDrop = FALSE;
    sendingChild = FALSE;
    draggedDict.setAutoDelete( TRUE );
    dropDict   .setAutoDelete( TRUE );

    menu = new QPopupMenu;
    menu->insertItem( "Open drop window", 1 );
    menu->insertItem( "Kill drop window", 2 );
    menu->insertItem( "Kill all drop windows", 3 );
    menu->insertSeparator();
//    menu->insertItem( "Send child home", 4 );
    menu->insertItem( "Send all children home", 5 );

    menu->connectItem( 1, this, SLOT(openDropWindow()) );
    menu->connectItem( 2, this, SLOT(killDropWindow()) );
    menu->connectItem( 3, this, SLOT(killAllDropWindows()) );
//    menu->connectItem( 4, this, SLOT(sendChildHome()) );
    menu->connectItem( 5, this, SLOT(sendAllChildrenHome()) );
}

QDragger::~QDragger()
{
    delete menu;
}

bool QDragger::notify( QObject *o, QEvent *e )
{
    if ( !o->isWidgetType() || o == menu )
      return FALSE;
    switch( e->type() ) {
      case QEvent::MouseMove:
          {
           QMouseEvent *tmp = (QMouseEvent*) e;
           if ( killingDrop )
               return killDropEvent( tmp );
           if ( sendingChild )
               return sendChildEvent( tmp );
           if ( tmp->state() & QMouseEvent::RightButton )
               return dragEvent( (QWidget*) o, tmp );
           break;
          }
      case QEvent::MouseButtonPress:
      case QEvent::MouseButtonRelease:
      case QEvent::MouseButtonDblClick:
          {
           QMouseEvent *tmp = (QMouseEvent*) e;
           if ( killingDrop )
               return killDropEvent( tmp );
```

```
       if ( sendingChild )
         return sendChildEvent( tmp );
       if ( tmp->button() == QMouseEvent::RightButton )
         return dragEvent( (QWidget*) o, tmp );
      }
      break;
   default:
      break;
  }
  return FALSE;
}

bool QDragger::isParentToDragged( QWidget *w )
{
  QIntDictIterator<DraggedInfo> iter( draggedDict );

  DraggedInfo *tmp;
  while( (tmp = iter.current()) ) {
   ++iter;
   if ( tmp->mother == w )
     return TRUE;
  }
  return FALSE;
}

bool QDragger::noWidgets( QWidget *w )
{
  const QObjectList *l = w->children();
  if ( !l )
   return TRUE;
  QObjectListIt iter( *l );
  QObject *tmp;
  while( (tmp = iter.current()) ) {
   ++iter;
   if ( tmp->isWidgetType() )
     return FALSE;
  }
  return TRUE;
}

void QDragger::sendAllChildrenHome( QWidget *w )
{
  const QObjectList *l = w->children();
  if ( !l )
   return;
  QObjectListIt iter( *l );
  QObject *tmp;
  while( (tmp = iter.current()) ) {
   ++iter;
   if ( tmp->isWidgetType() ) {
     sendAllChildrenHome( (QWidget*) tmp );
     DraggedInfo *di = draggedDict.find( (long) tmp );
     if ( di )
      sendChildHome( di );
   }
  }
}

bool QDragger::dragEvent( QWidget *w, QMouseEvent *e )
{
  switch( e->type() ) {
   case QEvent::MouseButtonDblClick:
   case QEvent::MouseButtonPress: {
     if ( !noWidgets( w ) || // has widget children
       isParentToDragged( w )|| // has had widget children
       w->parentWidget() == 0 ) {       // is top level window
      hostWidget = w;
      menu->popup( w->mapToGlobal( e->pos() ) );
      return TRUE;
     }
     if ( !draggedDict.find( (long) w ) ) {
      DraggedInfo *tmp = new DraggedInfo;
      tmp->w         = w;
      tmp->mother      = w->parentWidget();
      tmp->pos        = w->frameGeometry().topLeft();
      draggedDict.insert( (long) w, tmp );
     }
     dragBackground    = w->backgroundColor();
     dragForeground     = w->foregroundColor();
     dragInfo.w  = w;
     dragInfo.mother = w->parentWidget();
     dragInfo.pos     = w->frameGeometry().topLeft();
     clickOffset = e->pos();
     dragInfo.w  = w;
     QPoint p    = w->mapToGlobal(QPoint(0,0));
     w->reparent( 0, WType_Popup, p, TRUE );

     return TRUE;
   }
   case QEvent::MouseButtonRelease:
   case QEvent::MouseMove: {
     if ( dragInfo.w != 0 ) {
      QPoint p = QCursor::pos() - clickOffset;
      dragInfo.w->move( p );
      if ( e->type() == QEvent::MouseMove )
        return TRUE;
     } else {
      return FALSE;
     }
     if ( !dragInfo.w )
      return FALSE;
     if ( w != dragInfo.w )
      w = dragInfo.w;
     dragInfo.w = 0;
     w->hide();
     QPoint pos;
     QWidget *target = cursorWidget( &pos );
     pos = pos - clickOffset;
```

```
        QPoint p;
        if ( !target ) {
          target = openDropWindow( QRect( pos, w->size() ),
                      FALSE);
          p = QPoint( 0, 0 );
        }
        else
          p = target->mapFromGlobal( pos );
        w->reparent( target, 0, p, TRUE );
        DropInfo *tmp = dropDict.find( (long) dragInfo.mother );
        if ( tmp ) {
          if ( !tmp->userOpened && noWidgets( tmp->w ) )
            dropDict.remove( (long) tmp->w );
        }
        if ( !target->isVisible() )
          target->show();
      }
      return TRUE;
    default:
      return FALSE;
  }
}

bool QDragger::killDropEvent( QMouseEvent *e )
{
  switch( e->type() ) {
    case QEvent::MouseButtonDblClick:
    case QEvent::MouseButtonPress:
      clickedWidget = cursorWidget();
      return TRUE;
    case QEvent::MouseButtonRelease:
      hostWidget->releaseMouse();
      if ( clickedWidget ) {
        DropInfo *tmp = dropDict.find( (long) clickedWidget );
        if( tmp ) {
          killDropWindow( tmp );
          dropDict.remove( (long) tmp->w );
        }
      }
      grabFinished();
      return TRUE;
    case QEvent::MouseMove:
      return TRUE;
    default:
      break;
  }
  return FALSE;
}

bool QDragger::sendChildEvent( QMouseEvent *e )
{
  switch( e->type() ) {
    case QEvent::MouseButtonDblClick:
    case QEvent::MouseButtonPress:
     clickedWidget = cursorWidget();
     return TRUE;
   case QEvent::MouseButtonRelease:
     hostWidget->releaseMouse();
     if ( clickedWidget ) {
      DraggedInfo *tmp = draggedDict.find((long) clickedWidget);
      if( tmp ) {
        QWidget *parent = tmp->w->parentWidget();
        sendChildHome( tmp );
        DropInfo *dri = dropDict.find( (long) parent );
        if ( dri && noWidgets(dri->w) && !dri->userOpened ) {
         killDropWindow( dri );
         dropDict.remove( (long) dri );
        }
      }
      grabFinished();
     }
     return TRUE;
   case QEvent::MouseMove:
     return TRUE;
   default:
     break;
  }
  return FALSE;
}

bool QDragger::startGrab()
{
  if ( !hostWidget )
   return FALSE;
  clickedWidget = 0;
  cursor     = hostWidget->cursor();
  hostWidget->grabMouse();
  hostWidget->setCursor( QCursor( CrossCursor ) );
  return TRUE;
}

void QDragger::grabFinished()
{
  killingDrop = FALSE;
  sendingChild = FALSE;
  if(hostWidget)
   hostWidget->setCursor( cursor );
}

void QDragger::closeDropWindow( DropWindow *w )
{
  DropInfo *tmp = dropDict.find( (long) w);
  if( tmp )
   killDropWindow( tmp );
}

void QDragger::openDropWindow()
{
  QWidget *tmp = openDropWindow( QRect(100, 100, 300, 200), TRUE );
  tmp->show();
}

QWidget *QDragger::openDropWindow( const QRect &r, bool user )
{
  DropInfo *tmp = new DropInfo;
  DropWindow *w = new DropWindow;
  if ( user ) {
   tmp->userOpened = TRUE;
   w->setCaption( "Drop window" );
  } else {
   tmp->userOpened = FALSE;
   w->setCaption( "Auto drop window" );
  }
  tmp->w = w;
  w->master = this;
  w->setGeometry( r );
  dropDict.insert( (long) w, tmp );
  w->show();
  return w;
}

void QDragger::killDropWindow()
{
  if ( startGrab() )
   killingDrop   = TRUE;
}

void QDragger::killDropWindow( DropInfo *di )
{
  const QObjectList *l = di->w->children();
  if ( !l )
   return;
  QObjectListIt iter( *l );
  QObject *tmp;
  while( (tmp = iter.current()) ) {
   ++iter;
   if ( tmp->isWidgetType() ) {
     DraggedInfo *dri = draggedDict.find( (long) tmp );
     if ( dri ) {
      sendChildHome( dri );
      draggedDict.remove( (long) tmp );
     }
   }
  }
  di->w->hide();
}

void QDragger::killAllDropWindows()
{
  killAllDropWindows( FALSE );
}
```

```
void QDragger::killAllDropWindows( bool autoOnly )
{
   QIntDictIterator<DropInfo> iter( dropDict );

   DropInfo *tmp;
   while( (tmp = iter.current()) ) {
    ++iter;
    if( !autoOnly || !tmp->userOpened ) {
       killDropWindow( tmp );
       dropDict.remove( (long) tmp->w );
    }
   }
}

void QDragger::sendChildHome( DraggedInfo *i )
{
   i->w->reparent( i->mother, 0, i->pos, TRUE );
}

void QDragger::sendChildHome()
{
   if ( startGrab() )
    sendingChild  = TRUE;
}

void QDragger::sendAllChildrenHome()
{
   QIntDictIterator<DraggedInfo> iter( draggedDict );

   DraggedInfo *tmp;
   while( (tmp = iter.current()) ) {
    ++iter;
    sendChildHome( tmp );
    draggedDict.remove( (long) tmp->w );
   }
   killAllDropWindows( TRUE );
   draggedDict.clear();
}

QWidget *cursorWidget( QPoint *p )
{
   QPoint curpos = QCursor::pos();
   if ( p )
    *p = curpos;
   return QApplication::widgetAt( curpos );
}

#include "qdragapp.moc"
```

qdragapp.h

```
#ifndef QDRAGAPP_H
#define QDRAGAPP_H

#include "qapplication.h"
```

```
class QDragger;

class QDragApplication : public QApplication
{
   Q_OBJECT
public:
   QDragApplication( int &argc, char **argv );
   virtual ~QDragApplication();

   virtual bool notify( QObject *, QEvent * ); // event filter

private:
   QDragger *dragger;
};

#endif // QDRAGAPP_H
```

qtetrix.cpp

```
#include "qtetrix.h"
#include <qapplication.h>
#include <qlabel.h>
#include <qdatetime.h>

void drawTetrixButton( QPainter *p, int x, int y, int w, int h,  const QColor *color, QWidget *widg)
{
   if ( color ) {
      QPointArray a;
    a.setPoints( 3,  x,y+h-1, x,y, x+w-1,y );
    p->setPen( color->light() );
    p->drawPolyline( a );
    a.setPoints( 3, x+1,y+h-1, x+w-1,y+h-1, x+w-1,y+1 );
    p->setPen( color->dark() );
    p->drawPolyline( a );
    x++;
    y++;
    w -= 2;
    h -= 2;
    p->fillRect( x, y, w, h, *color );
   }
   else if(widg) {
    widg->erase(x, y, w, h);
   } else {
    p->fillRect(x, y, w, h, p->backgroundColor());
   }
}


ShowNextPiece::ShowNextPiece( QWidget *parent, const char *name ) : QFrame( parent, name )
{
   setFrameStyle( QFrame::Panel | QFrame::Sunken );
   xOffset = -1;     // -1 until first resizeEvent.
}
```

```
void ShowNextPiece::resizeEvent( QResizeEvent *e )
{
   QSize sz = e->size();
   blockWidth  = (sz.width()  - 3)/5;
   blockHeight = (sz.height() - 3)/6;
   xOffset     = (sz.width()  - 3)/5;
   yOffset     = (sz.height() - 3)/6;
}

void ShowNextPiece::paintEvent( QPaintEvent * )
{
   QPainter p( this );
   drawFrame( &p );
   p.end();            // explicit end() so any slots can paint too
   emit update();
}

void ShowNextPiece::drawNextSquare(int x, int y,QColor *color)
{
   if (xOffset == -1)        // Before first resizeEvent?
      return;

   QPainter paint;
   paint.begin(this);
   drawTetrixButton( &paint, xOffset+x*blockWidth, yOffset+y*blockHeight,
             blockWidth, blockHeight, color, this );
   paint.end();
}

QTetrix::QTetrix( QWidget *parent, const char *name )
   : QWidget( parent, name )
{
   QTime t = QTime::currentTime();
   TetrixPiece::setRandomSeed( (((double)t.hour())+t.minute()+t.second())/ (24+60+60) );

#define ADD_LABEL( str, x, y, w, h )            \
   { QLabel *label = new QLabel(str,this);           \
     label->setGeometry(x,y,w,h);              \
     label->setAlignment(AlignCenter|AlignVCenter); }

   ADD_LABEL( "NEXT", 50, 10, 78, 30 );
   ADD_LABEL( "SCORE", 330, 10, 178, 30 );
   ADD_LABEL( "LEVEL", 50, 130, 78, 30 );
   ADD_LABEL( "LINES REMOVED", 330, 130, 178, 30 );

   board       = new QTetrixBoard(this);
   showNext    = new ShowNextPiece(this);
#ifndef QT_NO_LCDNUMBER
   showScore   = new QLCDNumber(5,this);
   showLevel   = new QLCDNumber(2,this);
   showLines   = new QLCDNumber(5,this);
#else
   showScore   = new QLabel("0",this);
   showLevel   = new QLabel("0",this);
```

```
    showLines   = new QLabel("0",this);
    showScore->setAlignment(AlignCenter);
    showLines->setAlignment(AlignCenter);
    showLevel->setAlignment(AlignCenter);
    showScore->setFrameStyle(QFrame::Sunken|QFrame::Box);
    showLines->setFrameStyle(QFrame::Sunken|QFrame::Box);
    showLevel->setFrameStyle(QFrame::Sunken|QFrame::Box);
#endif
    quitButton  = new QPushButton("&Quit",this);
    startButton = new QPushButton("&New Game",this);
    pauseButton = new QPushButton("&Pause",this);

    // Don't let the buttons get keyboard focus
    quitButton->setFocusPolicy( QWidget::NoFocus );
    startButton->setFocusPolicy( QWidget::NoFocus );
    pauseButton->setFocusPolicy( QWidget::NoFocus );

    connect( board, SIGNAL(gameOverSignal()), SLOT(gameOver()) );
    connect( board, SIGNAL(drawNextSquareSignal(int,int,QColor*)), showNext,
          SLOT(drawNextSquare(int,int,QColor*)) );
    connect( showNext, SIGNAL(update()), board, SLOT(updateNext()) );
#ifndef QT_NO_LCDNUMBER
    connect( board, SIGNAL(updateScoreSignal(int)), showScore,  SLOT(display(int)) );
    connect( board, SIGNAL(updateLevelSignal(int)), showLevel, SLOT(display(int)));
    connect( board, SIGNAL(updateRemovedSignal(int)), showLines,  SLOT(display(int)));
#else
    connect( board, SIGNAL(updateScoreSignal(int)), showScore,  SLOT(setNum(int)) );
    connect( board, SIGNAL(updateLevelSignal(int)), showLevel,  SLOT(setNum(int)));
    connect( board, SIGNAL(updateRemovedSignal(int)), showLines,  SLOT(setNum(int)));
#endif
    connect( startButton, SIGNAL(clicked()), board, SLOT(start()) );
    connect( quitButton , SIGNAL(clicked()), SLOT(quit()));
    connect( pauseButton, SIGNAL(clicked()), board, SLOT(pause()) );

    board->setGeometry( 150, 20, 153, 333 );
    showNext->setGeometry( 50, 40, 78, 94 );
    showScore->setGeometry( 330, 40, 178, 93 );
    showLevel->setGeometry( 50, 160, 78, 93 );
    showLines->setGeometry( 330, 160, 178, 93 );
#ifndef QT_NO_LCDNUMBER
    showScore->display( 0 );
    showLevel->display( 0 );
    showLines->display( 0 );
#else
    showScore->setNum( 0 );
    showLevel->setNum( 0 );
    showLines->setNum( 0 );
#endif
    startButton->setGeometry( 46, 288, 90, 30 );
    quitButton->setGeometry( 370, 265, 90, 30 );
    pauseButton->setGeometry( 370, 310, 90, 30 );
    board->revealNextPiece(TRUE);

    resize( 550, 370 );
```

```
}

void QTetrix::gameOver()
{
}

void QTetrix::quit()
{
   qApp->quit();
}
```

qtetrix.h

```
#ifndef QTETRIX_H
#define QTETRIX_H

#include "qtetrixb.h"
#include <qframe.h>
#include <qlcdnumber.h>
#include <qlabel.h>
#include <qpushbutton.h>
#include <qpainter.h>

class ShowNextPiece : public QFrame
{
   Q_OBJECT
   friend class QTetrix;
public:
   ShowNextPiece( QWidget *parent=0, const char *name=0  );
public slots:
   void drawNextSquare( int x, int y,QColor *color );
signals:
   void update();
private:
   void paintEvent( QPaintEvent * );
   void resizeEvent( QResizeEvent * );

   int     blockWidth,blockHeight;
   int     xOffset,yOffset;
};

class QTetrix : public QWidget
{
   Q_OBJECT
public:
   QTetrix( QWidget *parent=0, const char *name=0 );
   void startGame() { board->startGame(); }

public slots:
   void gameOver();
   void quit();
private:
   void keyPressEvent( QKeyEvent *e ) { board->keyPressEvent(e); }

   QTetrixBoard  *board;
```

```
    ShowNextPiece *showNext;
#ifndef QT_NO_LCDNUMBER
    QLCDNumber    *showScore;
    QLCDNumber    *showLevel;
    QLCDNumber    *showLines;
#else
    QLabel    *showScore;
    QLabel    *showLevel;
    QLabel    *showLines;
#endif
    QPushButton   *quitButton;
    QPushButton   *startButton;
    QPushButton   *pauseButton;
};

void drawTetrixButton( QPainter *, int x, int y, int w, int h, const QColor *color, QWidget *widg);

#endif
```

qtetrixb.cpp

```
#include "qtetrixb.h"
#include "qtetrix.h"
#include <qtimer.h>
#include <qpainter.h>

const int waitAfterLineTime = 500;

QTetrixBoard::QTetrixBoard( QWidget *p, const char *name ) : QFrame( p, name )
{
    setFrameStyle( QFrame::Panel | QFrame::Sunken );
    paint = 0;
    paint_widget = 0;
    timer = new QTimer(this);
    connect( timer, SIGNAL(timeout()), SLOT(timeout()) );

    colors[0].setRgb(200,100,100);
    colors[1].setRgb(100,200,100);
    colors[2].setRgb(100,100,200);
    colors[3].setRgb(200,200,100);
    colors[4].setRgb(200,100,200);
    colors[5].setRgb(100,200,200);
    colors[6].setRgb(218,170,  0);

    xOffset          = -1;      // -1 until a resizeEvent is received.
    blockWidth       = 20;
    yOffset          = 30;
    blockHeight      = 20;
    noGame           = TRUE;
    isPaused         = FALSE;
    waitingAfterLine = FALSE;
    updateTimeoutTime();    // Sets timeoutTime
}

void QTetrixBoard::startGame(int gameType,int fillRandomLines)
```

```
{
   if ( isPaused )
      return;          // ignore if game is paused
   noGame = FALSE;
   GenericTetrix::startGame( gameType, fillRandomLines );
   // Note that the timer is started by updateLevel!
}

void QTetrixBoard::pause()
{
   if ( noGame )              // game not active
      return;
   isPaused = !isPaused;
   if ( isPaused ) {
    timer->stop();
      hideBoard();
   }
   else
    timer->start(timeoutTime);
   update();
}

void QTetrixBoard::drawSquare(int x,int y,int value)
{
   if (xOffset == -1)    // Before first resizeEvent?
      return;

   const int X = xOffset  + x*blockWidth;
   const int Y = yOffset  + (y - 1)*blockHeight;

   bool localPainter = paint == 0;
   QPainter *p;
   QWidget *w;
   if ( localPainter ) {
    p = new QPainter( this );
    w = this;
   } else {
    p = paint;
    w = paint_widget;
   }
   drawTetrixButton( p, X, Y, blockWidth, blockHeight,
             value == 0 ? 0 : &colors[value-1], w);
   /*
   if ( value != 0 ) {
    QColor tc, bc;
    tc = colors[value-1].light();
    bc = colors[value-1].dark();
    p->drawShadePanel( X, Y, blockWidth, blockHeight,
              tc, bc, 1, colors[value-1], TRUE );
   }
   else
    p->fillRect( X, Y, blockWidth, blockHeight, backgroundColor() );
    */
   if ( localPainter )
```

```
    delete p;
}

void QTetrixBoard::drawNextSquare( int x, int y, int value )
{
   if ( value == 0 )
      emit drawNextSquareSignal (x, y, 0 );
   else
      emit drawNextSquareSignal( x, y, &colors[value-1] );
}

void QTetrixBoard::updateRemoved( int noOfLines )
{
   if ( noOfLines > 0 ) {
      timer->stop();
      timer->start( waitAfterLineTime );
      waitingAfterLine = TRUE;
   }
   emit updateRemovedSignal( noOfLines );
}

void QTetrixBoard::updateScore( int newScore )
{
   emit updateScoreSignal( newScore );
}

void QTetrixBoard::updateLevel( int newLevel )
{
   timer->stop();
   updateTimeoutTime();
   timer->start( timeoutTime );
   emit updateLevelSignal( newLevel );
}

void QTetrixBoard::pieceDropped(int)
{
   if ( waitingAfterLine ) // give player a break if a line has been removed
      return;
   newPiece();
}

void QTetrixBoard::gameOver()
{
   timer->stop();
   noGame = TRUE;
   emit gameOverSignal();
}

void QTetrixBoard::timeout()
{
   if ( waitingAfterLine ) {
    timer->stop();
    waitingAfterLine = FALSE;
    newPiece();
```

```
     timer->start( timeoutTime );
   } else {
      oneLineDown();
   }
}

void QTetrixBoard::drawContents( QPainter *p )
{
   const char *text = "Press \"Pause\"";
   QRect r = contentsRect();
   paint = p;                    // set widget painter
   if ( isPaused ) {
     p->drawText( r, AlignCenter | AlignVCenter, text );
      return;
   }
   int x1,y1,x2,y2;
   x1 = (r.left() - xOffset) / blockWidth;
   if (x1 < 0)
      x1 = 0;
   if (x1 >= boardWidth())
      x1 = boardWidth() - 1;

   x2 = (r.right() - xOffset) / blockWidth;
   if (x2 < 0)
      x2 = 0;
   if (x2 >= boardWidth())
      x2 = boardWidth() - 1;

   y1 = (r.top() - yOffset) / blockHeight;
   if (y1 < 0)
      y1 = 0;
   if (y1 >= boardHeight())
      y1 = boardHeight() - 1;

   y2 = (r.bottom() - yOffset) / blockHeight;
   if (y2 < 0)
      y2 = 0;
   if (y2 >= boardHeight())
      y2 = boardHeight() - 1;

   updateBoard( x1, y1, x2, y2, TRUE );
   paint = 0;                    // reset widget painter
   return;
}

void QTetrixBoard::resizeEvent(QResizeEvent *e)
{
   QSize sz = e->size();
   blockWidth  = (sz.width() - 3)/10;
   blockHeight = (sz.height() - 3)/22;
   xOffset     = 1;
   yOffset     = 1;
}
```

```
void QTetrixBoard::keyPressEvent( QKeyEvent *e )
{
   if ( noGame || isPaused || waitingAfterLine )
      return;
   switch( e->key() ) {
    case Key_Left :
       moveLeft();
       break;
    case Key_Right :
       moveRight();
       break;
    case Key_Down :
       rotateRight();
       break;
    case Key_Up :
       rotateLeft();
       break;
    case Key_Space :
       dropDown();
       break;
    case Key_D :
       oneLineDown();
       break;
      default:
       return;
   }
   e->accept();
}

void QTetrixBoard::updateTimeoutTime()
{
   timeoutTime = 1000/(1 + getLevel());
}
```

qtetrixb.h

```
#ifndef QTETRIXB_H
#define QTETRIXB_H

#include "gtetrix.h"
#include <qframe.h>

class QTimer;

class QTetrixBoard : public QFrame, public GenericTetrix
{
   Q_OBJECT
public:
   QTetrixBoard( QWidget *parent=0, const char *name=0 );

   void     gameOver();
   void     startGame(int gameType = 0,int fillRandomLines = 0);

public slots:
   void     timeout();
```

```
    void     updateNext()  { GenericTetrix::updateNext(); }
    void     key(QKeyEvent *e) { keyPressEvent(e); }
    void     start()        { startGame(); }
    void     pause();

signals:
    void     gameOverSignal();
    void     drawNextSquareSignal(int x,int y,QColor *color1);
    void     updateRemovedSignal(int noOfLines);
    void     updateScoreSignal(int score);
    void     updateLevelSignal(int level);

public:       // until we have keyboard focus, should be protected
    void     keyPressEvent( QKeyEvent * );

private:
    void     drawContents( QPainter * );
    void     resizeEvent( QResizeEvent * );
    void     drawSquare(int x,int y,int value);
    void     drawNextSquare(int x,int y,int value);
    void     updateRemoved(int noOfLines);
    void     updateScore(int newScore);
    void     updateLevel(int newLlevel);
    void     pieceDropped(int dropHeight);
    void     updateTimeoutTime();

    QTimer   *timer;

    int     xOffset,yOffset;
    int     blockWidth,blockHeight;
    int     timeoutTime;
    bool     noGame;
    bool     isPaused;
    bool     waitingAfterLine;

    QColor    colors[7];
    QPainter *paint;
    QWidget *paint_widget;
};

#endif
```

tpiece.cpp

```
#include "tpiece.h"
#include "qstring.h"
#include <stdlib.h>

void TetrixPiece::rotateLeft()
{
    if ( pieceType == 5 )    // don't rotate square piece type
        return;
    int tmp;
    for (int i = 0 ; i < 4 ; i++) {
        tmp = getXCoord(i);
        setXCoord(i,getYCoord(i));
        setYCoord(i,-tmp);
    }
}

void TetrixPiece::rotateRight()
{
    if ( pieceType == 5 )    // don't rotate square piece type
        return;
    int tmp;
    for (int i = 0 ; i < 4 ; i++) {
        tmp = getXCoord(i);
        setXCoord(i,-getYCoord(i));
        setYCoord(i,tmp);
    }
}

int TetrixPiece::getMinX()
{
    int tmp = coordinates[0][0];
    for(int i = 1 ; i < 4 ; i++)
        if (tmp > coordinates[i][0])
            tmp = coordinates[i][0];
    return tmp;
}

int TetrixPiece::getMaxX()
{
    int tmp = coordinates[0][0];
    for(int i = 1 ; i < 4 ; i++)
        if (tmp < coordinates[i][0])
            tmp = coordinates[i][0];
    return tmp;

}

int TetrixPiece::getMinY()
{
    int tmp = coordinates[0][1];
    for(int i = 1 ; i < 4 ; i++)
        if (tmp > coordinates[i][1])
            tmp = coordinates[i][1];
    return tmp;
}

int TetrixPiece::getMaxY()
{
    int tmp = coordinates[0][1];
    for(int i = 1 ; i < 4 ; i++)
        if (tmp < coordinates[i][1])
            tmp = coordinates[i][1];
    return tmp;
}
```

```
void TetrixPiece::initialize(int type)
{
   static int pieceTypes[7][4][2] = {{{ 0,-1},{ 0, 0},{-1, 0},{-1, 1}},
                                     {{ 0,-1},{ 0, 0},{ 1, 0},{ 1, 1}},
                                     {{ 0,-1},{ 0, 0},{ 0, 1},{ 0, 2}},
                                     {{-1, 0},{ 0, 0},{ 1, 0},{ 0, 1}},
                                     {{ 0, 0},{ 1, 0},{ 0, 1},{ 1, 1}},
                                     {{-1,-1},{ 0,-1},{ 0, 0},{ 0, 1}},
                                     {{ 1,-1},{ 0,-1},{ 0, 0},{ 0, 1}}};
   if (type < 1 || type > 7)
      type = 1;
   pieceType = type;
   for(int i = 0 ; i < 4 ; i++) {
         coordinates[i][0] = pieceTypes[type - 1][i][0];
         coordinates[i][1] = pieceTypes[type - 1][i][1];
   }
}

/*
 *   Sigh, oh beautiful nostalgia! This random algorithm has
 *   been taken from the book "Adventures with your pocket calculator"
 *   and I used it in my first implemented and machine-
 *   run program of any size to speak of. Imagine how hungry I
 *   was after having programmed BASIC on paper for
 *   half a year?!!?!?!?!?!? The first program I typed in was a
 *   slot machine game and was made in BASIC on a SHARP
 *   PC-1211 with 1,47 KB RAM (one point four seven kilobytes) and
 *   a one-line LCD-display (I think it had 32 characters) in the
 *   year of our lord 1981. The man I had bought the machine from worked
 *   as a COBOL programmer and was amazed and impressed
 *   when I demonstrated the program 2 days after I had
 *   bought the machine, quote: "Gees, I have been looking so long
 *   for a "random" command in that BASIC, what is it called?"
 *   Oh, how I still get a thrill out of the thought of the
 *   explanation I then gave him...
 */

/*
 *   Sukk, aa vakre nostalgi! Denne random algoritmen er
 *   tatt fra boka "Adventures with your pocket calculator"
 *   og den brukte jeg i mitt foerste implementerte og maskin-
 *   kjoerte program av nevneverdig stoerrelse. Tror du jeg var
 *   noe sulten etter aa ha programmert BASIC paa papir i et
 *   halvt aar?!!?!?!?!?!? Programmet jeg tasta inn foerst var et
 *   "enarmet banditt" spill og ble laget i BASIC paa en SHARP
 *   PC-1211 med 1,47 KB RAM (en komma foertisju kilobyte) og
 *   et en-linjers LCD-display (tror det hadde 32 karakterer) i det
 *   herrens aar 1981. Mannen jeg kjoepte maskinen av jobbet til
 *   daglig med COBOL programmering og var forbloeffet og imponert
 *   da jeg demonstrerte programmet 2 dager etter at jeg hadde
 *   kjoept maskinen, sitat: "Joess, jeg som har leita saa lenge
 *   etter en random kommando i den BASICen, hva var det den
 *   het?" Aa, jeg frydes ennaa ved tanken paa forklaringen jeg
 *   deretter ga ham...
 */

double TetrixPiece::randomSeed = 0.33333;

void TetrixPiece::setRandomSeed(double seed)
{
   QCString buffer;
   if (seed < 0)
      seed = - seed;
   if (seed >= 1)
      seed = seed - (double) ((int) seed);
   buffer.sprintf("%1.5f",(float) seed);
   for (int i = 0 ; i < 5 ; i++)
      if ((buffer[i + 2] - '0') % 2 == 0)
       buffer[i + 2]++;
   randomSeed = atof(buffer);
}

int TetrixPiece::randomValue(int maxPlusOne)
{
   randomSeed = randomSeed*147;
   randomSeed = randomSeed - (double) ((int) randomSeed);
   return (int) (randomSeed*maxPlusOne);
}
```

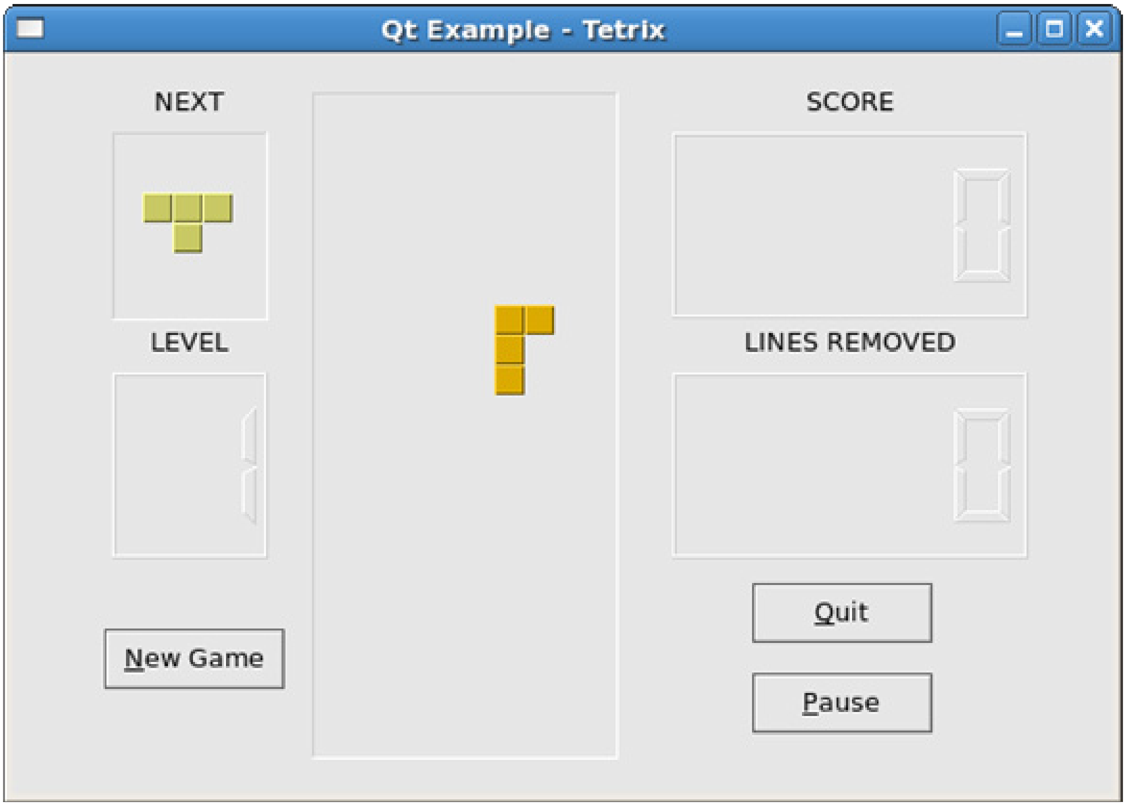
tpiece.h

```
#ifndef TPIECE_H
#define TPIECE_H

class TetrixPiece
{
public:
   TetrixPiece()                  {setRandomType();}
   TetrixPiece(int type)             {initialize(type % 7 + 1);}

   void setRandomType()              {initialize(randomValue(7) + 1);}

   void rotateLeft();
   void rotateRight();

   int  getType()                 {return pieceType;}
   int  getXCoord(int index)          {return coordinates[index][0];}
   int  getYCoord(int index)          {return coordinates[index][1];}
   void getCoord(int index,int &x,int&y){x = coordinates[index][0];
                               y = coordinates[index][1];}
   int  getMinX();
   int  getMaxX();
   int  getMinY();
   int  getMaxY();

   static void   setRandomSeed(double seed);
   static int    randomValue(int maxPlusOne);

private:
   void setXCoord(int index,int value)  {coordinates[index][0] = value;}
   void setYCoord(int index,int value)  {coordinates[index][1] = value;}
   void setCoords(int index,int x,int y){coordinates[index][0] = x;
                                         coordinates[index][1] = y;}
   void initialize(int type);

   int  pieceType;
   int  coordinates[4][2];

   static double randomSeed;
};

#endif
```

tetrix.cpp

```
#include "qtetrix.h"
#include "qdragapp.h"
#include "qfont.h"

int main( int argc, char **argv )
{
    QApplication::setColorSpec( QApplication::CustomColor );
    QDragApplication a(argc,argv);
    QTetrix *tetrix = new QTetrix;
    tetrix->setCaption("Tetrix");
    a.setMainWidget(tetrix);
    tetrix->setCaption("Qt Example - Tetrix");
    tetrix->show();
    return a.exec();
}
```

실행



64. 본문편집기실례

이 실례는 순수 C++로 씌여진 사용자대면부를 가지는 본문편집기를 현시한다.

textedit.pro

```
TEMPLATE  = app
TARGET      = textedit
CONFIG      += qt warn_on release
HEADERS        = textedit.h
SOURCES        = textedit.cpp \
          main.cpp
IMAGES      = editcopy.xpm editcut.xpm editpaste.xpm editredo.xpm editundo.xpm filenew.xpm
fileopen.xpm fileprint.xpm filesave.xpm textbold.xpm textcenter.xpm textitalic.xpm textjustify.xpm
textleft.xpm textright.xpm textunder.xpm
```

textedit.cpp

```
#include "textedit.h"
#include <qtextedit.h>
#include <qaction.h>
#include <qmenubar.h>
#include <qpopupmenu.h>
#include <qtoolbar.h>
#include <qtabwidget.h>
#include <qapplication.h>
#include <qfontdatabase.h>
#include <qcombobox.h>
```

```
#include <qlineedit.h>
#include <qfileinfo.h>
#include <qfile.h>
#include <qfiledialog.h>
#include <qprinter.h>
#include <qpaintdevicemetrics.h>
#include <qsimplerichtext.h>
#include <qcolordialog.h>
#include <qpainter.h>

TextEdit::TextEdit( QWidget *parent, const char *name )
   : QMainWindow( parent, name )
{
   setupFileActions();
   setupEditActions();
   setupTextActions();

   tabWidget = new QTabWidget( this );
   connect( tabWidget, SIGNAL( currentChanged( QWidget * ) ),
        this, SLOT( editorChanged( QWidget * ) ) );
   setCentralWidget( tabWidget );

   if ( qApp->argc() == 1 ) {
    load( "example.html" );
   } else {
    for ( int i = 1; i < qApp->argc(); ++i )
       load( qApp->argv()[ i ] );
   }
}

void TextEdit::setupFileActions()
{
   QToolBar *tb = new QToolBar( this );
   tb->setLabel( "File Actions" );
   QPopupMenu *menu = new QPopupMenu( this );
   menuBar()->insertItem( tr( "&File" ), menu );

   QAction *a;
   a = new QAction( QPixmap::fromMimeSource( "filenew.xpm" ), tr( "&New..." ), CTRL + Key_N,
this, "fileNew" );
   connect( a, SIGNAL( activated() ), this, SLOT( fileNew() ) );
   a->addTo( tb );
   a->addTo( menu );
   a = new QAction( QPixmap::fromMimeSource( "fileopen.xpm" ), tr( "&Open..." ), CTRL + Key_O,
this, "fileOpen" );
   connect( a, SIGNAL( activated() ), this, SLOT( fileOpen() ) );
   a->addTo( tb );
   a->addTo( menu );
   menu->insertSeparator();
   a = new QAction( QPixmap::fromMimeSource( "filesave.xpm" ), tr( "&Save..." ), CTRL + Key_S,
this, "fileSave" );
   connect( a, SIGNAL( activated() ), this, SLOT( fileSave() ) );
   a->addTo( tb );
   a->addTo( menu );
```

```
    a = new QAction( tr( "Save &As..." ), 0, this, "fileSaveAs" );
    connect( a, SIGNAL( activated() ), this, SLOT( fileSaveAs() ) );
    a->addTo( menu );
    menu->insertSeparator();
    a = new QAction( QPixmap::fromMimeSource( "fileprint.xpm" ), tr( "&Print..." ), CTRL + Key_P,
this, "filePrint" );
    connect( a, SIGNAL( activated() ), this, SLOT( filePrint() ) );
    a->addTo( tb );
    a->addTo( menu );
    a = new QAction( tr( "&Close" ), 0, this, "fileClose" );
    connect( a, SIGNAL( activated() ), this, SLOT( fileClose() ) );
    a->addTo( menu );
    a = new QAction( tr( "E&xit" ), 0, this, "fileExit" );
    connect( a, SIGNAL( activated() ), this, SLOT( fileExit() ) );
    a->addTo( menu );
}

void TextEdit::setupEditActions()
{
    QToolBar *tb = new QToolBar( this );
    tb->setLabel( "Edit Actions" );
    QPopupMenu *menu = new QPopupMenu( this );
    menuBar()->insertItem( tr( "&Edit" ), menu );

    QAction *a;
    a = new QAction( QPixmap::fromMimeSource( "editundo.xpm" ), tr( "&Undo" ), CTRL + Key_Z,
this, "editUndo" );
    connect( a, SIGNAL( activated() ), this, SLOT( editUndo() ) );
    a->addTo( tb );
    a->addTo( menu );
    a = new QAction( QPixmap::fromMimeSource( "editredo.xpm" ), tr( "&Redo" ), CTRL + Key_Y,
this, "editRedo" );
    connect( a, SIGNAL( activated() ), this, SLOT( editRedo() ) );
    a->addTo( tb );
    a->addTo( menu );
    menu->insertSeparator();
    a = new QAction( QPixmap::fromMimeSource( "editcopy.xpm" ), tr( "&Copy" ), CTRL + Key_C,
this, "editCopy" );
    connect( a, SIGNAL( activated() ), this, SLOT( editCopy() ) );
    a->addTo( tb );
    a->addTo( menu );
    a = new QAction( QPixmap::fromMimeSource( "editcut.xpm" ), tr( "Cu&t" ), CTRL + Key_X, this,
"editCut" );
    connect( a, SIGNAL( activated() ), this, SLOT( editCut() ) );
    a->addTo( tb );
    a->addTo( menu );
    a = new QAction( QPixmap::fromMimeSource( "editpaste.xpm" ), tr( "&Paste" ), CTRL + Key_V,
this, "editPaste" );
    connect( a, SIGNAL( activated() ), this, SLOT( editPaste() ) );
    a->addTo( tb );
    a->addTo( menu );
}

void TextEdit::setupTextActions()
```

```
{
   QToolBar *tb = new QToolBar( this );
   tb->setLabel( "Format Actions" );
   QPopupMenu *menu = new QPopupMenu( this );
   menuBar()->insertItem( tr( "F&ormat" ), menu );

   comboFont = new QComboBox( TRUE, tb );
   QFontDatabase db;
   comboFont->insertStringList( db.families() );
   connect( comboFont, SIGNAL( activated( const QString & ) ),
        this, SLOT( textFamily( const QString & ) ) );
   comboFont->lineEdit()->setText( QApplication::font().family() );

   comboSize = new QComboBox( TRUE, tb );
   QValueList<int> sizes = db.standardSizes();
   QValueList<int>::Iterator it = sizes.begin();
   for ( ; it != sizes.end(); ++it )
    comboSize->insertItem( QString::number( *it ) );
   connect( comboSize, SIGNAL( activated( const QString & ) ),
        this, SLOT( textSize( const QString & ) ) );
   comboSize->lineEdit()->setText( QString::number( QApplication::font().pointSize() ) );

   actionTextBold = new QAction( QPixmap::fromMimeSource( "textbold.xpm" ), tr( "&Bold" ), CTRL
+ Key_B, this, "textBold" );
   connect( actionTextBold, SIGNAL( activated() ), this, SLOT( textBold() ) );
   actionTextBold->addTo( tb );
   actionTextBold->addTo( menu );
   actionTextBold->setToggleAction( TRUE );
   actionTextItalic = new QAction( QPixmap::fromMimeSource( "textitalic.xpm" ), tr( "&Italic" ),
CTRL + Key_I, this, "textItalic" );
   connect( actionTextItalic, SIGNAL( activated() ), this, SLOT( textItalic() ) );
   actionTextItalic->addTo( tb );
   actionTextItalic->addTo( menu );
   actionTextItalic->setToggleAction( TRUE );
   actionTextUnderline = new QAction( QPixmap::fromMimeSource( "textunder.xpm" ),
tr( "&Underline" ), CTRL + Key_U, this, "textUnderline" );
   connect( actionTextUnderline, SIGNAL( activated() ), this, SLOT( textUnderline() ) );
   actionTextUnderline->addTo( tb );
   actionTextUnderline->addTo( menu );
   actionTextUnderline->setToggleAction( TRUE );
   menu->insertSeparator();

   QActionGroup *grp = new QActionGroup( this );
   connect( grp, SIGNAL( selected( QAction* ) ), this, SLOT( textAlign( QAction* ) ) );

   actionAlignLeft = new QAction( QPixmap::fromMimeSource( "textleft.xpm" ), tr( "&Left" ), CTRL
+ Key_L, grp, "textLeft" );
   actionAlignLeft->setToggleAction( TRUE );
   actionAlignCenter = new QAction( QPixmap::fromMimeSource( "textcenter.xpm" ), tr( "C&enter" ),
CTRL + Key_E, grp, "textCenter" );
   actionAlignCenter->setToggleAction( TRUE );
   actionAlignRight = new QAction( QPixmap::fromMimeSource( "textright.xpm" ), tr( "&Right" ),
CTRL + Key_R, grp, "textRight" );
   actionAlignRight->setToggleAction( TRUE );
   actionAlignJustify = new QAction( QPixmap::fromMimeSource( "textjustify.xpm" ), tr( "&Justify" ),
CTRL + Key_J, grp, "textjustify" );
   actionAlignJustify->setToggleAction( TRUE );

   grp->addTo( tb );
   grp->addTo( menu );

   menu->insertSeparator();

   QPixmap pix( 16, 16 );
   pix.fill( black );
   actionTextColor = new QAction( pix, tr( "&Color..." ), 0, this, "textColor" );
   connect( actionTextColor, SIGNAL( activated() ), this, SLOT( textColor() ) );
   actionTextColor->addTo( tb );
   actionTextColor->addTo( menu );
}

void TextEdit::load( const QString &f )
{
   if ( !QFile::exists( f ) )
    return;
   QTextEdit *edit = new QTextEdit( tabWidget );
   edit->setTextFormat( RichText );
   doConnections( edit );
   tabWidget->addTab( edit, QFileInfo( f ).fileName() );
   QFile file( f );
   if ( !file.open( IO_ReadOnly ) )
    return;
   QTextStream ts( &file );
   QString txt = ts.read();
   if ( !QStyleSheet::mightBeRichText( txt ) )
    txt = QStyleSheet::convertFromPlainText( txt, QStyleSheetItem::WhiteSpacePre );
   edit->setText( txt );
   tabWidget->showPage( edit );
   edit->viewport()->setFocus();
   filenames.replace( edit, f );
}

QTextEdit *TextEdit::currentEditor() const
{
   if ( tabWidget->currentPage() &&
     tabWidget->currentPage()->inherits( "QTextEdit" ) )
    return (QTextEdit*)tabWidget->currentPage();
   return 0;
}

void TextEdit::doConnections( QTextEdit *e )
{
   connect( e, SIGNAL( currentFontChanged( const QFont & ) ),
        this, SLOT( fontChanged( const QFont & ) ) );
   connect( e, SIGNAL( currentColorChanged( const QColor & ) ),
        this, SLOT( colorChanged( const QColor & ) ) );
   connect( e, SIGNAL( currentAlignmentChanged( int ) ),
        this, SLOT( alignmentChanged( int ) ) );
```

```
}

void TextEdit::fileNew()
{
   QTextEdit *edit = new QTextEdit( tabWidget );
   edit->setTextFormat( RichText );
   doConnections( edit );
   tabWidget->addTab( edit, tr( "noname" ) );
   tabWidget->showPage( edit );
   edit->viewport()->setFocus();
}

void TextEdit::fileOpen()
{
   QString fn = QFileDialog::getOpenFileName( QString::null, tr( "HTML-Files (*.htm *.html);;All
Files (*)" ), this );
   if ( !fn.isEmpty() )
    load( fn );
}

void TextEdit::fileSave()
{
   if ( !currentEditor() )
    return;
   QString fn;
   if ( filenames.find( currentEditor() ) == filenames.end() ) {
    fileSaveAs();
   } else {
    QFile file( *filenames.find( currentEditor() ) );
    if ( !file.open( IO_WriteOnly ) )
        return;
    QTextStream ts( &file );
    ts << currentEditor()->text();
   }
}

void TextEdit::fileSaveAs()
{
   if ( !currentEditor() )
    return;
   QString fn = QFileDialog::getSaveFileName( QString::null, tr( "HTML-Files (*.htm *.html);;All
Files (*)" ), this );
   if ( !fn.isEmpty() ) {
    filenames.replace( currentEditor(), fn );
    fileSave();
    tabWidget->setTabLabel( currentEditor(), QFileInfo( fn ).fileName() );
   }
}

void TextEdit::filePrint()
{
   if ( !currentEditor() )
    return;
#ifndef QT_NO_PRINTER
```

```
  QPrinter printer( QPrinter::HighResolution );
  printer.setFullPage(TRUE);
  if ( printer.setup( this ) ) {
   QPainter p( &printer );
   // Check that there is a valid device to print to.
   if ( !p.device() ) return;
   QPaintDeviceMetrics metrics( p.device() );
   int dpiy = metrics.logicalDpiY();
   int margin = (int) ( (2/2.54)*dpiy ); // 2 cm margins
   QRect view( margin, margin, metrics.width() - 2*margin, metrics.height() - 2*margin );
   QFont font( currentEditor()->QWidget::font() );
   font.setPointSize( 10 ); // we define 10pt to be a nice base size for printing

   QSimpleRichText richText( currentEditor()->text(), font,
             currentEditor()->context(),
             currentEditor()->styleSheet(),
             currentEditor()->mimeSourceFactory(),
             view.height() );
   richText.setWidth( &p, view.width() );
   int page = 1;
   do {
     richText.draw( &p, margin, margin, view, colorGroup() );
     view.moveBy( 0, view.height() );
     p.translate( 0 , -view.height() );
     p.setFont( font );
     p.drawText( view.right() - p.fontMetrics().width( QString::number( page ) ),
         view.bottom() + p.fontMetrics().ascent() + 5, QString::number( page ) );
     if ( view.top() - margin >= richText.height() )
      break;
     printer.newPage();
     page++;
   } while (TRUE);
  }
#endif
}

void TextEdit::fileClose()
{
  delete currentEditor();
  if ( currentEditor() )
   currentEditor()->viewport()->setFocus();
}

void TextEdit::fileExit()
{
  qApp->quit();
}

void TextEdit::editUndo()
{
  if ( !currentEditor() )
   return;
  currentEditor()->undo();
}
```

```
void TextEdit::editRedo()
{
   if ( !currentEditor() )
    return;
   currentEditor()->redo();
}

void TextEdit::editCut()
{
   if ( !currentEditor() )
    return;
   currentEditor()->cut();
}

void TextEdit::editCopy()
{
   if ( !currentEditor() )
    return;
   currentEditor()->copy();
}

void TextEdit::editPaste()
{
   if ( !currentEditor() )
    return;
   currentEditor()->paste();
}

void TextEdit::textBold()
{
   if ( !currentEditor() )
    return;
   currentEditor()->setBold( actionTextBold->isOn() );
}

void TextEdit::textUnderline()
{
   if ( !currentEditor() )
    return;
   currentEditor()->setUnderline( actionTextUnderline->isOn() );
}

void TextEdit::textItalic()
{
   if ( !currentEditor() )
    return;
   currentEditor()->setItalic( actionTextItalic->isOn() );
}

void TextEdit::textFamily( const QString &f )
{
   if ( !currentEditor() )
    return;
    currentEditor()->setFamily( f );
    currentEditor()->viewport()->setFocus();
}

void TextEdit::textSize( const QString &p )
{
    if ( !currentEditor() )
      return;
    currentEditor()->setPointSize( p.toInt() );
    currentEditor()->viewport()->setFocus();
}

void TextEdit::textColor()
{
    if ( !currentEditor() )
      return;
    QColor col = QColorDialog::getColor( currentEditor()->color(), this );
    if ( !col.isValid() )
      return;
    currentEditor()->setColor( col );
    QPixmap pix( 16, 16 );
    pix.fill( black );
    actionTextColor->setIconSet( pix );
}

void TextEdit::textAlign( QAction *a )
{
    if ( !currentEditor() )
      return;
    if ( a == actionAlignLeft )
      currentEditor()->setAlignment( AlignLeft );
    else if ( a == actionAlignCenter )
      currentEditor()->setAlignment( AlignHCenter );
    else if ( a == actionAlignRight )
      currentEditor()->setAlignment( AlignRight );
    else if ( a == actionAlignJustify )
      currentEditor()->setAlignment( AlignJustify );
}

void TextEdit::fontChanged( const QFont &f )
{
    comboFont->lineEdit()->setText( f.family() );
    comboSize->lineEdit()->setText( QString::number( f.pointSize() ) );
    actionTextBold->setOn( f.bold() );
    actionTextItalic->setOn( f.italic() );
    actionTextUnderline->setOn( f.underline() );
}

void TextEdit::colorChanged( const QColor &c )
{
    QPixmap pix( 16, 16 );
    pix.fill( c );
    actionTextColor->setIconSet( pix );
}
```

```
void TextEdit::alignmentChanged( int a )
{
   if ( ( a == AlignAuto ) || ( a & AlignLeft ))
    actionAlignLeft->setOn( TRUE );
   else if ( ( a & AlignHCenter ) )
    actionAlignCenter->setOn( TRUE );
   else if ( ( a & AlignRight ) )
    actionAlignRight->setOn( TRUE );
   else if ( ( a & AlignJustify ) )
    actionAlignJustify->setOn( TRUE );
}

void TextEdit::editorChanged( QWidget * )
{
   if ( !currentEditor() )
    return;
   fontChanged( currentEditor()->currentFont() );
   colorChanged( currentEditor()->color() );
   alignmentChanged( currentEditor()->alignment() );
}
```

textedit.h

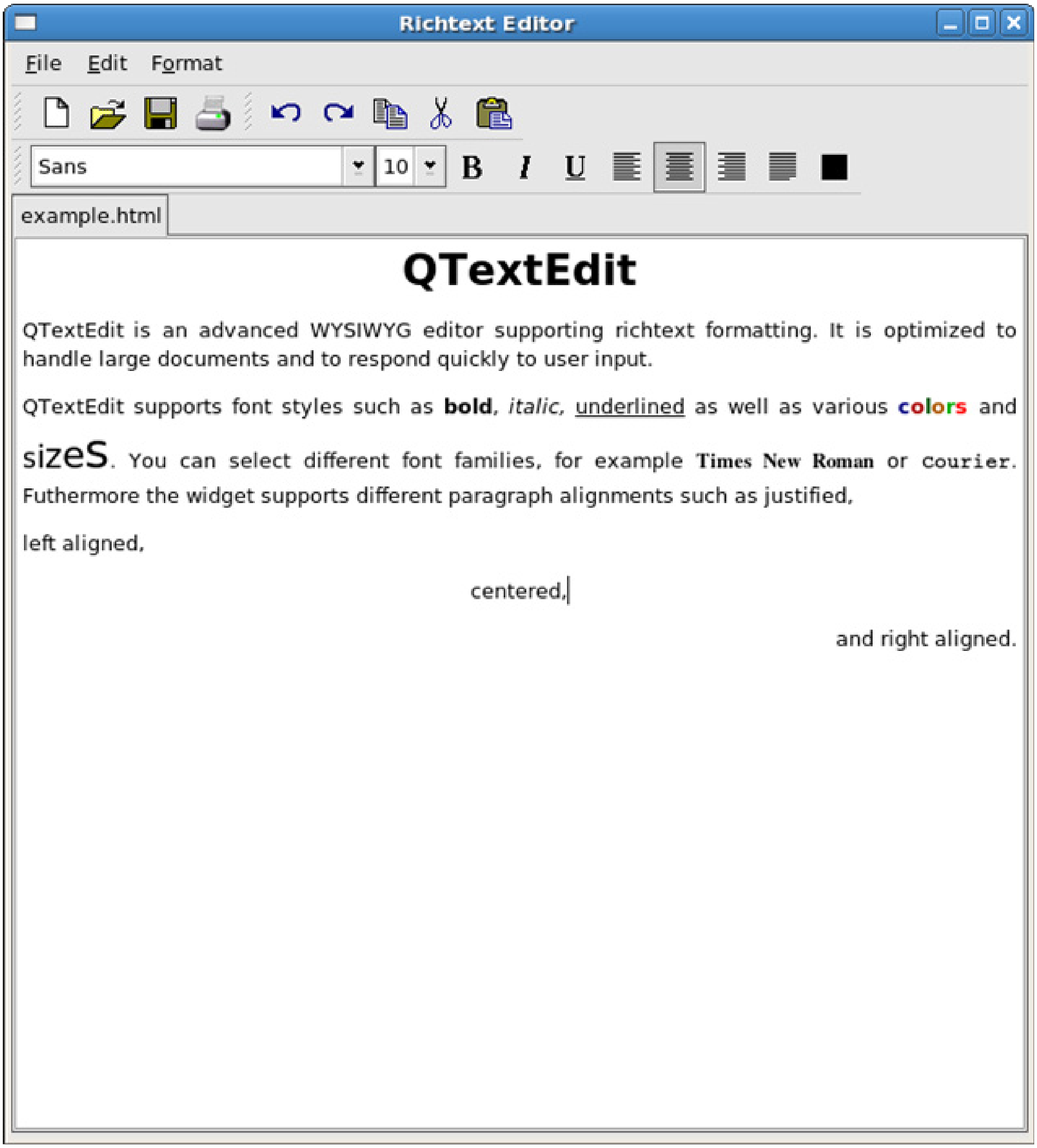
```
#ifndef TEXTEDIT_H
#define TEXTEDIT_H

#include <qmainwindow.h>
#include <qmap.h>

class QAction;
class QComboBox;
class QTabWidget;
class QTextEdit;

class TextEdit : public QMainWindow
{
   Q_OBJECT

public:
   TextEdit( QWidget *parent = 0, const char *name = 0 );

private:
   void setupFileActions();
   void setupEditActions();
   void setupTextActions();
   void load( const QString &f );
   QTextEdit *currentEditor() const;
   void doConnections( QTextEdit *e );

private slots:
   void fileNew();
   void fileOpen();
   void fileSave();
   void fileSaveAs();
```

```
    void filePrint();
    void fileClose();
    void fileExit();

    void editUndo();
    void editRedo();
    void editCut();
    void editCopy();
    void editPaste();

    void textBold();
    void textUnderline();
    void textItalic();
    void textFamily( const QString &f );
    void textSize( const QString &p );
    void textColor();
    void textAlign( QAction *a );

    void fontChanged( const QFont &f );
    void colorChanged( const QColor &c );
    void alignmentChanged( int a );
    void editorChanged( QWidget * );

private:
    QAction *actionTextBold,
      *actionTextUnderline,
      *actionTextItalic,
      *actionTextColor,
      *actionAlignLeft,
      *actionAlignCenter,
      *actionAlignRight,
      *actionAlignJustify;
    QComboBox
      *comboFont,
      *comboSize;
    QTabWidget *tabWidget;
    QMap<QTextEdit*, QString> filenames;

};

#endif
```

main.cpp

```
#include <qapplication.h>
#include "textedit.h"

int main( int argc, char ** argv )
{
    QApplication a( argc, argv );
    TextEdit * mw = new TextEdit();
    mw->setCaption( "Richtext Editor" );
    mw->resize( 640, 800 );
    mw->show();
    a.connect( &a, SIGNAL( lastWindowClosed() ), &a, SLOT( quit() ) );
    return a.exec();
}
```

실행



QTextEdit

QTextEdit is an advanced WYSIWYG editor supporting richtext formatting. It is optimized to handle large documents and to respond quickly to user input.

QTextEdit supports font styles such as **bold**, *italic*, underlined as well as various colors and sizes. You can select different font families, for example Times New Roman or courier. Futhermore the widget supports different paragraph alignments such as justified,

left aligned,

centered,

and right aligned.

65. Themes(형식)

이 실례는 창문부품들을 각이한 형식(주제)으로 그리는 방법을 보여준다. 실례로 나무모양이나 금속결정형태로 창문부품들을 그린다. 내리펼침차림표를 리용하여 실행시에 각이한 형식들사이를 절환할수 있다.

themes.pro

```
TEMPLATE  = app
TARGET      = themes
```

```
CONFIG       += qt warn_on release no_batch
HEADERS        = themes.h \
           ../buttongroups/buttongroups.h \
           ../lineedits/lineedits.h \
           ../listboxcombo/listboxcombo.h \
           ../checklists/checklists.h \
           ../progressbar/progressbar.h \
           ../rangecontrols/rangecontrols.h \
           ../richtext/richtext.h \
           wood.h \
           metal.h
SOURCES        = themes.cpp \
           main.cpp \
           ../buttongroups/buttongroups.cpp \
           ../lineedits/lineedits.cpp \
           ../listboxcombo/listboxcombo.cpp \
           ../checklists/checklists.cpp \
           ../progressbar/progressbar.cpp \
           ../rangecontrols/rangecontrols.cpp \
           ../richtext/richtext.cpp \
           wood.cpp \
           metal.cpp
```

metal.cpp

```
#include "metal.h"
#ifndef QT_NO_STYLE_WINDOWS

#include "qapplication.h"
#include "qcombobox.h"
#include "qpainter.h"
#include "qdrawutil.h" // for now
#include "qpixmap.h" // for now
#include "qpalette.h" // for now
#include "qwidget.h"
#include "qlabel.h"
#include "qimage.h"
#include "qpushbutton.h"
#include "qwidget.h"
#include "qrangecontrol.h"
#include "qscrollbar.h"
#include "qslider.h"
#include <limits.h>

/////////////////////////////////////////////////////////
//#include "stonedark.xpm"
#include "stone1.xpm"
#include "marble.xpm"
///////////////////////////////////////////////////////

MetalStyle::MetalStyle() : QWindowsStyle() { }

/*!
  Reimplementation from QStyle
 */
```

```
void MetalStyle::polish( QApplication *app)
{
    oldPalette = app->palette();

    // we simply create a nice QColorGroup with a couple of fancy
    // pixmaps here and apply to it all widgets

    QFont f("times", app->font().pointSize() );
    f.setBold( TRUE );
    f.setItalic( TRUE );
    app->setFont( f, TRUE, "QMenuBar");
    app->setFont( f, TRUE, "QPopupMenu");

    //    QPixmap button( stonedark_xpm );

    QColor gold("#B9B9A5A54040"); //same as topgrad below
    QPixmap button( 1, 1 ); button.fill( gold );

    QPixmap background(marble_xpm);
    QPixmap dark( 1, 1 ); dark.fill( red.dark() );
    QPixmap mid( stone1_xpm );
    QPixmap light( stone1_xpm );//1, 1 ); light.fill( green );

    QPalette op = app->palette();

    QColor backCol( 227,227,227 );

    // QPalette op(white);
    QColorGroup active (op.active().foreground(),
                QBrush(op.active().button(),button),
                QBrush(op.active().light(), light),
                QBrush(op.active().dark(), dark),
                QBrush(op.active().mid(), mid),
                op.active().text(),
                Qt::white,
                op.active().base(),//              QColor(236,182,120),
                QBrush(backCol, background)
                );
    active.setColor( QColorGroup::ButtonText,  Qt::white  );
    active.setColor( QColorGroup::Shadow,  Qt::black  );
    QColorGroup disabled (op.disabled().foreground(),
                QBrush(op.disabled().button(),button),
                QBrush(op.disabled().light(), light),
                op.disabled().dark(),
                QBrush(op.disabled().mid(), mid),
                op.disabled().text(),
                Qt::white,
                op.disabled().base(),//              QColor(236,182,120),
                QBrush(backCol, background)
                );

    QPalette newPalette( active, disabled, active );
    app->setPalette( newPalette, TRUE );
}
```

```
/*!
  Reimplementation from QStyle
 */
void MetalStyle::unPolish( QApplication *app)
{
    app->setPalette(oldPalette, TRUE);
    app->setFont( app->font(), TRUE );
}

/*!
  Reimplementation from QStyle
 */
void MetalStyle::polish( QWidget* w)
{

   // the polish function sets some widgets to transparent mode and
    // some to translate background mode in order to get the full
    // benefit from the nice pixmaps in the color group.

    if (w->inherits("QPushButton")){
      w->setBackgroundMode( QWidget::NoBackground );
      return;
    }

    if ( !w->isTopLevel() ) {
      if ( w->backgroundPixmap() )
          w->setBackgroundOrigin( QWidget::WindowOrigin );
    }
}

void MetalStyle::unPolish( QWidget* w)
{

   // the polish function sets some widgets to transparent mode and
    // some to translate background mode in order to get the full
    // benefit from the nice pixmaps in the color group.

    if (w->inherits("QPushButton")){
      w->setBackgroundMode( QWidget::PaletteButton );
      return;
    }
    if ( !w->isTopLevel() ) {
      if ( w->backgroundPixmap() )
          w->setBackgroundOrigin( QWidget::WidgetOrigin );
    }

}

void MetalStyle::drawPrimitive( PrimitiveElement pe,
                        QPainter *p,
                        const QRect &r,
                        const QColorGroup &cg,
                        SFlags flags, const QStyleOption& opt ) const
```

```
{
   switch( pe ) {
   case PE_HeaderSection:
    if ( flags & Style_Sunken )
       flags ^= Style_Sunken | Style_Raised;
    // fall through
   case PE_ButtonBevel:
   case PE_ButtonCommand:
       drawMetalButton( p, r.x(), r.y(), r.width(), r.height(),
                (flags & (Style_Sunken|Style_On|Style_Down)),  TRUE, !(flags & Style_Raised) );
       break;
   case PE_PanelMenuBar:
    drawMetalFrame( p, r.x(), r.y(), r.width(), r.height() );
    break;
   case PE_ScrollBarAddLine:
    drawMetalButton( p, r.x(), r.y(), r.width(), r.height(),
             flags & Style_Down, !( flags & Style_Horizontal ) );
    drawPrimitive( (flags & Style_Horizontal) ? PE_ArrowRight :PE_ArrowDown,
             p, r, cg, flags, opt );
    break;
   case PE_ScrollBarSubLine:
    drawMetalButton( p, r.x(), r.y(), r.width(), r.height(),
             flags & Style_Down, !( flags & Style_Horizontal ) );
    drawPrimitive( (flags & Style_Horizontal) ? PE_ArrowLeft : PE_ArrowUp,
             p, r, cg, flags, opt );
    break;


   case PE_ScrollBarSlider:
    drawMetalButton( p, r.x(), r.y(), r.width(), r.height(), FALSE, flags & Style_Horizontal );
    break;
   default:
    QWindowsStyle::drawPrimitive( pe, p, r, cg, flags, opt );
    break;
   }
}

void MetalStyle::drawControl( ControlElement element,  QPainter *p,  const QWidget *widget,
                 const QRect &r, const QColorGroup &cg,  SFlags how, const QStyleOption& opt ) const
{
   switch( element ) {
   case CE_PushButton:
    {
       const QPushButton *btn;
       btn = (const QPushButton*)widget;
       int x1, y1, x2, y2;

       r.coords( &x1, &y1, &x2, &y2 );

       p->setPen( cg.foreground() );
       p->setBrush( QBrush(cg.button(), NoBrush) );


       QBrush fill;
```

```
      if ( btn->isDown() )
        fill = cg.brush( QColorGroup::Mid );
      else if ( btn->isOn() )
        fill = QBrush( cg.mid(), Dense4Pattern );
      else
        fill = cg.brush( QColorGroup::Button );

      if ( btn->isDefault() ) {
        QPointArray a;
        a.setPoints( 9,  x1, y1, x2, y1, x2, y2, x1, y2, x1, y1+1,
                    x2-1, y1+1, x2-1, y2-1, x1+1, y2-1, x1+1, y1+1 );
        p->setPen( Qt::black );
        p->drawPolyline( a );
        x1 += 2;
        y1 += 2;
        x2 -= 2;
        y2 -= 2;
      }
      SFlags flags = Style_Default;
      if ( btn->isOn() )
        flags |= Style_On;
      if ( btn->isDown() )
        flags |= Style_Down;
      if ( !btn->isFlat() && !btn->isDown() )
        flags |= Style_Raised;
      drawPrimitive( PE_ButtonCommand, p, QRect( x1, y1, x2 - x1 + 1, y2 - y1 + 1), cg, flags, opt );

      if ( btn->isMenuButton() ) {
        flags = Style_Default;
        if ( btn->isEnabled() )
            flags |= Style_Enabled;

        int dx = ( y1 - y2 - 4 ) / 3;
        drawPrimitive( PE_ArrowDown, p,  QRect(x2 - dx, dx, y1, y2 - y1), cg, flags, opt );
      }
      if ( p->brush().style() != NoBrush )
        p->setBrush( NoBrush );
      break;
   }
 case CE_PushButtonLabel:
   {
      const QPushButton *btn;
      btn = (const QPushButton*)widget;
      int x, y, w, h;
      r.rect( &x, &y, &w, &h );

      int x1, y1, x2, y2;
      r.coords( &x1, &y1, &x2, &y2 );
      int dx = 0;
      int dy = 0;
      if ( btn->isMenuButton() )
        dx = ( y2 - y1 ) / 3;
      if ( btn->isOn() || btn->isDown() ) {
        dx--;
        dy--;
      }
      if ( dx || dy )
        p->translate( dx, dy );
      x += 2;
      y += 2;
      w -= 4;
      h -= 4;
      drawItem( p, QRect( x, y, w, h ),  AlignCenter|ShowPrefix,  cg, btn->isEnabled(),
          btn->pixmap(), btn->text(), -1,
          (btn->isDown() || btn->isOn())? &cg.brightText() : &cg.buttonText() );
      if ( dx || dy )
        p->translate( -dx, -dy );
      break;
   }
 default:
   QWindowsStyle::drawControl( element, p, widget, r, cg, how, opt );
   break;
 }
}
void MetalStyle::drawComplexControl( ComplexControl cc, QPainter *p,
             const QWidget *widget, const QRect &r, const QColorGroup &cg,
             SFlags how, SCFlags sub,  SCFlags subActive, const QStyleOption& opt ) const
{
 switch ( cc ) {
 case CC_Slider:
   {
      const QSlider *slider = ( const QSlider* ) widget;
      QRect handle = querySubControlMetrics( CC_Slider, widget,  SC_SliderHandle, opt);
      if ( sub & SC_SliderGroove )
        QWindowsStyle::drawComplexControl( cc, p, widget, r, cg, how,
                     SC_SliderGroove, subActive, opt );
      if ( (sub & SC_SliderHandle) && handle.isValid() )
        drawMetalButton( p, handle.x(), handle.y(), handle.width(),  handle.height(), FALSE,
              slider->orientation() == QSlider::Horizontal);
      break;
   }
 case CC_ComboBox:
   {
      // not exactly correct...
      const QComboBox *cmb = ( const QComboBox* ) widget;

      qDrawWinPanel( p, r.x(), r.y(), r.width(), r.height(), cg, TRUE,
           cmb->isEnabled() ? &cg.brush( QColorGroup::Base )
                     &cg.brush( QColorGroup::Background ) );
      drawMetalButton( p, r.x() + r.width() - 2 - 16, r.y() + 2, 16, r.height() - 4,
                    how & Style_Sunken, TRUE );
      drawPrimitive( PE_ArrowDown, p,  QRect( r.x() + r.width() - 2 - 16 + 2,
              r.y() + 2 + 2, 16 - 4, r.height() - 4 -4 ),  cg,
           cmb->isEnabled() ? Style_Enabled : Style_Default,  opt );
      break;
   }
 default:
   QWindowsStyle::drawComplexControl( cc, p, widget, r, cg, how, sub, subActive,  opt );
```

```
     break;
   }
}

/*!
  Draw a metallic button, sunken if \a sunken is TRUE, horizontal if
  /a horz is TRUE.
*/

void MetalStyle::drawMetalButton( QPainter *p, int x, int y, int w, int h,
                    bool sunken, bool horz, bool flat  ) const
{

   drawMetalFrame( p, x, y, w, h );
   drawMetalGradient( p, x, y, w, h, sunken, horz, flat );
}

void MetalStyle::drawMetalFrame( QPainter *p, int x, int y, int w, int h ) const
{
   QColor top1("#878769691515");
   QColor top2("#C6C6B4B44949");

   QColor bot2("#70705B5B1414");
   QColor bot1("#56564A4A0E0E"); //first from the bottom

   int x2 = x + w - 1;
   int y2 = y + h - 1;

   //frame:

   p->setPen( top1 );
   p->drawLine( x, y2, x, y );
   p->drawLine( x, y, x2-1, y );
   p->setPen( top2 );
   p->drawLine( x+1, y2 -1, x+1, y+1 );
   p->drawLine( x+1, y+1 , x2-2, y+1 );

   p->setPen( bot1 );
   p->drawLine( x+1, y2, x2, y2 );
   p->drawLine( x2, y2, x2, y );
   p->setPen( bot2 );
   p->drawLine( x+1, y2-1, x2-1, y2-1 );
   p->drawLine( x2-1, y2-1, x2-1, y+1 );
}

void MetalStyle::drawMetalGradient( QPainter *p, int x, int y, int w, int h,
                      bool sunken, bool horz, bool flat  ) const

{
   QColor highlight("#E8E8DDDD6565");
   QColor subh1("#CECEBDBD5151");
   QColor subh2("#BFBFACAC4545");

   QColor topgrad("#B9B9A5A54040");
```

```
    QColor botgrad("#89896C6C1A1A");

    if ( flat && !sunken ) {
        p->fillRect( x + 2, y + 2, w - 4,h -4, topgrad );
    } else {
     // highlight:
     int i = 0;
     int x1 = x + 2;
     int y1 = y + 2;
     int x2 = x + w - 1;
     int y2 = y + h - 1;
     if ( horz )
        x2 = x2 - 2;
     else
        y2 = y2 - 2;

#define DRAWLINE if (horz) \
                p->drawLine( x1, y1+i, x2, y1+i ); \
          else \
                p->drawLine( x1+i, y1, x1+i, y2 ); \
              i++;

     if ( !sunken ) {
        p->setPen( highlight );
        DRAWLINE;
        DRAWLINE;
        p->setPen( subh1 );
        DRAWLINE;
        p->setPen( subh2 );
        DRAWLINE;
     }
     // gradient:
     int ng = (horz ? h : w) - 8; // how many lines for the gradient?

     int h1, h2, s1, s2, v1, v2;
     if ( !sunken ) {
        topgrad.hsv( &h1, &s1, &v1 );
        botgrad.hsv( &h2, &s2, &v2 );
     } else {
        botgrad.hsv( &h1, &s1, &v1 );
        topgrad.hsv( &h2, &s2, &v2 );
     }

     if ( ng > 1 ) {
        for ( int j =0; j < ng; j++ ) {
         p->setPen( QColor( h1 + ((h2-h1)*j)/(ng-1),
                    s1 + ((s2-s1)*j)/(ng-1),
                    v1 + ((v2-v1)*j)/(ng-1),  QColor::Hsv ) );
         DRAWLINE;
        }
     } else if ( ng == 1 ) {
        p->setPen( QColor((h1+h2)/2, (s1+s2)/2, (v1+v2)/2, QColor::Hsv) );
        DRAWLINE;
     }
```

```
   if ( sunken ) {
     p->setPen( subh2 );
     DRAWLINE;

     p->setPen( subh1 );
     DRAWLINE;

     p->setPen( highlight );
     DRAWLINE;
     DRAWLINE;
   }
  }
}

int MetalStyle::pixelMetric( PixelMetric metric, const QWidget *w ) const
{
  switch ( metric ) {
  case PM_MenuBarFrameWidth:
   return 2;
  default:
   return QWindowsStyle::pixelMetric( metric, w );
  }
}

#endif
```

metal.h

```
#ifndef METAL_H
#define METAL_H

#include <qpalette.h>

#ifndef QT_NO_STYLE_WINDOWS

#include <qwindowsstyle.h>

class MetalStyle : public QWindowsStyle
{
public:
  MetalStyle();
  void polish( QApplication*);
  void unPolish( QApplication*);
  void polish( QWidget* );
  void unPolish( QWidget* );

  void drawPrimitive( PrimitiveElement pe, QPainter *p, const QRect &r, const QColorGroup &cg,
         SFlags flags = Style_Default, const QStyleOption& = QStyleOption::Default) const;

  void drawControl( ControlElement element, QPainter *p, const QWidget *widget,
         const QRect &r, const QColorGroup &cg, SFlags how = Style_Default,
         const QStyleOption& = QStyleOption::Default ) const;

  void drawComplexControl( ComplexControl cc,  QPainter *p, const QWidget *widget,
           const QRect &r, const QColorGroup &cg, SFlags how = Style_Default,
                    SCFlags sub = SC_All, SCFlags subActive = SC_None,
                    const QStyleOption& = QStyleOption::Default ) const;
    int pixelMetric( PixelMetric, const QWidget * ) const;


private:
    void drawMetalFrame(  QPainter *p, int x, int y, int w, int h ) const;
    void drawMetalGradient( QPainter *p, int x, int y, int w, int h,
                bool sunken, bool horz, bool flat=FALSE ) const;
    void drawMetalButton( QPainter *p, int x, int y, int w, int h,
                bool sunken, bool horz, bool flat=FALSE ) const;
    QPalette oldPalette;
};

#endif
#endif
```

themes.h

```
#ifndef THEMES_H
#define THEMES_H

#include <qmainwindow.h>
#include <qfont.h>

class QTabWidget;

class Themes: public QMainWindow
{
    Q_OBJECT

public:
    Themes( QWidget *parent = 0, const char *name = 0, WFlags f = WType_TopLevel );

protected:
    QTabWidget *tabwidget;

protected slots:
    void makeStyle(const QString &);
    void about();
    void aboutQt();

private:
    QFont appFont;
};

#endif
```

themes.cpp

```
#include "themes.h"
#include "wood.h"
#include "metal.h"
#include "../buttongroups/buttongroups.h"
#include "../lineedits/lineedits.h"
#include "../listboxcombo/listboxcombo.h"
```

```
#include "../checklists/checklists.h"
#include "../progressbar/progressbar.h"
#include "../rangecontrols/rangecontrols.h"
#include "../richtext/richtext.h"

#include <qtabwidget.h>
#include <qapplication.h>
#include <qpopupmenu.h>
#include <qmenubar.h>
#include <qmessagebox.h>
#include <qfont.h>
#include <qstylefactory.h>
#include <qaction.h>
#include <qsignalmapper.h>
#include <qdict.h>

Themes::Themes( QWidget *parent, const char *name, WFlags f ) : QMainWindow( parent, name, f )
{
   appFont = QApplication::font();
   tabwidget = new QTabWidget( this );

   tabwidget->addTab( new ButtonsGroups( tabwidget ), "Buttons/Groups" );
   QHBox *hbox = new QHBox( tabwidget );
   hbox->setMargin( 5 );
   (void)new LineEdits( hbox );
   (void)new ProgressBar( hbox );
   tabwidget->addTab( hbox, "Lineedits/Progressbar" );
   tabwidget->addTab( new ListBoxCombo( tabwidget ), "Listboxes/Comboboxes" );
   tabwidget->addTab( new CheckLists( tabwidget ), "Listviews" );
   tabwidget->addTab( new RangeControls( tabwidget ), "Rangecontrols" );
   tabwidget->addTab( new MyRichText( tabwidget ), "Fortune" );

   setCentralWidget( tabwidget );

   QPopupMenu *style = new QPopupMenu( this );
   style->setCheckable( TRUE );
   menuBar()->insertItem( "&Style" , style );

   style->setCheckable( TRUE );
   QActionGroup *ag = new QActionGroup( this, 0 );
   ag->setExclusive( TRUE );
   QSignalMapper *styleMapper = new QSignalMapper( this );
   connect( styleMapper, SIGNAL( mapped( const QString& ) ), this, SLOT( makeStyle( const
QString& ) ) );
   QStringList list = QStyleFactory::keys();
   list.sort();
#ifndef QT_NO_STYLE_WINDOWS
   list.insert(list.begin(), "Norwegian Wood");
   list.insert(list.begin(), "Metal");
#endif
   QDict<int> stylesDict( 17, FALSE );
   for ( QStringList::Iterator it = list.begin(); it != list.end(); ++it ) {
    QString styleStr = *it;
    QString styleAccel = styleStr;
    if ( stylesDict[styleAccel.left(1)] ) {
       for ( uint i = 0; i < styleAccel.length(); i++ ) {
         if ( !stylesDict[styleAccel.mid( i, 1 )] ) {
            stylesDict.insert(styleAccel.mid( i, 1 ), (const int *)1);
            styleAccel = styleAccel.insert( i, '&' );
            break;
         }
       }
    } else {
       stylesDict.insert(styleAccel.left(1), (const int *)1);
       styleAccel = "&"+styleAccel;
    }
    QAction *a = new QAction( styleStr, QIconSet(), styleAccel, 0, ag, 0, ag->isExclusive() );
    connect( a, SIGNAL( activated() ), styleMapper, SLOT(map()) );
    styleMapper->setMapping( a, a->text() );
   }
   ag->addTo(style);
   style->insertSeparator();
   style->insertItem("&Quit", qApp, SLOT( quit() ), CTRL | Key_Q );

   QPopupMenu * help = new QPopupMenu( this );
   menuBar()->insertSeparator();
   menuBar()->insertItem( "&Help", help );
   help->insertItem( "&About", this, SLOT(about()), Key_F1);
   help->insertItem( "About &Qt", this, SLOT(aboutQt()));

#ifndef QT_NO_STYLE_WINDOWS
   qApp->setStyle( new NorwegianWoodStyle );
#endif
}

void Themes::makeStyle(const QString &style)
{
   if(style == "Norwegian Wood") {
#ifndef QT_NO_STYLE_WINDOWS
    qApp->setStyle( new NorwegianWoodStyle );
#endif
   } else if( style == "Metal" ) {
#ifndef QT_NO_STYLE_WINDOWS
    qApp->setStyle( new MetalStyle );
#endif
   } else {
    qApp->setStyle(style);
    if(style == "Platinum") {
       QPalette p( QColor( 239, 239, 239 ) );
       qApp->setPalette( p, TRUE );
       qApp->setFont( appFont, TRUE );
    } else if(style == "Windows") {
       qApp->setFont( appFont, TRUE );
    } else if(style == "CDE") {
       QPalette p( QColor( 75, 123, 130 ) );
       p.setColor( QPalette::Active, QColorGroup::Base, QColor( 55, 77, 78 ) );
       p.setColor( QPalette::Inactive, QColorGroup::Base, QColor( 55, 77, 78 ) );
       p.setColor( QPalette::Disabled, QColorGroup::Base, QColor( 55, 77, 78 ) );
```

```
     p.setColor( QPalette::Active, QColorGroup::Highlight, Qt::white );
     p.setColor( QPalette::Active, QColorGroup::HighlightedText, QColor( 55, 77, 78 ) );
     p.setColor( QPalette::Inactive, QColorGroup::Highlight, Qt::white );
     p.setColor( QPalette::Inactive, QColorGroup::HighlightedText, QColor( 55, 77, 78 ) );
     p.setColor( QPalette::Disabled, QColorGroup::Highlight, Qt::white );
     p.setColor( QPalette::Disabled, QColorGroup::HighlightedText, QColor( 55, 77, 78 ) );
     p.setColor( QPalette::Active, QColorGroup::Foreground, Qt::white );
     p.setColor( QPalette::Active, QColorGroup::Text, Qt::white );
     p.setColor( QPalette::Active, QColorGroup::ButtonText, Qt::white );
     p.setColor( QPalette::Inactive, QColorGroup::Foreground, Qt::white );
     p.setColor( QPalette::Inactive, QColorGroup::Text, Qt::white );
     p.setColor( QPalette::Inactive, QColorGroup::ButtonText, Qt::white );
     p.setColor( QPalette::Disabled, QColorGroup::Foreground, Qt::lightGray );
     p.setColor( QPalette::Disabled, QColorGroup::Text, Qt::lightGray );
     p.setColor( QPalette::Disabled, QColorGroup::ButtonText, Qt::lightGray );
     qApp->setPalette( p, TRUE );
     qApp->setFont( QFont( "times", appFont.pointSize() ), TRUE );
   } else if(style == "Motif" || style == "MotifPlus") {
     QPalette p( QColor( 192, 192, 192 ) );
     qApp->setPalette( p, TRUE );
     qApp->setFont( appFont, TRUE );
   }
  }
}

void Themes::about()
{
  QMessageBox::about( this, "Qt Themes Example",
         "<p>This example demonstrates the concept of "
         "<b>generalized GUI styles </b> first introduced "
         " with the 2.0 release of Qt.</p>" );
}

void Themes::aboutQt()
{
  QMessageBox::aboutQt( this, "Qt Themes Example" );
}
```

wood.cpp

```
#include "wood.h"

#ifndef QT_NO_STYLE_WINDOWS

#include "qapplication.h"
#include "qcombobox.h"
#include "qpainter.h"
#include "qdrawutil.h" // for now
#include "qpixmap.h" // for now
#include "qpalette.h" // for now
#include "qwidget.h"
#include "qlabel.h"
#include "qimage.h"
#include "qpushbutton.h"
#include "qwidget.h"
```

```
#include "qrangecontrol.h"
#include "qscrollbar.h"
#include <limits.h>
#include "qstylefactory.h"

/* XPM */
static const char *polish_xpm[] = {
/* width height num_colors chars_per_pixel */
"    96    96      254            2",
/* colors */
".. c #9c4a34",
".# c #a4825c",
".a c #bc5e2c",
…
w.wbc.w#G#G#G#G#G#Ga0#P.1.r"
};


/* XPM */
static const char *button_xpm[] = {
/* width height num_colors chars_per_pixel */
"    96    96      254            2",
/* colors */
".. c #9c3218",
".# c #a4733e",
".a c #bc450a",
".b c #d4700c",
…
VbVbV#s#s.e.Oba.K.4aT.k.0"
};

static void drawroundrect( QPainter *p, QCOORD x, QCOORD y,
                   QCOORD w, QCOORD h, QCOORD d );

static inline int buttonthickness( int d );

static QRegion roundRectRegion( const QRect& g, int r );

static void get_combo_parameters( const QRect &r,  int &ew, int &awh, int &ax,
                        int &ay, int &sh, int &dh,  int &sy );

static int get_combo_extra_width( int h, int *return_awh = 0 );

enum { PointUp, PointDown, PointLeft, PointRight };

NorwegianWoodStyle::NorwegianWoodStyle() : QWindowsStyle()
{
}

/*!
  Reimplementation from QStyle
 */
void NorwegianWoodStyle::polish( QApplication *app)
{
```

```
oldPalette = app->palette();

// we simply create a nice QColorGroup with a couple of fancy wood
// pixmaps here and apply to it all widgets

QImage img(button_xpm);
QImage orig = img;
orig.detach();
QPixmap button;
button.convertFromImage(img);

int i;
for (i=0; i<img.numColors(); i++) {
 QRgb rgb = img.color(i);
 QColor c(rgb);
 rgb = c.dark(120).rgb();
 img.setColor(i,rgb);
}
QPixmap mid;
mid.convertFromImage(img);

img = orig;
img.detach();
for (i=0; i<img.numColors(); i++) {
 QRgb rgb = img.color(i);
 QColor c(rgb);
 rgb = c.light().rgb();
 img.setColor(i,rgb);
}
QPixmap light;
light.convertFromImage(img);

img = orig;
img.detach();
for (i=0; i<img.numColors(); i++) {
 QRgb rgb = img.color(i);
 QColor c(rgb);
 rgb = c.dark(180).rgb();
 img.setColor(i,rgb);
}
QPixmap dark;
dark.convertFromImage(img);

QImage bgimage(polish_xpm);
QPixmap background;
background.convertFromImage(bgimage);

img = bgimage;
img.detach();
for (i=0; i<img.numColors(); i++) {
 QRgb rgb = img.color(i);
 QColor c(rgb);
 rgb = c.dark(180).rgb();
```

```
     img.setColor(i,rgb);
    }
    sunkenDark = new QPixmap;
    sunkenDark->convertFromImage(img);

    img = bgimage;
    img.detach();
    for (i=0; i<img.numColors(); i++) {
     QRgb rgb = img.color(i);
     QColor c(rgb);
     rgb = c.light(130).rgb();
     img.setColor(i,rgb);
    }
    sunkenLight= new QPixmap;
    sunkenLight->convertFromImage(img);

    QPalette op(QColor(212,140,95));
    // QPalette op(white);
    QColorGroup active (op.active().foreground(),  QBrush(op.active().button(),button),
              QBrush(op.active().light(), light),  QBrush(op.active().dark(), dark),
              QBrush(op.active().mid(), mid),  op.active().text(),  Qt::white, QColor(236,182,120),
              QBrush(op.active().background(), background)  );
    QColorGroup disabled (op.disabled().foreground(), QBrush(op.disabled().button(),button),
              QBrush(op.disabled().light(), light), op.disabled().dark(), QBrush(op.disabled().mid(), mid),
              op.disabled().text(), Qt::white, QColor(236,182,120),
              QBrush(op.disabled().background(), background)   );

   app->setPalette(QPalette(active, disabled, active), TRUE );
}

void NorwegianWoodStyle::unPolish( QApplication *app)
{
    app->setPalette(oldPalette, TRUE);
}

/*!
  Reimplementation from QStyle
 */
void NorwegianWoodStyle::polish( QWidget* w)
{

    // the polish function sets some widgets to transparent mode and
    // some to translate background mode in order to get the full
    // benefit from the nice pixmaps in the color group.

    if ( !w->isTopLevel() ) {
     if ( w->inherits("QPushButton")  || w->inherits("QToolButton")  || w->inherits("QComboBox") ) {
        w->setAutoMask( TRUE );
        return;
     }
     if ( w->backgroundPixmap() )
        w->setBackgroundOrigin( QWidget::WindowOrigin );
    }
}
```

```
void NorwegianWoodStyle::unPolish( QWidget* w)
{
   // the polish function sets some widgets to transparent mode and
   // some to translate background mode in order to get the full
   // benefit from the nice pixmaps in the color group.
   if ( !w->isTopLevel() ) {
    if ( w->inherits("QPushButton")  || w->inherits("QToolButton")  || w->inherits("QComboBox") ) {
       w->setAutoMask( FALSE );
       return;
    }
    if ( w->backgroundPixmap() )
       w->setBackgroundOrigin( QWidget::WidgetOrigin );
   }
}

void NorwegianWoodStyle::drawPrimitive( PrimitiveElement pe, QPainter *p, const QRect &r,
                    const QColorGroup &cg, SFlags flags, const QStyleOption& opt ) const
{
   int x, y, w, h;
   r.rect( &x, &y, &w, &h );
   switch ( pe ) {
   case PE_ButtonCommand:
    {
       int d = QMIN( w, h ) / 2;
       int b = buttonthickness( d );

       QRegion internR = roundRectRegion( QRect(x + b, y + b,  w - 2 * b, h - 2 * b), d - b );
       QPen oldPen = p->pen();

       QBrush brush( flags & Style_Sunken ? cg.brush(QColorGroup::Mid) :
              cg.brush(QColorGroup::Button) );
       p->setClipRegion( internR );
       p->fillRect( r, brush );

       int e = QMIN( w, h ) / 2;
       QPoint p2( x + w - 1 - e, y + e );
       QPoint p3( x + e, y + h - 1 - e );

       QPointArray a;
       a.setPoints( 5, x,y, x+w-1, y, p2.x(), p2.y(), p3.x(), p3.y(), x, y + h - 1 );
       p->setClipRegion( QRegion(a) - internR );

       p->fillRect( r, (flags & Style_Sunken ? QBrush( cg.dark(), *sunkenDark)
                        : cg.brush(QColorGroup::Light)) );

       // A little inversion is needed the buttons
       // ( but not flat)
       if ( flags & Style_Raised || flags & Style_Sunken ) {
        a.setPoint( 0, x + w - 1, y + w - 1 );
        p->setClipRegion( QRegion( a ) - internR );

        p->fillRect( r, (flags & Style_Sunken ? QBrush( cg.light(), *sunkenLight) :
cg.brush( QColorGroup::Dark ) ) );
```

```
     }
     p->setClipRegion( internR );
     p->setClipping( FALSE );
     p->setPen( cg.foreground() );
     drawroundrect( p, x, y, w, h, d );
     p->setPen( oldPen );
     break;
   }
  case PE_ScrollBarAddLine:
   if ( flags & Style_Horizontal )
     drawSemicircleButton( p, r, PointRight, flags & Style_Down, cg );
   else
     drawSemicircleButton( p, r, PointDown, flags & Style_Down, cg );
   break;
  case PE_ScrollBarSubLine:
   if ( flags & Style_Horizontal )
     drawSemicircleButton( p, r, PointLeft, flags & Style_Down, cg );
   else
     drawSemicircleButton( p, r, PointUp, flags & Style_Down, cg );
   break;
  default:
   QWindowsStyle::drawPrimitive( pe, p, r, cg, flags, opt );
   break;
  }
}

void NorwegianWoodStyle::drawControl( ControlElement element, QPainter *p,
                 const QWidget *widget,  const QRect &r,  const QColorGroup &cg,
                 SFlags how, const QStyleOption& opt ) const
{
  switch( element ) {
  case CE_PushButton:
   {
     const QPushButton *btn;
     btn = ( const QPushButton * )widget;
     QColorGroup myCg( cg );
     SFlags flags = Style_Default;
     if ( btn->isOn() )
      flags |= Style_On;
     if ( btn->isDown() )
      flags |= Style_Down;
     if ( btn->isOn() || btn->isDown() )
      flags |= Style_Sunken;
     if ( btn->isDefault() )
      flags |= Style_Default;
     if ( ! btn->isFlat() && !(flags & Style_Down) )
      flags |= Style_Raised;

     int x1, y1, x2, y2;
     r.coords( &x1, &y1, &x2, &y2 );

     p->setPen( cg.foreground() );
     p->setBrush( QBrush( cg.button(), NoBrush ) );
```

```
        QBrush fill;
        if ( btn->isDown() )
          fill = cg.brush( QColorGroup::Mid );
        else if ( btn->isOn() )
          fill = QBrush( cg.mid(), Dense4Pattern );
        else
          fill = cg.brush( QColorGroup::Button );
        myCg.setBrush( QColorGroup::Mid, fill );

        if ( btn->isDefault() ) {
          x1 += 2;
          y1 += 2;
          x2 -= 2;
          y2 -= 2;
        }

        drawPrimitive( PE_ButtonCommand, p,  QRect( x1, y1, x2 - x1 + 1, y2 - y1 + 1),
                myCg, flags, opt );

        if ( btn->isDefault() ) {
          QPen pen( Qt::black, 4 );
          pen.setCapStyle( Qt::RoundCap );
          pen.setJoinStyle( Qt::RoundJoin );
          p->setPen( pen );
          drawroundrect( p, x1 - 1, y1 - 1, x2 - x1 + 3, y2 - y1 + 3, 8 );
        }

        if ( btn->isMenuButton() ) {
          int dx = ( y1 - y2 - 4 ) / 3;

          // reset the flags
          flags = Style_Default;
          if ( btn->isEnabled() )
              flags |= Style_Enabled;
          drawPrimitive( PE_ArrowDown, p, QRect( x2 - dx, dx, y1, y2 - y1), myCg, flags, opt );
        }

        if ( p->brush().style() != NoBrush )
          p->setBrush( NoBrush );
        break;
      }
    case CE_PushButtonLabel:
      {
        const QPushButton *btn;
        btn = (const QPushButton*)widget;
        int x, y, w, h;
        r.rect( &x, &y, &w, &h );

        int x1, y1, x2, y2;
        r.coords( &x1, &y1, &x2, &y2 );
        int dx = 0;
        int dy = 0;
        if ( btn->isMenuButton() )
          dx = ( y2 - y1 ) / 3;
        if ( dx || dy )
          p->translate( dx, dy );

        x += 2;
        y += 2;
        w -= 4;
        h -= 4;
        drawItem( p, QRect( x, y, w, h ), AlignCenter | ShowPrefix, cg, btn->isEnabled(),
                  btn->pixmap(), btn->text(), -1, (btn->isDown() || btn->isOn()) ? &cg.brightText()
                  : &cg.buttonText() );
        if ( dx || dy )
          p->translate( -dx, -dy );
        break;
      }
   default:
     QWindowsStyle::drawControl( element, p, widget, r, cg, how, opt );
     break;
   }
}

void NorwegianWoodStyle::drawControlMask( ControlElement element,  QPainter *p,
                         const QWidget *widget, const QRect &r, const QStyleOption& opt ) const
{
   switch( element ) {
   case CE_PushButton:
     {
        int d = QMIN( r.width(), r.height() ) / 2;
        p->setPen( color1 );
        p->setBrush( color1 );
        drawroundrect( p, r.x(), r.y(), r.width(), r.height(), d );
        break;
     }
   default:
     QWindowsStyle::drawControlMask( element, p, widget, r, opt );
     break;
   }
}

void NorwegianWoodStyle::drawComplexControl( ComplexControl cc,  QPainter *p,
    const QWidget *widget, const QRect &r, const QColorGroup &cg,  SFlags how,
    SCFlags sub,  SCFlags subActive,  const QStyleOption& opt ) const
{
   switch( cc ) {
   case CC_ComboBox:
     {
        const QComboBox *cmb;
        cmb = (const QComboBox*)widget;

        int awh, ax, ay, sh, sy, dh, ew;
        get_combo_parameters( subRect(SR_PushButtonContents, widget), ew, awh, ax, ay, sh, dh, sy );
        drawPrimitive( PE_ButtonCommand, p, r, cg, Style_Raised, opt );
        QStyle *mstyle = QStyleFactory::create( "Motif" );
        if ( mstyle )
          mstyle->drawPrimitive( PE_ArrowDown, p, QRect(ax, ay, awh, awh), cg, how, opt );
        else
          drawPrimitive( PE_ArrowDown, p, QRect(ax, ay, awh, awh), cg, how, opt );

        QPen oldPen = p->pen();
        p->setPen( cg.light() );
        p->drawLine( ax, sy, ax + awh - 1, sy );
        p->drawLine( ax, sy, ax, sy + sh - 1 );
        p->setPen( cg.dark() );
        p->drawLine( ax + 1, sy + sh - 1, ax + awh - 1, sy + sh - 1 );
        p->drawLine( ax + awh - 1, sy + 1, ax + awh - 1, sy + sh - 1 );
        p->setPen( oldPen );

        if ( cmb->editable() ) {
          QRect r( querySubControlMetrics(CC_ComboBox, widget, SC_ComboBoxEditField, opt) );
          qDrawShadePanel( p, r, cg, TRUE, 1, &cg.brush(QColorGroup::Button) );
        }

        break;
     }
   default:
     QWindowsStyle::drawComplexControl( cc, p, widget, r, cg, how,
                                sub, subActive, opt );
     break;
   }
}

void NorwegianWoodStyle::drawComplexControlMask( ComplexControl control, QPainter *p,
             const QWidget *widget, const QRect &r, const QStyleOption& opt ) const
{
   switch ( control ) {
   case CC_ComboBox:
     {
        int d = QMIN( r.width(), r.height() ) / 2;
        p->setPen( color1 );
        p->setBrush( color1 );
        drawroundrect( p, r.x(), r.y(), r.width(), r.height(), d );
        break;
     }
   default:
     QWindowsStyle::drawComplexControlMask( control, p, widget, r, opt );
     break;
   }
}

QRect NorwegianWoodStyle::querySubControlMetrics( ComplexControl control,
                                    const QWidget *widget,  SubControl sc,  const QStyleOption& opt ) const
{
   QRect rect;
   switch ( control ) {
   case CC_ComboBox:
     {
        switch( sc ) {
        case SC_ComboBoxEditField:
          {
```

```
        rect = subRect( SR_PushButtonContents, widget );
        int ew = get_combo_extra_width( rect.height(), 0 );
        rect.setRect( rect.x() + 1, rect.y() + 1,
                 rect.width() - 2 - ew, rect.height() - 2 );
        break;
      }
     default:
      rect = QWindowsStyle::querySubControlMetrics( control, widget,  sc, opt );
      break;
     }
     break;
   }
  case CC_ScrollBar:
   {
     const QScrollBar* sb;
     sb = (const QScrollBar*)widget;
     bool horz = sb->orientation() == QScrollBar::Horizontal;
     int b = 2;
     int w = horz ? sb->height() : sb->width();

     switch ( sc ) {
     case SC_ScrollBarAddLine:
      rect.setRect( b, b, w - 2 * b, w - 2 * b );
      if ( horz )
        rect.moveBy( sb->width() - w, 0 );
      else
        rect.moveBy( 0, sb->height() - w );
      break;
     case SC_ScrollBarSubLine:
      rect.setRect( b, b, w - 2 * b, w - 2 * b );
      break;
     default:
      rect = QWindowsStyle::querySubControlMetrics( control, widget,  sc, opt );
      break;
     }
     break;
   }
  default:
   rect = QWindowsStyle::querySubControlMetrics( control, widget,  sc, opt );
   break;
  }
  return rect;
}

QRect NorwegianWoodStyle::subRect( SubRect sr, const QWidget * widget ) const
{
  QRect r;
  switch ( sr ) {
  case SR_PushButtonContents:
   {
     const QPushButton *btn;
     btn = (const QPushButton*)widget;
     r = btn->rect();
     int d = QMIN( r.width(), r.height() ) / 2;
```

```
        int b = buttonthickness( d );

        d -= b;
        b++;

        if ( r.width() < r.height() )
          r.setRect( r.x() + b, r.y() + d,  r.width() - 2 * b, r.height() - 2 * d );
        else
          r.setRect( r.x() + d, r.y() + b,  r.width() - 2 * d, r.height() - 2 * b );
        break;
     }
   case SR_ComboBoxFocusRect:
     {
        r = subRect( SR_PushButtonContents, widget );
        int ew = get_combo_extra_width( r.height() );
        r.setRect( r.x() + 1, r.y() + 1, r.width() - 2 - ew, r.height() - 2 );
        break;
     }
   default:
     r = QWindowsStyle::subRect( sr, widget );
     break;
   }
   return r;
}

static void drawroundrect( QPainter *p, QCOORD x, QCOORD y,
                  QCOORD w, QCOORD h, QCOORD d )
{
   int rx = (200*d)/w;
   int ry = (200*d)/h;
   p->drawRoundRect( x, y, w, h, rx, ry );
}

static QRegion roundRectRegion( const QRect& g, int r )
{
   QPointArray a;
   a.setPoints( 8, g.x()+r, g.y(), g.right()-r, g.y(), g.right(), g.y()+r, g.right(), g.bottom()-r,
         g.right()-r, g.bottom(), g.x()+r, g.bottom(), g.x(), g.bottom()-r, g.x(), g.y()+r );
   QRegion reg( a );
   int d = r*2-1;
   reg += QRegion( g.x(),g.y(),r*2,r*2, QRegion::Ellipse );
   reg += QRegion( g.right()-d,g.y(),r*2,r*2, QRegion::Ellipse );
   reg += QRegion( g.x(),g.bottom()-d,r*2,r*2, QRegion::Ellipse );
   reg += QRegion( g.right()-d,g.bottom()-d,r*2,r*2, QRegion::Ellipse );
   return reg;
}

static int get_combo_extra_width( int h, int *return_awh )
{
   int awh;
   if ( h < 8 ) {
     awh = 6;
   } else if ( h < 14 ) {
     awh = h - 2;
```

```
    } else {
     awh = h/2;
    }
    if ( return_awh )
     *return_awh = awh;
    return awh*3/2;
}


static void get_combo_parameters( const QRect &r, int &ew, int &awh, int &ax,
                         int &ay, int &sh, int &dh, int &sy )
{
    ew = get_combo_extra_width( r.height(), &awh );

    sh = (awh+3)/4;
    if ( sh < 3 )
     sh = 3;
    dh = sh/2 + 1;

    ay = r.y() + (r.height()-awh-sh-dh)/2;
    if ( ay < 0 ) {
     //panic mode
     ay = 0;
     sy = r.height();
    } else {
     sy = ay+awh+dh;
    }
    ax = r.x() + r.width() - ew +(ew-awh)/2;
}

static inline int buttonthickness( int d )
{ return  d > 20 ? 5 : ( d < 10 ? 2: 3 ); }

void NorwegianWoodStyle::drawSemicircleButton( QPainter *p, const QRect &r,  int dir, bool sunken,
                              const QColorGroup &g ) const
{
    int b =  pixelMetric( PM_ScrollBarExtent ) > 20 ? 3 : 2;

     QRegion extrn(  r.x(),   r.y(),   r.width(),     r.height(),     QRegion::Ellipse );
     QRegion intern( r.x()+b, r.y()+b, r.width()-2*b, r.height()-2*b, QRegion::Ellipse );
    int w2 = r.width()/2;
    int h2 = r.height()/2;

    int bug = 1; //off-by-one somewhere!!!???

    switch( dir ) {
    case PointRight:
     extrn +=  QRegion( r.x(),  r.y(),   w2,     r.height() );
     intern += QRegion( r.x()+b,r.y()+b, w2-2*b, r.height()-2*b );
     break;
    case PointLeft:
     extrn +=  QRegion( r.x()+w2,  r.y(),   w2,     r.height() );
     intern += QRegion( r.x()+w2+b,r.y()+b, w2-2*b, r.height()-2*b );
     break;
```

```
  case PointUp:
   extrn += QRegion( r.x(), r.y()+h2,  r.width(),   h2 );
   intern += QRegion( r.x()+b,r.y()+h2+b, r.width()-2*b-bug, h2-2*b-bug );
   break;
  case PointDown:
   extrn += QRegion( r.x(), r.y(),  r.width(),   h2 );
   intern += QRegion( r.x()+b,r.y()+b, r.width()-2*b-bug, h2-2*b-bug );
   break;
  }

  extrn = extrn - intern;
  QPointArray a;
  a.setPoints( 3, r.x(), r.y(), r.x(), r.bottom(), r.right(), r.top() );

  QRegion oldClip = p->clipRegion();
  bool bReallyClip = p->hasClipping(); // clip only if we really want.
  p->setClipRegion( intern );
  p->fillRect( r, g.brush( QColorGroup::Button ) );

  p->setClipRegion( QRegion(a)&extrn );
  p->fillRect( r, sunken ? g.dark() : g.light() );

  a.setPoints( 3, r.right(), r.bottom(), r.x(), r.bottom(),
       r.right(), r.top() );
  p->setClipRegion( QRegion(a) & extrn );
  p->fillRect( r, sunken ? g.light() : g.dark() );

  p->setClipRegion( oldClip );
  p->setClipping( bReallyClip );
}

#endif
```

wood.h

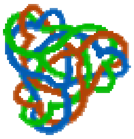
```
#ifndef WOOD_H
#define WOOD_H

#include <qpalette.h>

#ifndef QT_NO_STYLE_WINDOWS

#include <qwindowsstyle.h>

class NorwegianWoodStyle : public QWindowsStyle
{
public:
  NorwegianWoodStyle();
  void polish( QApplication*);
  void polish( QWidget* );
  void unPolish( QWidget* );
  void unPolish( QApplication*);

  void drawPrimitive( PrimitiveElement pe, QPainter *p, const QRect &r, const QColorGroup &cg,
        SFlags flags = Style_Default, const QStyleOption& = QStyleOption::Default ) const;
```

```
   void drawControl( ControlElement element, QPainter *p, const QWidget *widget,
             const QRect &r, const QColorGroup &cg, SFlags how = Style_Default,
             const QStyleOption& = QStyleOption::Default ) const;

   void drawControlMask( ControlElement element, QPainter *p, const QWidget *widget,
              const QRect &r,  const QStyleOption& = QStyleOption::Default ) const;

   void drawComplexControl( ComplexControl cc, QPainter *p,  const QWidget *widget,
                 const QRect &r, const QColorGroup &cg,  SFlags how = Style_Default,
                 SCFlags sub = SC_All,  SCFlags subActive = SC_None,
                 const QStyleOption& = QStyleOption::Default ) const;

   void drawComplexControlMask( ComplexControl control, QPainter *p, const QWidget *widget,
                  const QRect &r, const QStyleOption& = QStyleOption::Default ) const;

   QRect querySubControlMetrics( ComplexControl control,  const QWidget *widget,
                   SubControl sc, const QStyleOption& = QStyleOption::Default ) const;

   QRect subRect( SubRect r, const QWidget *widget ) const;

private:
   void drawSemicircleButton(QPainter *p, const QRect &r, int dir,
                  bool sunken, const QColorGroup &g ) const;
   QPalette oldPalette;
   QPixmap *sunkenDark;
   QPixmap *sunkenLight;

};

#endif

#endif
```

main.cpp

```
#include <qapplication.h>
#include <qwindowsstyle.h>
#include "themes.h"
#include "metal.h"

int main( int argc, char ** argv )
{
   QApplication::setColorSpec( QApplication::ManyColor );
   QApplication a( argc, argv );

   Themes themes;
   themes.setCaption( "Qt Example - Themes (QStyle)" );
   themes.resize( 640, 400 );
   a.setMainWidget( &themes );
   themes.show();

   return a.exec();
}
```

trolltech.gif



66. 스레드

1) Thread-prodcons

prodcons.pro

```
TEMPLATE  = app
TARGET      = prodcons
CONFIG      += qt warn_on
SOURCES         = prodcons.cpp
CLEAN_FILES  = prodcons.out
```

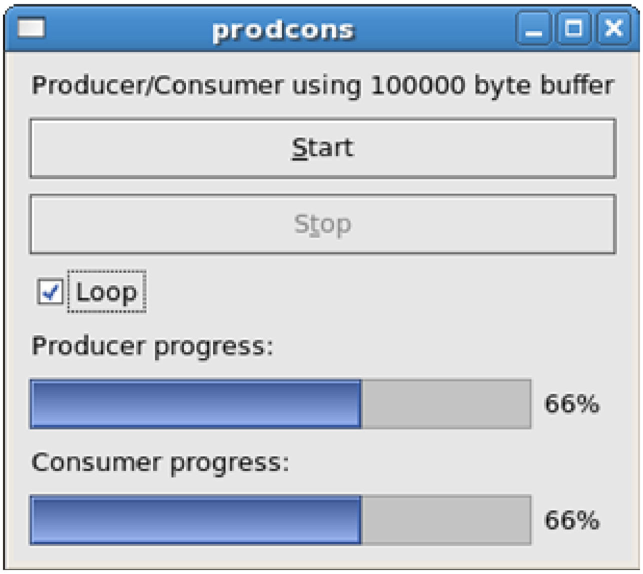
prodcons.cpp

```
#include <qthread.h>
#include <qwaitcondition.h>
#include <qmutex.h>
#include <qapplication.h>
#include <qwidget.h>
#include <qpushbutton.h>
#include <qcheckbox.h>
#include <qprogressbar.h>
#include <qlayout.h>
#include <qevent.h>
#include <qlabel.h>
#include <qcstring.h>
#include <qtextstream.h>
#include <qfile.h>

#include <stdio.h>

// 50kb buffer
#define BUFSIZE (100*1000)
#define PRGSTEP (BUFSIZE / 50)
#define BLKSIZE (8)
QByteArray bytearray;

class ProdEvent : public QCustomEvent
{
public:
    ProdEvent(long s, bool d)
     : QCustomEvent(QEvent::User + 100), sz(s), dn(d)
    { ; }

    long size() const { return sz; }
    bool done() const { return dn; }

private:
    long sz;
    bool dn;
};

class ProdThread : public QThread
{
public:
    ProdThread(QObject *r, QMutex *m, QWaitCondition *c);

    void stop();
    void run();

private:
    QObject *receiver;
    QMutex *mutex;
    QWaitCondition *condition;

    bool done;
};

ProdThread::ProdThread(QObject *r, QMutex *m, QWaitCondition *c)
    : receiver(r), mutex(m), condition(c), done(FALSE)
{
}

void ProdThread::stop()
{
    mutex->lock();
    done = TRUE;
    mutex->unlock();
}

void ProdThread::run()
{
    bool stop = FALSE;
    done = FALSE;

    uchar *buffer = new uchar[BUFSIZE];
    int pos = 0, oldpos = 0;
    int loop = 1;
    int lastpostedpos = 0;

    ProdEvent *pe = new ProdEvent(pos, done);
    QApplication::postEvent(receiver, pe);

    while (! stop) {
     oldpos = pos;
     int i;
     for (i = 0; i < BLKSIZE && pos < BUFSIZE; i++) {
         buffer[pos++] = (loop % 2) ? 'o' : 'e';
     }

     mutex->lock();

     if (pos == BUFSIZE) {
```

```
     done = TRUE;
   }

   while (! bytearray.isNull() && ! stop) {
      condition->wakeOne();
      condition->wait(mutex);

     stop = done;
   }

   stop = done;
   bytearray.duplicate((const char *) (buffer + oldpos), pos - oldpos);
   condition->wakeOne();

   mutex->unlock();

   if ( pos - lastpostedpos > PRGSTEP || stop ) {
     lastpostedpos = pos;
     ProdEvent *pe = new ProdEvent(pos, stop);
     QApplication::postEvent(receiver, pe);
   }

   loop++;
  }

  condition->wakeOne();

  delete [] buffer;
}

class ConsEvent : public QCustomEvent
{
public:
  ConsEvent(long s)
   : QCustomEvent(QEvent::User + 101), sz(s)
  { ; }

  long size() const { return sz; }

private:
  long sz;
};

class ConsThread : public QThread
{
public:
  ConsThread(QObject *r, QMutex *m, QWaitCondition *c);

  void stop();
  void run();


private:
  QObject *receiver;
```

```
    QMutex *mutex;
    QWaitCondition *condition;

    bool done;
};

ConsThread::ConsThread(QObject *r, QMutex *m, QWaitCondition *c)
    : receiver(r), mutex(m), condition(c), done(FALSE)
{
}

void ConsThread::stop()
{
    mutex->lock();
    done = TRUE;
    mutex->unlock();
}

void ConsThread::run()
{
    bool stop = FALSE;
    done = FALSE;

    QFile file("prodcons.out");
    file.open(IO_WriteOnly);

    long size = 0;
    long lastsize = 0;

    ConsEvent *ce = new ConsEvent(size);
    QApplication::postEvent(receiver, ce);

    while (! stop) {
     mutex->lock();

     while (bytearray.isNull() && ! stop) {
          condition->wakeOne();
          condition->wait(mutex);

        stop = done;
     }

     if (size < BUFSIZE) {
        file.writeBlock(bytearray.data(), bytearray.size());
        size += bytearray.size();
        bytearray.resize(0);
     }

     stop = done || size >= BUFSIZE;

     mutex->unlock();

     if ( size - lastsize > 1000 || stop ) {
        lastsize = size;
        ConsEvent *ce = new ConsEvent(size);
        QApplication::postEvent(receiver, ce);
      }
    }

    file.flush();
    file.close();
}

class ProdCons : public QWidget
{
    Q_OBJECT

public:
    ProdCons();
    ~ProdCons();

    void customEvent(QCustomEvent *);

public slots:
    void go();
    void stop();

private:
    QMutex mutex;
    QWaitCondition condition;

    ProdThread *prod;
    ConsThread *cons;

    QPushButton *startbutton, *stopbutton;
    QCheckBox *loopcheckbox;
    QProgressBar *prodbar, *consbar;
    bool stopped;
    bool redraw;
};

ProdCons::ProdCons()
    : QWidget(0, "producer consumer widget"),
      prod(0), cons(0), stopped(FALSE), redraw(TRUE)
{
    startbutton = new QPushButton("&Start", this);
    connect(startbutton, SIGNAL(clicked()), SLOT(go()));

    stopbutton = new QPushButton("S&top", this);
    connect(stopbutton, SIGNAL(clicked()), SLOT(stop()));
    stopbutton->setEnabled(FALSE);

    loopcheckbox = new QCheckBox("Loop", this);
    loopcheckbox->setChecked(FALSE);

    prodbar = new QProgressBar(BUFSIZE, this);
    consbar = new QProgressBar(BUFSIZE, this);
    QVBoxLayout *vbox = new QVBoxLayout(this, 8, 8);
    vbox->addWidget(new QLabel(QString("Producer/Consumer using %1 byte buffer").
                    arg(BUFSIZE), this));
    vbox->addWidget(startbutton);
    vbox->addWidget(stopbutton);
    vbox->addWidget(loopcheckbox);
    vbox->addWidget(new QLabel("Producer progress:", this));
    vbox->addWidget(prodbar);
    vbox->addWidget(new QLabel("Consumer progress:", this));
    vbox->addWidget(consbar);
}

ProdCons::~ProdCons()
{
    stop();

    if (prod) {
     delete prod;
     prod = 0;
    }

    if (cons) {
     delete cons;
     cons = 0;
    }
}

void ProdCons::go()
{
    stopped = FALSE;

    mutex.lock();

    if ( redraw ) {
     startbutton->setEnabled(FALSE);
     stopbutton->setEnabled(TRUE);
    }

    // start the consumer first
    if (! cons)
        cons = new ConsThread(this, &mutex, &condition);
    cons->start();

    // wait for consumer to signal that it has started
    condition.wait(&mutex);

    if (! prod)
        prod = new ProdThread(this, &mutex, &condition);
    prod->start();
    mutex.unlock();
}

void ProdCons::stop()
{
    if (prod && prod->running()) {
     prod->stop();
       condition.wakeAll();
     prod->wait();
    }

    if (cons && cons->running()) {
     cons->stop();
       condition.wakeAll();
     cons->wait();
    }

    if ( redraw ) {
     // no point in repainting these buttons so many times is we are looping...
     startbutton->setEnabled(TRUE);
     stopbutton->setEnabled(FALSE);
    }

    stopped = TRUE;
}

void ProdCons::customEvent(QCustomEvent *e)
{
    switch (e->type()) {
    case QEvent::User + 100:
     {
        // ProdEvent
        ProdEvent *pe = (ProdEvent *) e;

        if (pe->size() == 0 ||
         pe->size() == BUFSIZE ||
         pe->size() - prodbar->progress() >= PRGSTEP)
         prodbar->setProgress(pe->size());

        // reap the threads
        if (pe->done()) {
         bool loop = (loopcheckbox->isChecked() && ! stopped);
         bool save_redraw = redraw;
         redraw = !loop;

         stop();

         if (loop)
             go();

         redraw = save_redraw;
        }

        break;
     }

    case QEvent::User + 101:
     {
        // ConsEvent
     ConsEvent *ce = (ConsEvent *) e;

     if (ce->size() == 0 ||
      ce->size() == BUFSIZE ||
      ce->size() - consbar->progress() >= PRGSTEP)
      consbar->setProgress(ce->size());

     break;
   }

  default:
   {
     ;
   }
  }
}

int main(int argc, char **argv)
{
  QApplication app(argc, argv);
  ProdCons prodcons;
  app.setMainWidget(&prodcons);
  prodcons.show();
  return app.exec();
}

#include "prodcons.moc"
```

실 행



Producer/Consumer using 100000 byte buffer

2) semaphores

semaphores.pro

```
TEMPLATE  = app
TARGET      = semaphores
```

```
CONFIG       += qt warn_on release thread
HEADERS       =
SOURCES       = main.cpp
INTERFACES    =
```

main.cpp

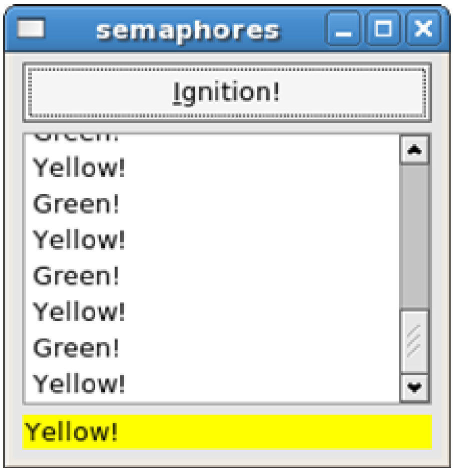
```
#include <qapplication.h>
#include <qwidget.h>
#include <qpushbutton.h>
#include <qmultilineedit.h>
#include <qthread.h>
#include <qsemaphore.h>
#include <qmutex.h>
#include <qlayout.h>
#include <qmessagebox.h>
#include <qlabel.h>

#if defined(QT_NO_THREAD)
#  error Thread support not enabled.
#endif

// Use pointers to create semaphores after QApplication object!
QSemaphore* yellowSem, *greenSem;

class YellowThread : public QThread
{
public:
   YellowThread(QWidget *o) : receiver(o), stopped(FALSE)
   { ; }

   void run();
   void stop();


private:
   QWidget *receiver;
   QMutex mutex;
   bool stopped;
};

void YellowThread::run()
{
   for (int i = 0; i < 20; i++) {
    (*yellowSem)++;

    QCustomEvent *event = new QCustomEvent(12345);
    event->setData(new QString("Yellow!"));
    QApplication::postEvent(receiver, event);
    msleep(200);

    (*greenSem)--;

    mutex.lock();
    if (stopped) {
        stopped = FALSE;
        mutex.unlock();
        break;
     }
     mutex.unlock();
   }

   (*yellowSem)++;

   QCustomEvent *event = new QCustomEvent(12346);
   event->setData(new QString("Yellow!"));
   QApplication::postEvent(receiver, event);

   (*greenSem)--;
}

void YellowThread::stop()
{
   mutex.lock();
   stopped = TRUE;
   mutex.unlock();
}

class GreenThread: public QThread
{
public:
  GreenThread(QWidget *o) : receiver(o), stopped( FALSE )
   { ; }

   void run();
   void stop();


private:
   QWidget *receiver;
   QMutex mutex;
   bool stopped;
};

void GreenThread::run()
{
   for (int i = 0; i < 20; i++) {
     (*greenSem)++;

     QCustomEvent *event = new QCustomEvent(12345);
     event->setData(new QString("Green!"));
     QApplication::postEvent(receiver, event);
     msleep(200);

     (*yellowSem)--;

     mutex.lock();
     if (stopped) {
        stopped = FALSE;
```

```
       mutex.unlock();
       break;
     }
    mutex.unlock();
   }

   (*greenSem)++;

   QCustomEvent *event = new QCustomEvent(12346);
   event->setData(new QString("Green!"));
   QApplication::postEvent(receiver, event);
   msleep(10);

   (*yellowSem)--;
}

void GreenThread::stop()
{
   mutex.lock();
   stopped = TRUE;
   mutex.unlock();
}

class SemaphoreExample : public QWidget
{
   Q_OBJECT
public:
   SemaphoreExample();
   ~SemaphoreExample();

   void customEvent(QCustomEvent *);

public slots:
   void startExample();

protected:

private:
   QMultiLineEdit *mlineedit;
   QPushButton *button;
   QLabel *label;

   YellowThread yellowThread;
   GreenThread greenThread;
};

SemaphoreExample::SemaphoreExample()  : QWidget(), yellowThread(this), greenThread(this)
{
   yellowSem = new QSemaphore(1);
   greenSem = new QSemaphore(1);

   button = new QPushButton("&Ignition!", this);
   connect(button, SIGNAL(clicked()), SLOT(startExample()));
```

```
  mlineedit = new QMultiLineEdit(this);
  label = new QLabel(this);

  QVBoxLayout *vbox = new QVBoxLayout(this, 5);
  vbox->addWidget(button);
  vbox->addWidget(mlineedit);
  vbox->addWidget(label);
}

SemaphoreExample::~SemaphoreExample()
{
  bool stopYellow = yellowThread.running(),
    stopGreen = greenThread.running();
  if (stopYellow)
   yellowThread.stop();
  if (greenThread.running())
   greenThread.stop();
  if (stopYellow)
   yellowThread.wait();
  if (stopGreen)
   greenThread.wait();
  delete yellowSem;
  delete greenSem;
}

void SemaphoreExample::startExample()
{
  if (yellowThread.running() || greenThread.running()) {
   QMessageBox::information(this, "Sorry",
             "The threads have not completed yet, and must finish before "
             "they can be started again.");

   return;
  }

  mlineedit->clear();

  while (yellowSem->available() < yellowSem->total()) (*yellowSem)--;
  (*yellowSem)++;

  yellowThread.start();
  greenThread.start();
}

void SemaphoreExample::customEvent(QCustomEvent *event) {
  switch (event->type()) {
  case 12345:
   {
     QString *s = (QString *) event->data();

     mlineedit->append(*s);

     if (*s == "Green!")
      label->setBackgroundColor(green);
```

```
          else
            label->setBackgroundColor(yellow);
          label->setText(*s);

          delete s;

          break;
      }

    case 12346:
      {
          QString *s = (QString *) event->data();

          QMessageBox::information(this, (*s) + " - Finished",
                                   "The thread creating the \"" + *s +
                                   "\" events has finished.");
          delete s;

          break;
      }

    default:
      {
          qWarning("Unknown custom event type: %d", event->type());
      }
    }
}

int main(int argc, char **argv)
{
    QApplication app(argc, argv);
    SemaphoreExample se;
    app.setMainWidget(&se);
    se.show();
    return app.exec();
}

#include "main.moc"
```

실행



Green!
Yellow!
Green!
Yellow!
Green!
Yellow!
Green!
Yellow!

Yellow!

67. 세목놓기

이것은 세목놓기(Tic-tac-toe)게임의 실례이다.

tictac.pro

```
TEMPLATE  = app
TARGET      = tictac
CONFIG      += qt warn_on release
HEADERS        = tictac.h
SOURCES        = main.cpp \
          tictac.cpp
```

tictac.cpp

```
#include "tictac.h"
#include <qapplication.h>
#include <qpainter.h>
#include <qdrawutil.h>
#include <qcombobox.h>
#include <qcheckbox.h>
#include <qlabel.h>
#include <qlayout.h>
#include <stdlib.h>                    // rand() function
#include <qdatetime.h>                    // seed for rand()

//* TicTacButton member functions
// Creates a TicTacButton
TicTacButton::TicTacButton( QWidget *parent ) : QPushButton( parent )
{
   t = Blank;                          // initial type
}

// Paints TicTacButton
void TicTacButton::drawButtonLabel( QPainter *p )
{
   QRect r = rect();
   p->setPen( QPen( white,2 ) );          // set fat pen
   if ( t == Circle ) {
     p->drawEllipse( r.left()+4, r.top()+4, r.width()-8, r.height()-8 );
   } else if ( t == Cross ) {              // draw cross
     p->drawLine( r.topLeft()   +QPoint(4,4), r.bottomRight()-QPoint(4,4));
     p->drawLine( r.bottomLeft()+QPoint(4,-4),r.topRight()   -QPoint(4,-4));
   }
}

//* TicTacGameBoard member functions
// Creates a game board with N x N buttons and connects the "clicked()"
// signal of all buttons to the "buttonClicked()" slot.
TicTacGameBoard::TicTacGameBoard( int n, QWidget *parent, const char *name )
   : QWidget( parent, name )
{
   st = Init;                         // initial state
   nBoard = n;
   n *= n;                        // make square
   comp_starts = FALSE;               // human starts
   buttons = new TicTacButtons(n);          // create real buttons
   btArray = new TicTacArray(n);       // create button model
   QGridLayout * grid = new QGridLayout( this, nBoard, nBoard, 4 );
   QPalette p( blue );
   for ( int i=0; i<n; i++ ) {             // create and connect buttons
     TicTacButton *ttb = new TicTacButton( this );
     ttb->setPalette( p );
     ttb->setEnabled( FALSE );
     connect( ttb, SIGNAL(clicked()), SLOT(buttonClicked()) );
     grid->addWidget( ttb, i%nBoard, i/nBoard );
     buttons->insert( i, ttb );
     btArray->at(i) = TicTacButton::Blank;// initial button type
   }
   QTime t = QTime::currentTime();        // set random seed
   srand( t.hour()*12+t.minute()*60+t.second()*60 );
}

TicTacGameBoard::~TicTacGameBoard()
{
   delete buttons;
   delete btArray;
}

// TicTacGameBoard::computerStarts( bool v )
// Computer starts if v=TRUE. The human starts by default.
void TicTacGameBoard::computerStarts( bool v )
{
   comp_starts = v;
}

// TicTacGameBoard::newGame()
// Clears the game board and prepares for a new game
void TicTacGameBoard::newGame()
{
   st = HumansTurn;
   for ( int i=0; i<nBoard*nBoard; i++ )
```

```
      btArray->at(i) = TicTacButton::Blank;
    if ( comp_starts )
      computerMove();
    else
      updateButtons();
}

// TicTacGameBoard::buttonClicked()       - SLOT
// This slot is activated when a TicTacButton emits the signal "clicked()",
// i.e. the user has clicked on a TicTacButton.
void TicTacGameBoard::buttonClicked()
{
    if ( st != HumansTurn )               // not ready
      return;
    int i = buttons->findRef( (TicTacButton*)sender() );
    TicTacButton *b = buttons->at(i);         // get piece that was pressed
    if ( b->type() == TicTacButton::Blank ) {   // empty piece?
      btArray->at(i) = TicTacButton::Circle;
      updateButtons();
      if ( checkBoard( btArray ) == 0 )    // not a winning move?
          computerMove();
      int s = checkBoard( btArray );
      if ( s ) {                     // any winners yet?
          st = s == TicTacButton::Circle ? HumanWon : ComputerWon;
          emit finished();
      }
    }
}

// TicTacGameBoard::updateButtons()
// Updates all buttons that have changed state
void TicTacGameBoard::updateButtons()
{
    for ( int i=0; i<nBoard*nBoard; i++ ) {
      if ( buttons->at(i)->type() != btArray->at(i) )
          buttons->at(i)->setType( (TicTacButton::Type)btArray->at(i) );
      buttons->at(i)->setEnabled( buttons->at(i)->type() ==  TicTacButton::Blank );
    }
}

// TicTacGameBoard::checkBoard()
// Checks if one of the players won the game, works for any board size.
// Returns:
//  - TicTacButton::Cross  if the player with X buttons won
//  - TicTacButton::Circle if the player with O buttons won
//  - Zero (0) if there is no winner yet
int TicTacGameBoard::checkBoard( TicTacArray *a )
{
    int  t = 0;
    int  row, col;
    bool won = FALSE;
    for ( row=0; row<nBoard && !won; row++ ) {// check horizontal
      t = a->at(row*nBoard);
      if ( t == TicTacButton::Blank )
```

```
     continue;
   col = 1;
   while ( col<nBoard && a->at(row*nBoard+col) == t )
     col++;
   if ( col == nBoard )
     won = TRUE;
  }
  for ( col=0; col<nBoard && !won; col++ ) {    // check vertical
   t = a->at(col);
   if ( t == TicTacButton::Blank )
     continue;
   row = 1;
   while ( row<nBoard && a->at(row*nBoard+col) == t )
     row++;
   if ( row == nBoard )
     won = TRUE;
  }
  if ( !won ) {                   // check diagonal top left
   t = a->at(0);                  //   to bottom right
   if ( t != TicTacButton::Blank ) {
     int i = 1;
     while ( i<nBoard && a->at(i*nBoard+i) == t )
      i++;
     if ( i == nBoard )
      won = TRUE;
   }
  }
  if ( !won ) {                   // check diagonal bottom left
   int j = nBoard-1;              //   to top right
   int i = 0;
   t = a->at(i+j*nBoard);
   if ( t != TicTacButton::Blank ) {
     i++; j--;
     while ( i<nBoard && a->at(i+j*nBoard) == t ) {
      i++; j--;
     }
     if ( i == nBoard )
      won = TRUE;
   }
  }
  if ( !won )                     // no winner
   t = 0;
  return t;
}

// TicTacGameBoard::computerMove()
// Puts a piece on the game board. Very, very simple.
void TicTacGameBoard::computerMove()
{
  int numButtons = nBoard*nBoard;
  int *altv = new int[numButtons];      // buttons alternatives
  int altc = 0;
  int stopHuman = -1;
  TicTacArray a = btArray->copy();
```

```
    int i;
    for ( i=0; i<numButtons; i++ ) {       // try all positions
      if ( a[i] != TicTacButton::Blank )   // already a piece there
          continue;
      a[i] = TicTacButton::Cross;       // test if computer wins
      if ( checkBoard(&a) == a[i] ) {       // computer will win
          st = ComputerWon;
          stopHuman = -1;
          break;
      }
      a[i] = TicTacButton::Circle;      // test if human wins
      if ( checkBoard(&a) == a[i] ) {       // oops...
          stopHuman = i;                // remember position
          a[i] = TicTacButton::Blank;       // restore button
          continue;                     // computer still might win
      }
      a[i] = TicTacButton::Blank;       // restore button
      altv[altc++] = i;                 // remember alternative
    }
    if ( stopHuman >= 0 )               // must stop human from winning
      a[stopHuman] = TicTacButton::Cross;
    else if ( i == numButtons ) {       // tried all alternatives
      if ( altc > 0 )                   // set random piece
          a[altv[rand()%(altc--)]] = TicTacButton::Cross;
      if ( altc == 0 ) {                // no more blanks
          st = NobodyWon;
          emit finished();
      }
    }
    *btArray = a;                       // update model
    updateButtons();                    // update buttons
    delete[] altv;
}

//* TicTacToe member functions
// Creates a game widget with a game board and two push buttons, and connects
// signals of child widgets to slots.
TicTacToe::TicTacToe( int boardSize, QWidget *parent, const char *name )
    : QWidget( parent, name )
{
    QVBoxLayout * l = new QVBoxLayout( this, 6 );

    // Create a message label

    message = new QLabel( this );
    message->setFrameStyle( QFrame::WinPanel | QFrame::Sunken );
    message->setAlignment( AlignCenter );
    l->addWidget( message );

    // Create the game board and connect the signal finished() to this
    // gameOver() slot

    board = new TicTacGameBoard( boardSize, this );
    connect( board, SIGNAL(finished()), SLOT(gameOver()) );
    l->addWidget( board );

    // Create a horizontal frame line

    QFrame *line = new QFrame( this );
    line->setFrameStyle( QFrame::HLine | QFrame::Sunken );
    l->addWidget( line );

    // Create the combo box for deciding who should start, and
    // connect its clicked() signals to the buttonClicked() slot

    whoStarts = new QComboBox( this );
    whoStarts->insertItem( "Computer starts" );
    whoStarts->insertItem( "Human starts" );
    l->addWidget( whoStarts );

    // Create the push buttons and connect their clicked() signals
    // to this right slots.

    newGame = new QPushButton( "Play!", this );
    connect( newGame, SIGNAL(clicked()), SLOT(newGameClicked()) );
    quit = new QPushButton( "Quit", this );
    connect( quit, SIGNAL(clicked()), qApp, SLOT(quit()) );
    QHBoxLayout * b = new QHBoxLayout;
    l->addLayout( b );
    b->addWidget( newGame );
    b->addWidget( quit );

    newState();
}

// TicTacToe::newGameClicked()              - SLOT
// This slot is activated when the new game button is clicked.
void TicTacToe::newGameClicked()
{
    board->computerStarts( whoStarts->currentItem() == 0 );
    board->newGame();
    newState();
}

// TicTacToe::gameOver()               - SLOT
// This slot is activated when the TicTacGameBoard emits the signal
// "finished()", i.e. when a player has won or when it is a draw.
void TicTacToe::gameOver()
{
    newState();                        // update text box
}

// Updates the message to reflect a new state.
void TicTacToe::newState()
{
    static const char *msg[] = {        // TicTacGameBoard::State texts
      "Click Play to start", "Make your move",
      "You won!", "Computer won!", "It's a draw" };
    message->setText( msg[board->state()] );
    return;
}
```

tictac.h

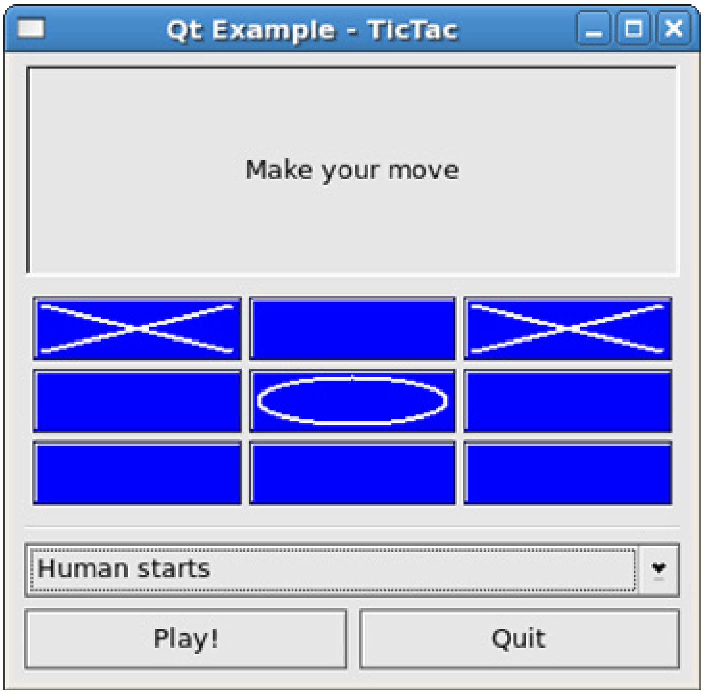
```
#ifndef TICTAC_H
#define TICTAC_H

#include <qpushbutton.h>
#include <qptrvector.h>

class QComboBox;
class QLabel;

// TicTacButton implements a single tic-tac-toe button
class TicTacButton : public QPushButton
{
    Q_OBJECT
public:
    TicTacButton( QWidget *parent );
    enum Type { Blank, Circle, Cross };
    Type     type() const          { return t; }
    voidsetType( Type type ) { t = type; repaint(); }
    QSizePolicy sizePolicy() const
    { return QSizePolicy( QSizePolicy::Preferred, QSizePolicy::Preferred ); }
    QSize sizeHint() const { return QSize( 32, 32 ); }
    QSize minimumSizeHint() const { return QSize( 10, 10 ); }
protected:
    voiddrawButtonLabel( QPainter * );
private:
    Type t;
};

// Using template vector to make vector-class of TicTacButton.
// This vector is used by the TicTacGameBoard class defined below.

typedef QPtrVector<TicTacButton>    TicTacButtons;
typedef QMemArray<int>         TicTacArray;

// TicTacGameBoard implements the tic-tac-toe game board.
// TicTacGameBoard is a composite widget that contains N x N TicTacButtons.
// N is specified in the constructor.
class TicTacGameBoard : public QWidget
{
    Q_OBJECT
public:
    TicTacGameBoard( int n, QWidget *parent=0, const char *name=0 );
   ~TicTacGameBoard();
    enum    State { Init, HumansTurn, HumanWon, ComputerWon, NobodyWon };
    State     state() const          { return st; }
    voidcomputerStarts( bool v );
    void        newGame();
signals:
    voidfinished();           // game finished
private slots:
    voidbuttonClicked();
private:
    void       setState( State state ) { st = state; }
    voidupdateButtons();
    int   checkBoard( TicTacArray * );
    voidcomputerMove();
    State     st;
    int        nBoard;
    boolcomp_starts;
    TicTacArray *btArray;
    TicTacButtons *buttons;
};

// TicTacToe implements the complete game.
// TicTacToe is a composite widget that contains a TicTacGameBoard and
// two push buttons for starting the game and quitting.
class TicTacToe : public QWidget
{
    Q_OBJECT
public:
    TicTacToe( int boardSize=3, QWidget *parent=0, const char *name=0 );
private slots:
    voidnewGameClicked();
    voidgameOver();
private:
    voidnewState();
    QComboBox   *whoStarts;
    QPushButton *newGame;
    QPushButton *quit;
    QLabel *message;
    TicTacGameBoard *board;
};

#endif // TICTAC_H
```

main.cpp

```
#include <qapplication.h>
#include <stdlib.h>
#include "tictac.h"

int main( int argc, char **argv )
{
   QApplication a( argc, argv );
   int n = 3;
   if ( argc == 2 )              // get board size n
      n = atoi(argv[1]);
   if ( n < 3 || n > 10 ) {          // out of range
      qWarning( "%s: Board size must be from 3x3 to 10x10", argv[0] );
      return 1;
   }
   TicTacToe ttt( n );              // create game
   a.setMainWidget( &ttt );
   ttt.setCaption("Qt Example - TicTac");
```

```
    ttt.show();                     // show widget
    return a.exec();                    // go
}
```

실행



68. 도구암시의 고급한 사용

이 실례창문부품은 창문부품안에 정적 및 동적령역을 위한 도구암시를 사용하는 방법을 보여준다. 이것은 두개의 청색과 하나의 적색 직4각형을 현시한다. 청색직4각형은 그것들을 찰칵할 때마다 이동하고 적색은 정적이다. 청색직4각형들우에 동적도구암시들이 있고 적색직4각형우에 정적도구암시가 있다.

tooltip.pro

```
TEMPLATE  = app
TARGET    = tooltip
CONFIG    += qt warn_on release
HEADERS       = tooltip.h
SOURCES       = main.cpp \
          tooltip.cpp
```

tooltip.cpp

```
#include "tooltip.h"
#include <qapplication.h>
#include <qpainter.h>
#include <stdlib.h>

DynamicTip::DynamicTip( QWidget * parent )
    : QToolTip( parent )
{
    // no explicit initialization needed
}

void DynamicTip::maybeTip( const QPoint &pos )
{
    if ( !parentWidget()->inherits( "TellMe" ) )
      return;

    QRect r( ((TellMe*)parentWidget())->tip(pos) );
    if ( !r.isValid() )
      return;

    QString s;
    s.sprintf( "position: %d,%d", r.center().x(), r.center().y() );
    tip( r, s );
}

TellMe::TellMe( QWidget * parent , const char * name  )
    : QWidget( parent, name )
{
    setMinimumSize( 30, 30 );
    r1 = randomRect();
    r2 = randomRect();
    r3 = randomRect();

    t = new DynamicTip( this );

    QToolTip::add( this, r3, "this color is called red" ); // <- helpful
}

TellMe::~TellMe()
{
    delete t;
    t = 0;
}

void TellMe::paintEvent( QPaintEvent * e )
{
    QPainter p( this );

    // I try to be efficient here, and repaint only what's needed

    if ( e->rect().intersects( r1 ) ) {
      p.setBrush( blue );
      p.drawRect( r1 );
    }

    if ( e->rect().intersects( r2 ) ) {
      p.setBrush( blue );
      p.drawRect( r2 );
    }

    if ( e->rect().intersects( r3 ) ) {
      p.setBrush( red );
```

```
     p.drawRect( r3 );
   }
}

void TellMe::mousePressEvent( QMouseEvent * e )
{
   if ( r1.contains( e->pos() ) )
     r1 = randomRect();
   if ( r2.contains( e->pos() ) )
     r2 = randomRect();
   repaint();
}

void TellMe::resizeEvent( QResizeEvent * )
{
   if ( !rect().contains( r1 ) )
      r1 = randomRect();
   if ( !rect().contains( r2 ) )
      r2 = randomRect();
}

QRect TellMe::randomRect()
{
   return QRect( ::rand() % (width() - 20), ::rand() % (height() - 20),
          20, 20 );
}

QRect TellMe::tip( const QPoint & p )
{
   if ( r1.contains( p ) )
     return r1;
   else if ( r2.contains( p ) )
     return r2;
   else
     return QRect( 0,0, -1,-1 );
}
```

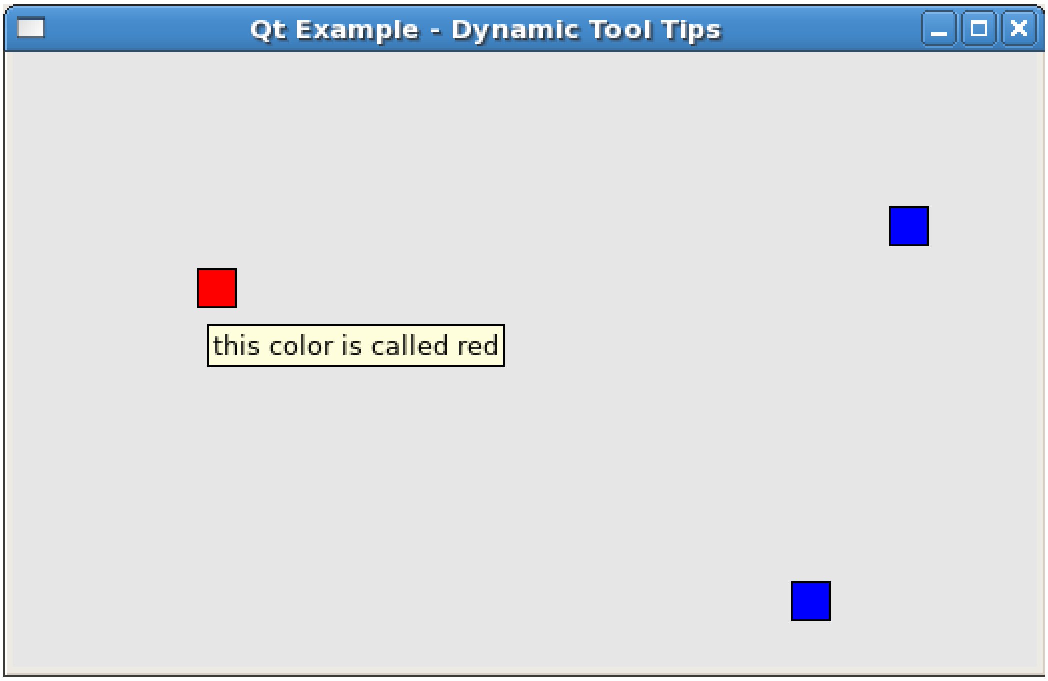
tooltip.h

```
#include <qwidget.h>
#include <qtooltip.h>

class DynamicTip : public QToolTip
{
public:
   DynamicTip( QWidget * parent );

protected:
   void maybeTip( const QPoint & );
};

class TellMe : public QWidget
{
   Q_OBJECT
public:
   TellMe( QWidget * parent = 0, const char * name = 0 );
   ~TellMe();

   QRect tip( const QPoint & );

protected:
   void paintEvent( QPaintEvent * );
   void mousePressEvent( QMouseEvent * );
   void resizeEvent( QResizeEvent * );

private:
   QRect randomRect();

   QRect r1, r2, r3;
   DynamicTip * t;
};
```

main.cpp

```
#include <qapplication.h>
#include "tooltip.h"

int main( int argc, char ** argv )
{
    QApplication a( argc, argv );

    TellMe mw;
    mw.setCaption( "Qt Example - Dynamic Tool Tips" );
    a.setMainWidget( &mw );
    mw.show();

    return a.exec();
}
```

실행



69. 최상위창문부품

이 실례는 전용창문장식을 가진 최상위창문부품들을 제공하기 위하여 Qt의 창문부품을 사용하는 방법을 보여준다.

이것은 창문부품장식(decoration) 혹은 동작을 위한 각이한 항목들을 선택하기 위한 도형방식 사용자대면부를 제공하며 적당한 창문부품기발들을 QWidget구성자에 넘긴다. QWidget::reparent()는 실행시에 창문부품기발들을 변경시키는데 쓰인다.

경고: 창문부품기발들의 해석과 기능은 응용프로그람을 실행할 때 사용된 창문관리기에 의존한다. 수많은 창문관리기는 매개의 가능한 기발결합을 유지하지 않는다.

각이한 선택들을 제공하는 사용자대면부는 Qt Designer에 의해 창조되였다. 각이한 선택들을 도구암시와 What's This방조의 사용을 통하여 사용자대면부에서 설명한다. 자세한 정보를 보려면 options.ui 파일을 Qt Designer에 적재하시오.

main함수는 사용자대면부용 대화칸을 창조하고 현시한다. 이 대화칸은 이행금지대화칸이다.

이 실례와 관련한 코드는 options.ui.h 파일에 있다.

apply()처리부는 창문부품기발변수 f를 선언하고 다음 값들로 초기화한다

- WDestructiveClose – 창문부품이 닫길 때 자동적으로 해체된다,
- WType_TopLevel – 창문부품이 어미창문부품을 가진다면 그것은 최상위준위이다,
- WStyle_Customize – 기발들은 기정값들을 무시한다.

사용자대면부에서 선택된 항목들에 따라 다른 기발들이 사용된다.

창문은 선택항목에 따라 표준 혹은 대화칸테두리선을 얻는다.

적당한 항목들이 검사되였으면 조종요소들을 가진 제목띠가 제공된다.

창문이 틀선을 가지지 말아야 한다면 제목띠를 가질수 없다. 자체의(실례로 주제) 창문장식을 제공하는 창문부품들은 이 기발을 사용해야 한다.

최상위창문부품이 어미를 가진다면 과제띠입구를 가지지 않으며 대부분의 창문관리기들에 대하여 항상 어미창문부품의 꼭대기에 상주한다. 이것은 대화칸들 특히 이행허용대화칸과 다른 2차최상위창문부품들의 표준동작이다.

과제띠입구를 제공하기 위하여 창문부품은 어미가 없어야 하며 그 경우에 WGroupLeader기발을 리용하여 이행금지기본대화칸을 통한 폐색을 방지하여야 한다. 동시에 열려진 여러 준위 창문들을 가질수 있는 응용프로그람들은 이 결합을 사용해야 한다.

최상위창문부품은 창문관리기가 이 기능을 지원한다면 전체탁상의 꼭대기에 머무를수 있다.(일부 X11창문관리기들도 역시 추가적으로 WX11BypassWM기발을 설정할것을 요구하지만 다른 X11창문관리기들은 이 기발이 설정되면 문제를 일으킨다.)

중요한 혹은 실제시간정보(즉 IRC의뢰기들)을 현시하는 창문부품들은 그 기발을 리용하면 덕을 볼수 있다.

튀여나오기창문부품은 자동적으로 닫기는 간단한 살아있는 이행금지창문부품이다. 튀여나오기차림표는 그러한 창문부품들의 전형적인 실례이다.

이행금지창문부품은 다른 최상위창문부품들이 각이한 이행금지그룹(WGroupLeader참고)안에 있지 않으면 그것들에 대한 입력을 폐색한다. 대화칸은 흔히 이행금지이고 QDialog클라스는 간단한 API를 제공하여 이 기발을 명시적으로 사용하지 않고 대화칸을 창조하여 현시한다.

도구창문은 (지어 어미창문부품을 가지지 않는다 해도) 과제띠입구를 가지지 않으며 흔히 더 작은 창문장식을 가진다. 도구창문은 흔히 이행허용대화칸대신에 사용된다.

창문부품이 아직 창조되지 않았거나 닫겨졌으면(WDestructiveClose기발을 사용하므로) 그 창문부품이 창조된다. 창문은 아직 볼수 없다.(실례는 QGuardedPtr를 사용하여 WDestructiveClose기발로 인하여 창문부품객체가 해체될 때 지적자가 0으로 재설정되는가 확인한다.)

창문부품이 이미 창조되였으면 reparent()함수가 창문부품의 기발들을 수정하는데 사용된다. 창문부품의 기하는 달라지지 않으며 창문은 다시 표시되지 않는다.

끝으로 창문제목과 그림기호과 같은 높은 준위속성들이 설정된다. 창문투명성은 미끄럼띠값에 따라 설정된다. 이것이 최상위창문용으로 이 특성을 유지하는 체계에만 영향을 가진다는것을 알아두시오.

끝으로 창문은 새 특성으로 표시된다.

실례를 구축하려면 등록부 QTDIR/examples/toplevel(QTDIR는 Qt가 설치되는 등록부)로 가서 qmake를 실행하여 makefile을 생성하고 make도구로 서고를 구축한다.

toplevel.pro

```
TEMPLATE = app
LANGUAGE = C++
CONFIG  += qt warn_on release
unix {
 UI_DIR = .ui
 MOC_DIR = .moc
 OBJECTS_DIR = .obj
}
SOURCES    += main.cpp
FORMS    = options.ui
```

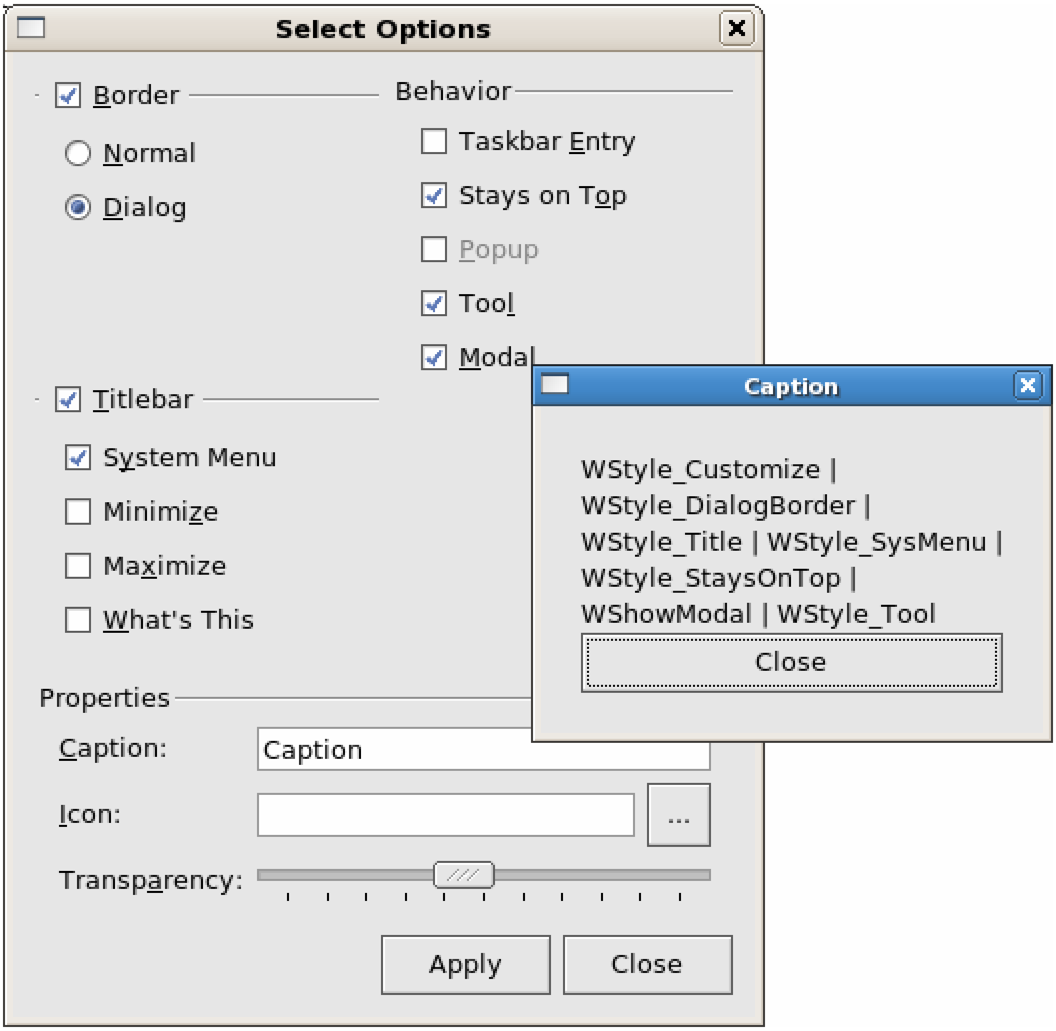
options.ui

```
<!DOCTYPE UI><UI version="3.3" stdsetdef="1">
<class>OptionsDialog</class>
<widget class="QDialog">
   <property name="name">
      <cstring>OptionsDialog</cstring>
   </property>
…
   <slot access="protected">pickIcon()</slot>
</slots>
<layoutdefaults spacing="6" margin="11"/>
</UI>
```

main.cpp

```
#include <qapplication.h>
#include "options.h"

int main( int argc, char ** argv )
{
   QApplication a( argc, argv );
   OptionsDialog dlg;
   return dlg.exec();
}
```

실행



☑ Border

○ Normal

● Dialog

Behavior

□ Taskbar Entry

☑ Stays on Top

□ Popup

☑ Tool

☑ Modal

☑ Titlebar

☑ System Menu

□ Minimize

□ Maximize

□ What's This

Properties

Caption: Caption

Icon:

Transparency:

WStyle_Customize |
WStyle_DialogBorder |
WStyle_Title | WStyle_SysMenu |
WStyle_StaysOnTop |
WShowModal | WStyle_Tool

70. Tux

tux.pro

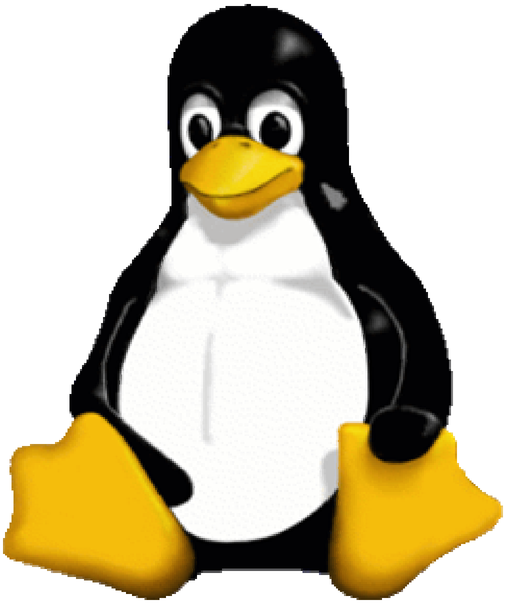
```
TEMPLATE  = app
TARGET      = tux
CONFIG       += qt warn_on release
```

```
HEADERS       =
SOURCES       = tux.cpp
INTERFACES    =
```

tux.cpp

```
#include <qapplication.h>
#include <qwidget.h>
#include <qimage.h>
#include <qpixmap.h>
#include <qbitmap.h>
#include <qfile.h>

#include <stdlib.h>

class MoveMe : public QWidget
{
public:
   MoveMe( QWidget *parent=0, const char *name=0, WFlags f = 0)
     :QWidget(parent,name, f) {}

protected:
   void mousePressEvent( QMouseEvent *);
   void mouseMoveEvent( QMouseEvent *);
private:
   QPoint clickPos;
};

void MoveMe::mousePressEvent( QMouseEvent *e )
{
   clickPos = e->pos();
}

void MoveMe::mouseMoveEvent( QMouseEvent *e )
{
   move( e->globalPos() - clickPos );
}

int main( int argc, char **argv )
{
   QApplication a( argc, argv );

   QString fn="tux.png";

   if ( argc >= 2 )
     fn = argv[1];

   if ( ! QFile::exists( fn ) )
     exit( 1 );

   QImage img( fn );
   QPixmap p;
   p.convertFromImage( img );
   if ( !p.mask() )
     if ( img.hasAlphaBuffer() ) {
       QBitmap bm;
       bm = img.createAlphaMask();
       p.setMask( bm );
     } else {
       QBitmap bm;
       bm = img.createHeuristicMask();
       p.setMask( bm );
     }
   MoveMe w(0,0,Qt::WStyle_Customize|Qt::WStyle_NoBorder);
   w.setBackgroundPixmap( p );
   w.setFixedSize( p.size() );
   if ( p.mask() )
     w.setMask( *p.mask() );
   w.show();
   a.setMainWidget(&w);

   return a.exec();
}
```

tux.png



71. 창문부품실례

이 실례는 동작하고있는 대부분의 Qt창문부품들을 보여준다. 이것은 $QTDIR/examples/demo안의 시위실례와 비슷하다. 프로그람을 실행하고 마우스의 오른쪽 단추+Ctrl을 눌러서 창문부품을 식별한다.

widgets.pro

```
TEMPLATE    = app
TARGET      = widgets
CONFIG      += qt warn_on release
INCLUDEPATH += ../aclock ../dclock
```

```
HEADERS         = widgets.h ../aclock/aclock.h ../dclock/dclock.h
SOURCES         = main.cpp widgets.cpp ../aclock/aclock.cpp ../dclock/dclock.cpp
```

widgets.cpp

```
#include <qmessagebox.h>
#include <qpixmap.h>
#include <qlayout.h>
#include <qapplication.h>

// Standard Qt widgets

#include <qtoolbar.h>
#include <qmenubar.h>
#include <qpopupmenu.h>
#include <qbuttongroup.h>
#include <qcheckbox.h>
#include <qcombobox.h>
#include <qframe.h>
#include <qgroupbox.h>
#include <qlabel.h>
#include <qlcdnumber.h>
#include <qmultilineedit.h>
#include <qlineedit.h>
#include <qlistbox.h>
#include <qpushbutton.h>
#include <qradiobutton.h>
#include <qslider.h>
#include <qtooltip.h>
#include <qspinbox.h>
#include <qstatusbar.h>
#include <qwhatsthis.h>
#include <qtoolbutton.h>
#include <qvbox.h>
#include <qtabbar.h>
#include <qtabwidget.h>
#include <qwidgetstack.h>
#include <qprogressbar.h>
#include <qsplitter.h>
#include <qlistview.h>
#include <qheader.h>
#include <qtextbrowser.h>
#include <qfiledialog.h>
#include <qaccel.h>
#include <qmetaobject.h>
#include <qpainter.h>
#include "widgets.h"

// Some sample widgets

#include "../aclock/aclock.h"
#include "../dclock/dclock.h"

#define MOVIEFILENAME "trolltech.gif"
```

```
#include "../application/fileopen.xpm"
#include "../application/filesave.xpm"
#include "../application/fileprint.xpm"

class MyWhatsThis : public QWhatsThis
{
public:
   MyWhatsThis( QListBox* lb)
    : QWhatsThis( lb ) { listbox = lb; };
   ~MyWhatsThis(){};

   QString text( const QPoint & p) {
    QListBoxItem* i = listbox->itemAt( p );
    if ( i && i->pixmap() ) {
        return "Isn't that a <em>wonderful</em> pixmap? <br>" \
          "Imagine, you could even decorate a" \
          " <b>red</b> pushbutton with it! :-)";
    }
    return "This is a QListBox.";
   }

private:
   QListBox* listbox;
};

class MyMenuItem : public QCustomMenuItem
{
public:
   MyMenuItem( const QString& s, const QFont& f )
    : string( s ), font( f ){};
   ~MyMenuItem(){}

   void paint( QPainter* p, const QColorGroup& /*cg*/, bool /*act*/,
         bool /*enabled*/, int x, int y, int w, int h )
   {
    p->setFont ( font );
    p->drawText( x, y, w, h,
              AlignAuto | AlignVCenter | ShowPrefix | DontClip,
              string );
   }

   QSize sizeHint()
   {
    return QFontMetrics( font ).size( AlignAuto | AlignVCenter |
                         ShowPrefix | DontClip,  string );
   }
private:
   QString string;
   QFont font;
};

// Construct the WidgetView with children

WidgetView::WidgetView( QWidget *parent, const char *name )
```

```
    : QMainWindow( parent, name )
{
    QColor col;

    // Set the window caption/title
    setCaption( "Qt Example - Widgets Demo Application" );

    // create a toolbar
    QToolBar *tools = new QToolBar( this, "toolbar" );

    // put something in it
    QPixmap openIcon( fileopen );
    QToolButton * toolb = new QToolButton( openIcon, "toolbutton 1",
                          QString::null, this, SLOT(open()),
                          tools, "open file" );
    QWhatsThis::add( toolb, "This is a <b>QToolButton</b>. It lives in a "
            "QToolBar. This particular button doesn't do anything "
            "useful." );

    QPixmap saveIcon( filesave );
    toolb = new QToolButton( saveIcon, "toolbutton 2", QString::null,
                this, SLOT(dummy()),
                tools, "save file" );
    QWhatsThis::add( toolb, "This is also a <b>QToolButton</b>." );

    QPixmap  printIcon( fileprint );
    toolb = new QToolButton( printIcon, "toolbutton 3", QString::null,
                this, SLOT(dummy()),
                tools, "print file" );
    QWhatsThis::add( toolb, "This is the third <b>QToolButton</b>.");

    toolb = QWhatsThis::whatsThisButton( tools );
    QWhatsThis::add( toolb, "This is a <b>What's This</b> button "
            "It enables the user to ask for help "
            "about widgets on the screen.");

    // Install an application-global event filter to catch control+leftbutton
    qApp->installEventFilter( this );

    //make a central widget to contain the other widgets
    central = new QWidget( this );
    setCentralWidget( central );

    // Create a layout to position the widgets
    QHBoxLayout *topLayout = new QHBoxLayout( central, 10 );

    // Create a grid layout to hold most of the widgets
    QGridLayout *grid = new QGridLayout( 0, 3 ); //3 wide and autodetect number of rows
    topLayout->addLayout( grid, 1 );

    // Create an easter egg
    QToolTip::add( menuBar(), QRect( 0, 0, 2, 2 ), "easter egg" );

    QPopupMenu* popup;
```

```
popup = new QPopupMenu( this );
menuBar()->insertItem( "&File", popup );
int id;
id = popup->insertItem( "&New" );
popup->setItemEnabled( id, FALSE );
id = popup->insertItem( openIcon, "&Open...", this, SLOT( open() ) );

popup->insertSeparator();
popup->insertItem( "Quit", qApp, SLOT(quit()), CTRL+Key_Q );

textStylePopup = popup = new QPopupMenu( this );
menuBar()->insertItem( "&Edit", popup );

plainStyleID = id = popup->insertItem( "&Plain" );
popup->setAccel( CTRL+Key_T, id );

popup->insertSeparator();
QFont f = font();
f.setBold( TRUE );
id = popup->insertItem( new MyMenuItem( "&Bold", f ) );
popup->setAccel( CTRL+Key_B, id );
f = font();
f.setItalic( TRUE );
id = popup->insertItem( new MyMenuItem( "&Italic", f ) );
popup->setItemChecked( id, TRUE );
popup->setAccel( CTRL+Key_I, id );
f = font();
f.setUnderline( TRUE );
id = popup->insertItem( new MyMenuItem( "&Underline", f ) );
popup->setAccel( CTRL+Key_U, id );
f = font();
f.setStrikeOut( TRUE );
id = popup->insertItem( new MyMenuItem( "&Strike", f ) );
connect( textStylePopup, SIGNAL(activated(int)),
     this, SLOT(popupSelected(int)) );

// Create an analog and a digital clock
AnalogClock  *aclock = new AnalogClock( central );
aclock->setAutoMask( TRUE );
DigitalClock *dclock = new DigitalClock( central );
dclock->setMaximumWidth(200);
grid->addWidget( aclock, 0, 2 );
grid->addWidget( dclock, 1, 2 );

// Give the dclock widget a blue palette
col.setRgb( 0xaa, 0xbe, 0xff );
dclock->setPalette( QPalette( col ) );

// make tool tips for both of them
QToolTip::add( aclock, "custom widget: analog clock" );
QToolTip::add( dclock, "custom widget: digital clock" );

// Create a push button.
QPushButton *pb;
```

```
pb = new QPushButton( "&Push button 1", central, "button1" );
grid->addWidget( pb, 0, 0, AlignVCenter );
connect( pb, SIGNAL(clicked()), SLOT(button1Clicked()) );
QToolTip::add( pb, "push button 1" );
QWhatsThis::add( pb, "This is a <b>QPushButton</b>.<br>"
          "Click it and watch...<br>"
          "The wonders of modern technology.");

QPixmap pm;
bool pix = pm.load("qt.png");
if ( !pix ) {
  QMessageBox::information( 0, "Qt Widgets Example",
                "Could not load the file \"qt.png\", which\n"
                "contains an icon used...\n\n"
                "The text \"line 42\" will be substituted.",
                QMessageBox::Ok + QMessageBox::Default );
}

// Create a label containing a QMovie
movie = QMovie( MOVIEFILENAME );
movielabel = new QLabel( central, "label0" );
movie.connectStatus(this, SLOT(movieStatus(int)));
movie.connectUpdate(this, SLOT(movieUpdate(const QRect&)));
movielabel->setFrameStyle( QFrame::Box | QFrame::Plain );
movielabel->setMovie( movie );
movielabel->setFixedSize( 128+movielabel->frameWidth()*2,
                64+movielabel->frameWidth()*2 );
grid->addWidget( movielabel, 0, 1, AlignCenter );
QToolTip::add( movielabel, "movie" );
QWhatsThis::add( movielabel, "This is a <b>QLabel</b> "
          "that contains a QMovie." );

// Create a group of check boxes
bg = new QButtonGroup( central, "checkGroup" );
bg->setTitle( "Check Boxes" );
grid->addWidget( bg, 1, 0 );

// Create a layout for the check boxes
QVBoxLayout *vbox = new QVBoxLayout(bg, 10);

vbox->addSpacing( bg->fontMetrics().height() );

cb[0] = new QCheckBox( bg );
cb[0]->setText( "&Read" );
vbox->addWidget( cb[0] );
cb[1] = new QCheckBox( bg );
cb[1]->setText( "&Write" );
vbox->addWidget( cb[1] );
cb[2] = new QCheckBox( bg );
cb[2]->setText( "&Execute" );
vbox->addWidget( cb[2] );

connect( bg, SIGNAL(clicked(int)), SLOT(checkBoxClicked(int)) );
```

```
QToolTip::add( cb[0], "check box 1" );
QToolTip::add( cb[1], "check box 2" );
QToolTip::add( cb[2], "check box 3" );

// Create a group of radio buttons
QRadioButton *rb;
bg = new QButtonGroup( central, "radioGroup" );
bg->setTitle( "Radio buttons" );

grid->addWidget( bg, 1, 1 );

// Create a layout for the radio buttons
vbox = new QVBoxLayout(bg, 10);

vbox->addSpacing( bg->fontMetrics().height() );
rb = new QRadioButton( bg );
rb->setText( "&AM" );
rb->setChecked( TRUE );
vbox->addWidget(rb);
QToolTip::add( rb, "radio button 1" );
rb = new QRadioButton( bg );
rb->setText( "F&M" );
vbox->addWidget(rb);
QToolTip::add( rb, "radio button 2" );
rb = new QRadioButton( bg );
rb->setText( "&Short Wave" );
vbox->addWidget(rb);

connect( bg, SIGNAL(clicked(int)), SLOT(radioButtonClicked(int)) );
QToolTip::add( rb, "radio button 3" );

// Create a list box
QListBox *lb = new QListBox( central, "listBox" );
for ( int i=0; i<100; i++ ) {          // fill list box
  QString str;
  str.sprintf( "line %d", i );
  if ( i == 42 && pix )
     lb->insertItem( pm );
  else
     lb->insertItem( str );
}
grid->addMultiCellWidget( lb, 2, 4, 0, 0 );
connect( lb, SIGNAL(selected(int)), SLOT(listBoxItemSelected(int)) );
QToolTip::add( lb, "list box" );
(void)new MyWhatsThis( lb );

vbox = new QVBoxLayout(8);
grid->addLayout( vbox, 2, 1 );

// Create a slider
QSlider *sb = new QSlider( 0, 300, 30, 100, QSlider::Horizontal,
                central, "Slider" );
sb->setTickmarks( QSlider::Below );
sb->setTickInterval( 10 );
```

```
sb->setFocusPolicy( QWidget::TabFocus );
vbox->addWidget( sb );

connect( sb, SIGNAL(valueChanged(int)), SLOT(sliderValueChanged(int)) );
QToolTip::add( sb, "slider" );
QWhatsThis::add( sb, "This is a <b>QSlider</b>. "
          "The tick marks are optional."
          " This slider controls the speed of the movie." );
// Create a combo box
QComboBox *combo = new QComboBox( FALSE, central, "comboBox" );
combo->insertItem( "darkBlue" );
combo->insertItem( "darkRed" );
combo->insertItem( "darkGreen" );
combo->insertItem( "blue" );
combo->insertItem( "red" );
vbox->addWidget( combo );
connect( combo, SIGNAL(activated(int)),
    this, SLOT(comboBoxItemActivated(int)) );
QToolTip::add( combo, "read-only combo box" );

// Create an editable combo box
QComboBox *edCombo = new QComboBox( TRUE, central, "edComboBox" );
QListBox *edComboLst = new QListBox(this);
edCombo->setListBox(edComboLst);
edComboLst->insertItem( "Permutable" );
edComboLst->insertItem( "Malleable" );
edComboLst->insertItem( "Adaptable" );
edComboLst->insertItem( "Alterable" );
edComboLst->insertItem( "Inconstant" );
vbox->addWidget( edCombo );
connect( edCombo, SIGNAL(activated(const QString&)),
    this, SLOT(edComboBoxItemActivated(const QString&)) );
QToolTip::add( edCombo, "editable combo box" );

edCombo->setAutoCompletion( TRUE );

vbox = new QVBoxLayout(8);
grid->addLayout( vbox, 2, 2 );

// Create a spin box
QSpinBox *spin = new QSpinBox( 0, 10, 1, central, "spin" );
spin->setSuffix(" mm");
spin->setSpecialValueText( "Auto" );
connect( spin, SIGNAL( valueChanged(const QString&) ),
    SLOT( spinBoxValueChanged(const QString&) ) );
QToolTip::add( spin, "spin box" );
QWhatsThis::add( spin, "This is a <b>QSpinBox</b>. "
          "You can chose values in a given range "
          "either by using the arrow buttons "
          "or by typing them in." );
vbox->addWidget( spin );

vbox->addStretch( 1 );
```

```
    // Create a tabwidget that switches between multi line edits
    tabs = new QTabWidget( central );
    //tabs->setTabPosition( QTabWidget::Bottom );
    tabs->setMargin( 4 );
    grid->addMultiCellWidget( tabs, 3, 3, 1, 2 );
    QMultiLineEdit *mle = new QMultiLineEdit( tabs, "multiLineEdit" );
    edit = mle;
    mle->setWordWrap( QMultiLineEdit::WidgetWidth );
    mle->setText("This is a QMultiLineEdit widget, "
            "useful for small multi-line "
          "input fields.");
    QToolTip::add( mle, "multi line editor" );

    tabs->addTab( mle, "F&irst");

    mle = new QMultiLineEdit( tabs, "multiLineEdit" );
    QString mleText = "This is another QMultiLineEdit widget.";
#if 1
    mleText += "\n";
    mleText += "Japanese: ";
    mleText += QChar((ushort)0x6a38); // Kanji
    mleText += "\n";
    mleText += "Russian: ";
    mleText += QChar((ushort)0x042e); // Cyrillic
    mleText += "\n";
    mleText += "Norwegian: ";
    mleText += QChar((ushort)0x00d8); // Norwegian
    mleText += "\n";
    mleText += "Unicode (black square): ";
    mleText += QChar((ushort)0x25A0); // BLACK SQUARE
    mleText += "\n";
#endif
    mle->setText( mleText );
    QToolTip::add( mle, "second multi line editor" );
    tabs->addTab( mle, "Se&cond");

    // Create a single line edit
    QLineEdit *le = new QLineEdit( central, "lineEdit" );


    grid->addMultiCellWidget( le, 4, 4, 1, 2 );
    connect( le, SIGNAL(textChanged(const QString&)),
         SLOT(lineEditTextChanged(const QString&)) );
    QToolTip::add( le, "single line editor" );
    QWhatsThis::add( le, "This is a <b>QLineEdit</b>, you can enter a "
             "single line of text in it. "
              "It also it accepts text drops." );

    grid->setRowStretch(0,0);
    grid->setRowStretch(1,0);
    grid->setRowStretch(2,0);
    grid->setRowStretch(3,1);
    grid->setRowStretch(4,0);
```

```
grid->setColStretch(0,1);
grid->setColStretch(1,1);
grid->setColStretch(2,1);

QSplitter *split = new QSplitter( Vertical, central, "splitter" );
split->setOpaqueResize( TRUE );
topLayout->addWidget( split, 1 );
QListView *lv = new MyListView( split );
connect(lv, SIGNAL(selectionChanged() ),
    this, SLOT( selectionChanged() ) );
connect(lv, SIGNAL(selectionChanged(QListViewItem*) ),
    this, SLOT( selectionChanged(QListViewItem*) ) );
connect(lv, SIGNAL(clicked(QListViewItem*) ),
    this, SLOT( clicked(QListViewItem*) ) );
connect(lv, SIGNAL(mySelectionChanged(QListViewItem*) ),
    this, SLOT( mySelectionChanged(QListViewItem*) ) );
lv->addColumn( "One" );
lv->addColumn( "Two" );
lv->setAllColumnsShowFocus( TRUE );

QListViewItem *lvi=  new QListViewItem( lv, "Text", "Text" );
lvi=  new QListViewItem( lv, "Text", "Other Text" );
lvi=  new QListViewItem( lv, "Text", "More Text" );
lvi=  new QListViewItem( lv, "Text", "Extra Text" );
lvi->setOpen(TRUE);
(void)new QListViewItem( lvi, "SubText", "Additional Text" );
lvi=  new QListViewItem( lvi, "SubText", "Side Text" );
lvi=  new QListViewItem( lvi, "SubSubText", "Complimentary Text" );

QToolTip::add( lv, "list view" );
QWhatsThis::add( lv, "This is a <b>QListView</b>, you can display lists "
        "(or outline lists) of multiple-column data in it." );

lv = new QListView( split );
lv->addColumn( "Choices" );
(void) new QCheckListItem( lv, "Onion", QCheckListItem::CheckBox );
(void) new QCheckListItem( lv, "Artichoke", QCheckListItem::CheckBox );
(void) new QCheckListItem( lv, "Pepper", QCheckListItem::CheckBox );
(void) new QCheckListItem( lv, "Habaneros", QCheckListItem::CheckBox );
(void) new QCheckListItem( lv, "Pineapple", QCheckListItem::CheckBox );
(void) new QCheckListItem( lv, "Ham", QCheckListItem::CheckBox );
(void) new QCheckListItem( lv, "Pepperoni", QCheckListItem::CheckBox );
(void) new QCheckListItem( lv, "Garlic", QCheckListItem::CheckBox );

QCheckListItem *lit = new QCheckListItem( lv, "Cheese" );
lit->setOpen( TRUE );
(void) new QCheckListItem( lit, "Cheddar", QCheckListItem::RadioButton );
(void) new QCheckListItem( lit, "Mozarella", QCheckListItem::RadioButton );
(void) new QCheckListItem( lit, "Jarlsberg", QCheckListItem::RadioButton );

QToolTip::add( lv, "list view" );
QWhatsThis::add( lv, "This is also a <b>QListView</b>, with "
        "interactive items." );
```

```
   QTextBrowser *browser =  new QTextBrowser( split );
   browser->setText( "<h1>QTextBrowser</h1>"
          "<p>Qt supports formatted rich text, such "
          "as the heading above, <em>emphasized</em> and "
          "<b>bold</b> text, via an XML subset.</p> "
          "<p><a href=\"nogo://some.where.com\">Hypertext navigation</a> and style sheets are
supported.</p>", "" );
   browser->setFont(QFont("Charter",11));
   browser->setFrameStyle( QFrame::WinPanel | QFrame::Sunken );
   connect( browser, SIGNAL(linkClicked(const QString&)), browser, SLOT(setText(const
QString&)) );

   // Create an label and a message in the status bar
   // The message is updated when buttons are clicked etc.
   msg = new QLabel( statusBar(), "message" );
   msg->setAlignment( AlignCenter );
   QFont boldfont; boldfont.setWeight(QFont::Bold);
   msg->setFont( boldfont );
   statusBar()->addWidget( msg, 4 );
   QToolTip::add( msg, "Message area" );

   QAccel* a = new QAccel( this );
   a->connectItem(  a->insertItem( Key_F9 ),
            this, SLOT( showProperties() ) );

   prog = new QProgressBar( statusBar(), "progress" );
   prog->setTotalSteps( 100 );
   progress = 64;
   prog->setProgress( progress );
   statusBar()->addWidget( prog , 1 );
   QWhatsThis::add( prog, "This is a <b>QProgressBar</b> "
            "You can use it to show that a lengthy "
            " process is progressing. "
            "In this program, nothing much seems to happen." );
   statusBar()->message( "Welcome to Qt", 2000 );
}

void WidgetView::setStatus(const QString& text)
{
   msg->setText(text);
}

void WidgetView::button1Clicked()
{
   msg->setText( "The push button was clicked" );
   prog->setProgress( ++progress );
}

void WidgetView::movieUpdate( const QRect& )
{
   // Uncomment this to test animated icons on your window manager
   //setIcon( movie.framePixmap() );
}
```

```
void WidgetView::movieStatus( int s )
{
    switch ( s ) {
     case QMovie::SourceEmpty:
     case QMovie::UnrecognizedFormat:
     {
         QPixmap pm("tt-logo.png");
         movielabel->setPixmap(pm);
         movielabel->setFixedSize(pm.size());
     }
     break;
     default:
     if ( movielabel->movie() )          // for flicker-free animation:
         movielabel->setBackgroundMode( NoBackground );
    }
}

void WidgetView::popupSelected( int selectedId )
{
    if ( selectedId == plainStyleID ) {
     for ( int i = 0; i < int(textStylePopup->count()); i++ ) {
         int id = textStylePopup->idAt( i );
         textStylePopup->setItemChecked( id, FALSE);
     }
    } else {
     textStylePopup->setItemChecked( selectedId, TRUE );
    }
}

void WidgetView::checkBoxClicked( int id )
{
    QString str;
    str = tr("Check box %1 clicked : ").arg(id);
    QString chk = "---";
    if ( cb[0]->isChecked() )
     chk[0] = 'r';
    if ( cb[1]->isChecked() )
     chk[1] = 'w';
    if ( cb[2]->isChecked() )
     chk[2] = 'x';
    str += chk;
    msg->setText( str );
}

void WidgetView::edComboBoxItemActivated( const QString& text)
{
    QString str = tr("Editable Combo Box set to ");
    str += text;
    msg->setText( str );
}

void WidgetView::radioButtonClicked( int id )
{
    msg->setText( tr("Radio button #%1 clicked").arg(id) );
}

void WidgetView::listBoxItemSelected( int index )
{
  msg->setText( tr("List box item %1 selected").arg(index) );
}

void WidgetView::sliderValueChanged( int value )
{
  msg->setText( tr("Movie set to %1% of normal speed").arg(value) );
  movie.setSpeed( value );
}

void WidgetView::comboBoxItemActivated( int index )
{
  msg->setText( tr("Combo box item %1 activated").arg(index) );
  switch ( index ) {
  default:
  case 0:
   QApplication::setWinStyleHighlightColor( darkBlue );
   break;
  case 1:
   QApplication::setWinStyleHighlightColor( darkRed );
   break;
  case 2:
   QApplication::setWinStyleHighlightColor( darkGreen );
   break;
  case 3:
   QApplication::setWinStyleHighlightColor( blue );
   break;
  case 4:
   QApplication::setWinStyleHighlightColor( red );
   break;
  }
}

void WidgetView::lineEditTextChanged( const QString& newText )
{
  QString str( "Line edit text: ");
  str += newText;
  if ( newText.length() == 1 ) {
   QString u;
   u.sprintf(" (U%02x%02x)", newText[0].row(), newText[0].cell() );
   str += u;
  }
  msg->setText( str );
}

void WidgetView::spinBoxValueChanged( const QString& valueText )
{
  QString str( "Spin box value: " );
  str += valueText;
  msg->setText( str );
}
```

```
// All application events are passed through this event filter.
// We're using it to display some information about a clicked
// widget (right mouse button + CTRL).

bool WidgetView::eventFilter( QObject *obj, QEvent *event )
{
   static bool identify_now = TRUE;
   if ( event->type() == QEvent::MouseButtonPress && identify_now ) {
     QMouseEvent *e = (QMouseEvent*)event;
     if ( e->button() == QMouseEvent::RightButton &&
          (e->state() & QMouseEvent::ControlButton) != 0 ){
         QString str = "The clicked widget is a\n";
         str += obj->className();
         str += "\nThe widget's name is\n";
         if ( obj->name() )
           str += obj->name();
         else
           str += "<no name>";
         identify_now = FALSE;        // don't do it in message box
         QMessageBox::information( (QWidget*)obj, "Identify Widget", str );
         identify_now = TRUE;         // allow it again
     }
   }
   return QMainWindow::eventFilter( obj, event ); // don't eat event
}

void WidgetView::open()
{
   QFileDialog::getOpenFileName( QString::null, "Textfiles (*.txt)", this );
}

void WidgetView::dummy()
{
   QMessageBox::information( this, "Sorry", "This function is not implemented" );
}

void WidgetView::selectionChanged()
{
   //qDebug("selectionChanged");
}
void WidgetView::selectionChanged( QListViewItem* /*item*/)
{
   //qDebug("selectionChanged %p", item );
}

void WidgetView::clicked( QListViewItem* /*item*/ )
{
   //qDebug("clicked %p", item );
}

void WidgetView::mySelectionChanged( QListViewItem* /*item*/ )
{
   //qDebug("mySelectionChanged %p", item );
}

void WidgetView::showProperties()
{
   if ( !qApp->focusWidget() )
     return;
   QCString output;
   output.sprintf( "Properties for class '%s'",
             qApp->focusWidget()->className() );
   int i = 0;
   while( i < (int) qApp->focusWidget()->metaObject()->numProperties( TRUE ) ) {
     const QMetaProperty* p
         = qApp->focusWidget()->metaObject()->property( i, TRUE );
     QCString tmp;
     tmp.sprintf( "\n %2d: %s (read-%s, %s)", ++i, p->name(),
             p->writable() ? "write" : "only", p->type() );
     output += tmp;
   }
   qDebug( output );
}
```

widgets.h

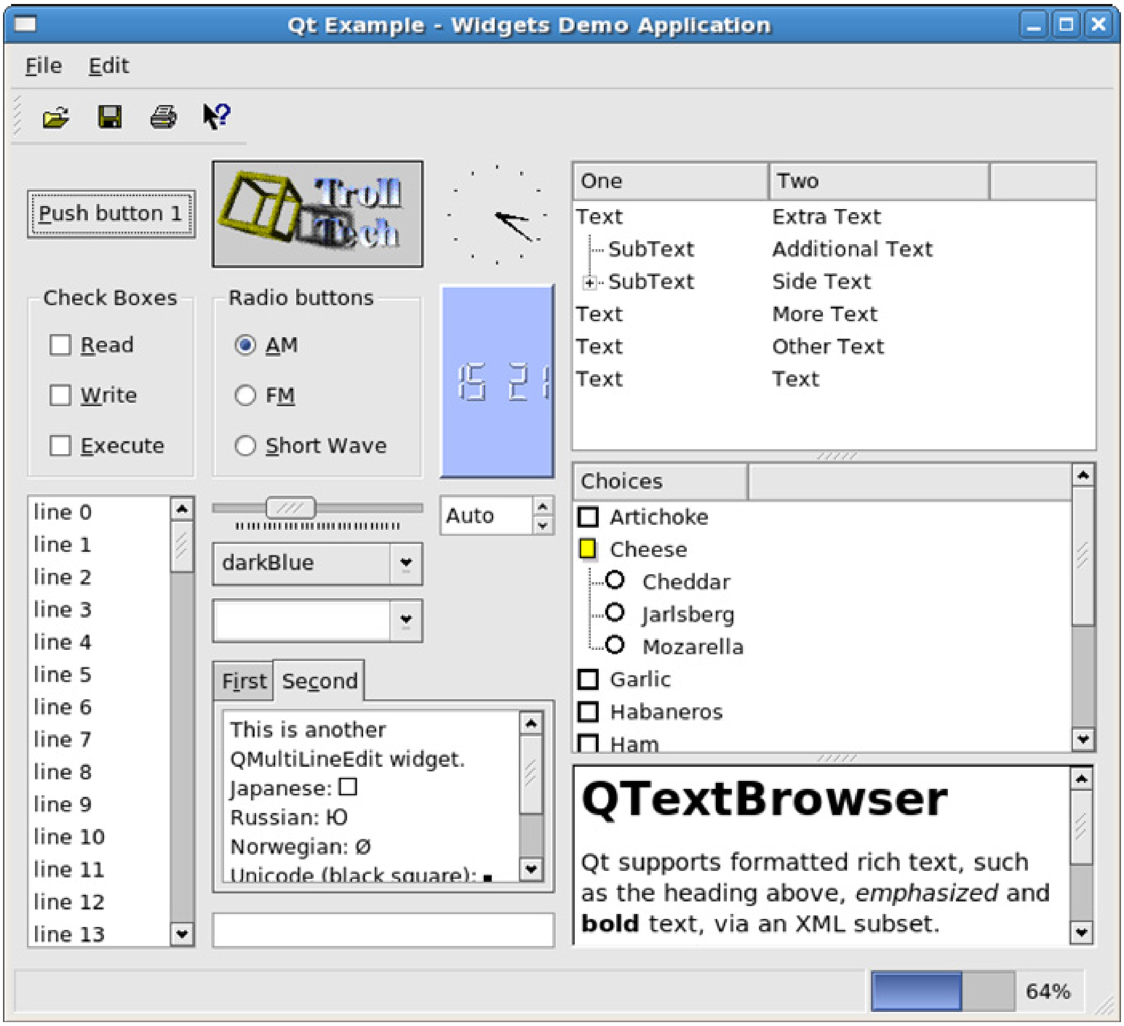
```
#ifndef WIDGETS_H
#define WIDGETS_H

#include <qmainwindow.h>
#include <qmovie.h>
#include <qlistview.h>
class QLabel;
class QCheckBox;
class QProgressBar;
class QTabWidget;
class QGroupBox;
class QMultiLineEdit;
class QPopupMenu;

class MyListView : public QListView
{
    Q_OBJECT
public:
    MyListView( QWidget * parent = 0, const char *name = 0 )
      : QListView( parent, name ), selected(0)
    {}
    ~MyListView()
    {}
protected:

    void contentsMousePressEvent( QMouseEvent * e )
    {
      selected = selectedItem();
      QListView::contentsMousePressEvent( e );
    }
    void contentsMouseReleaseEvent( QMouseEvent * e )
    {
```

```
   QListView::contentsMouseReleaseEvent( e );
   if ( selectedItem() != selected ) {
     emit mySelectionChanged( selectedItem() );
     emit mySelectionChanged();
   }
  }

signals:
  void mySelectionChanged();
  void mySelectionChanged( QListViewItem* );

private:
  QListViewItem* selected;

};

// WidgetView contains lots of Qt widgets.
class WidgetView : public QMainWindow
{
  Q_OBJECT
public:
  WidgetView( QWidget *parent=0, const char *name=0 );

public slots:
  voidsetStatus(const QString&);
  void selectionChanged();
  void selectionChanged( QListViewItem* );
  void clicked( QListViewItem* );
  void mySelectionChanged( QListViewItem* );

protected slots:
 virtual void button1Clicked();
private slots:
  voidcheckBoxClicked( int );
  voidradioButtonClicked( int );
  voidsliderValueChanged( int );
  voidlistBoxItemSelected( int );
  voidcomboBoxItemActivated( int );
  voidedComboBoxItemActivated( const QString& );
  voidlineEditTextChanged( const QString& );
  voidmovieStatus( int );
  voidmovieUpdate( const QRect& );
  voidspinBoxValueChanged( const QString& );
  voidpopupSelected( int );

  voidopen();
  voiddummy();
  voidshowProperties();

private:
  booleventFilter( QObject *, QEvent * );
  QLabel     *msg;
  QCheckBox  *cb[3];
  QGroupBox* bg;
```

```
    QLabel     *movielabel;
    QMovie      movie;
    QWidget *central;
    QProgressBar *prog;
    int progress;
    QTabWidget* tabs;
    QMultiLineEdit* edit;
    QPopupMenu *textStylePopup;
    int plainStyleID;
    QWidget* bla;
};

#endif
```

qt.png

실행



One	Two
Text	Extra Text
SubText	Additional Text
SubText	Side Text
Text	More Text
Text	Other Text
Text	Text

Choices
□ Artichoke
□ Cheese
○ Cheddar
○ Jarlsberg
○ Mozarella
□ Garlic
□ Habaneros
□ Ham

This is another
QMultiLineEdit widget.
Japanese: □
Russian: Ю
Norwegian: Ø
Unicode (black square): ■

QTextBrowser

Qt supports formatted rich text, such as the heading above, *emphasized* and **bold** text, via an XML subset.

72. 위자드

이 실례는 Qt의 위자드클라스의 사용법을 보여준다. 위자드는 복잡한 작용으로 사용자를 방조하는데 사용된다.

wizard.pro

```
TEMPLATE  = app
TARGET    = wizard
CONFIG    += qt warn_on release
HEADERS       = wizard.h
SOURCES       = main.cpp \
          wizard.cpp
```

wizard.cpp

```
#include "wizard.h"
#include <qwidget.h>
#include <qhbox.h>
#include <qvbox.h>
#include <qlabel.h>
#include <qlineedit.h>
#include <qpushbutton.h>
#include <qvalidator.h>
#include <qapplication.h>

Wizard::Wizard( QWidget *parent, const char *name )
   : QWizard( parent, name, TRUE )
{
   setupPage1();
   setupPage2();
   setupPage3();

   key->setFocus();
}

void Wizard::setupPage1()
{
   page1 = new QHBox( this );
   page1->setSpacing(8);

   QLabel *info = new QLabel( page1 );
   info->setMargin( 11 );
   info->setPalette( yellow );
   info->setText( "Enter your personal\n"
               "key here.\n\n"
               "Your personal key\n"
               "consists of 4 digits" );
   info->setMaximumWidth( info->sizeHint().width() );

   QVBox *page = new QVBox( page1 );

   QHBox *row1 = new QHBox( page );

   (void)new QLabel( "Key:", row1 );
```

```
    key = new QLineEdit( row1 );
    key->setMaxLength( 4 );
    key->setValidator( new QIntValidator( 1000, 9999, key ) );

    connect( key, SIGNAL( textChanged( const QString & ) ),
          this, SLOT( keyChanged( const QString & ) ) );

    addPage( page1, "Personal Key" );

    setNextEnabled( page1, FALSE );
    setHelpEnabled( page1, FALSE );
}

void Wizard::setupPage2()
{
    page2 = new QHBox( this );
    page2->setSpacing(8);

    QLabel *info = new QLabel( page2 );
    info->setMargin( 11 );
    info->setPalette( yellow );
    info->setText( "\n"
                "Enter your personal\n"
                "data here.\n\n"
                "The required fields are\n"
                "First Name, Last Name \n"
                "and E-Mail.\n" );
    info->setMaximumWidth( info->sizeHint().width() );

    QVBox *page = new QVBox( page2 );

    QHBox *row1 = new QHBox( page );
    QHBox *row2 = new QHBox( page );
    QHBox *row3 = new QHBox( page );
    QHBox *row4 = new QHBox( page );
    QHBox *row5 = new QHBox( page );

    QLabel *label1 = new QLabel( " First Name: ", row1 );
    label1->setAlignment( Qt::AlignVCenter );
    QLabel *label2 = new QLabel( " Last Name: ", row2 );
    label2->setAlignment( Qt::AlignVCenter );
    QLabel *label3 = new QLabel( " Address: ", row3 );
    label3->setAlignment( Qt::AlignVCenter );
    QLabel *label4 = new QLabel( " Phone Number: ", row4 );
    label4->setAlignment( Qt::AlignVCenter );
    QLabel *label5 = new QLabel( " E-Mail: ", row5 );
    label5->setAlignment( Qt::AlignVCenter );

    label1->setMinimumWidth( label4->sizeHint().width() );
    label2->setMinimumWidth( label4->sizeHint().width() );
    label3->setMinimumWidth( label4->sizeHint().width() );
    label4->setMinimumWidth( label4->sizeHint().width() );
    label5->setMinimumWidth( label4->sizeHint().width() );
    firstName = new QLineEdit( row1 );
    lastName = new QLineEdit( row2 );
    address = new QLineEdit( row3 );
    phone = new QLineEdit( row4 );
    email = new QLineEdit( row5 );

    connect( firstName, SIGNAL( textChanged( const QString & ) ),
          this, SLOT( dataChanged( const QString & ) ) );
    connect( lastName, SIGNAL( textChanged( const QString & ) ),
          this, SLOT( dataChanged( const QString & ) ) );
    connect( email, SIGNAL( textChanged( const QString & ) ),
          this, SLOT( dataChanged( const QString & ) ) );

    addPage( page2, "Personal Data" );

    setHelpEnabled( page2, FALSE );
}

void Wizard::setupPage3()
{
    page3 = new QHBox( this );
    page3->setSpacing(8);

    QLabel *info = new QLabel( page3 );
    info->setPalette( yellow );
    info->setText( "\n"
                "Look here to see of\n"
                "the data you entered\n"
                "is correct. To confirm,\n"
                "press the [Finish] button\n"
                "else go back to correct\n"
                "mistakes." );
    info->setMargin( 11 );
    info->setAlignment( AlignTop|AlignLeft );
    info->setMaximumWidth( info->sizeHint().width() );

    QVBox *page = new QVBox( page3 );

    QHBox *row1 = new QHBox( page );
    QHBox *row2 = new QHBox( page );
    QHBox *row3 = new QHBox( page );
    QHBox *row4 = new QHBox( page );
    QHBox *row5 = new QHBox( page );
    QHBox *row6 = new QHBox( page );

    QLabel *label1 = new QLabel( " Personal Key: ", row1 );
    label1->setAlignment( Qt::AlignVCenter );
    QLabel *label2 = new QLabel( " First Name: ", row2 );
    label2->setAlignment( Qt::AlignVCenter );
    QLabel *label3 = new QLabel( " Last Name: ", row3 );
    label3->setAlignment( Qt::AlignVCenter );
    QLabel *label4 = new QLabel( " Address: ", row4 );
    label4->setAlignment( Qt::AlignVCenter );
    QLabel *label5 = new QLabel( " Phone Number: ", row5 );
    label5->setAlignment( Qt::AlignVCenter );
    QLabel *label6 = new QLabel( " E-Mail: ", row6 );
    label6->setAlignment( Qt::AlignVCenter );

    label1->setMinimumWidth( label1->sizeHint().width() );
    label2->setMinimumWidth( label1->sizeHint().width() );
    label3->setMinimumWidth( label1->sizeHint().width() );
    label4->setMinimumWidth( label1->sizeHint().width() );
    label5->setMinimumWidth( label1->sizeHint().width() );
    label6->setMinimumWidth( label1->sizeHint().width() );

    lKey = new QLabel( row1 );
    lFirstName = new QLabel( row2 );
    lLastName = new QLabel( row3 );
    lAddress = new QLabel( row4 );
    lPhone = new QLabel( row5 );
    lEmail = new QLabel( row6 );

    addPage( page3, "Finish" );

    setFinishEnabled( page3, TRUE );
    setHelpEnabled( page3, FALSE );
}

void Wizard::showPage( QWidget* page )
{
    if ( page == page1 ) {
    } else if ( page == page2 ) {
    } else if ( page == page3 ) {
        lKey->setText( key->text() );
        lFirstName->setText( firstName->text() );
        lLastName->setText( lastName->text() );
        lAddress->setText( address->text() );
        lPhone->setText( phone->text() );
        lEmail->setText( email->text() );
    }

    QWizard::showPage(page);

    if ( page == page1 ) {
        keyChanged( key->text() );
        key->setFocus();
    } else if ( page == page2 ) {
        dataChanged( firstName->text() );
        firstName->setFocus();
    } else if ( page == page3 ) {
        finishButton()->setEnabled( TRUE );
        finishButton()->setFocus();
    }
}

void Wizard::keyChanged( const QString &text )
{
    QString t = text;
    int p = 0;
    bool on = ( key->validator()->validate(t, p) == QValidator::Acceptable );
    nextButton()->setEnabled( on );
}

void Wizard::dataChanged( const QString & )
{
    if ( !firstName->text().isEmpty() &&
         !lastName->text().isEmpty() &&
         !email->text().isEmpty() )
        nextButton()->setEnabled( TRUE );
    else
        nextButton()->setEnabled( FALSE );
}
```

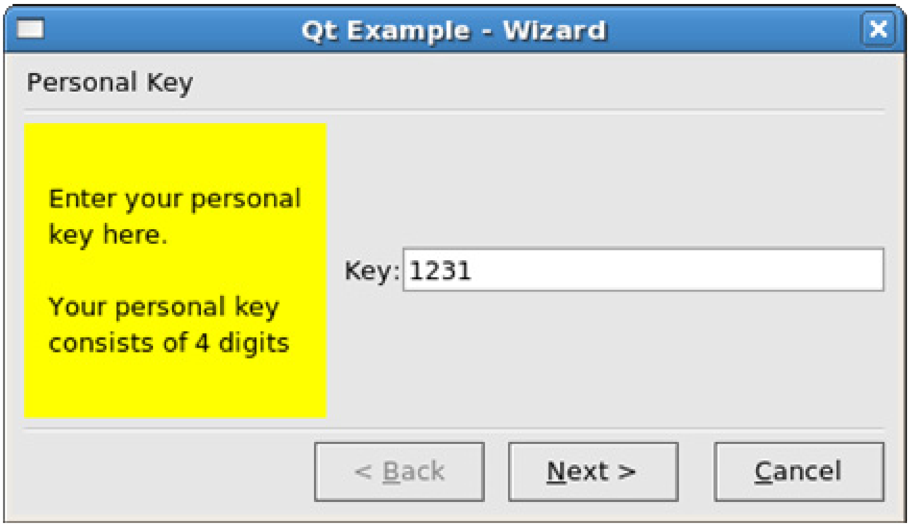
wizard.h

```
#ifndef WIZARD_H
#define WIZARD_H

#include <qwizard.h>

class QWidget;
class QHBox;
class QLineEdit;
class QLabel;

class Wizard : public QWizard
{
    Q_OBJECT

public:
    Wizard( QWidget *parent = 0, const char *name = 0 );

    void showPage(QWidget* page);

protected:
    void setupPage1();
    void setupPage2();
    void setupPage3();

    QHBox *page1, *page2, *page3;
    QLineEdit *key, *firstName, *lastName, *address, *phone, *email;
    QLabel *lKey, *lFirstName, *lLastName, *lAddress, *lPhone, *lEmail;

protected slots:
    void keyChanged( const QString & );
    void dataChanged( const QString & );

};

#endif
```

main.cpp

```
#include "wizard.h"
#include <qapplication.h>

int main(int argc,char **argv)
{
   QApplication a(argc,argv);

   Wizard wizard;
   wizard.setCaption("Qt Example - Wizard");
   return wizard.exec();
}
```

실행



Personal Key

Enter your personal key here.

Your personal key consists of 4 digits

Key: 1231

73. 도형변환프로그람

이 실례는 사용자가 본문과 그라픽스를 임의의 회전시키고 잘라내고 확대할수 있게 한다.

xform.pro

```
TEMPLATE  = app
TARGET     = xform
CONFIG     += qt warn_on release
HEADERS        =
SOURCES        = xform.cpp
```

xform.cpp

```
#include <qapplication.h>
#include <qdialog.h>
#include <qlabel.h>
#include <qlineedit.h>
#include <qpushbutton.h>
#include <qcheckbox.h>
#include <qradiobutton.h>
#include <qbuttongroup.h>
#include <qlcdnumber.h>
#include <qslider.h>
```

```
#include <qmenubar.h>
#include <qfontdialog.h>
#include <qlayout.h>
#include <qvbox.h>
#include <qwidgetstack.h>

#include <qpainter.h>
#include <qpixmap.h>
#include <qpicture.h>

#include <stdlib.h>

class ModeNames {
public:
    enum Mode { Text, Image, Picture };
};


class XFormControl : public QVBox, public ModeNames
{
    Q_OBJECT
public:
    XFormControl( const QFont &initialFont, QWidget *parent=0, const char *name=0 );
   ~XFormControl() {}

    QWMatrix matrix();

signals:
    void newMatrix( QWMatrix );
    void newText( const QString& );
    void newFont( const QFont & );
    void newMode( int );
private slots:
    void newMtx();
    void newTxt(const QString&);
    void selectFont();
    void fontSelected( const QFont & );
    void changeMode(int);
    void timerEvent(QTimerEvent*);
private:
    Mode mode;
    QSlider  *rotS;            // Rotation angle scroll bar
    QSlider  *shearS;          // Shear value scroll bar
    QSlider  *magS;                 // Magnification value scroll bar
    QLCDNumber  *rotLCD;        // Rotation angle LCD display
    QLCDNumber  *shearLCD;          // Shear value LCD display
    QLCDNumber  *magLCD;            // Magnification value LCD display
    QCheckBox      *mirror;         // Checkbox for mirror image on/of
    QWidgetStack* optionals;
    QLineEdit   *textEd;          // Inp[ut field for xForm text
    QPushButton  *fpb;                 // Select font push button
    QRadioButton *rb_txt;          // Radio button for text
    QRadioButton *rb_img;          // Radio button for image
    QRadioButton *rb_pic;          // Radio button for picture
    QFont currentFont;
};

/*
  ShowXForm displays a text or a pixmap (QPixmap) using a coordinate
  transformation matrix (QWMatrix)
*/

class ShowXForm : public QWidget, public ModeNames
{
    Q_OBJECT
public:
    ShowXForm( const QFont &f, QWidget *parent=0, const char *name=0 );
   ~ShowXForm() {}
    void showIt();            // (Re)displays text or pixmap

    Mode mode() const { return m; }
public slots:
    void setText( const QString& );
    void setMatrix( QWMatrix );
    void setFont( const QFont &f );
    void setPixmap( QPixmap );
    void setPicture( const QPicture& );
    void setMode( int );
private:
    QSizePolicy sizePolicy() const;
    QSize sizeHint() const;
    void paintEvent( QPaintEvent * );
    void resizeEvent( QResizeEvent * );
    QWMatrix  mtx;              // coordinate transform matrix
    QString   text;            // text to be displayed
    QPixmap   pix;             // pixmap to be displayed
    QPicture  picture;              // text to be displayed
    QRect     eraseRect;         // covers last displayed text/pixmap
    Mode      m;
};

XFormControl::XFormControl( const QFont &initialFont, QWidget *parent, const char *name )
      : QVBox( parent, name )
{
    setSpacing(6);
    setMargin(6);
    currentFont = initialFont;
    mode = Image;

    rotLCD = new QLCDNumber( 4, this, "rotateLCD" );
    rotS = new QSlider( QSlider::Horizontal, this,
                    "rotateSlider" );
    shearLCD  = new QLCDNumber( 5,this, "shearLCD" );
    shearS  = new QSlider( QSlider::Horizontal, this,
                    "shearSlider" );
    mirror  = new QCheckBox( this, "mirrorCheckBox" );
    rb_txt = new QRadioButton( this, "text" );
    rb_img = new QRadioButton( this, "image" );
```

```
rb_pic = new QRadioButton( this, "picture" );
optionals = new QWidgetStack(this);
QVBox* optionals_text = new QVBox(optionals);
optionals_text->setSpacing(6);
QVBox* optionals_other = new QVBox(optionals);
optionals_other->setSpacing(6);
optionals->addWidget(optionals_text,0);
optionals->addWidget(optionals_other,1);
fpb       = new QPushButton( optionals_text, "text" );
textEd  = new QLineEdit( optionals_text, "text" );
textEd->setFocus();

rotLCD->display( "  0'" );

rotS->setRange( -180, 180 );
rotS->setValue( 0 );
connect( rotS, SIGNAL(valueChanged(int)), SLOT(newMtx()) );

shearLCD->display( "0.00" );

shearS->setRange( -25, 25 );
shearS->setValue( 0 );
connect( shearS, SIGNAL(valueChanged(int)), SLOT(newMtx()) );

mirror->setText( tr("Mirror") );
connect( mirror, SIGNAL(clicked()), SLOT(newMtx()) );

QButtonGroup *bg = new QButtonGroup(this);
bg->hide();
bg->insert(rb_txt,0);
bg->insert(rb_img,1);
bg->insert(rb_pic,2);
rb_txt->setText( tr("Text") );
rb_img->setText( tr("Image") );
rb_img->setChecked(TRUE);
rb_pic->setText( tr("Picture") );
connect( bg, SIGNAL(clicked(int)), SLOT(changeMode(int)) );

fpb->setText( tr("Select font...") );
connect( fpb, SIGNAL(clicked()), SLOT(selectFont()) );

textEd->setText( "Troll" );
connect( textEd, SIGNAL(textChanged(const QString&)),
          SLOT(newTxt(const QString&)) );

magLCD = new QLCDNumber( 4,optionals_other, "magLCD" );
magLCD->display( "100" );
magS = new QSlider( QSlider::Horizontal, optionals_other,
               "magnifySlider" );
magS->setRange( 0, 800 );
connect( magS, SIGNAL(valueChanged(int)), SLOT(newMtx()) );
magS->setValue( 0 );
connect( magS, SIGNAL(valueChanged(int)), magLCD, SLOT(display(int)));
```

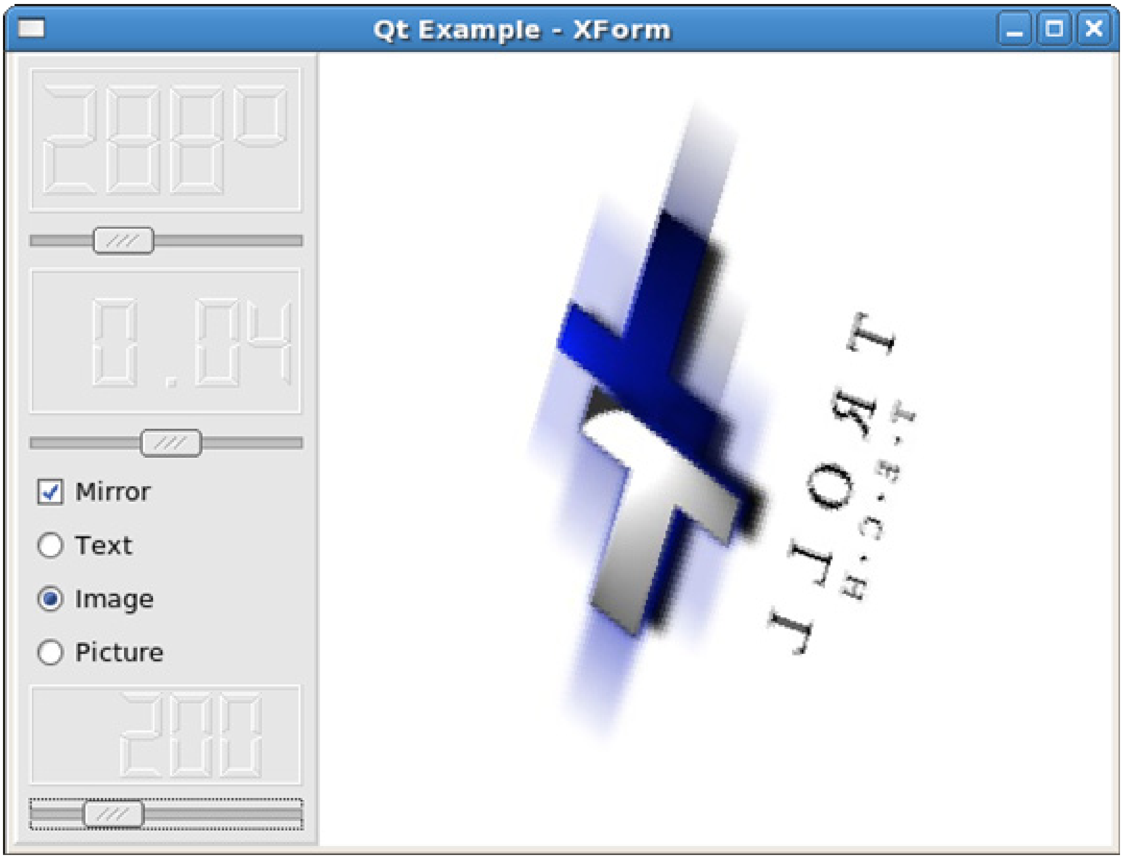
```
    optionals_text->adjustSize();
    optionals_other->adjustSize();
    changeMode(Image);

    startTimer(20); // start an initial animation
}

void XFormControl::timerEvent(QTimerEvent*)
{
    int v = magS->value();
    v = (v+2)+v/10;
    if ( v >= 200 ) {
      v = 200;
      killTimers();
    }
    magS->setValue(v);
}



/*
    Called whenever the user has changed one of the matrix parameters
    (i.e. rotate, shear or magnification)
*/
void XFormControl::newMtx()
{
    emit newMatrix( matrix() );
}

void XFormControl::newTxt(const QString& s)
{
    emit newText(s);
    changeMode(Text);
}

/*
    Calculates the matrix appropriate for the current controls, and updates the displays.
*/
QWMatrix XFormControl::matrix()
{
    QWMatrix m;
    if (mode != Text) {
      double magVal = 1.0*magS->value()/100;
      m.scale( magVal, magVal );
    }
    double shearVal = 1.0*shearS->value()/25;
    m.shear( shearVal, shearVal );
    m.rotate( rotS->value() );
    if ( mirror->isChecked() ) {
      m.scale( 1, -1 );
      m.rotate( 180 );
    }

    QString tmp;
    tmp.sprintf( "%1.2f", shearVal  );
    if ( shearVal >= 0 )
      tmp.insert( 0, " " );
    shearLCD->display( tmp );

    int rot = rotS->value();
    if ( rot < 0 )
      rot = rot + 360;
    tmp.sprintf( "%3i'", rot );
    rotLCD->display( tmp );
    return m;
}

void XFormControl::selectFont()
{
    bool ok;
    QFont f = QFontDialog::getFont( &ok, currentFont );
    if ( ok ) {
      currentFont = f;
      fontSelected( f );
    }
}

void XFormControl::fontSelected( const QFont &font )
{
    emit newFont( font );
    changeMode(Text);
}

/*
    Sets the mode - Text, Image, or Picture.
*/

void XFormControl::changeMode(int m)
{
    mode = (Mode)m;

    emit newMode( m );
    newMtx();
    if ( mode == Text ) {
      optionals->raiseWidget(0);
      rb_txt->setChecked(TRUE);
    } else {
      optionals->raiseWidget(1);
      if ( mode == Image )
          rb_img->setChecked(TRUE);
      else
          rb_pic->setChecked(TRUE);
    }
    qApp->flushX();
}

ShowXForm::ShowXForm( const QFont &initialFont, QWidget *parent, const char *name )
      : QWidget( parent, name, WResizeNoErase )
```

```
{
   setFont( initialFont );
   setBackgroundColor( white );
   m = Text;
   eraseRect = QRect( 0, 0, 0, 0 );
}

QSizePolicy ShowXForm::sizePolicy() const
{
   return QSizePolicy( QSizePolicy::Expanding, QSizePolicy::Expanding );
}

QSize ShowXForm::sizeHint() const
{
   return QSize(400,400);
}

void ShowXForm::paintEvent( QPaintEvent * )
{
   showIt();
}

void ShowXForm::resizeEvent( QResizeEvent * )
{
   eraseRect = QRect( width()/2, height()/2, 0, 0 );
   repaint(rect());
}

void ShowXForm::setText( const QString& s )
{
   text = s;
   showIt();
}

void ShowXForm::setMatrix( QWMatrix w )
{
   mtx = w;
   showIt();
}

void ShowXForm::setFont( const QFont &f )
{
   m = Text;
   QWidget::setFont( f );
}

void ShowXForm::setPixmap( QPixmap pm )
{
   pix   = pm;
   m    = Image;
   showIt();
}

void ShowXForm::setPicture( const QPicture& p )
```

```
{
   picture = p;
   m = Picture;
   showIt();
}

void ShowXForm::setMode( int mode )
{
   m = (Mode)mode;
}

void ShowXForm::showIt()
{
   QPainter p;
   QRect r;       // rectangle covering new text/pixmap in virtual coordinates
   QWMatrix um;  // copy user specified transform
   int textYPos = 0; // distance from boundingRect y pos to baseline
   int textXPos = 0; // distance from boundingRect x pos to text start
   QRect br;
   QFontMetrics fm( fontMetrics() );    // get widget font metrics
   switch ( mode() ) {
     case Text:
    br = fm.boundingRect( text );    // rectangle covering text
    r  = br;
    textYPos = -r.y();
    textXPos = -r.x();
    br.moveTopLeft( QPoint( -br.width()/2, -br.height()/2 ) );
      break;
     case Image:
    r = QRect(0, 0, pix.width()+1, pix.height()+1);
      break;
     case Picture:
    // ### need QPicture::boundingRect()
    r = QRect(0,0,1000,1000);
      break;
   }
   r.moveTopLeft( QPoint(-r.width()/2, -r.height()/2) );
   r.moveBy( -1, -1 ); // add border for matrix round off
   r.setSize( QSize( r.width() + 2,r.height() + 2 ) );
      // compute union of new and old rect
      // the resulting rectangle will cover what is already displayed
      // and have room for the new text/pixmap
   eraseRect = eraseRect.unite( mtx.mapRect(r) );
   int pw = QMIN(eraseRect.width(),width());
   int ph = QMIN(eraseRect.height(),height());
   QPixmap pm( pw, ph );          // off-screen drawing pixmap
   pm.fill( backgroundColor() );

   p.begin( &pm );
   um.translate( pw/2, ph/2 );    // 0,0 is center
   um = mtx * um;
   p.setWorldMatrix( um );
   switch ( mode() ) {
     case Text:
    p.setFont( font() );          // use widget font
    p.drawText( r.left() + textXPos, r.top() + textYPos, text );
#if 0
    p.setPen( red );
    p.drawRect( br );
#endif
    break;
   case Image:
    p.drawPixmap( -pix.width()/2, -pix.height()/2, pix );
    break;
    case Picture:
    // ### need QPicture::boundingRect()
    p.scale(0.25,0.25);
    p.translate(-230,-180);
    p.drawPicture( picture );
   }
   p.end();

   int xpos = width()/2  - pw/2;
   int ypos = height()/2 - ph/2;
   bitBlt( this, xpos, ypos,               // copy pixmap to widget
       &pm, 0, 0, -1, -1 );
   eraseRect =      mtx.map( r );
}

/*
   Grand unifying widget, putting ShowXForm and XFormControl
   together.
*/

class XFormCenter : public QHBox, public ModeNames
{
   Q_OBJECT
public:
   XFormCenter( QWidget *parent=0, const char *name=0 );
public slots:
   void setFont( const QFont &f ) { sx->setFont( f ); }
   void newMode( int );
private:
   ShowXForm    *sx;
   XFormControl *xc;
};

void XFormCenter::newMode( int m )
{
   static bool first_i = TRUE;
   static bool first_p = TRUE;

   if ( sx->mode() == m )
    return;
   if ( m == Image && first_i ) {
    first_i = FALSE;
    QPixmap pm;
    if ( pm.load( "image.any" ) )
```

```
          sx->setPixmap( pm );
      return;
    }
    if ( m == Picture && first_p ) {
      first_p = FALSE;
      QPicture p;
      if (p.load( "picture.any" ))
          sx->setPicture( p );
      return;
    }
    sx->setMode(m);
}

XFormCenter::XFormCenter( QWidget *parent, const char *name )
    : QHBox( parent, name )
{
    QFont f( "Charter", 36, QFont::Bold );

    xc = new XFormControl( f, this );
    sx = new ShowXForm( f, this );
    setStretchFactor(sx,1);
    xc->setFrameStyle( QFrame::Panel | QFrame::Raised );
    xc->setLineWidth( 2 );
    connect( xc, SIGNAL(newText(const QString&)), sx,
            SLOT(setText(const QString&)) );
    connect( xc, SIGNAL(newMatrix(QWMatrix)),
           sx, SLOT(setMatrix(QWMatrix)) );
    connect( xc, SIGNAL(newFont(const QFont&)), sx,
            SLOT(setFont(const QFont&)) );
    connect( xc, SIGNAL(newMode(int)), SLOT(newMode(int)) );
    sx->setText( "Troll" );
    newMode( Image );
    sx->setMatrix(xc->matrix());
}

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    XFormCenter *xfc = new XFormCenter;

    a.setMainWidget( xfc );
    xfc->setCaption("Qt Example - XForm");
    xfc->show();
    return a.exec();
}

#include "xform.moc"                  // include metadata generated by the moc
```

실행



74. XML

다음의 실례프로그람들은 Qt XML클라스사용법을 보여준다.

1) DOM 의 사용을 보여주는 개요프로그람

이 실례는 자그마한 개요프로그람을 제시하여 DOM클라스들의 기본사용법을 보여준다. 개요형식은 http://www.opml.org/spec에서 서술된것과 같은 OPML형식이다.

이 실례는 XML파일로부터 DOM나무를 적재하는 방법과 그 횡단방법을 보여준다.

outliner.pro

```
TEMPLATE  = app
TARGET    = outliner
CONFIG    += qt warn_on release
HEADERS      = outlinetree.h
SOURCES      = main.cpp \
        outlinetree.cpp
INTERFACES   =
```

outlinetree.cpp

```
#include "outlinetree.h"
#include <qfile.h>
#include <qmessagebox.h>

OutlineTree::OutlineTree( const QString fileName, QWidget *parent, const char *name )
```

```
   : QListView( parent, name )
{
   // div. configuration of the list view
   addColumn( "Outlines" );
   setSorting( -1 );
   setRootIsDecorated( TRUE );

   // read the XML file and create DOM tree
   QFile opmlFile( fileName );
   if ( !opmlFile.open( IO_ReadOnly ) ) {
    QMessageBox::critical( 0,
         tr( "Critical Error" ),
         tr( "Cannot open file %1" ).arg( fileName ) );
    return;
   }
   if ( !domTree.setContent( &opmlFile ) ) {
    QMessageBox::critical( 0,
         tr( "Critical Error" ),
         tr( "Parsing error for file %1" ).arg( fileName ) );
    opmlFile.close();
    return;
   }
   opmlFile.close();

   // get the header information from the DOM
   QDomElement root = domTree.documentElement();
   QDomNode node;
   node = root.firstChild();
   while ( !node.isNull() ) {
    if ( node.isElement() && node.nodeName() == "head" ) {
        QDomElement header = node.toElement();
        getHeaderInformation( header );
        break;
    }
    node = node.nextSibling();
   }
   // create the tree view out of the DOM
   node = root.firstChild();
   while ( !node.isNull() ) {
    if ( node.isElement() && node.nodeName() == "body" ) {
        QDomElement body = node.toElement();
        buildTree( 0, body );
        break;
    }
    node = node.nextSibling();
   }
}

OutlineTree::~OutlineTree()
{
}

void OutlineTree::getHeaderInformation( const QDomElement &header )
{
  // visit all children of the header element and look if you can make
  // something with it
  QDomNode node = header.firstChild();
  while ( !node.isNull() ) {
   if ( node.isElement() ) {
     // case for the different header entries
     if ( node.nodeName() == "title" ) {
      QDomText textChild = node.firstChild().toText();
      if ( !textChild.isNull() ) {
        setColumnText( 0, textChild.nodeValue() );
      }
     }
   }
   node = node.nextSibling();
  }
}

void OutlineTree::buildTree( QListViewItem *parentItem, const QDomElement &parentElement )
{
  QListViewItem *thisItem = 0;
  QDomNode node = parentElement.firstChild();
  while ( !node.isNull() ) {
   if ( node.isElement() && node.nodeName() == "outline" ) {
     // add a new list view item for the outline
     if ( parentItem == 0 )
      thisItem = new QListViewItem( this, thisItem );
     else
      thisItem = new QListViewItem( parentItem, thisItem );
     thisItem->setText( 0, node.toElement().attribute( "text" ) );
     // recursive build of the tree
     buildTree( thisItem, node.toElement() );
   }
   node = node.nextSibling();
  }
}
```

outlinetree.h

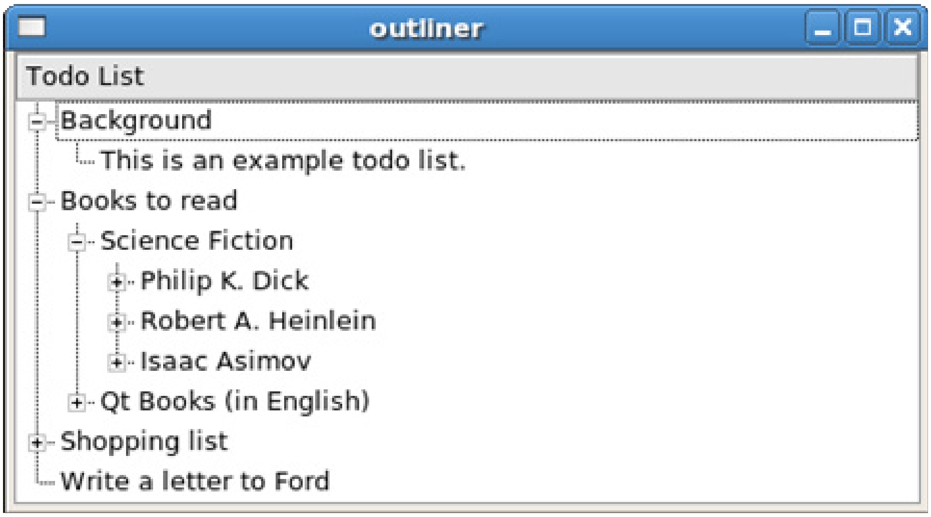
```
#ifndef OUTLINETREE_H
#define OUTLINETREE_H

#include <qlistview.h>
#include <qdom.h>

class OutlineTree : public QListView
{
  Q_OBJECT

public:
  OutlineTree( const QString fileName, QWidget *parent = 0, const char *name = 0 );
  ~OutlineTree();

private:
  QDomDocument domTree;
  void getHeaderInformation( const QDomElement &header );
```

```
    void buildTree( QListViewItem *parentItem, const QDomElement &parentElement );
};

#endif // OUTLINETREE_H
```

main.cpp

```
#include <qapplication.h>
#include "outlinetree.h"

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    OutlineTree outline( "todos.opml" );
    a.setMainWidget( &outline );
    outline.show();

    return a.exec();
}
```

실행



```
Todo List
Background
    This is an example todo list.
Books to read
    Science Fiction
        Philip K. Dick
        Robert A. Heinlein
        Isaac Asimov
    Qt Books (in English)
Shopping list
Write a letter to Ford
```

2) 간단한 SAX2 문법해석기

이 실레는 지령행에 XML문서안의 모든 요소들의 이름을 출력하는 간단한 SAX2읽기프로그람을 제공한다. 겹치는 요소이름들을 처리할수 있게 되여있다.

tagreader.pro

```
TEMPLATE  = app
TARGET       = tagreader
CONFIG     += qt console warn_on release
HEADERS        = structureparser.h
SOURCES        = tagreader.cpp \
              structureparser.cpp
INTERFACES    =
```

structureparser.cpp

```
#include "structureparser.h"
```

```
#include <stdio.h>
#include <qstring.h>

bool StructureParser::startDocument()
{
   indent = "";
   return TRUE;
}

bool StructureParser::startElement( const QString&, const QString&, const QString& qName,
                                    const QXmlAttributes& )
{
   printf( "%s%s\n", (const char*)indent, (const char*)qName );
   indent += "    ";
   return TRUE;
}

bool StructureParser::endElement( const QString&, const QString&, const QString& )
{
   indent.remove( (uint)0, 4 );
   return TRUE;
}
```

structureparser.h

```
#ifndef STRUCTUREPARSER_H
#define STRUCTUREPARSER_H

#include <qxml.h>

class QString;

class StructureParser : public QXmlDefaultHandler
{
public:
   bool startDocument();
   bool startElement( const QString&, const QString&, const QString& , const QXmlAttributes& );
   bool endElement( const QString&, const QString&, const QString& );

private:
   QString indent;
};

#endif
```

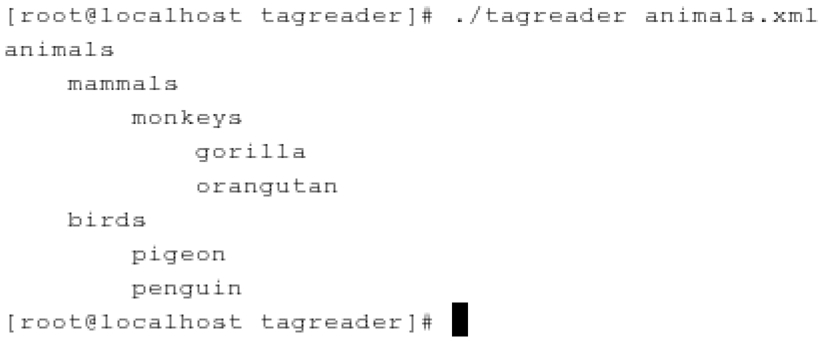
tagreader.cpp

```
#include "structureparser.h"
#include <qfile.h>
#include <qxml.h>
#include <qwindowdefs.h>

int main( int argc, char **argv )
{
   if ( argc < 2 ) {
     fprintf( stderr, "Usage: %s <xmlfile> [<xmlfile> ...]\n", argv[0] );
     return 1;
   }
   StructureParser handler;
   QXmlSimpleReader reader;
   reader.setContentHandler( &handler );
   for ( int i=1; i < argc; i++ ) {
      QFile xmlFile( argv[i] );
      QXmlInputSource source( &xmlFile );
      reader.parse( source );
   }
   return 0;
}
```

animals.xml

```
<animals>
<mammals>
  <monkeys> <gorilla/> <orangutan/> </monkeys>
</mammals>
<birds> <pigeon/> <penguin/> </birds>
</animals>
```

실행



```
[root@localhost tagreader]# ./tagreader animals.xml
animals
    mammals
        monkeys
            gorilla
            orangutan
    birds
        pigeon
        penguin
[root@localhost tagreader]# █
```

3) SAX2 기능의 시위

이 실례는 XML파일안의 수식된 이름들과 모든 요소들과 특성들의 개개의 이름공간URI들을 출력하는 작은 SAX2읽기프로그람을 제공한다. 또한 문서의 나무구조가 현시된다.

3개의 목록보기에서 SAX2 기능들인 *http://xml.org/sax/features/namespaces*와 *http://xml.org/sax/features/namespace-prefixes* 를 어떻게 설정하는가에 따라 읽기프로그람의 각이한 출력을 보여준다.

tagreader-with-features.pro

```
TEMPLATE  = app
TARGET         = tagreader-with-features
CONFIG       += qt warn_on release
HEADERS        = structureparser.h
SOURCES        = tagreader.cpp \
              structureparser.cpp
INTERFACES    =
```

structureparser.cpp

```
#include "structureparser.h"
#include <qstring.h>
```

```
#include <qlistview.h>

StructureParser::StructureParser( QListView * t )
               : QXmlDefaultHandler()
{
    setListView( t );
}

void StructureParser::setListView( QListView * t )
{
    table = t;
    table->setSorting( -1 );
    table->addColumn( "Qualified name" );
    table->addColumn( "Namespace" );
}

bool StructureParser::startElement( const QString& namespaceURI,
                                    const QString& ,
                                    const QString& qName,
                                    const QXmlAttributes& attributes)
{
    QListViewItem * element;

    if ( ! stack.isEmpty() ){
     QListViewItem *lastChild = stack.top()->firstChild();
     if ( lastChild ) {
         while ( lastChild->nextSibling() )
           lastChild = lastChild->nextSibling();
     }
     element = new QListViewItem( stack.top(), lastChild, qName, namespaceURI );
    } else {
     element = new QListViewItem( table, qName, namespaceURI );
    }
    stack.push( element );
    element->setOpen( TRUE );

    if ( attributes.length() > 0 ) {
     for ( int i = 0 ; i < attributes.length(); i++ ) {
         new QListViewItem( element, attributes.qName(i), attributes.uri(i) );
     }
    }
    return TRUE;
}

bool StructureParser::endElement( const QString&, const QString&,
                                  const QString& )
{
    stack.pop();
    return TRUE;
}
```

structureparser.h

```
#ifndef STRUCTUREPARSER_H
#define STRUCTUREPARSER_H
```

```
#include <qxml.h>
#include <qptrstack.h>

class QListView;
class QListViewItem;
class QString;

class StructureParser: public QXmlDefaultHandler
{
public:
   StructureParser( QListView * );
   bool startElement( const QString&, const QString&, const QString& ,
                  const QXmlAttributes& );
   bool endElement( const QString&, const QString&, const QString& );

   void setListView( QListView * );

private:
   QPtrStack<QListViewItem> stack;
   QListView * table;
};

#endif
```

tagreader.cpp

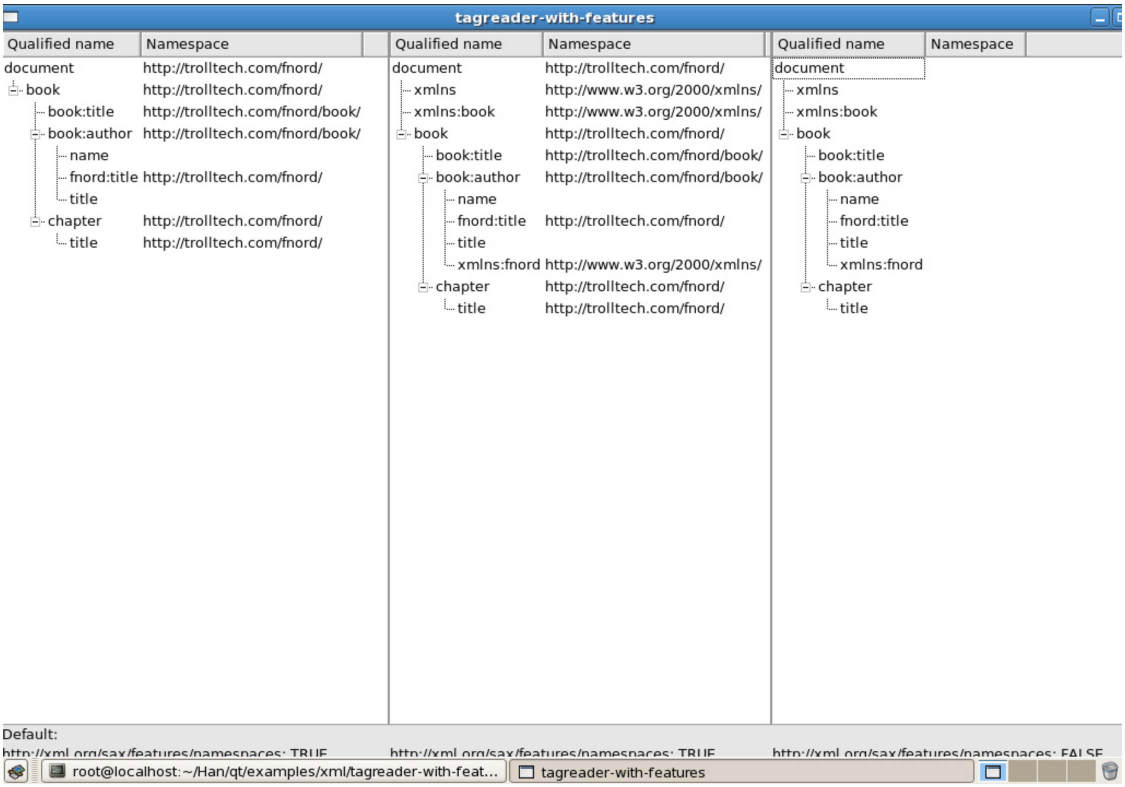
```
#include "structureparser.h"
#include <qapplication.h>
#include <qfile.h>
#include <qxml.h>
#include <qlistview.h>
#include <qgrid.h>
#include <qmainwindow.h>
#include <qlabel.h>

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

   QFile xmlFile( argc == 2 ? argv[1] : "fnord.xml" );
   QXmlInputSource source( &xmlFile );

   QXmlSimpleReader reader;

   QGrid * container = new QGrid( 3 );

   QListView * nameSpace = new QListView( container, "table_namespace" );
   StructureParser * handler = new StructureParser( nameSpace );
   reader.setContentHandler( handler );
   reader.parse( source );

   QListView * namespacePrefix = new QListView( container,
                                   "table_namespace_prefix" );
   handler->setListView( namespacePrefix );
```

```
    reader.setFeature( "http://xml.org/sax/features/namespace-prefixes",
                       TRUE );
    source.reset();
    reader.parse( source );

    QListView * prefix = new QListView( container, "table_prefix");
    handler->setListView( prefix );
    reader.setFeature( "http://xml.org/sax/features/namespaces", FALSE );
    source.reset();
    reader.parse( source );

    // namespace label
    (void) new QLabel(
            "Default:\n"
            "http://xml.org/sax/features/namespaces: TRUE\n"
            "http://xml.org/sax/features/namespace-prefixes: FALSE\n",
            container );

    // namespace prefix label
    (void) new QLabel(
            "\n"
            "http://xml.org/sax/features/namespaces: TRUE\n"
            "http://xml.org/sax/features/namespace-prefixes: TRUE\n",
            container );

    // prefix label
    (void) new QLabel(
            "\n"
            "http://xml.org/sax/features/namespaces: FALSE\n"
            "http://xml.org/sax/features/namespace-prefixes: TRUE\n",
            container );

    app.setMainWidget( container );
    container->show();
    return app.exec();
}
```

fnord.xml

```
<document xmlns:book = 'http://trolltech.com/fnord/book/'
          xmlns      = 'http://trolltech.com/fnord/' >
<book>
  <book:title>Practical XML</book:title>
  <book:author xmlns:fnord = 'http://trolltech.com/fnord/'
               title="Ms"
               fnord:title="Goddess"
               name="Eris Kallisti"/>
  <chapter>
    <title>A Namespace Called fnord</title>
  </chapter>
</book>
</document>
```

실행



Qualified name	Namespace
document	http://trolltech.com/fnord/
book	http://trolltech.com/fnord/
book:title	http://trolltech.com/fnord/book/
book:author	http://trolltech.com/fnord/book/
name	
fnord:title	http://trolltech.com/fnord/
title	
chapter	http://trolltech.com/fnord/
title	http://trolltech.com/fnord/

Qualified name	Namespace
document	http://trolltech.com/fnord/
xmlns	http://www.w3.org/2000/xmlns/
xmlns:book	http://www.w3.org/2000/xmlns/
book	http://trolltech.com/fnord/
book:title	http://trolltech.com/fnord/book/
book:author	http://trolltech.com/fnord/book/
name	
fnord:title	http://trolltech.com/fnord/
title	
xmlns:fnord	http://www.w3.org/2000/xmlns/
chapter	http://trolltech.com/fnord/
title	http://trolltech.com/fnord/

Qualified name	Namespace
document	
xmlns	
xmlns:book	
book	
book:title	
book:author	
name	
fnord:title	
title	
xmlns:fnord	
chapter	
title	

Default:
http://xml.org/sax/features/namespaces: TRUE
http://xml.org/sax/features/namespaces: TRUE
http://xml.org/sax/features/namespaces: FALSE

75. 끌기와 놓기(2)

이것은 Qt 의 끌기 및 놓기기능의 아주 간단한 실례를 제공한다.

simple_dd.pro

```
TEMPLATE = app
CONFIG  += qt warn_on release
HEADERS      = main.h
SOURCES      = main.cpp
```

main.h

```
#include <qapplication.h>
#include <qcursor.h>
#include <qsplitter.h>
#include <qlistbox.h>
#include <qiconview.h>
#include <qpixmap.h>

class QDragEnterEvent;
class QDragDropEvent;

class DDListBox : public QListBox
{
   Q_OBJECT
public:
   DDListBox( QWidget * parent = 0, const char * name = 0, WFlags f = 0 );
   // Low-level drag and drop
   void dragEnterEvent( QDragEnterEvent *evt );
```

```
    void dropEvent( QDropEvent *evt );
    void mousePressEvent( QMouseEvent *evt );
    void mouseMoveEvent( QMouseEvent * );
private:
    int dragging;
};


class DDIconViewItem : public QIconViewItem
{
public:
    DDIconViewItem( QIconView *parent, const QString& text, const QPixmap& icon ) :
      QIconViewItem( parent, text, icon ) {}
    DDIconViewItem( QIconView *parent, const QString &text ) :
      QIconViewItem( parent, text ) {}
    // High-level drag and drop
    bool acceptDrop( const QMimeSource *mime ) const;
    void dropped( QDropEvent *evt, const QValueList<QIconDragItem>& );
};


class DDIconView : public QIconView
{
    Q_OBJECT
public:
    DDIconView( QWidget * parent = 0, const char * name = 0, WFlags f = 0 ) :
      QIconView( parent, name, f ) {}
    // High-level drag and drop
    QDragObject *dragObject();
public slots:
    void slotNewItem( QDropEvent *evt, const QValueList<QIconDragItem>& list );
};
```

main.cpp

```
#include "main.h"
const char* red_icon[]={
"16 16 2 1",
"r c red",
". c None",
"................",
"................",
"..rrrrrrrrrrrr..",
"..rrrrrrrrrrrr..",
"..rrrrrrrrrrrr..",
"..rrr......rrr..",
"..rrr......rrr..",
"..rrr......rrr..",
"..rrr......rrr..",
"..rrr......rrr..",
"..rrr......rrr..",
"..rrrrrrrrrrrr..",
"..rrrrrrrrrrrr..",
"..rrrrrrrrrrrr..",
"................",
"................"};

const char* blue_icon[]={
"16 16 2 1",
"b c blue",
". c None",
"................",
"................",
"..bbbbbbbbbbbb..",
"..bbbbbbbbbbbb..",
"..bbbbbbbbbbbb..",
"..bbb......bbb..",
"..bbb......bbb..",
"..bbb......bbb..",
"..bbb......bbb..",
"..bbb......bbb..",
"..bbb......bbb..",
"..bbbbbbbbbbbb..",
"..bbbbbbbbbbbb..",
"..bbbbbbbbbbbb..",
"................",
"................"};

const char* green_icon[]={
"16 16 2 1",
"g c green",
". c None",
"................",
"................",
"..gggggggggggg..",
"..gggggggggggg..",
"..gggggggggggg..",
"..ggg......ggg..",
"..ggg......ggg..",
"..ggg......ggg..",
"..ggg......ggg..",
"..ggg......ggg..",
"..ggg......ggg..",
"..gggggggggggg..",
"..gggggggggggg..",
"..gggggggggggg..",
"................",
"................"};

// ListBox -- low level drag and drop

DDListBox::DDListBox( QWidget * parent, const char * name, WFlags f ) :
   QListBox( parent, name, f )
{
   setAcceptDrops( TRUE );
   dragging = FALSE;
}

void DDListBox::dragEnterEvent( QDragEnterEvent *evt )
```

```
{
   if ( QTextDrag::canDecode( evt ) )
    evt->accept();
}

void DDListBox::dropEvent( QDropEvent *evt )
{
   QString text;

   if ( QTextDrag::decode( evt, text ) )
    insertItem( text );
}

void DDListBox::mousePressEvent( QMouseEvent *evt )
{
   QListBox::mousePressEvent( evt );
   dragging = TRUE;
}

void DDListBox::mouseMoveEvent( QMouseEvent * )
{
   if ( dragging ) {
    QDragObject *d = new QTextDrag( currentText(), this );
    d->dragCopy(); // do NOT delete d.
    dragging = FALSE;
   }
}

// IconViewIcon -- high level drag and drop

bool DDIconViewItem::acceptDrop( const QMimeSource *mime ) const
{
   if ( mime->provides( "text/plain" ) )
    return TRUE;
   return FALSE;
}

void DDIconViewItem::dropped( QDropEvent *evt, const QValueList<QIconDragItem>& )
{
   QString label;

   if ( QTextDrag::decode( evt, label ) )
    setText( label );
}

// IconView -- high level drag and drop

QDragObject *DDIconView::dragObject()
{
 return new QTextDrag( currentItem()->text(), this );
}

void DDIconView::slotNewItem( QDropEvent *evt, const QValueList<QIconDragItem>& )
{
   QString label;

   if ( QTextDrag::decode( evt, label ) ) {
    DDIconViewItem *item = new DDIconViewItem( this, label );
    item->setRenameEnabled( TRUE );
   }
}

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

   // Create and show the widgets
   QSplitter *split = new QSplitter();
   DDIconView *iv   = new DDIconView( split );
   (void)           new DDListBox( split );
   app.setMainWidget( split );
   split->resize( 600, 400 );
   split->show();

   // Set up the connection so that we can drop items into the icon view
   QObject::connect(
    iv, SIGNAL(dropped(QDropEvent*, const QValueList<QIconDragItem>&)),
    iv, SLOT(slotNewItem(QDropEvent*, const QValueList<QIconDragItem>&)));

   // Populate the QIconView with icons
   DDIconViewItem *item;
   item = new DDIconViewItem( iv, "Red",   QPixmap( red_icon ) );
   item->setRenameEnabled( TRUE );
   item = new DDIconViewItem( iv, "Green", QPixmap( green_icon ) );
   item->setRenameEnabled( TRUE );
   item = new DDIconViewItem( iv, "Blue",  QPixmap( blue_icon ) );
   item->setRenameEnabled( TRUE );

   return app.exec();
}
```

76. Qt 의 간단한 실례

demo/demo.pro

```
TEMPLATE  = app
TARGET      = demo
CONFIG      += qt warn_off release
unix:LIBS+= -lm
DEFINES     += QT_INTERNAL_ICONVIEW
DEFINES     += QT_INTERNAL_WORKSPACE
DEFINES     += QT_INTERNAL_CANVAS
INCLUDEPATH+= .

HEADERS         = frame.h \
            categoryinterface.h \
            qthumbwheel.h \
                display.h \
            textdrawing/textedit.h \
            textdrawing/helpwindow.h \
            dnd/dnd.h \
            dnd/styledbutton.h \
            dnd/iconview.h \
            dnd/listview.h \
            i18n/i18n.h \
            i18n/wrapper.h \
            ../aclock/aclock.h
SOURCES           = frame.cpp \
            qthumbwheel.cpp \
                 display.cpp \
            textdrawing/textedit.cpp \
            textdrawing/helpwindow.cpp \
            dnd/dnd.cpp \
            dnd/styledbutton.cpp \
            dnd/iconview.cpp \
            dnd/listview.cpp \
            i18n/i18n.cpp \
            ../aclock/aclock.cpp \
            main.cpp

FORMS        = dnd/dndbase.ui

include( ../../src/qt_professional.pri )

canvas {
   HEADERS      +=graph.h \
            qasteroids/toplevel.h \
            qasteroids/view.h \
            qasteroids/ledmeter.h
   SOURCES      +=graph.cpp \
            qasteroids/toplevel.cpp \
            qasteroids/view.cpp \
            qasteroids/ledmeter.cpp
}

opengl {
   HEADERS      +=opengl/glworkspace.h \
            opengl/glcontrolwidget.h \
            opengl/gltexobj.h \
            opengl/glbox.h \
            opengl/glgear.h \
            opengl/gllandscape.h \
            opengl/fbm.h \
            opengl/glinfo.h \
            opengl/glinfotext.h
   SOURCES      +=opengl/glworkspace.cpp \
            opengl/glcontrolwidget.cpp \
            opengl/gltexobj.cpp \
            opengl/glbox.cpp \
            opengl/glgear.cpp \
            opengl/gllandscape.cpp \
            opengl/fbm.c
   win32 {
    SOURCES +=opengl/glinfo_win.cpp
   } mac {
    SOURCES +=opengl/glinfo_mac.cpp
    LIBS   +=-framework Carbon
   } else:unix {
    SOURCES +=opengl/glinfo_x11.cpp
   }

   FORMS     +=opengl/printpreview.ui \
            opengl/gllandscapeviewer.ui

   CONFIG -= dlopen_opengl
}

sql {
   FORMS     +=sql/connect.ui \
            sql/sqlex.ui
}

table {
   FORMS     +=widgets/widgetsbase.ui
}

!table {
   FORMS     +=widgets/widgetsbase_pro.ui
}

TRANSLATIONS    = translations/demo_ar.ts \
            translations/demo_de.ts \
            translations/demo_fr.ts \
            translations/demo_he.ts

PRECOMPILED_HEADER = demo_pch.h
```

demo/categoryinterface.h

```
#ifndef CATEGORYINTERFACE_H
#define CATEGORYINTERFACE_H

#include <qstring.h>
#include <qiconset.h>
#include <qobject.h>

class QWidgetStack;

class CategoryInterface : public QObject
{
    Q_OBJECT

public:
    CategoryInterface( QWidgetStack *s ) : stack( s ) {}
    virtual ~CategoryInterface() {}
    virtual QString name() const = 0;
    virtual QIconSet icon() const = 0;
    virtual int numCategories() const = 0;
    virtual QString categoryName( int i ) const = 0;
    virtual QIconSet categoryIcon( int i ) const = 0;
    virtual int categoryOffset() const = 0;

public slots:
    virtual void setCurrentCategory( int i ) = 0;

protected:
    QWidgetStack *stack;

};

#endif
```

demo/display.cpp

```
#include "display.h"

#include <qpainter.h>
#include <qlayout.h>
#include <qtimer.h>
#include <qpushbutton.h>
#include <qframe.h>
#include <qdial.h>
#include <qlcdnumber.h>
#include <qprogressbar.h>
#include <qspinbox.h>

#include <math.h>

Screen::Screen(  QWidget *parent, const char *name )
    : QFrame( parent, name )
{
    setLineWidth( FrameWidth );
    setFrameStyle( Panel | Sunken );
    setBackgroundMode( PaletteBase );
    setSizePolicy( QSizePolicy::MinimumExpanding, QSizePolicy::MinimumExpanding );
    setPaletteBackgroundColor( black );
    setPaletteForegroundColor( blue );

    yval = new int[width()];
    memset( yval, 0, sizeof(int)*width() );
    pos0 = 0;
    t0 = 0;
    step = 0;
}

Screen::~Screen()
{
    delete yval;
}

void Screen::resizeEvent( QResizeEvent *e )
{
    delete yval;
```

```
   int w = e->size().width();
   yval = new int[w];
   memset( yval, 0, sizeof(int)*w);
}

void Screen::animate()
{
   if ( step == 0 )
    return;

   int t = t0;
   int p = pos0;
   if ( step < 0 ) {
    t += width() + step;
   } else {
    t -= step;
    p -= step;
    if ( p < 0 )
        p += width();
  }

   for ( int i = 0; i < QABS( step ); i++ ) {
    int y = (int)((height()-FrameWidth)/2 * sin( 3.1415*(double)t/180.0 ));
    yval[ p ] = y;
    ++t;
    t %= 360;
    ++p;
    p %= width();
   }
   t0 -= step;
   if ( t0 < 0 )
    t0 += 360;
   pos0 = (pos0 - step) % width();
   if ( pos0 < 0 )
    pos0 += width();

   scroll( step, 0, QRect( FrameWidth, FrameWidth, width()-2*FrameWidth, height()-2*FrameWidth ));
}

void Screen::setStep( int s )
{
   step = s;
}

void Screen::drawContents( QPainter *p )
{
   QRect r = p->hasClipping() ?
         p->clipRegion().boundingRect() : contentsRect();

   int vp = ( r.left() - FrameWidth + pos0 ) % width();
   int y0 = FrameWidth + height()/2;

   for ( int x = r.left(); x <= r.right(); x++ ) {
    p->drawLine( x, y0 + yval[ vp ], x, r.bottom());
    ++vp;
    vp %= width();
   }
}

/************************************************************************/

Curve::Curve( QWidget *parent, const char *name )
   : QFrame( parent, name )
{
   setLineWidth( FrameWidth );
   setFrameStyle( Panel | Sunken );
   setBackgroundMode( PaletteBase );
   setPaletteBackgroundColor(black);
   setPaletteForegroundColor(red);
   setSizePolicy( QSizePolicy::MinimumExpanding, QSizePolicy::MinimumExpanding );

   shift = 0;
   n = 1;
}

void Curve::drawContents( QPainter *p )
{
   p->moveTo( width()/2, height()/2 + (int)(90.0*sin( double(shift)*3.1415/180.0)));

   for ( double a = 0.0; a < 360.0; a += 1.0 ) {
    double rad = 3.1415 / 180.0 * a;
    double x = width()/2 + 90.0 * sin(rad);
    double y = height()/2 + 90.0 * sin(n * rad + double(shift)*3.1415/180.0);
    p->lineTo( int(x), int(y) );
   }
}

void Curve::animate()
{
   shift = (shift + 1) % 360;
   update( FrameWidth, FrameWidth, width() - 2*FrameWidth, height() - 2*FrameWidth );
}

void Curve::setFactor( int f )
{
   n = f;
}

/************************************************************************/

DisplayWidget::DisplayWidget( QWidget *parent, const char *name )
   : QWidget( parent, name )
{
   timer = 0;

   QVBoxLayout *vbox = new QVBoxLayout( this, 10 );

   QHBoxLayout *hbox = new QHBoxLayout( vbox );
```

```
    screen = new Screen( this );
    dial = new QDial( this );
    dial->setNotchesVisible( TRUE );
    dial->setRange( -10, 10 );
    dial->setValue( 1 );
    screen->setStep( dial->value() );
    connect( dial, SIGNAL( valueChanged( int )),
         screen, SLOT( setStep( int )));
    lcd = new QLCDNumber( 2, this );
    lcd->setSizePolicy( QSizePolicy::MinimumExpanding, QSizePolicy::Preferred );
    lcdval = 0;

    hbox->addWidget( screen );

    QVBoxLayout *vb2 = new QVBoxLayout( hbox );

    curve = new Curve( this );
    spin = new QSpinBox( 1, 10, 1, this );
    connect( spin, SIGNAL( valueChanged( int )), curve, SLOT( setFactor( int )));
    spin->setValue( 2 );
    vb2->addWidget( curve );
    vb2->addWidget( spin );

    QHBoxLayout *hbox2 = new QHBoxLayout( vb2 );

    hbox2->addWidget( dial );
    hbox2->addWidget( lcd );

    bar = new QProgressBar( 10, this );
    tbar = 0;

    vbox->addWidget( bar );
}

void DisplayWidget::run()
{
    if ( !timer ) {
     timer = new QTimer( this );
     connect( timer, SIGNAL( timeout() ), SLOT( tick() ) );
    }

    timer->start( 5 );
}

void DisplayWidget::stop()
{
    timer->stop();
}

void DisplayWidget::tick()
{
    // sine
    screen->animate();
    // Lissajous
    curve->animate();
    // lcd display
    lcd->display( ++lcdval % 100 );
    // progress bar
    bar->setProgress( 5 + (int)(5*sin( 3.1415 * (double)tbar / 180.0 )));
    ++tbar;
    tbar %= 360;
}

void DisplayWidget::showEvent( QShowEvent * )
{
    run();
    screen->repaint();
}

void DisplayWidget::hideEvent( QHideEvent * )
{
    stop();
}
```

demo/display.h

```
#ifndef DISPLAY_H
#define DISPLAY_H

#ifndef QT_H
#include <qwidget.h>
#include <qframe.h>
#endif // QT_H

class QTimer;
class QDial;
class QLCDNumber;
class QProgressBar;
class QSpinBox;
class Screen;
class Curve;

class DisplayWidget : public QWidget {
    Q_OBJECT
public:
    DisplayWidget( QWidget *parent=0, const char *name=0 );

    void run();
    void stop();

protected:
    virtual void showEvent( QShowEvent * );
    virtual void hideEvent( QHideEvent * );

private slots:
    void tick();

private:
    Screen *screen;
    QDial *dial;
    Curve *curve;
    QSpinBox *spin;
    QLCDNumber *lcd;
    int lcdval;
    QProgressBar *bar;
    int tbar;
    QTimer *timer;
};

class Screen : public QFrame {
    Q_OBJECT
public:
    enum { FrameWidth = 3 };
    Screen( QWidget *parent=0, const char *name=0 );
    ~Screen();

    void animate();

public slots:
    void setStep( int s );

protected:
    virtual void drawContents( QPainter * );
    virtual void resizeEvent( QResizeEvent * );

private:
    int *yval;
    int pos0; // buffer pointer for x == 0
    int t0;   // time parameter at x == 0
    int step;
};

class Curve : public QFrame {
    Q_OBJECT
    enum { FrameWidth = 3 };
public:
    Curve( QWidget *parent=0, const char *name=0 );

    void animate();
public slots:
    void setFactor( int );

protected:
    virtual void drawContents( QPainter * );

private:
    int shift, n;
};

#endif // PLOT_H
```

demo/frame.cpp

```
#include "frame.h"
```

```
#include <qapplication.h>
#include <qpopupmenu.h>
#include <qmenubar.h>
#include <qaccel.h>
#include <qtoolbox.h>
#include <qpainter.h>
#include <qwidgetstack.h>
#include <qstylefactory.h>
#include <qaction.h>
#include <qsignalmapper.h>
#include <qdict.h>
#include <qdir.h>
#include <qtextcodec.h>
#include <stdlib.h>
#include <qbuttongroup.h>
#include <qtoolbutton.h>

static QTranslator *translator = 0;
static QTranslator *qt_translator = 0;

Frame::Frame( QWidget *parent, const char *name )
    : QMainWindow( parent, name )
{
    QMenuBar *mainMenu = menuBar();
    QPopupMenu *fileMenu = new QPopupMenu( this, "file" );
    fileMenu->insertItem( tr( "&Exit" ), this, SLOT( close() ),
                QAccel::stringToKey( tr( "Ctrl+Q" ) ) );

    QPopupMenu *styleMenu = new QPopupMenu( this, "style" );
    styleMenu->setCheckable( TRUE );
    QActionGroup *ag = new QActionGroup( this, 0 );
    ag->setExclusive( TRUE );
    QSignalMapper *styleMapper = new QSignalMapper( this );
    connect( styleMapper, SIGNAL( mapped( const QString& ) ),
         this, SLOT( setStyle( const QString& ) ) );

    QStringList list = QStyleFactory::keys();
    list.sort();
    QDict<int> stylesDict( 17, FALSE );
    for ( QStringList::Iterator it = list.begin(); it != list.end(); ++it ) {
      QString style = *it;
      QString styleAccel = style;
      if ( stylesDict[styleAccel.left(1)] ) {
          for ( uint i = 0; i < styleAccel.length(); i++ ) {
            if ( !stylesDict[styleAccel.mid( i, 1 )] ) {
                stylesDict.insert(styleAccel.mid( i, 1 ), (const int *)1);
                styleAccel = styleAccel.insert( i, '&' );
                break;
            }
          }
      } else {
          stylesDict.insert(styleAccel.left(1), (const int *)1);
          styleAccel = "&"+styleAccel;
```

```
   }
   QAction *a = new QAction( style, QIconSet(),
               styleAccel, 0, ag, 0, ag->isExclusive() );
   connect( a, SIGNAL( activated() ), styleMapper, SLOT(map()) );
   styleMapper->setMapping( a, a->text() );
  }
  ag->addTo( styleMenu );

  mainMenu->insertItem( tr( "&File" ), fileMenu );
  mainMenu->insertItem( tr( "St&yle" ), styleMenu );

  stack = new QWidgetStack( this );

  setCentralWidget( stack );
}

void Frame::setCategories( const QPtrList<CategoryInterface> &l )
{
  categories = l;
  QDockWindow *dw = new QDockWindow( QDockWindow::InDock, this );
  dw->setResizeEnabled( TRUE );
  dw->setVerticalStretchable( TRUE );
  addDockWindow( dw, DockLeft );
  setDockEnabled( dw, DockTop, FALSE );
  setDockEnabled( dw, DockBottom, FALSE );
  dw->setCloseMode( QDockWindow::Always );

  toolBox = new QToolBox( dw );
  dw->setWidget( toolBox );

  dw->setCaption( tr( "Demo Categories" ) );

  for ( int i = 0; i < categories.count(); ++i )
   toolBox->addItem( createCategoryPage( categories.at(i) ),
            categories.at(i)->icon(),
            categories.at(i)->name() );

  categories.first()->setCurrentCategory( 0 );
}

QWidget *Frame::createCategoryPage( CategoryInterface *c )
{
  QButtonGroup *g = new QButtonGroup( 1, Horizontal, toolBox );
  g->setFrameStyle( QFrame::NoFrame );
  g->setEraseColor(green);
  g->setBackgroundMode(PaletteBase);
  for ( int i = 0; i < c->numCategories(); ++i ) {
   QToolButton *b = new QToolButton( g );
   b->setBackgroundMode(PaletteBase);
   b->setTextLabel( c->categoryName( i ) );
   b->setIconSet( c->categoryIcon( i ) );
   b->setAutoRaise( TRUE );
   b->setTextPosition( QToolButton::Right );
   b->setUsesTextLabel( TRUE );
```

```
      g->insert( b, i + c->categoryOffset() );
      connect( g, SIGNAL( clicked( int ) ), c, SLOT( setCurrentCategory( int ) ) );
    }
    return g;
}

void Frame::setStyle( const QString& style )
{
    QStyle *s = QStyleFactory::create( style );
    if ( s )
      QApplication::setStyle( s );
}

void Frame::updateTranslators()
{
    if ( !qt_translator ) {
      qt_translator = new QTranslator( qApp );
      translator = new QTranslator( qApp );
      qApp->installTranslator( qt_translator );
      qApp->installTranslator( translator );
    }

    QString base = QDir("../../translations").absPath();
    qt_translator->load( QString( "qt_%1" ).arg( QTextCodec::locale() ), base );
    translator->load( QString( "translations/demo_%1" ).arg( QTextCodec::locale() ) );
}

bool Frame::event( QEvent *e )
{
    if ( e->type() == QEvent::LocaleChange )
      updateTranslators();

    return QMainWindow::event( e );
}
```

demo/frame.h

```
#include <qmainwindow.h>
#include <qintdict.h>
#include "categoryinterface.h"

class QToolBox;
class QStyle;
class QWidgetStack;

class Frame : public QMainWindow
{
    Q_OBJECT

public:
    Frame( QWidget *parent=0, const char *name=0 );
    void setCategories( const QPtrList<CategoryInterface> &l );

    static void updateTranslators();
    QWidgetStack *widgetStack() const { return stack; }

private slots:
    void setStyle( const QString& );

protected:
    bool event( QEvent *e );

private:
    QWidget *createCategoryPage( CategoryInterface *c );

private:
    QToolBox *toolBox;
    QWidgetStack *stack;
    QIntDict<QWidget> categoryPages;
    QPtrList<CategoryInterface> categories;

};
```

demo/graph.cpp

```
#include "graph.h"
#include <qcanvas.h>
#include <stdlib.h>
#include <qdatetime.h>
#include <qhbox.h>
#include <qpushbutton.h>
#include <qslider.h>
#include <qlabel.h>
#include <qlayout.h>

const int bounce_rtti = 1234;

// We use a global variable to save memory - all the brushes and pens in
// the mesh are shared.
static QBrush *tb = 0;
static QPen *tp = 0;

class EdgeItem;
class NodeItem;
class FigureEditor;
typedef QValueList<NodeItem*> NodeItemList;
typedef QValueList<EdgeItem*> EdgeItemList;

#define SPEED2ADVANCE(x) (301-x)

class GraphWidgetPrivate
{
public:
    GraphWidgetPrivate() {
      moving = 0;
      speed = 275;
    }
    ~GraphWidgetPrivate() {
      delete canvas;
    }
    NodeItemList nodeItems;
    FigureEditor* editor;
    QCanvas* canvas;
    QCanvasItem* moving;
    int speed;
};

class EdgeItem: public QCanvasLine
{
public:
    EdgeItem( NodeItem*, NodeItem*, QCanvas* );
    void setFromPoint( int x, int y ) ;
    void setToPoint( int x, int y );
    void moveBy(double dx, double dy);

    NodeItem* from;
    NodeItem* to;
};

class NodeItem: public QCanvasEllipse
{
public:
    NodeItem( GraphWidgetPrivate* g );
    ~NodeItem() {}

    void addInEdge( EdgeItem *edge ) { inList.append( edge ); }
    void addOutEdge( EdgeItem *edge ) { outList.append( edge ); }

    void moveBy(double dx, double dy);

    void calcForce();
    void advance( int stage );

private:
    GraphWidgetPrivate* graph;
    EdgeItemList inList;
    EdgeItemList outList;
};

void EdgeItem::moveBy(double, double)
{
    //nothing
}

EdgeItem::EdgeItem( NodeItem *fromItem, NodeItem *toItem, QCanvas *canvas )
    : QCanvasLine( canvas )
{
    from = fromItem;
    to = toItem;
    setPen( *tp );
    setBrush( *tb );
    from->addOutEdge( this );
    to->addInEdge( this );
    setPoints( int(from->x()), int(from->y()), int(to->x()), int(to->y()) );
    setZ( 127 );
}

void EdgeItem::setFromPoint( int x, int y )
{
    setPoints( x,y, endPoint().x(), endPoint().y() );
}

void EdgeItem::setToPoint( int x, int y )
{
    setPoints( startPoint().x(), startPoint().y(), x, y );
}


void NodeItem::moveBy(double dx, double dy)
{
    double nx = x() + dx;
    double ny = y() + dy;
    if ( graph->moving != this ) {
     nx = QMAX( width()/2, nx );
     ny = QMAX( height()/2, ny );
     nx = QMIN( canvas()->width() - width()/2, nx );
     ny = QMIN( canvas()->height() - height()/2, ny );
    }
    QCanvasEllipse::moveBy( nx-x(), ny-y() );
    EdgeItemList::Iterator it;
    for (  it = inList.begin(); it != inList.end(); ++it )
     (*it)->setToPoint( int(x()), int(y()) );
    for (  it = outList.begin(); it != outList.end(); ++it )
     (*it)->setFromPoint( int(x()), int(y()) );
}

NodeItem::NodeItem( GraphWidgetPrivate* g )
    : QCanvasEllipse( 32, 32, g->canvas )
{
    graph = g;
    graph->nodeItems.append( this );
    setPen( *tp );
    setBrush( *tb );
    setZ( 128 );
}

void NodeItem::advance( int stage ) {
     switch ( stage ) {
     case 0:
        calcForce();
        break;
     case 1:
        QCanvasItem::advance(stage);
        break;
     }
}
```

```
void NodeItem::calcForce() {
   if ( graph->moving == this ) {
    setVelocity( 0, 0 );
    return;
   }
   double xvel = 0;
   double yvel = 0;
   for ( NodeItemList::Iterator it = graph->nodeItems.begin(); it != graph->nodeItems.end(); ++it ) {
    NodeItem* n = (*it);
    if ( n == this )
       continue;
    double dx  = x() - n->x();
    double dy  = y() - n->y();
    double l = 2 * ( dx * dx + dy * dy );
    if ( l > 0 ) {
       xvel = xvel + dx*260 / l;
       yvel = yvel + dy*260 / l;
    }
   }
   double w = 1 + outList.count() + inList.count();
   w *= 10;
   EdgeItemList::Iterator it2;
   EdgeItem * e;
   NodeItem* n;
   for ( it2 = outList.begin(); it2 != outList.end(); ++it2 ) {
    e = (*it2);
    n = e->to;
    xvel = xvel - ( x() - n->x() ) / w;
    yvel = yvel - ( y() - n->y() ) / w;
   }
   for ( it2 = inList.begin(); it2 != inList.end(); ++it2 ) {
    e = (*it2);
    n = e->from;
    xvel = xvel - ( x() - n->x() ) / w;
    yvel = yvel - ( y() - n->y() ) / w;
   }
   if ( QABS( xvel ) < .1 && QABS( yvel ) < .1 )
    xvel = yvel = 0;
  setVelocity( xvel, yvel );
}

class FigureEditor : public QCanvasView {
public:
   FigureEditor( GraphWidgetPrivate *g, QWidget* parent=0, const char* name=0, WFlags f=0);

   QSize sizeHint() const;

protected:
   void contentsMousePressEvent(QMouseEvent*);
   void contentsMouseReleaseEvent(QMouseEvent*);
   void contentsMouseMoveEvent(QMouseEvent*);


   void resizeEvent( QResizeEvent* );
```

```
    void showEvent( QShowEvent* );
    void hideEvent( QHideEvent* e);

private:
    void initialize();
    QPoint moving_start;
    GraphWidgetPrivate* graph;
};

FigureEditor::FigureEditor(
      GraphWidgetPrivate* g, QWidget* parent,
      const char* name, WFlags f) :
    QCanvasView(g->canvas, parent,name,f)
{
    graph = g;
}

void FigureEditor::contentsMousePressEvent(QMouseEvent* e)
{
    QCanvasItemList l=canvas()->collisions(e->pos());
    for (QCanvasItemList::Iterator it=l.begin(); it!=l.end(); ++it) {
      if ((*it)->rtti()==bounce_rtti )
          continue;
      graph->moving = *it;
      moving_start = e->pos();
      return;
    }
    graph->moving = 0;
}

void FigureEditor::contentsMouseReleaseEvent(QMouseEvent* )
{
    if ( graph->moving )
      graph->moving = 0;
}

void FigureEditor::contentsMouseMoveEvent(QMouseEvent* e)
{
    if ( graph->moving ) {
      graph->moving->moveBy(e->pos().x() - moving_start.x(),
                e->pos().y() - moving_start.y());
      moving_start = e->pos();
      canvas()->update();
    }
}

class BouncyText : public QCanvasText {
    void initPos();
    void initSpeed();
public:
    int rtti() const;
    BouncyText(const QString&, QFont, QCanvas*);
    void advance(int);
};
```

```
BouncyText::BouncyText( const QString& text, QFont f, QCanvas* canvas) :
   QCanvasText(text, f, canvas)
{
   setAnimated(TRUE);
   initPos();
}


int BouncyText::rtti() const
{
   return bounce_rtti;
}

void BouncyText::initPos()
{
   initSpeed();
   int trial=1000;
   do {
    move(rand()%(canvas()->width()-boundingRect().width()),
        rand()%(canvas()->height()-boundingRect().height()));
    advance(0);
   } while (trial-- && xVelocity()==0.0 && yVelocity()==0.0);
}

void BouncyText::initSpeed()
{
   const double speed = 2.0;
   double d = (double)(rand()%1024) / 1024.0;
   double e = (double)(rand()%1024) / 1024.0;

   if ( d < .5 )
    d = -1 - d;
   else
    d = d + 1;
   if ( e < .5 )
    e = -1 - e;
   else
    e = e + 1;

   setVelocity( d*speed, e * speed );
}

void BouncyText::advance( int stage )
{
   switch ( stage ) {
    case 0: {
    double vx = xVelocity();
    double vy = yVelocity();

    if ( vx == 0.0 && vy == 0.0 ) {
       // stopped last turn
       initSpeed();
       vx = xVelocity();
```

```
        vy = yVelocity();
     }

     QRect r = boundingRect();
     r.moveBy( int(vx), int(vy) );

     if ( r.left() < 0 || r.right() > canvas()->width() )
        vx = -vx;
     if ( r.top() < 0 || r.bottom() > canvas()->height() )
        vy = -vy;

     r = boundingRect();
     r.moveBy( int(vx), int(vy) );
     if ( r.left() < 0 || r.right() > canvas()->width() )
        vx = 0;
     if ( r.top() < 0 || r.bottom() > canvas()->height() )
        vy = 0;

     setVelocity( vx, vy );
     } break;
     case 1:
     QCanvasItem::advance( stage );
     break;
   }
}

GraphWidget::GraphWidget( QWidget *parent, const char *name)
   : QWidget( parent, name )
{
   d = new GraphWidgetPrivate;
   d->canvas = 0;
   QVBoxLayout* vb = new QVBoxLayout(  this, 11, 6 );
   d->editor = new FigureEditor( d, this  );
   vb->addWidget( d->editor );
   QHBoxLayout* hb = new QHBoxLayout(  vb );
   hb->addWidget( new QLabel("Slow", this ) );
   QSlider* slider = new QSlider( 0, 300, 25, d->speed, Horizontal, this );
   connect( slider, SIGNAL( valueChanged(int) ), this, SLOT( setSpeed(int) ) );
   hb->addWidget( slider );
   hb->addWidget( new QLabel("Fast", this ) );
   hb->addSpacing( 10 );
   QPushButton* btn = new QPushButton( "Shuffle Nodes", this );
   connect( btn, SIGNAL( clicked() ), this, SLOT( shuffle() ) );
   hb->addWidget( btn );
}

GraphWidget::~GraphWidget()
{
   delete d;
}

void GraphWidget::setSpeed(int s)
{
   d->speed = s;
```

```
    if ( isVisible() && d->canvas )
     d->canvas->setAdvancePeriod( SPEED2ADVANCE( s ) );
}

void GraphWidget::shuffle()
{

    for ( NodeItemList::Iterator it = d->nodeItems.begin(); it != d->nodeItems.end(); ++it ) {
     NodeItem* ni = (*it);
     ni->move(rand()%(d->canvas->width()-ni->width()),rand()%(d->canvas->height()-ni->height()));
    }
}

QSize FigureEditor::sizeHint() const
{
    return QSize( 600, 400 );
}

void FigureEditor::resizeEvent( QResizeEvent* e )
{
    if ( canvas() )
     canvas()->resize( contentsRect().width(), contentsRect().height() );
    QCanvasView::resizeEvent( e );
}

void FigureEditor::showEvent( QShowEvent* )
{
    initialize();
    canvas()->setAdvancePeriod( SPEED2ADVANCE(graph->speed) );
}

void FigureEditor::hideEvent( QHideEvent* )
{
    initialize();
    canvas()->setAdvancePeriod( -10 );
}

void FigureEditor::initialize()
{
    if ( canvas() )
     return;
    resize( sizeHint() );
    graph->canvas = new QCanvas( contentsRect().width(), contentsRect().height() );
    if ( !tb ) tb = new QBrush( Qt::red );
    if ( !tp ) tp = new QPen( Qt::black );
    srand( QTime::currentTime().msec() );
    int nodecount = 0;

    int rows = 3;
    int cols = 3;

    QMemArray<NodeItem*> lastRow(cols);
    for ( int r = 0; r < rows; r++ ) {
     NodeItem *prev = 0;
```

```
   for ( int c = 0; c < cols; c++ ) {
     NodeItem *ni = new NodeItem( graph );
     ni->setAnimated( TRUE );
     nodecount++;
     ni->move(rand()%(graph->canvas->width()-ni->width()),rand()%(graph->canvas->height()-ni-
>height()));

     if ( r > 0 )
      (new EdgeItem( lastRow[c], ni, graph->canvas ))->show();
     if ( prev )
      (new EdgeItem( prev, ni, graph->canvas ))->show();
     prev = ni;
     lastRow[c] = ni;
     ni->show();
   }
  }

  graph->canvas->advance();

  QCanvasItem* i = new BouncyText( tr( "Drag the nodes around!" ), QFont("helvetica", 24), graph-
>canvas);
  i->show();
  setCanvas( graph->canvas );
  setMinimumSize( 600, 400 );
  setSizePolicy( QSizePolicy::MinimumExpanding, QSizePolicy::MinimumExpanding );
}
```

demo/graph.h

```
#include <qwidget.h>
class QStyle;
class QListBox;
class QListBoxItem;
class QWidgetStack;

class GraphWidgetPrivate;

class GraphWidget : public QWidget
{
  Q_OBJECT
public:
  GraphWidget( QWidget *parent=0, const char *name=0 );
  ~GraphWidget();

private slots:
  void shuffle();
  void setSpeed(int);

private:
  GraphWidgetPrivate* d;
};
```

demo/icons.h

```
/* XPM */
const char *widgeticon[] = {
```

```
/* columns rows colors chars-per-pixel */
"48 48 64 1",
"  c #e7e7e7",
". c Gray59",
"X c #e9d3b7",
"o c #a79783",
"O c #968775",
"+ c #cfbba2",
"@ c #bbbbbb",
"# c Gray70",
"$ c #6f6557",
"% c #838383",
"& c #7b7b7b",
"* c #baa891",
"= c #837666",
"- c #a2a2a2",
"; c #c9b59d",
": c Gray54",
"> c Gray67",
", c Gray42",
"< c #e4ceb2",
"1 c #cbcbcb",
"2 c #f9f9f9",
"3 c #c1c1c1",
"4 c #070706",
"5 c #4a443b",
"6 c #3c3731",
"7 c #dac5ab",
"8 c #dddddd",
"9 c #d2d2d2",
"0 c #484848",
"q c #b6a58f",
"w c #2c2823",
"e c #f1f1f1",
"r c #38332d",
"t c #555454",
"y c #665d51",
"u c #5b5348",
"i c #433d36",
"p c #ddc8ad",
"a c #c0ad96",
"s c #534b41",
"d c #d3bfa6",
"f c #867f76",
"g c #9e9d9c",
"h c #d7c3a9",
"j c #a8a198",
"k c #e0caaf",
"l c #312d29",
"z c #9f8f7c",
"x c #919090",
"c c #8e8d8c",
"v c #7f7263",
"b c #af9e89",
"n c #edd7bd",
"m c #b7b7b6",
"M c #796d5e",
"N c #e0cbb2",
"B c #8d7f6e",
"V c #252220",
"C c #191714",
"Z c #a7a7a7",
"A c #bdb3a7",
"S c #afafb0",
"D c Gray53",
"F c None",
/* pixels */
"FFFFFFFe81S.%,t06li FFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"F28@g&t0irr666rrrlV,2FFFFFFFFFFFFFFFFFFFFFFFFFFF",
">,t005ii50tt,%xg-ZS#mFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"#.,y,&x->m313S-.x%&9SFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"91191@>-gxD&&&&&&&&,S@2FFFFFFFFFFFFFFFFFFFFFFFFF",
"129-D%DD:ccc:D%%%%,Sm2FFFFFFFFFFFFFFFFFFFFFFFFFF",
"3e@-.xcc:DDD%%%%%&,mm2FFFFFFFFFFFFFFFFFFFFFFFFFF",
"3 >.cc::::&&%%%%%&,@#FFFFFFFFFFFFFFFFFFFFFFFFFFF",
"@ Zxcc::D%t0:%%%&&,3#FFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"m Zxc:::%,0tc%%%%&,1#FFFFFFFFFFFFFFFFFFFFFFFFFFF",
"# Zxc::D%c05tlw6y&&9SFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"S Zx::DD%:ilrV444V, #FFFFFFFFFFFFFFFFFFFFFFFFFFF",
"Ze-c:DDD%%,$t0lC44V.&@ FFFFFFFFFFFFFFFFFFFFFFFFF",
"ge-:DDD%%%%O=y5VCwwwwVw8FFFFFFFFFFFFFFFFFFFFFFFF",
"ge-D%D%%%%&Bbzv$v==$uiCC.2FFFFFFFFFFFFFFFFFFFFFF",
"g -%%%%D:cxco;*bbbbzvy5l4,eFFFFFFFFFFFFFFFFFFFFF",
"Z #.g-Z>SS#Sj*;**aaqzB$s6Ct FFFFFFFFFFFFFFFFFFFF",
"Z81333@@mmm#Sjbob*;+aoB$siCl1FFFFFFF29>1FF89eFFF",
">89913@@mmSg>S-cfzod+*zBys6C4D8FFFF>t6wV,tC4V,9F",
"#91 13@@@-t0tS>Z-gDqh+qOMs6V444t3 %5ssirC44CC44g",
"@93 13@@@,,StcSZZ-%chh;bBys56VC4C665ss5irV44CC44",
"11@ 13@@@,%3,,SZ--D:jddaoOOvy5wwwlr6i5s5i6lC4444",
"91# 93@@@,&3&,SZ--c%3Ad;*a;qO$iwwwlr6i555i6wC44C",
"83S893@m@&t@,,SZ--.&8 q++hph*Bu6rrrr6ii55i6rwC4C",
"em#983@mm>0tt->Z-g.& F3a7kkd*zvys5i66666ii66rlCC",
"F>#@8@@mmm>.Z>>ZZ>-&FFFgkkpdaoBvyus5i6666666rlwV",
"FmSZ83@mmmmm#>-.cD.3FFFmdkkdabO=M$yu5i66rr66rlwl",
"FeS1 1@@S-.xxg#1 2FFFFF8*Nkh;qoB=M$yusi6rrrrllwl",
"FF9SZx.Z#1 2FFFFFFFFFFF2jNN7+abzB=M$yysirrrrlwVl",
"FFFFFFFFFFFFFFFFFFFFFFFFmdNpd;*bzB=vM$yu56rllwVl",
"FFFFFFFFFFFFFFFFFFFFFFFF q<k7+aqozO=vM$yu56rlwVl",
"FFFFFFFFFFFFFFFFFFFFFFFFFjkkph+aqbzOBvM$yusirwwl",
"FFFFFFFFFFFFFFFFFFFFFFFFF1;kk7d;aqozOB=M$yusirwr",
"FFFFFFFFFFFFFFFFFFFFFFFFF2q7kph+;*bozOB=M$yusirr",
"FFFFFFFFFFFFFFFFFFFFFFFFF f+kkph+a*bozOB=M$yus5i",
"FFFFFFFFFFFFFFFFFFFFFF2g,lwopkk7d;a*bozOB=M$yuss",
"FFFFFFFFFFFFFFFFFFFFFFxM5V4ydkNk7d;a*bozOB=M$yuu",
"FFFFFFFFFFFFFFFFFFFFF>OBsC4w*k<<k7d;aqbozOB=M$yy",
"FFFFFFFFFFFFFFFFFFFFeO*z$w44vp<<<k7d;aqbozOB=M$$",
"FFFFFFFFFFFFFFFFFFFF8b;bB5C4s7<<<<k7d;aqbozOB=MM",
"FFFFFFFFFFFFFFFFFFFF8*dao$r4id<<XX<k7+;aqbozOB==",
"FFFFFFFFFFFFFFFFFFFFeb7;qBuVi+<<XXX<k7d;aqbozOBB",
"FFFFFFFFFFFFFFFFFFFFFgp+;o=5s;N<XXXX<N7d;aqbozOO",
"FFFFFFFFFFFFFFFFFFFFF1;d+*zM=;k<XXXXX<N7d;aqbozz",
"FFFFFFFFFFFFFFFFFFFFFFj7+abo*+7<XXXXXXXN7d;a*boo",
"FFFFFFFFFFFFFFFFFFFFFF@+d;aa;d7<XXXXnnXXN7d;a*bb",
"FFFFFFFFFFFFFFFFFFFFFF2oh;+;+hp<<XXXnnnnX<pd+a*q",
"FFFFFFFFFFFFFFFFFFFFFFF@;dhhhpN<XXnnnnnnnnXN7+;;"
};

/* XPM */
const char *widgeticon_sel[] = {
/* columns rows colors chars-per-pixel */
"48 48 64 1",
"  c #706557",
". c #818181",
"X c #100f0e",
"o c Gray73",
"O c #797979",
"+ c #e9d3b7",
"@ c #c9b59d",
"# c #cfbba2",
"$ c #847766",
"% c Gray91",
"& c Gray84",
"* c Gray64",
"= c #b4b4b4",
"- c #8b8b8b",
"; c #baa891",
": c #e3cdb2",
"> c Gray42",
", c #a9a9a9",
"< c #3b3732",
"1 c #dac5ab",
"2 c #4c443b",
"3 c #c1c1c1",
"4 c #b7a58f",
"5 c #2b2823",
"6 c #9a9a9b",
"7 c #464646",
"8 c #cbcbcb",
"9 c Gray33",
"0 c #38332c",
"q c #928472",
"w c #665d51",
"e c #929292",
"r c #9d8d7a",
"t c #5c5348",
"y c #aa9985",
"u c #433d35",
"i c #ddc8ad",
"p c #c0ae96",
"a c #a59581",
"s c #544b41",
"d c #d3bfa5",
"f c #e0caaf",
"g c #a2927e",
"h c #847c73",
"j c #ae9d88",
"k c #928d86",
"l c #302d29",
"z c #796d5e",
"x c #7f7263",
"c c #b1a08a",
"v c #252320",
"b c #d7c2a8",
"n c #8c7e6d",
"m c #968775",
"M c #dfcab1",
"N c #1d1b19",
"B c #a6a099",
"V c #9e907d",
"C c #9a8a78",
"Z c #a79782",
"A c Gray94",
"S c #afafaf",
"D c #868686",
"F c None",
/* pixels */
"FFFFFFFFFFFFXXXNvNNXFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFXXXN5l0<<<<00lvlFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"N77777uu7799>.e6*,S=0FFFFFFFFFFFFFFFFFFFFFFFFFFF",
"66>w>Oe*So383S*6eDO&OFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"388&8o,*6eD.O.OO..>SDFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"3A&*DDDD-----DD...>S.FFFFFFFFFFFFFFFFFFFFFFFFFFF",
"*Ao*ee---DDDD.....>=.FFFFFFFFFFFFFFFFFFFFFFFFFFF",
"6%,e------.OD....O>oOFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"6%,e----D.97-D...O>3>FFFFFFFFFFFFFFFFFFFFFFFFFFF",
".%,e----.>79-..D..>8>FFFFFFFFFFFFFFFFFFFFFFFFFFF",
"O%*e---DD-779l5<wOO&>FFFFFFFFFFFFFFFFFFFFFFFFFFF",
"O%*e--DDD-ul0vXFFv>%9FFFFFFFFFFFFFFFFFFFFFFFFFFF",
"9A*--DDDDD> 97lXFFve7XXFFFFFFFFFFFFFFFFFFFFFFFFF",
"7A*-DDDDD..m$w2vN5555vXXFFFFFFFFFFFFFFFFFFFFFFFF",
"<A*DDDD....qcgx x$$ tuNXNFFFFFFFFFFFFFFFFFFFFFFF",
"l%*.DDDD--eky@;yjjjrxw2lXNFFFFFFFFFFFFFFFFFFFFFF",
"5%Se6**,SS=SB;@;;pp4gn s<XNXFFFFFFFFFFFFFFFFFFFF",
"v%8333ooo===SBcyj;p#;an suNXNFFFFFFFFXXXFFXXFFFF",
"N&&&83ooooS*,S*khVyd#;gnws<NFNXFFFFl<05NXXXFXNXF",
"X&8%83ooo*979S,,*6D4b#4Czs<NFFFXNX<2ssu0XFFXXFFN",
"X&3%83oo3>>S9-S,**Dkbb@ynws2<vXXNl<2ss2u0vXXXXFF",
"X8o%83ooo>.3>>S,**DDhddpamCxw25vvl0<u2s2u<lNXXFF",
"F3=%83ooo>O3O>S***-ONzd@;p@4q u555l0<u222u<5NXFF",
"F=S&&3oooO9o>>S***eOXFm##bib;nt<0000<uu22u<05NXF",
"F6=8&3ooo,799*,***6>FFN;1ffd;Cxw22u<<<<uuuu<0lNF",
"F9=3&ooo==,6,S,,,,*<FFF2ffidpaqxwts2u<<<<<<<0l5X",
"FNS*&3ooo==o=,*eO90FFFFXdffdpjC$z wt2u<<00<<0l5N",
"FF-8%8ooS6.w<NXFFFFFFFFFc:fb@4gq$z wtsu<0000ll5v",
"FFFN97<NXFFFFFFFFFFFFFFF ::1#pcgq$z wwsu0000l5vN",
"FFFFFFFFFFFFFFFFFFFFFFFF5d:id@;jVq$xz wt2<0ll5vN",
"FFFFFFFFFFFFFFFFFFFFFFFFFg:f1#p4yVq$xz wt2<0l5vN",
"FFFFFFFFFFFFFFFFFFFFFFFFFuffib#p4jgqnxz wt2u055v",
"FFFFFFFFFFFFFFFFFFFFFFFFFX@ff1d@p4yrmn$z wtsu05v",
"FFFFFFFFFFFFFFFFFFFFFFFFFFq1fib#@;cZVmn$z wtsu05",
"FFFFFFFFFFFFFFFFFFFFFFFXs#ffib#p;cygmn$z wtsu0",
"FFFFFFFFFFFFFFFFFFFFFFFXNv5yiff1d@p;cyVmn$z wtsu",
"FFFFFFFFFFFFFFFFFFFFFvz2NXwdf:f1d@p;cZVmn$z wt2",
"FFFFFFFFFFFFFFFFFFFFFXCnsXFv4f::f1d@p4jZrmn$z ws",
"FFFFFFFFFFFFFFFFFFFFF ;g 5FFxi:::f1d@p4jarmn$z t",
"FFFFFFFFFFFFFFFFFFFFFZ@cn2XFs1::::f1d@p4jarqn$zw",
"FFFFFFFFFFFFFFFFFFFFFcdpa 0Fud::++:f1d@p4jarqn$ ",
"FFFFFFFFFFFFFFFFFFFFF$1@4qtNu#::+++:f1d@p4jarmnz",
"FFFFFFFFFFFFFFFFFFFFF<i#@y$2s@::++++:M1d@p4jZVm$",
"FFFFFFFFFFFFFFFFFFFFFX@d#;rz$@f:++++++:1d@p4jZVn",
"FFFFFFFFFFFFFFFFFFFFFt1#pcy;#1:+++++++:1d@p;cZm",
"FFFFFFFFFFFFFFFFFFFFFFFX#d@pp@d1:++++++++:1d@p;cV",
"FFFFFFFFFFFFFFFFFFFFFFF b@#@#bi:+++++++++:id#p;y",
"FFFFFFFFFFFFFFFFFFFFFFFXcp@@@@#d11iiMMMMMib#@p;a"
};

/* XPM */
const char *dbicon[] = {
/* columns rows colors chars-per-pixel */
"48 48 128 2",
"   c #9b8471",
".  c #c4a48e",
"X  c #ab806c",
"o  c #93918c",
"O  c #6a534a",
"+  c #9c7d69",
"@  c #a27d69",
"#  c #b4a69a",
"$  c #a4816d",
"%  c #6f6d6d",
"&  c #b1826e",
"*  c #91816d",
"=  c #a59687",
"-  c #c5baaf",
";  c #8e7e6a",
":  c #94816e",
">  c #a28571",
",  c #e8e7e5",
"<  c #ad4e34",
"1  c #99806c",
"2  c Gray99",
"3  c #393534",
"4  c #dad9d8",
"5  c #cbc6c1",
"6  c #9e816d",
"7  c #c5937d",
"8  c #ba8a75",
"9  c #c8c1ba",
"0  c #f4f3f2",
"q  c #ab8974",
"w  c #b38672",
"e  c #d6927d",
"r  c #4a6299",
"t  c #d19b85",
"y  c #595754",
"u  c #292828",
"i  c #aa8470",
"p  c #abb9d7",
"a  c #917d6a",
"s  c #8c6856",
"d  c #874c32",
"f  c #7a472d",
"g  c #967d69",
"h  c #b89a86",
"j  c #b98671",
"k  c #915c44",
"l  c #c38974",
"z  c #744a31",
"x  c #a87f6b",
"c  c #b28975",
"v  c #ba937e",
"b  c #9f4c31",
"n  c #52453d",
"m  c #6e462d",
"M  c #9598b1",
"N  c #73758b",
"B  c #907f6c",
"V  c #bbb3ab",
"C  c #7586b4",
"Z  c #897b77",
"A  c #c58d79",
"S  c #bcbcbc",
"D  c #934a30",
"F  c #7c5f51",
"G  c #9b7b67",
"H  c #8e7f6c",
"J  c #bd8d78",
"K  c #8d7966",
"L  c #93715e",
"P  c #cc8e79",
"I  c #e1deda",
"U  c #81472e",
"Y  c #c45035",
"T  c #9f8d7e",
"R  c #85624c",
"E  c #b8836f",
"W  c #926c5b",
"Q  c #bbaca0",
"!  c #647aab",
"~  c #d0ccc7",
"^  c #654931",
"/  c #927864",
"(  c #adabb7",
")  c #8d8477",
"_  c #423c3a",
"`  c #a79f95",
"'  c #e0a690",
"]  c #a27666",
"[  c #98624c",
"{  c #8b7d6a",
"}  c #a58773",
"|  c #d8d4e3",
" . c #a55941",
".. c #7e7164",
"X. c #a27b67",
"o. c #2f2e2e",
"O. c #ac8e79",
"+. c #927b68",
"@. c #947f74",
"#. c #d7d4cf",
"$. c #333130",
"%. c #947f6c",
"&. c #181715",
"*. c #f8f7f7",
"=. c #af806c",
"-. c #8d7c68",
";. c #835941",
":. c #2d2f34",
">. c #c98c77",
",. c #cf907b",
"<. c #a89180",
"1. c #ab7a68",
"2. c #c5c4d8",
"3. c #9ea4bf",
"4. c #e49f88",
"5. c #eeedeb",
"6. c #8a7564",
"7. c #817d84",
"8. c #a05f48",
"9. c #c08873",
"0. c #977462",
"q. c #9b705d",
"w. c #8b7e6f",
"e. c #74543d",
"r. c #98836f",
"t. c #586997",
"y. c #8e806c",
"u. c None",
/* pixels */
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.*.I 5 Q # = <.<.h # V 5 I *.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.*.#.# > G G X.X.@ @ @ @ X.X.X.+ T Q #.*.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.*.9 T G G + @ @ @ @ @ @ @ @ @ @ @ + + G T 9 0
u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.2 5 T G G + + + @ @ @ x x x x x x @ @ @ @ + + G G > - 0 u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.5.` g G G G + @ @ @ @ x x x X X X X x x @ @ @ + + + + G <.I u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u., T g g g + + + @ @ @ x x X X X X X X X x x x @ @ + + + g g g - 2 u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.4   +.g g g + + + @ @ $ x X =.=.=.=.=.=.X X X x x @ @ + + g g g +.# *.u.u.u.u.u.u.u.",
"u.u.u.u.u.I : +.+.g g g + + 6 $ $ x X =.=.=.=.& & =.X X X X x $ 6 + + 1 g g +.+.` *.u.u.u.u.u.u.",
"u.u.u.u.0 T +.a a g g g + 1 6 $ $ x X X =.=.& & & =.=.X X x $ $ 6 6 1 1 g g a a -.# 2 u.u.u.u.u.",
"u.u.u.u.V K a a a a g g 1 1 6 6 $ $ x X =.=.& & & & =.X X x $ $ 6 6 1 1 g g a a a -.- u.u.u.u.u.",
"u.u.u.#.K -.; a a %.%.%.1 6 6 6 6 $ x X X & & & & & =.X x $ $ 6 6 6 1 1 %.%.a a ; ; -.4 u.u.u.u.",
"u.u.2 = -.; ; ; a a %.%.1 1 1 6 6 $ $ x X =.& & & & =.X x $ $ 6 6 1 1 1 1 %.a a ; ; -.= *.u.u.u.",
"u.u.#.-.-.; ; ; B B %.%.1 1 1 1 6 6 $ $ X & w & E w & X $ $ 6 6 1 1 %.%.%.%.B a ; ; ; -.9 u.u.u.",
"u.u.` -.{ ; ; ; B B B %.%.%.1 1 1 6 $ i $ L R F F s X.i $ 6 6 1 1 %.%.%.%.%.B B ; ; ; ; r.5.u.u.",
"u.5.y.{ { ; ; ; H B B B B %.%.%.1 1 6 L e.^ m U U m z [ + 6 1 1 %.%.%.B B B B B ; ; ; ; -.9 u.u.",
"u.~ { { { { ; H B B B B B B B * %.r.W z m D b < Y b U f ;.g 1 %.%.B B B B B B B H ; ; ; ; = 2 u.",
"u.V { { { ; H H B B B B B B B B * K d m f d < Y Y < d U f k : %.* B B B B B H H H H H ; ; ; I u.",
"u.` { { { ; H H B H H H B B B B * W U ^ z d d d D D d z ^ d q.* B B B H H H H H H H H H H { 9 u.",
"u.= ; ; H H H H H y.y.y.y.y.y.y.y.8.U ^ z z n m U ^ m m z f 8.y.H H H H H H H H H H H H H ; Q u.",
"u.T H H H H H H y.y.y.y.y.y.y.y.y. .U ^ z ^ m f d ^ ^ m ^ m  .B y.y.y.y.H H H y.y.y.y.H H H ` u.",
"u.T y.H H y.y.y.y.y.y.y.y.y.y.y.y.y.[ D z z m b < Y < z m ^ f  .y.y.y.y.y.y.y.y.y.y.y.y.y.y.y.y.= u.",
"u.) y.y.y.y.y.y.y.y.y.y.y.y.y.B B B * s b U z f d < Y b f z ^ D [ * y.y.y.y.y.y.y.y.y.y.y.y.y.y.y.= u.",
"u.w.: B B B B B B B B B * * * * : * ;.b D d d b b D D d D b / * * B B B y.y.B y.y.y.y.y.* * = u.",
"u.w.: * * * * * * * B * * : : : :  / d D b < < < < < < D s r.: : : * * * * * B * * * * * * ` u.",
"u.o : * * * : : : : * : : : : 1 1 6 > G ;.f D b < b D d s 6 r.1 : : : : : * * * * * * * * * Q u.",
"u.` -.: : : : : : : : : 1 1 1 1 6 6 $ i i q.R ;.;.;.s X.i $ 6 1 1 1 1 : : : : : : * : : : : 9 u.",
"u.9 ..r.r.r.: : : : r.w.Z 1 6 6 6 $ $ i & w j E E j j & i $ $ 6 6 1 1 1 1 : : : : r.: : r.r.I u.",
"u., ..1 r.r.r.r.r.r.  @.t.t.Z $ i i i X & & E E E E E & X i $ $ $ 6 6 1 1 1 r.r.r.r.r.r.r.<.*.u.",
"u.u.o K       r.      6 @.t.r 7.6.G w & & & E E E E E & & i i $ $ $ 6 6 6 6 r.r.r.r.r.r.r.9 u.u.",
"u.u.~ ..6             6 > Z N % :.3 F w j E E E E E E E E & & i i $ $ $ 6 6 6            <.5.u.u.",
"u.u.*.) K }   > > > > > > > q ..:.u u o.n 0.9.j E E E E E & & & i i $ $ > > > > > >       > 9 u.u.u.",
"u.u.u.~ ..6 } } } } } } } i q 0.o.u O X.n $.O 1.l j E E E E & & & i i i > > > } > > > > > = 0 u.u.u.",
"u.u.u.u.` 6.> q q q q q i i c R u s J 8 ] n $.F E l E E E & & i i i i i i } } } } } } } ~ u.u.u.u.",
"u.u.u.u., { K q q q q c c w c 8 O o.X.9.9.9.s 3 _ W l j j E w w w w w q q q q q q q h *.u.u.u.u.",
"u.u.u.u.u.~ K g c c c c c c c 8 j _ n 9.9.9.>.E O $.O =.l 8 j j w w c c c c c q c } 4 u.u.u.u.u.",
"u.u.u.u.u.u.9 +.1 c J 8 8 8 8 8 A =.3 F >.l l l P 1.O _ F 9.A 8 8 8 8 8 c c c J } 5 u.u.u.u.u.u.",
"u.u.u.u.u.u.u.9 1 6 q J J J J J A ,.q.u W ,.>.>.>.P P ] n n ] P A J J J 8 J v q - u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.5 > } O.7 7 A A A A ,.O o.1.e >.>.>.>.,.P F $.O j ,.7 A 7 7 O.V u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.I ` q O.v 7 t ,.P ,.,.O 3 9.e ,.,.,.,.,.e 1.3 3 F 7 t v <.~ u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.*.9 O.v v h t t t t ,._ O e e e e e e e 4.] u u _ y o , u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u., V h h h . . . ' v 3 s ' ' ' ' t ' 7 O u o.u u ` u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u., 9 Q . . . . . _ u O > O.>   ..n &.&.$.o.o.o.% 4 u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.2 , ~ - - T u u u 3 y y % o 9 y u $.o.o.o.u $.V u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.0 % u o.o.o 5.2 u.u.7.u o.u o.3 $.u u S u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u., n u o.o.y % % y 3 o.3 N M 3.( 7._ $.4 u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.4 3 u $.o.o.o.o.o.3 M C t.r ! 2.( 3 y *.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.S u o.$.$.$.$.u % M r r ! ! 3.| 7.u 5 u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.` $._ _ _ _ 3 7.C r t.p p C | ( 3 o u.u.u.u.u.u."
};

/* XPM */
const char *dbicon_sel[] = {
/* columns rows colors chars-per-pixel */
"48 48 128 2",
"   c #a3816d",
".  c #2a2929",
"X  c #b1826e",
"o  c #91816d",
"O  c #796a5b",
"+  c #8e7e6a",
"@  c #4b433b",
"#  c #94816d",
"$  c #9d8571",
"%  c #ad4e34",
"&  c #9a816d",
"*  c #a28571",
"=  c #9e816d",
"-  c #0b0b0a",
";  c #756c73",
":  c #c4937d",
">  c #3a3633",
",  c #ad816d",
"<  c #ba8a75",
"1  c #ab8974",
"2  c #574d44",
"3  c #cdc9da",
"4  c #b38672",
"5  c #876858",
"6  c #d6927d",
"7  c #d19984",
"8  c #aa8570",
"9  c #917d6a",
"0  c #a9816d",
"q  c #4b6399",
"w  c #874c32",
"e  c #63574c",
"r  c #7a472d",
"t  c #957d69",
"y  c #b98671",
"u  c #a17d6a",
"i  c #915c44",
"p  c #c38974",
"a  c #9a7d6a",
"s  c #744a31",
"d  c #aa7f6b",
"f  c #b28975",
"g  c #9d7d69",
"h  c #9f4c31",
"j  c #6e462d",
"k  c #b8937e",
"l  c #8491ba",
"z  c #a57e6a",
"x  c #907f6c",
"c  c #c58d79",
"v  c #735749",
"b  c #c5a28c",
"n  c #934a30",
"m  c #6b7397",
"M  c #bd9b85",
"N  c #9a7a66",
"B  c #897b77",
"V  c #8e7f6c",
"C  c #647aa9",
"Z  c #9da9c6",
"A  c #bd8d78",
"S  c #8d7966",
"D  c #817261",
"F  c #93705e",
"G  c #cc8e79",
"H  c #2f2d2c",
"J  c #81472e",
"K  c #353230",
"L  c #c45035",
"P  c #88644d",
"I  c #433c37",
"U  c #927864",
"Y  c #b8836f",
"T  c #8a7361",
"R  c #916958",
"E  c #5a6c98",
"W  c #795e52",
"Q  c #654930",
"!  c #dba993",
"~  c #43598f",
"^  c #c5a892",
"/  c #daa28b",
"(  c #e4a58f",
")  c #a27666",
"_  c #98624c",
"`  c #8b7d6a",
"'  c #645d52",
"]  c #a55941",
"[  c #a58773",
"{  c #a37a66",
"}  c #adaab7",
"|  c #ab8e78",
" . c #907b68",
".. c #947f74",
"X. c #947f6c",
"o. c #634d45",
"O. c #232120",
"+. c #b0806c",
"@. c #a47a68",
"#. c #6e6358",
"$. c #a5806c",
"%. c #9e7b68",
"&. c #8d7c68",
"*. c #835941",
"=. c #2d2f34",
"-. c #c98c77",
";. c #997c68",
":. c #cf907b",
">. c #99836f",
",. c #a27c69",
"<. c #9c9bab",
"1. c #31302f",
"2. c #ab7a68",
"3. c #e49f88",
"4. c #987f6c",
"5. c #967561",
"6. c #7b5f4a",
"7. c #9ca1bb",
"8. c #807c84",
"9. c #a05f48",
"0. c #987262",
"q. c #181816",
"w. c #96836f",
"e. c #c08873",
"r. c #947b68",
"t. c #9b705d",
"y. c #8e806c",
"u. c None",
/* pixels */
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.- O.I 2 e W O W W e 2 > O.- u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.- . e T %.%.%.,.,.,.,.,.,.,.%.g 5 o.H - u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.- > O N %.%.,.,.u u ,.z z z z ,.,.,.%.%.N 5 I - u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.> O ;.;.g %.g u ,.z z z z z z z z z z z z u g g ;.;.T @ - u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.q.e r.;.;.;.g u u z z z d d d d d d d d z z ,.u g g a a ;.O O.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.q.O r.t ;.a g g u u z z d d d , , , d d d z z u u g g a t t r.@ u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.. D r.r.t t a g u u z $.0 d , , +.+.+., , , d d z u u g a ;.;.t r.e - u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.O.T  .r.r.t t a g =  $.0 d , +.+.+.+.+.+., , d d $.  = g a a a t 9  .e - u.u.u.u.u.u.u.",
"u.u.u.u.- O  .9 t t t 4.a a =    0 d , +.+.+.X X +., , d 0 $.  = = a a 4.t t 9  .e u.u.u.u.u.u.u.",
"u.u.u.u.2  . . .9 9 t t 4.& = =  $.0 , , +.X X X +., , d $.$.  = = 4.4.4.t 9 9 9 &.2 u.u.u.u.u.",
"u.u.u.. S &.+ 9 9 X.X.X.4.= = = =  $.d , +.X X X +., , 0 $.$.= = = & 4.X.X.9 9 + + &.H u.u.u.u.",
"u.u.u.#.&.+ + + 9 9 X.X.4.& & = =  $.0 , +.X X X +., , $.$.  = = 4.4.4.4.X.9 9 + + &.O - u.u.u.",
"u.u.H &.&.+ + + x x X.X.4.4.4.& = =  $., X 4 X Y 4 X 0 $.  = = & 4.X.X.X.X.x 9 + + + &.@ u.u.u.",
"u.u.' &.` + + + x x x X.X.X.4.4.& &  0 $.F P 6.6.R @.8  = & & 4.X.X.X.X.X.x x + + + + D q.u.u.",
"u.q.D ` ` + + + V x x x x X.X.X.4.4.= F v Q j J J j s _ g = & 4.X.X.X.x x x x x + + + + &.@ u.u.",
"u.K ` ` ` ` + V x x x x x x x x o X.w.F s j n h % L h J r *.;.& X.X.x x x x x x x x V + + + + O - u.",
"u.2 ` ` ` + V V x x x x x x x x x o  .w j r w % L L % w J r i # X.o x x x x x V V V V V + + + O.u.",
"u.e ` ` ` + V V x V V V x x x x o R J Q s w w w n n w s Q w t.o x x x V V V V V V V V V V ` @ u.",
"u.#.+ + V V V V V y.y.y.y.y.y.y.y.9.J Q s s @ j J Q j j s r 9.y.V V V V V V V V V V V V V + e u.",
"u.O V V V V V V y.y.y.y.y.y.y.y.y.] J Q s Q j r w Q Q j Q j ] x y.y.y.y.V V V y.y.y.y.V V V #.u.",
"u.O y.V V y.y.y.y.y.y.y.y.y.y.y.y.y._ n s s j h % L % s j Q r ] y.y.y.y.y.y.y.y.y.y.y.y.y.y.y.y.O u.",
"u.O y.y.y.y.y.y.y.y.y.y.y.y.x x x o 5 h J s r w % L h r s Q n _ o y.y.y.y.y.y.y.y.y.y.y.y.y.y.y.D u.",
"u.#.# x x x x x x x x x x o o o o # o *.h n w w h h n n w n h U o o x x x y.y.x y.y.y.y.y.o o O u.",
"u.' # o o o o o o o x o o # # # # >.U w n h % % % % % % n R w.# # # o o o o o x o o o o o o #.u.",
"u.2 # o o o # # # # o # # # # # & & * N *.r n h % h n w R = & & # # # # # o o o o o o o o o e u.",
"u.@ &.w.# # # # # # # # & # & & & =  8 , t.P *.*.*.P { 8  = & & & # # # # # # # # o # # # # 2 u.",
"u.1.D >.w.w.w.w.w.w.>.B B & = & =    0 , 4 y Y Y y y X 0    = = & & & # # # # # w.# # w.w.H u.",
"u.q.O & >.>.>.>.>.>.>...E E B  8 8 0 , X X Y Y Y Y Y X , 0     = & & & & w.w.w.w.w.w.w.w.T - u.",
"u.u.e S $ >.>.>.$ $ >.= ..q q 8.T %.4 X X X Y Y Y Y Y X X , 0     = = & & >.>.>.>.>.>.>.2 u.u.",
"u.u.1.O & $ $ $ $ $ $ = * B m ; =.K W 4 y Y Y Y Y Y Y Y X X , 8 0    = = = $ $ $ $ $ $ t q.u.u.",
"u.u.- ' S [ $ * * * * * * 8 ; =.. . H @ 0.e.y Y Y Y Y Y X X X , 0    * * * * * $ $ $ $ 2 u.u.u.",
"u.u.u.H D & [ [ [ [ [ [ 8 1 5.H . v { o.K o.2.p y Y Y Y Y X X X 8 8 8 * * * [ * * * * U - u.u.u.",
"u.u.u.u.e T * 1 1 1 1 1 8 8 f 5 . 5 A < ) 2 K W Y p Y Y Y X X 8 8 8 8 8 8 8 [ [ [ [ [ I u.u.u.u.",
"u.u.u.u.q.D S 1 1 1 1 f f 4 f < o.H @.e.e.e.R > I F p y y Y 4 4 4 4 4 1 1 1 1 1 1 1 T - u.u.u.u.",
"u.u.u.u.u.> S t f f f f f f f < y I @ e.e.e.-.Y v K o.+.p < y y 4 4 f f f f f 1 f [ H u.u.u.u.u.",
"u.u.u.u.u.u.I  .& f A < < < < < c +.> v -.p p p G 2.o.I W e.c < < < < < f f f A [ I u.u.u.u.u.u.",
"u.u.u.u.u.u.u.> & = 1 A A A A A c :.t.. R :.-.-.-.G G ) @ @ ) G c A A A < A A 1 @ u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.> = [ | : : c c c c :.v H 2.6 -.-.-.-.:.G W 1.e y :.c c : : | 2 u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.O.O 1 | k : 7 :.G :.:.o.> e.6 :.:.:.:.:.6 2.K > 5 : 7 k >.> u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.- @ 1 k k M b 7 7 7 :.I o.6 6 6 6 6 6 6 6 3.) . . I ' 2 q.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.q.#.k M M b b b ! k > 5 ( ( ( / / ! : v . H . . . u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.q.2 X.M b ^ ^ ^ I . e * | $ >.D 2 - O.1.H H H 1.q.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.q.@ #.$ ` . . . > @ @ I K O.q.O.1.1.H H . H O.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.- 1.. H 1.. - u.u.u.. . H O.1.> K . . O.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.q.H . H H 1.. O.H 1.H > 8.<.7.} 8.I . O.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.O.H . 1.1.1.1.1.H K <.l E q C 3 } > 1.- u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.. . H 1.1.1.1.. #.l ~ q C C Z 3 8.. O.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.H . H H H H . ; E ~ ~ Z Z C 3 <.K H u.u.u.u.u."
};

/* XPM */
const char *internicon[] = {
/* columns rows colors chars-per-pixel */
"48 48 32 1",
"  c #54706a",
". c #32494f",
"X c #a5b6b0",
"o c #1f6796",
"O c #56758a",
"+ c #273133",
"@ c #465450",
"# c #173547",
"$ c #5e7968",
"% c #7192af",
"& c #6e8c86",
"* c #112836",
"= c #688370",
"- c #2d5161",
"; c #5885a9",
": c #134c6e",
"> c #53605e",
", c #080f14",
"< c #656860",
"1 c #405c65",
"2 c #2e3f44",
"3 c #131b21",
"4 c #44677a",
"5 c #17425c",
"6 c #8e9185",
"7 c #3f4944",
"8 c #3979a3",
"9 c #797770",
"0 c #202527",
"q c #363936",
"w c #7e9c93",
"e c None",
/* pixels */
"eeeeeeeeeeeeeeeeeeee%&&OOOO&eeeeeeeeeeeeeeeeeee",
"eeeeeeeeeeeeeeeeewOO 4OOOO4411..9eeeeeeeeeeeeeeee",
"eeeeeeeeeeeeeX&OOwwO4OOOO411-.2q2@9eeeeeeeeeeeee",
"eeeeeeeeeeee;OO%XXwO444411.22.qqq+07eeeeeeeeeeee",
"eeeeeeeeeeewO;%XXX&O4411--..222q2++000>eeeeeeeeee",
"eeeeeeeee%;;XXXwOO444411-..222qq++00+0qeeeeeeeee",
"eeeeeeee%;;%%%%OOOOO4411-..22++++0000q3+eeeeeeee",
"eeeeeee8%;;;%%%OOOOO4>>11-.77q+++003300,3eeeeeee",
"eeeeee8%%;;;%%;OOOO4>><>>@@77qqq++000000,3eeeeee",
"eeeee88%%;;%%;;OOO>>>>>>>>@77q+q+0000+033,0eeeee",
"eeee%8%%%;%%;;OOOO $411>>@777qq++000q2+033,<eeee",
"eeee88;;;%%;;;OOO4O 4>>>@@7722qqq+++7q++033,eeee",
"eee8o88;%;;%;OOOO   <>@@77777qqqqq++q0q+00,0eee",
"eeXooo8%;;;;;OO   <<>>@@@@7@7qqqqq+077q+q03,9ee",
"ee;o8;%;8;;;8O4411>>>1@@>@7@77qqqq++q@70+703,0ee",
"eeoo88888;;&;&44>>><<>>>>@772#22+++q77307q0,,,ee",
"e%ooo88888;=  &$>$>>>@@@.5.2##22+++21+07q0003,@e",
"e8ooo88888&&44OOO>---.555#5@7##2+++2-.77+00+3,3e",
"eoooo8oooo444O4----.---..@@72*22++2.2@q0+000,,e",
"e8;oooooo8OOO=4-:-:-5->@@@@@.2#22+2.*0q7q++003,e",
"%88oooooo $&6=$ ::::5:-..@@@@7722+220330000003,>",
";oooooo8O =&==&$-:::55555@@7@.722+0++000300303,7",
"8ooooo;w&&&w&&&= 1: 1-:5-@@777.7.++++000300333,+",
"8oooo8w66=&w&&=$$  =$$ 11@@@@7#22+2+**0000033,,0",
"8oooo;www&ww=  $$  $$$$>-.@@772###****0000033,,0",
"8oooo&XXXwww=  $= $$$$$ .5..77722.+***3++0033,,0",
";oooo&XXXw&=$$$$$$>$$ $ @5-.777777+***00q0033,,+",
"%o4O4&66&&&==$$$$1@$$$$ >25.277@77+****0+03033,@",
"eo4OO&996===$ ===  $$$   @#5277@7#****00000333,9",
"eo4ooO=99$$=$==$=$$$$$$ >>.5-772******+000003,,e",
"e4::::4$ >  $=$$$===$$ <>><2-7###****++00003,,3e",
"e%::::-141>>>> $9999<<>>@<>22#52#***+++0003,3,7e",
"ee:::::-1111>-199<<9<<@><<@77272******0003333,9e",
"ee45:::::-11111><<<<>@>><@@q2q7###+++**0333333ee",
"eee5::::::-111--1>11>-@>@..+227##22+++*033,3,<ee",
"eee455:5555------1-1>@@@...22222222++**33,,,3eee",
"eeee5555555555-11-1><<>@..2.2.2.772+**33,,,,&eee",
"eeee%#555-5555--@@><<<>@-.q2.22222+***3333,7eeee",
"eeeee1#555--555--@@@>@.7.7q22###2#***3333,,eeeee",
"eeeeee.#555--555-....@@.7772####*****333,,6eeeee",
"eeeeeee-##55--55-......77772####*****33,,9eeeeee",
"eeeeeeee-*##555--....222.772###+****03,,6eeeeeee",
"eeeeeeeee1####5......2222222#2+*****3,3eeeeeeeee",
"eeeeeeeeee 2###55.77..2++##222+****3,0eeeeeeeeee",
"eeeeeeeeeeee@####2.772.+#####+++0033>eeeeeeeeeee",
"eeeeeeeeeeeeew@###2@@72+**#****330>eeeeeeeeeeeee",
"eeeeeeeeeeeeeeee>2##++q2#****0+79eeeeeeeeeeeeeee",
"eeeeeeeeeeeeeeeeeee$>@27722@>eeeeeeeeeeeeeeeeeee"
};

/* XPM */
const char *internicon_sel[] = {
/* columns rows colors chars-per-pixel */
"48 48 32 1",
"  c #526f6b",
". c #2d4750",
"X c #226a98",
"o c #a7b7ae",
"O c #293233",
"+ c #53748a",
"@ c #44534e",
"# c #1c3848",
"$ c #112a39",
"% c #2e5263",
"& c #5d7a67",
"* c #688472",
"= c #688cae",
"- c #457fa9",
"; c #6f9383",
": c #144c6e",
"> c #506462",
", c #0e1419",
"< c #686c63",
"1 c #405c65",
"2 c #313d3f",
"3 c #151e23",
"4 c #16425c",
"5 c #43667d",
"6 c #617f93",
"7 c #8f8d7a",
"8 c #3d4844",
"9 c #829e9a",
"0 c #7f7769",
"q c #212728",
"w c #575c58",
"e c None",
/* pixels */
"eeeeeeeeeeeeeeeeeeee@1 66+6>8eeeeeeeeeeeeeeeeeeee",
"eeeeeeeeeeeeeeeee +6+5+6++5551.2Oeeeeeeeeeeeeeeeee",
"eeeeeeeeeeeeee>+6699+5+6++511%.282O3eeeeeeeeeeeeee",
"eeeeeeeeeeeee+669oo96555511..282222Oqeeeeeeeeeeeee",
"eeeeeeeeeee6669ooo6+5511%%...2222OOqqO3eeeeeeeeeee",
"eeeeeeeeee6==ooo96+55551%%..222OOOOqqOq,eeeeeeeee",
"eeeeeeeee6=-=9==6++++551%%...2OOOOqq3qOq3eeeeeeee",
"eeeeeee-========6++++5>>11%8882OOqqq33qq3,eeeeeee",
"eeeeeeX========6+++5>>>>www@8822OOqq3qqq33,eeeeee",
"eeeeeX-=======66+++>>>>>>www@882OOOqqqqOq33,,eeeee",
"eeee%-========6++++ <511ww@8882OOOqqq22qq,,,,eeee",
"eeeeX-========6+++++ 5>>w@@@82222OOqO82qO33,,eeee",
"eeeXXX--=-===++++   <w@@@@8882222OOOOqOqq3,3eee",
"eeeXXX-=--==-++   <<>w@@@@8@82222OqO88OO2q,,eee",
"ee5X--=------+5511>>>1@@w@8@88222OOq2@8322q3,,ee",
"eeXXX------666  >>><<wwww@882#22OOO2@23q8Oq,,,ee",
"e%XXXXX---6*  *&><>>>@@@.4.2##.2OOq2@qq82qq3,,3e",
"e:XXXXX-XX;* 5+++>%%%.444#.@8##2OOO2%288qqqO3,,e",
"eXXXXXXXXX5555+5%%%1..%%.%@@82$22OO2.2@OqOqq3,,e",
"eX=XXXXXXX++6*5%:%:%4%w@1@@@.2#.2O2.$q28OOqq3,,e",
"e--XXXXXX  ;7*& ::::4:...@@@@8822O22q33qqqqq33,e",
"%XXXXXXX-+ *;**;&%:::44444@@@@@822OqOO3q33qq33,,,",
":XXXXX-9;;;;;;;* 1% 1%:4%@@888.82OOOOqqq3qq33,,,",
"5XXXXX99;**;;**&& +*&&>1>@@@@8#.2O2$$qqqqqq3,,,,",
"5XXXX-99;;9;*  &&> &&&&>..@@@82$##$$33qqqqq3,,,,",
"1XXXX6ooo99;*  &* &&&&&&>.4.8@888.2$$$33Oqqq3,,,,",
"%XXXX6ooo9;*&&&&&&>&& & @4%8888@88O$$$3qOq33,,,,",
"eX5+5677;;;**&&&&1>&&&& >.4.888@88O$$$qqOq333,,e",
"eX5++6*07***& &**  &&&  @#.28@@2$$$$$qqqq33,,,e",
"e%55X5*00<&&&**&*&&&&&&& >>.4%8@2$$$$$qOqqq3,,,e",
"e%::::5& >  &*&&&**&&& >www#%8#$$$$$$OOqqq3,,,,e",
"e1:::::1>>>>>> &0000<<www<w2.#.2$$$$OOOq33,,3,,e",
"ee:::::%1111>%1<0<<0<w@w<<@88282$$$$$qq333,33,ee",
"ee%::::::%11>11><<<<wwww<w@2228#$##OOqq33333,,ee",
"eee::::::4%%11%%1>11>%@w@..O228##22OOq333,,3,,ee",
"eee.44::444%%%%%%1%1>@@@8..22222222OO$33,,,,,eee",
"eeee4444444444%11%1>www@%8282..8882$$$3,,,,,eeee",
"eeee$4444.44444%1@wwwww@%828..2222O$$333,,,,eeee",
"eeeee#4444%%444%%@@@w@8@882..####$$$3333,,,eeeee",
"eeeeee.4444%%444....8@@8@28###$$$$$$333,,,eeeeee",
"eeeeeee##444.%44%......8888##$$$$$$$333,,eeeeeee",
"eeeeeeee###44..%%.......8@82#$#O$$$q33,,eeeeeeee",
"eeeeeeeee####44.........22.2##O$$$$33,,eeeeeeeee",
"eeeeeeeeeeO.###4..8@8..OO##222q$$$q33,eeeeeeeeee",
"eeeeeeeeeeee.#####8@8..O#####OOOqq33,eeeeeeeeeee",
"eeeeeeeeeeeee2####2@@82$$$$$$$$qq3,eeeeeeeeeeeee",
"eeeeeeeeeeeeeeeeO##22222#$$$$qq33eeeeeeeeeeeeeee",
"eeeeeeeeeeeeeeeeeeeeqqOO882q33eeeeeeeeeeeeeeeeeee"
};



/* XPM */
const char *texticon[] = {
/* columns rows colors chars-per-pixel */
"48 48 128 2",
"   c #c1ad96",
".  c #b3a28e",
"X  c #cdc4ba",
"o  c #c6b9ac",
"O  c #aaaaaa",
"+  c #645a4f",
"@  c #dddddd",
"#  c #151613",
"$  c #696054",
"%  c #d3d2d3",
"&  c #4b4c4b",
"*  c #747474",
"=  c #f6f6f6",
"-  c #faf9f9",
";  c #49433a",
":  c #9a9a9a",
">  c #948473",
",  c #c9b59c",
"<  c Gray64",
"1  c #6e6e6d",
"2  c Gray70",
"3  c #bcbcbc",
"4  c #434342",
"5  c #cccccb",
"6  c #959595",
"7  c #597046",
"8  c Gray52",
"9  c #38322b",
"0  c Gray99",
"q  c #5c5248",
"w  c #c6b299",
"e  c Gray95",
"r  c #e4e4e3",
"t  c #a89884",
"y  c #7d7e7d",
"u  c #555555",
"i  c #9a8b79",
"p  c #a39481",
"a  c #eeeeee",
"s  c Gray77",
"d  c #3d3e3d",
"f  c #7d7162",
"g  c #bfac95",
"h  c #74695b",
"j  c #8b7d6c",
"k  c #c1af99",
"l  c #ac9c88",
"z  c #544c42",
"x  c #7a6e60",
"c  c #a3b3a2",
"v  c #d2cac1",
"b  c #087106",
"n  c #c3b098",
"m  c #807464",
"M  c #8b8b8b",
"N  c #c4b29c",
"B  c #b6a692",
"V  c #bda991",
"C  c #232422",
"Z  c #323231",
"A  c #c6b49e",
"S  c #c1c1c0",
"D  c #bead99",
"F  c #94897c",
"G  c #bdab94",
"H  c #443e36",
"J  c #807a72",
"K  c #2d2924",
"L  c #322d27",
"P  c #093809",
"I  c #2c2c2a",
"U  c #e9e9e9",
"Y  c #786c5e",
"T  c #3c3935",
"R  c #626261",
"E  c #877a6a",
"W  c #25201d",
"Q  c #6f6457",
"!  c #a1927f",
"~  c #c2b19c",
"^  c #c3ae96",
"/  c #726658",
"(  c #2b2621",
")  c #d7d5d3",
"_  c #968776",
"`  c #9e9488",
"'  c #40403f",
"]  c #9e8f7d",
"[  c #c6b097",
"{  c #005000",
"}  c #c1ac93",
"|  c #dad8d6",
" . c #988a78",
".. c #beb7b1",
"X. c #bfa991",
"o. c #bbab96",
"O. c #272826",
"+. c #8f8171",
"@. c #8b7f70",
"#. c #1b1c1a",
"$. c #3b362f",
"%. c #1f211f",
"&. c #baa892",
"*. c #cebba3",
"=. c #6c6862",
"-. c #847767",
";. c #dedcd9",
":. c #363933",
">. c #ececeb",
",. c #e6e6e5",
"<. c #2c7f24",
"1. c #a0988f",
"2. c #aa9284",
"3. c #a9a197",
"4. c #908677",
"5. c #b09f8b",
"6. c #fbfbfb",
"7. c #b9a58e",
"8. c #dae1da",
"9. c #c0b2a2",
"0. c #484847",
"q. c #c7c6c6",
"w. c #b7afa5",
"e. c #737875",
"r. c #777777",
"t. c #7f7e73",
"y. c Gray0",
"u. c None",
/* pixels */
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
": :.& 0.0.0.0.0.& & & 0.0.0.0.0.0.0.& & & & & & & & & & & & & 0.' a u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"1 6 = = = 0 u.u.u.u.u.u.u.u.- = = = - u.u.u.u.u.u.u.u.u.6.- - >.O.% u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"R O u.u.u.5 O : 6 6 6 6 6 : a u.u.u.e : 6 6 6 6 6 : O 5 u.u.u.- :.5 u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u 2 u.@ & u 6 < O O O O O & < u.u.u.q.O.< O O O O < 6 =.0.s u.= d s u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"& s r Z M 0 u.u.u.u.u.u.u.8 < a ,.= s & u.u.u.u.u.u.u.u.2 0.s u.d 3 u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"4 % =.8 u.u.u.u.u.u.u.a 5 1 =.@.> ` 4.J ;.- u.u.u.u.u.u.u.5 :.@ 0.2 u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"d 6 0.- u.u.u.u.u.6.q.+ z z z /  .! ! _ E F ..;.- u.u.u.u.u.y 1 u 2 u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"Z ' M u.u.u.u.8.c t.q H 9 L W 9 Y _ t l i E -.E ` ) = u.u.u.,.0.4 O u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"#.d 8 2 3 | 2 :.$.H H K # %.#.C $.Q > t t ] E -.j i B w.) 6.u.& C O u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"I Z y I %.I #.# W ( W L =.1 : U 4 L Q +.i _ > i p 5.V 7.l v 0 8 %.2 u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"e e u.@ u #.#.P b 7 q q 1.y < u.,.I L $ m j l   n g   n n } v e - 0 u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.= q.y { b <.> h R u : - u.r 0.9 $ f l N w n     k [ ^ v - u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.- 4 { b 7 2.Q I d =.O a 5 W q Y ! k w n   g g n w X.o = u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.% # b 7 > t H I C O.' R O.z h p ~ A k   g g   w [ 7.X u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.r.L Q f l _ y y Z O.C O.Z ; x . *.*.n g     n w   7.e u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.U L q Y   .. s u.@ O 1 Z O.%.O.' $ 4.B , w n n w [ X.X u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.8 9 / _   ..u.u.u.% I K 9 C #.#.%.:.$  .7.w , , [ 7.r u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.>.T q +.D , v = u.% $.#.H + q H C # #.C :.+ @.7., w 9.= u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.2 L Q ] D n 9.r q.( y.L $ m  . .$ $.#.# #.C Z $ i 5.X u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.0 y L + j l g &.F z 9 T + h E t N k ] Q T #.# #.O.T q 3 a u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.0 : L q f p g &.i Q q + $ x > 5.N *.*.n _ z I #.# #.d y e u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.r.L z h @.B . f $ $ $ Q -.! B N A A , w . +.z C & e.e u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.8 u T z $ E ] E Y Q Q / f > t o.N A N n n , , l 4.3.0 u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.r.O % K ; + x -.m f Y Y m j  .5.D A A N   g ^ w , V ;.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.r.: u.3 ( ; + / x f m f f E +.! . k A A n   g   n } 9.- u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.r.: u.u.3 H $.z $ h Y x x m E > p B ~ A A n   g   ^ X.| u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.r.: u.u.u.3 O.K H q Q h h Y f E _ t B ~ A N k g g   ^ ~ = u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.r.: u.u.u.5 4 : & K ; + Q / h Y E i l o.N A N k g g n ^ v u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.r.: u.u.u.S 0.u.= O K K z + $ Q h -.i l o.N A N   G   [   >.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.r.: u.u.u.S 4 6.u.u.@ 4 #.H q + $ Q m i l o.N N n g G k ^ o 0 u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.r.: u.u.u.S 4 - u.u.u.e 8 # K ; z q + m _ 5.D N N k G G n } X u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.r.: u.u.u.S 4 - u.u.u.u.0 2 T # 9 ; z + f _ l D N ~ g G g ^ V % u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.r.: u.u.u.S 4 - u.u.u.u.u.u.e M # #.$.; q Y +.l k ~   g g g   V a u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.* : u.u.u.S ' - u.u.u.u.u.u.u.u.% R # ( $.q Y _ B N A , , k   } X 0 u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.R O u.u.u.5 Z = u.u.u.u.u.u.u.u.u.0 s I # 9 q f i t ! ] ! B G , [ | u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.0 d s u.u.u.r O.a u.u.u.u.u.u.u.u.u.u.u.U 1 # 9 L #.# y.y.y.# ( ; m D = ",
"u.u.u.u.u.u.u.u.u.u.u.u.u.3 C e u.u.u.0 & 2 u.u.u.u.u.u.u.u.u.u.u.u.e O.y.y.y.y.y.y.y.y.y.y.y.( S ",
"u.u.u.u.u.u.u.0 ) q.3 8 ' 2 u.u.u.u.u.| u 8 3 q.) u.u.u.u.u.u.u.u.u.;.#.y.y.y.y.y.y.y.y.y.y.y.y.T ",
"u.u.u.u.u.u.u.;.# T ' u 2 ,.@ @ @ @ @ r s R 4 T # S u.u.u.u.u.u.u.u.u.) Z y.y.y.y.y.y.y.y.y.y.Z q.",
"u.u.u.u.u.u.u.U 1 1 * y * 1 1 1 1 1 1 1 * y * 1 R % u.u.u.u.u.u.u.u.u.u.e O 1 Z # # # d * S = u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u."
};

/* XPM */
const char *texticon_sel[] = {
/* columns rows colors chars-per-pixel */
"48 48 128 2",
"   c #5a534a",
".  c Gray35",
"X  c #6a6153",
"o  c Gray77",
"O  c #49443a",
"+  c #acacac",
"@  c #9c8d7b",
"#  c Gray71",
"$  c #645b4f",
"%  c #938573",
"&  c #b3a28d",
"*  c Gray42",
"=  c #cfbaa2",
"-  c #897d6c",
";  c #979797",
":  c #a59582",
">  c #554e44",
",  c #aa9a86",
"<  c #cacaca",
"1  c Gray70",
"2  c Gray72",
"3  c #36312b",
"4  c #e9e9e9",
"5  c #7d7162",
"6  c #bfac95",
"7  c #bebebe",
"8  c #c9b69e",
"9  c #727272",
"0  c #817464",
"q  c Gray55",
"w  c #0b4d09",
"e  c #b5a490",
"r  c #057905",
"t  c #2d2e2d",
"y  c #75695b",
"u  c #1a1c1b",
"i  c Gray23",
"p  c #a4a4a4",
"a  c Gray26",
"s  c gainsboro",
"d  c #c5b39d",
"f  c #0b0b0a",
"g  c #141413",
"h  c Gray47",
"j  c Gray38",
"k  c #222422",
"l  c Gray29",
"z  c #a0917e",
"x  c #2e2a25",
"c  c #c2af98",
"v  c #433e37",
"b  c #bca991",
"n  c #ad9d88",
"m  c #3d3a34",
"M  c #c1af99",
"N  c #312d27",
"B  c #716c64",
"V  c #092708",
"C  c #25221e",
"Z  c #71675a",
"A  c #bcaa94",
"S  c #2e312a",
"D  c #d5d5d5",
"F  c #333332",
"G  c #c4b098",
"H  c #c6b49e",
"J  c #908272",
"K  c Gray1",
"L  c #c2ae97",
"P  c #c6b098",
"I  c Gray50",
"U  c #786d5e",
"Y  c #bead98",
"T  c #baa995",
"R  c #86796a",
"E  c #c0ac95",
"W  c #c8b299",
"Q  c #c0ae96",
"!  c #c5b29b",
"~  c #c0ad96",
"^  c #c6b39b",
"/  c #9e8f7d",
"(  c #9a8b78",
")  c #c5af97",
"_  c Gray51",
"`  c #cdb69d",
"'  c #50483e",
"]  c #c2b19c",
"[  c #201e1b",
"{  c #cab49b",
"}  c #2c2622",
"|  c #c6b199",
" . c #c4b19a",
".. c #171918",
"X. c #262725",
"o. c #c5b098",
"O. c #1d1a16",
"+. c #0b120c",
"@. c #12100f",
"#. c #8e806f",
"$. c #171615",
"%. c #202220",
"&. c #282a28",
"*. c #0e0e0e",
"=. c #837767",
"-. c #080706",
";. c #080908",
":. c #978676",
">. c #988977",
",. c #a0a0a0",
"<. c #060606",
"1. c #957e73",
"2. c #4b813b",
"3. c #b8a793",
"4. c #514a42",
"5. c #2f4a24",
"6. c #bfb2a3",
"7. c #6e6456",
"8. c #a89885",
"9. c #7e7b79",
"0. c #676867",
"q. c #bca792",
"w. c #817f65",
"e. c #373835",
"r. c #3d362f",
"t. c #8c7e6e",
"y. c #b0a08b",
"u. c None",
/* pixels */
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"0.< 1 2 2 2 2 2 # 1 1 # 2 2 2 2 2 # # # # # # # # # 1 1 1 1 1 < 9 u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"; 0.f f ;.<.u.u.u.u.u.u.u.u.-.;.;.;.K u.u.u.u.u.u.u.u.K <.<.<.. p u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"; . u.u.u...l 0.* * * * *   u.u.u.u.g j * * * * * j i f u.u.u.l p u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"p l u.[ ; + q I h h h h h D t u.u.u.j 7 h h h h h I q ,.I ;.u.a + u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"1 i *.o q f u.u.u.u.u.u.u.+ &.K <.u.  * u.u.u.u.u.u.u.g ; ; K e.# K u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"2 F ; _ u.u.u.u.u.u.u.u.K g 9.4.  Z > Z > K u.u.u.u.u.u.u.u.q h &.7 <.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"o 9 < f u.u.u.u.u.u.g v > O   0 z : ( 0 X v u <.u.u.u.u.u.u.*.o j 7 ;.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"< # 9 u.u.u.u.<.+.x   ' r.3 x ' R / , : % - t.7.x <.u.u.u.u.q + 2 f u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"s 2 t u.K u.f S O 4.' N g @.<.O.  =.z n 8.J R - % 5 m $.K u.l o < *.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"4 2 %.%.X.&.[ C } } } C [ h &.@.C > =.@ / >.#.J / & 6 , y f K s 4 *.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"l a K f k k [ V w O r.O.u.+ k u.g C > 5 - J , b A A c | o.#.@.X.a K u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.<.g u w r 2.1.O X.,.X.u.K ..C 4.Z t.3.^ W Q ~ c G ` J ;.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.K V r r w.:.v F &.[ f u.@.r.$ 5 , !  .c 6 6 ~ | { :.@.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.*.w r 7.z - %.t S F k u 3   0 , !  .L 6 6 6 G | P % f u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.<.+.5.Z R n B e.S &.&.&.S O Z z 8 8 G ~ 6 ~ L | ) ) v u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.g 3 7.5 : 6.[ g k S &.X.X.S > J 3.8 8 c Q ~ G | o.( <.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.K C   y / T g u.u.;.e.i &.%.u k F   - e | { | | o.P 4.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.f r.7.z 8 U f u.u.a   [ O m X.....%.e.$ - & { ` W b u u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.K [ > #.A ` #.u u.F m @.' Z U $ m %...u k F X @ b ` J K u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.;.x $ J y.8 y.  F [ -.v X 0 z e : Z 3 u ..u X.m $ ( N u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.<.x > 5 z Y 6 % 7.' ' $ y - , ] = Y J   S ....u &.e.t g u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.-.C 4.y :.3.M ( 7.  $ X 5 >.& ] 8 = 8 e 5 O %.g &.0.F u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.. 3 ' 7.=.y.: 5 X X X Z - : T d H ^ ! { G : $ O * &.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.+ t ..' $ 0 J R U Z Z U 0 % n Y H H  .Q Q ^ ` b & v u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.K 1 u u.C ' $ U 0 5 5 U 5 =.t.@ y.Y H H  .Q 6 L | | / f u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.K 1 t u.<.} O   7.U 5 5 5 0 - % z e ] H d  .~ 6 L c | > u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.K 1 i u.u.K [ 3 > X y U U U 0 R % 8.3.] H d c 6 6 L P y.f u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.K 1 i u.u.u.u a } O   7.Z y U 5 - ( n T d H ! M 6 E c {   u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.K 1 i u.u.u.. . u.$.N ' $ 7.7.Z U t./ y.Y d H ! ~ 6 ~ | c u u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.K 1 i u.u.u.0.9 u.u.f C r.> $ X 7.U t.z & Y H d G 6 6 c { R u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.K 1 i u.u.u.0.I u.u.u.<.$.x O   $ $ Z - z y.M d ! M 6 ~ | W x u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.K 1 i u.u.u.0.I u.u.u.u.K f [ r.O >   X R / e ] d ] ~ A ~ | n $.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.K 1 i u.u.u.0.I u.u.u.u.u.u.<.$.C r.O > X =./ e ] ! L A A ~ W : -.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.K 1 i u.u.u.0.I u.u.u.u.u.u.u.K @.O.x m O $ 0 ( T ] M ~ 6 6 E { > u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.K # i u.u.u.0._ u.u.u.u.u.u.u.u.u.;.g [ 3 v $ =.8.Y d H 8 ^ E ) & @.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.;.< t u.u.u.j q u.u.u.u.u.u.u.u.u.<.@.$.} v X % , 8.: : n A { = - K u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.$.s ..u.u.u.l p u.u.u.u.u.u.u.u.u.u.u.u.<.$.x v 3 C f -.<.;.C ' 5 q.N u.",
"u.u.u.u.u.u.u.u.u.u.u.u.. o K u.u.u.u D $.u.u.u.u.u.u.u.u.u.u.u.u.<.$.;.u.u.u.u.u.u.u.u.u.u.O.4.K ",
"u.u.u.u.u.u.u.u.*.g %.j o l u.u.u.u.u.9 + a ..g <.u.u.u.u.u.u.u.u.u.u.;.u.u.u.u.u.u.u.u.u.u.u.u.u.f ",
"u.u.u.u.u.u.u.F D 7 7 + a K f f f f f K . + 7 o 1 <.u.u.u.u.u.u.u.u.u.K f K u.u.u.u.u.u.u.u.u.K ;.<.",
"u.u.u.u.u.u.u.i s o 7 1 + 2 2 2 2 2 2 2 + # 7 < < ;.u.u.u.u.u.u.u.u.u.u.u.f *.-.K u.u.K -.*.f u.u.",
"u.u.u.u.u.u.u.K *.g g g $.$.$.$.$.$.$.$.$.g g g f u.u.u.u.u.u.u.u.u.u.u.u.u.u.<.-.<.<.-.K u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u."
};
/* XPM */
const char *twodicon[] = {
/* columns rows colors chars-per-pixel */
"48 48 128 2",
"   c #959292",
".  c #6d6a6a",
"X  c #797575",
"o  c #eae9e8",
"O  c #a4a1a1",
"+  c #c1bfbf",
"@  c #6a6767",
"#  c #bcbaba",
"$  c #070808",
"%  c #8c8989",
"&  c #555352",
"*  c #acaaa9",
"=  c #585555",
"-  c #9e9b9b",
";  c #b2afaf",
":  c #383736",
">  c #5c5959",
",  c #656161",
"<  c #a9a6a6",
"1  c #615e5d",
"2  c #222121",
"3  c #817e7e",
"4  c #121211",
"5  c #2a2929",
"6  c #e2e0e0",
"7  c #4c4a4a",
"8  c #e0dede",
"9  c #b8b5b5",
"0  c #262525",
"q  c #494646",
"w  c #878383",
"e  c #1d1d1d",
"r  c #3c3b3a",
"t  c #918d8e",
"y  c #c6c4c4",
"u  c #191918",
"i  c #aeacac",
"p  c #7c7979",
"a  c #898686",
"s  c #444242",
"d  c #2d2c2c",
"f  c #3f3d3c",
"g  c #747171",
"h  c #302e2e",
"j  c #151515",
"k  c #323130",
"l  c #514e4e",
"z  c #363434",
"x  c #4e4c4b",
"c  c #3f3e3d",
"v  c #3a3838",
"b  c #343232",
"n  c #434140",
"m  c #2c2b2b",
"M  c #e5e4e4",
"N  c #999696",
"B  c #848181",
"V  c #1c1b1b",
"C  c #c8c6c6",
"Z  c #e7e6e6",
"A  c #f4f3f3",
"S  c #474444",
"D  c #cbc9c9",
"F  c #fbfbfb",
"G  c #fafaf9",
"H  c #f0eeee",
"J  c #5f5c5b",
"K  c #dbd9d9",
"L  c #878585",
"P  c #736f6f",
"I  c #d7d5d5",
"U  c #42403f",
"Y  c #e5e3e3",
"T  c #7e7b7b",
"R  c #706d6d",
"E  c #625f5e",
"W  c Gray95",
"Q  c #575454",
"!  c #d0cdcd",
"~  c #b7b4b4",
"^  c #dcdada",
"/  c #535150",
"(  c #d2d0cf",
")  c #7b7878",
"_  c #cccaca",
"`  c #353433",
"'  c #272626",
"]  c #f2f1f0",
"[  c #757272",
"{  c #676463",
"}  c #b5b3b3",
"|  c #cecccb",
" . c #dedcdb",
".. c #242423",
"X. c #d5d3d3",
"o. c #31302f",
"O. c #cac8c7",
"+. c #52504f",
"@. c #777373",
"#. c #5f5c5c",
"$. c #aba8a8",
"%. c #d8d6d6",
"&. c #b3b0b0",
"*. c #7f7c7c",
"=. c #4b4848",
"-. c #9b9797",
";. c #4a4847",
":. c #464443",
">. c #d3d1d1",
",. c #dad8d7",
"<. c #636060",
"1. c #939090",
"2. c #999695",
"3. c #eeeded",
"4. c #5a5756",
"5. c #ecebeb",
"6. c #838080",
"7. c #6f6c6c",
"8. c #989494",
"9. c #8b8888",
"0. c #bab8b7",
"q. c #a19e9e",
"w. c #5b5958",
"e. c #f7f6f6",
"r. c #8e8b8b",
"t. c #1f1f1e",
"y. c #716e6e",
"u. c None",
/* pixels */
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.9 p = s v 7 J L y u.u.u.u.u.u.u.( > q & 1 . g g y.{ 1 > x q q & { t C u.u.u.u.u.u.u.",
"u.u.u.u.O k 4 : @ 3 9.X <.m 4 v C u.u.u.u.u.& : p . E & q q =.> { R p 3 *.. Q ..4 h O u.u.u.u.u.",
"u.u.u.P j / C W _ < O } 6 u.&.;.u 3 u.u.u.u.S & D | ! G u.u.u.u.; 8.a   O D u.u.i > t.x 8 u.u.u.",
"u.u.r.$ O u.a m 4 u e j 4 & O.u.P $ i u.u.u.O e 4 4 4 5 ! u.] x 4 ' 5 ..V 4 e , X.u.; ..r A u.u.",
"u.~ $   u.: ..T y Y M K $.S $   u.P o.F u.u.u.Y _ _ C S & u.; $ # H A 5.M C   U 4 = u.Z d 7 u.u.",
"u.+.h u.7 ` K u.u.u.u.u.u.u.X 2 ( H ..w u.u.u.u.u.u.u.B : u.N u 5.u.u.u.u.u.u.u.  ' J u.+ $ L u.",
"H 5 1.u.$ s o u.u.u.u.u.u.u.u.q E u.Q ` u.u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.I u y.u.R ..W ",
"6 ' K F <.0 +.G u.u.u.u.u.u.u.w o.u.@.5 e.u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.u.[ j Y W 5 < ",
"6 ' I u.u.D ' Y u.u.u.u.u.u.u.  0 u.p 5 A u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.u.^ u   u./ <.",
"A m #.u.u.9 ' o u.u.u.u.u.u.u.u.{ s u.P m G u.u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.u.u.u.u.u.4.4.u.3 s ",
"u.L 4 4.{ 5 { u.u.u.u.u.u.u.u.D ..-.u.q U u.u.u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.u.u.u.u.u.L s u.i ` ",
"u.u.~ 4.+.% u.u.u.u.u.u.u.6 #.4 & u. .u < u.u.u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.u.u.u.u.* : F %.5 ",
"u.u.u.u.u.u.u.u.u.u.Z -.7 2 ` *.u.o s S u.u.u.u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.u.u.u.u.+ o.] M ' ",
"u.u.u.u.u.u.u.u.X.) ..j c 2.u.u.; 5 2 >.u.u.u.u.u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.u.u.u.u.C h H Z 5 ",
"u.u.u.u.u.%.S j h a  .u.u.% ` 4 #.,.u.u.u.u.u.u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.u.u.u.u.u.y h H Z 5 ",
"u.u.u.u.X ..c   u.u.< & 5 ' +.i u.u.u.u.u.u.u.u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.u.u.u.u.~ b A 6 ' ",
"u.u.u.E $ ) u.A a 5 4 : 3 3.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.u.u.u.u.- r u._ 5 ",
"u.G 1 ..| K Q 2 r p C u.u.u.u.i :.: E X.u.u.u.u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.u.u.u.u.[ 7 u.- r ",
"u.% $ # 9 ' r O u.u.u.u.u.u.u.I t.c > ..5 Z u.u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.u.u.u.u.c . u.. x ",
"W ..@ ( 4 7 R 4.{ C u.u.u.u.B c u.u.Y t.X u.u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.u.u.u.} $ # u.U X ",
"q.d u.1 $ 2 z n ` 2 s O.u.u.r.o.+   O.*.S u.u.u.u.u.u.u.B : u.N V M u.u.u.u.u.u.u.u.u.u.u.;.d u.+ 2 D ",
"J 4.u.` h } u.u.u.+ : 2 $.u.o k 4 $ E ; r G u.u.u.u.u.u.w z u.2.e Z u.u.u.u.u.u.u.u.u.O 4 ; u.: f u.",
"r T F 7.N p . t Z u.u.. u J ^ u.>.q , i c G u.u.u.u.u.u.T r u.- j o u.u.u.u.u.u.u.o . u r.u.w 4 9 u.",
"z w u.r.4 5 h u 5 O u.u.i 2 t.X @ u + T q u.u.+ < D I f <.u.C 4 X C K 8 ( i p d u w u.# 4 t u.u.",
"n g u., $ B F ^ P 4 v M u.M L U q O H e T u.1.4 4 4 4 h ,.u.u.N t.4 4 4 4 4 k 6.] u.) $ . u.u.u.",
"@.r u.u.& z u.u.u.; f d 8.u.u.u.u.u.u.x e 8 u.:.@ A C # u.u.u.u.u.F | # 9 + M u.3.  s 2 @.u.u.u.u.",
"I u > 0.n : u.u.u.u.u.+.$ 7 N + O =.V &.u.u.= : X [ @ J 7 q 7 w.<.. X ) X @ l e u U ! u.u.u.u.u.",
"u.* c o.U ~ u.u.u.u.u.u.u.+ > v h : = ! u.u.u.^ { / Q > <.R g R , J 4.& / & = @ -  .u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u."
};

/* XPM */
const char *twodicon_sel[] = {
/* columns rows colors chars-per-pixel */
"48 48 128 2",
"   c #868282",
".  c #b8b6b6",
"X  c #757272",
"o  c #918d8e",
"O  c #adaaaa",
"+  c #7d7a7a",
"@  c #a29f9f",
"#  c #8a8787",
"$  c #cdcaca",
"%  c #605d5d",
"&  c #bebcbc",
"*  c #6a6666",
"=  c #a8a5a5",
"-  c #dad8d8",
";  c #555252",
":  c #989595",
">  c #3f3d3c",
",  c #595656",
"<  c #eae9e9",
"1  c #514e4d",
"2  c #d3d0d0",
"3  c #4d4a4a",
"4  c #5c5a59",
"5  c #222121",
"6  c #6e6b6b",
"7  c #e3e1e1",
"8  c #a4a1a1",
"9  c #969393",
"0  c #9d9999",
"q  c #b1aeae",
"w  c #0f0f0e",
"e  c #2d2c2c",
"r  c #151515",
"t  c #181818",
"y  c #b3b1b1",
"u  c #9f9b9b",
"i  c #111111",
"p  c #939091",
"a  c #484645",
"s  c #b5b2b2",
"d  c #1d1d1d",
"f  c #040404",
"g  c #272626",
"h  c #535050",
"j  c #636060",
"k  c #787474",
"l  c #c9c7c7",
"z  c #343232",
"x  c #575454",
"c  c #393736",
"v  c #2f2e2d",
"b  c #434140",
"n  c #323130",
"m  c #2c2b2a",
"M  c #2a2929",
"N  c #3a3838",
"B  c Gray3",
"V  c #454242",
"C  c #3f3e3d",
"Z  c #3c3a39",
"A  c #a3a0a0",
"S  c #4e4c4b",
"D  c #d0cdcd",
"F  c #dddbdb",
"G  c #42403f",
"H  c #8e8b8b",
"J  c #e0dedd",
"K  c #201f1f",
"L  c #736f6f",
"P  c #5f5c5c",
"I  c #7a7777",
"U  c #c3c1c1",
"Y  c #4b4948",
"T  c #6c6969",
"R  c #373535",
"E  c #cecccc",
"W  c #706d6c",
"Q  c #bab8b8",
"!  c #676463",
"~  c #dedddd",
"^  c #c7c5c5",
"/  c #afacac",
"(  c #e8e6e6",
")  c #1b1b1a",
"_  c #6b6868",
"`  c #bdbbbb",
"'  c #838080",
"]  c #d6d4d4",
"[  c #0b0b0b",
"{  c #c2c0bf",
"}  c #31302f",
"|  c #4b4847",
" . c #a6a4a3",
".. c #7b7878",
"X. c #474544",
"o. c #797676",
"O. c #c0bebe",
"+. c #292827",
"@. c #fafaf9",
"#. c #5a5857",
"$. c #d5d3d3",
"%. c #e5e3e3",
"&. c #edebeb",
"*. c #20201f",
"=. c #8b8787",
"-. c #cac8c8",
";. c #b7b5b4",
":. c #c5c4c3",
">. c #c4c2c2",
",. c #8f8c8b",
"<. c #f0efef",
"1. c #e1e0df",
"2. c #7f7c7c",
"3. c #282727",
"4. c #716e6e",
"5. c #878585",
"6. c #5b5958",
"7. c #716e6d",
"8. c #efeded",
"9. c #4f4c4c",
"0. c #83807f",
"q. c #d1cfcf",
"w. c #807d7d",
"e. c #d8d5d5",
"r. c #e6e5e5",
"t. c #353333",
"y. c #464443",
"u. c None",
/* pixels */
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.m h k o u   W Y 5 u.u.u.u.u.u.u.) 4.=...6 j #.6.4 * 6 L ' =.# I ! V 5 u.u.u.u.u.u.u.",
"u.u.u.u.R = 7 A j 1 a x _ s 1.u 5 u.u.u.u.u.I A ; % T + =.=.5.4.! P ; 1 h j k U 7 / c u.u.u.u.u.",
"u.u.u.4 - + 5 f K t.R e i u.v # 2 1 u.u.u.u.H I K d d f u.u.u.u.v G | G c K u.u.n L $ ' r u.u.u.",
"u.u.X.( R u.| s J 2 E - J ..*.u.4 &.} u.u.u.R $ F F F . d u.f ' F & . U q.~ $ * t u.v O.u f u.u.",
"u.m <.G u.8 U h 5 i w r z H ( G u.4 O u.u.u.u.i K K 5 H I u.v &.+.B f [ w 5 G p %.k u.w y ' u.u.",
"u.2./ u.  = r u.u.u.u.u.u.u.x ^ ) B U Y u.u.u.u.u.u.u.S A u.> 2 [ u.u.u.u.u.u.u.G & 7.u.g <.Y u.",
"B Q b u.< o [ u.u.u.u.u.u.u.u.# T u.o.= u.u.u.u.u.u.u.S 8 u.> D w u.u.u.u.u.u.u.u.t e.P u.P O.f ",
"i ` r u._ & 2.f u.u.u.u.u.u.u.3 O u., . f u.u.u.u.u.u.S A u.> D w u.u.u.u.u.u.u.u.u., - i f ;.t.",
"i & t u.u.K & i u.u.u.u.u.u.u.G & u.h . f u.u.u.u.u.u.S A u.> D w u.u.u.u.u.u.u.u.u.r ] G u.+ _ ",
"f s 6 u.u.m & [ u.u.u.u.u.u.u.! o u.4 s f u.u.u.u.u.u.S A u.> D w u.u.u.u.u.u.u.u.u.u.X X u.1 o ",
"u.Y J X * . ! u.u.u.u.u.u.u.K { > u.# 9 u.u.u.u.u.u.u.S A u.> D w u.u.u.u.u.u.u.u.u.u.Y o u.} = ",
"u.u.m X + a u.u.u.u.u.u.i 6 F I u.r 2 t.u.u.u.u.u.u.u.S A u.> D w u.u.u.u.u.u.u.u.u.u.n A u.t Q ",
"u.u.u.u.u.u.u.u.u.w >   :.= 1 u.[ o H u.u.u.u.u.u.u.u.S A u.> D w u.u.u.u.u.u.u.u.u.u.g O f w ` ",
"u.u.u.u.u.u.u.t ; U - : C u.u.v . ^ ) u.u.u.u.u.u.u.u.S A u.> D w u.u.u.u.u.u.u.u.u.u.5 q B w Q ",
"u.u.u.u.u.t H - q | r u.u.X.= 7 6 t u.u.u.u.u.u.u.u.u.S A u.> D w u.u.u.u.u.u.u.u.u.u.5 q B w Q ",
"u.u.u.u.x { 9 G u.u.t...Q ` + n u.u.u.u.u.u.u.u.u.u.u.S A u.> D w u.u.u.u.u.u.u.u.u.u.m = f i ` ",
"u.u.u.T ( ; u.f | . ~ @ 1 B u.u.u.u.u.u.u.u.u.u.u.u.u.S A u.> D w u.u.u.u.u.u.u.u.u.u.Z 0 u.K . ",
"u.f 6 O.d r o.^ 0 ; 5 u.u.u.u.} ,.8 T ) u.u.u.u.u.u.u.S A u.> D w u.u.u.u.u.u.u.u.u.u., ' u.Z u ",
"u.a @.+.m & u c u.u.u.u.u.u.t -.9 L U Q w u.u.u.u.u.u.S A u.> D w u.u.u.u.u.u.u.u.u.u.: % u.% 0.",
"f { j ) J ' P X ! 5 u.u.u.u.S 9 u.u.i l x u.u.u.u.u.u.S A u.> D w u.u.u.u.u.u.u.u.u.e 8.3.u.p ; ",
"N y u.6 @.>. .p = ^ o *.u.u.y.O g G *.h H u.u.u.u.u.u.S A u.> D w u.u.u.u.u.u.u.u.u.# y u.g l K ",
"W X u.= q e u.u.u.g 8 ^ z u.[ O F &.T v 0 f u.u.u.u.u.3 8 u.C E w u.u.u.u.u.u.u.u.c 7 v u.@ : u.",
"0 h u.P > ; % V w u.u.% 2 W r u.) =.* } : f u.u.u.u.u.h 0 u.Z - [ u.u.u.u.u.u.[ % $.X.u.3 1.M u.",
" .Y u.X.%.;.q $.Q c u.u.} :.$ , j $.g h # u.u.g t.K t : * u.5 %.x 5 r r ) n h y ] Y u.+.~ V u.u.",
"p #.u.* @.S u.r 4 r.u w u.w Y p =.R B E h u.b J F F ~ / t u.u.> -.F F F ~ J O 9.f u.; < j u.u.u.",
", 0 u.u.I  .u.u.u.v : y C u.u.u.u.u.' E i u.,.j f 5 3.u.u.u.u.u.u.d +.m g w u.B G o >., u.u.u.u.",
"t ] L M p @ u.u.u.u.u.+ (   > g c 5.D v u.u.k 8 , , j W   =.  L * % x ; ; j w.E ] p d u.u.u.u.u.",
"u.n : O p m u.u.u.u.u.u.g L u / @ k d u.u.u.r ! + o.L _ P #.P * W X + + + k j Z r u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.",
"u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u.u."
};

/* XPM */
const char * threedicon[] = {
```

```
/* columns rows colors chars-per-pixel */
"48 48 16 1",
"  c #a2a5a5",
". c #f3f2f2",
"X c #cccbc9",
"o c #5d5257",
"O c #904857",
"+ c #b3b2af",
"@ c #8e9393",
"# c #c3c4bf",
"$ c #b5838c",
"% c #dad7d7",
"& c #697377",
"* c #bcbcb8",
"= c #dac0c5",
"- c #7f8184",
"; c #e4e4e4",
": c None",
/* pixels */
"::::::::::::::::::::::::::::::::::::::::::::::::",
"::::::::::::::::::::::::::::::::::::::::::::::::",
"::::::::::::::::::::::::::::::::::::::::::::::::",
"::::::::::::::::::::::::::::::::::::::::::::::::",
"::::::::::;+ @@+%:::::::::::::::::::::::::::::::",
"::::::::.#-&--&o&-X:::::::::::::::::::::::::::::",
":::::::;@ &&--o--&& .:::::::::::::::::::::::::::",
":::::::o@    --&&&--%:::::::::::::::::::::::::::",
"::::::+o   +++*+@&@  -.:::::::::::::::::::::::::",
"::::::@@    +++**     @:::::::::::::::::::::::::",
"::::::+    +*+++++*+   &#:::::::::::::::::::::::",
"::::::.# #..:;#++++++@--&.::.;::::::::::::::::::",
"::::::::;::::::X++++@-&&%::%& .:::::::::::::::::",
"::::::::::::::::*+++ o X*::;&&&*.:::::::::::::::",
"::::::::::. @;::#++++++:;*::;&&&&-%:::::::::::::",
":::::.::::%+@&@%#**++.:X%::.-&&&&o-;::::::::::::",
"::X@-@@%:.**+ &@*****::+:::;*@&&-ooo .::::::::::",
":@&@@@- ;%+**++*****%;@;::.###+@-oooo-#.::::::::",
"#&++*+@-%;*+******#+$@;:::.X####+&ooo-@@%:::::::",
" @++++@@X:.#+**##***%:::::: +####*@oo@ @-+.:::::",
"* ++++@@%:::#**%:X&.:::::%&####*## @@@@@@@;::::",
"X+++++@@%:::.***%#&X:::::-@#********+ @@  -%:::",
"; +++*+@+:::.****+@@:::::;o #****++++++   + -.::",
":++++*   +.:.****%#-.::::@@###*# @;*++++   +@&.:",
":;+++*   @-@*****;.@;:::%-X###**-;:.#+++   @&-+:",
"::* ++*+ @-@*****;: %:::-*X####  :::%++  @-@-&.",
":::  +++*++******.:*:::X %#####-.::::#++++@-&OX",
":::.@-+*+*******;:.%:::@%X###X  :::::;++++ OOo+",
"::::.@& +*******.:.:::X+%#####-;::::::.*++++-&- ",
"::::::#&o-   $$$%::::@%####X  :::::::.***++  +@",
":::::::.+OOOOOO$.::.% +%#####-.:::::::.*****+ @ ",
"::::::::.;====.:.@--&X####X *::::::::.*****+@ +",
"::::::::::::::::*++ @#####*@::::::::;***** @@*",
":::::::::::::::*&+#**#####*#::::::::X###***++%",
":::::::::::::::&o #****###%*;:::::::;#####*##*.",
"::::::::::::::::.* +******X+@- ;.::;#######*  X:",
":::::::::::::::::;+ +****# @@@-o--@*#######*++.:",
"::::::::::::::::::::X  ****#+@-oo&@#X#####*+ %::",
":::::::::::::::::::::.+@+***##*++*X######*@@ :::",
":::::::::::::::::::::::%@@+***###########$O&;:::",
"::::::::::::::::::::::::.+- +##*######X*$OO=::::",
":::::::::::::::::::::::::::;#-- *###X#*$OOO%::::",
":::::::::::::::::::::::::::::; oo-$ $$OOO$.:::::",
":::::::::::::::::::::::::::::::.X $OO$$=;:::::::",
"::::::::::::::::::::::::::::::::::::::::::::::::",
"::::::::::::::::::::::::::::::::::::::::::::::::",
"::::::::::::::::::::::::::::::::::::::::::::::::",
"::::::::::::::::::::::::::::::::::::::::::::::::"
};

/* XPM */
const char *threedicon_sel[] = {
/* columns rows colors chars-per-pixel */
"48 48 64 1",
"  c #9b495a",
". c Gray99",
"X c #728b91",
"o c #abaaa6",
"O c #dbabb3",
"+ c #4e4f4f",
"@ c #c1c2bd",
"# c #92a9b0",
"$ c #9b9996",
"% c #0d0c0c",
"& c #b9bab5",
"* c #bdbdb8",
"= c #2d2d2c",
"- c #ac7881",
"; c #b2b2ae",
": c #c4c5c0",
"> c #b09598",
", c #7d7679",
"< c #b5b4b0",
"1 c #b8a4a8",
"2 c #dbdbd8",
"3 c #848687",
"4 c #6a6968",
"5 c #82989d",
"6 c #637e84",
"7 c #6b323e",
"8 c #c9cac5",
"9 c #89a0a5",
"0 c #5c7276",
"q c #b9b6b3",
"w c #7a414d",
"e c #403a3c",
"r c #ffccdd",
"t c #69757a",
"y c #c5c8c2",
"u c #bfbfba",
"i c #b7b8b3",
"p c #a2a19e",
"a c #8f8d8a",
"s c #eae9e9",
"d c #a1a9ae",
"f c #7b5960",
"g c #c7c6c2",
"h c #706b6b",
"j c #979391",
"k c #cecdc9",
"l c #83807f",
"z c #afafaa",
"x c #261317",
"c c #727f83",
"v c #828e91",
"b c #7a6c70",
"n c #a3b4b5",
"m c #72716f",
"M c Gray10",
"N c #36191f",
"B c #b4aab6",
"V c #fd87a5",
"C c #5a5758",
"Z c #91b5bc",
"A c #484645",
"S c #918a8e",
"D c #616162",
"F c None",
/* pixels */
"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFM=+CA=MFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFM6X6XX60,+=FFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFF%vZ66X6Dcv6t+%FFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFF+9d$pd93vt6t5,MFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFF=mpoooz<&zvt5#dl%FFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFCpoooz;;<iio##d#4FFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFF4<ooo$pqi<i&<dZ#3=FFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFMmz,M%%ej*qqqn5XXDFFF%MFFFFFFFFFFFFFFFFFFF",
"FFFFFFFF=%FFFF%,&<<<9X,,MFF3cAMFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFF%;i<<pf12=Fe266te%FFFFFFFFFFFFFFFF",
"FFFFFFFFFFM4AMFFj&<<zq.s=Fks660t4=FFFFFFFFFFFFFF",
"FFFFFFFFFFpu$he=p*&qq..2M%.sX00t,DCMFFFFFFFFFFFF",
"FF=e+,b=FM@*;pmju**&u..jFFs2&960,CADA%FFFFFFFFFF",
"FAlj$jadMm@ii;<*&*uu2s$%FF;k@g<v,CeeCce%FFFFFFFF",
"=lz<&oajAA&*i&&q&*@q>h%FFFMp:@::obee+X5mMFFFFFFF",
"Dpzz<za$eFM$u&&g@&<C=%FFFFFC*@@@:*S+CX99vA%FFFFF",
"azoz<;jpeFF%$u&2.kD%FFFFFFMly@@@uu:ov595954MFFFF",
"liz;;ij$AFFF=:*&2:,MFFFFFF+$yuuuuu***;9999zSMFFF",
"eiz;<uqpX%FFM*u*&;$+FFFFFMho:uuu*iu&<<zd9dB#c%FF",
"%oq;<&p#n0%F=@u**2:,%FFFFApg@uu@z+A;&;zzod#Z9CFF",
"FA*;<qop#9cel@**&2.aMFFFMak:@@u@l%FM3izoodZXcv=F",
"FFl;<q&;p5,j@***&2.$MFFFCg8::@:;eFFFFD<ooo$c5XD%",
"FF%3zqqqq<<*&&*&*..$FFFMo2::::83%FFFFFl<zoo9c,f=",
"FFFM3j;&qq&ii&&&2..AFFFm28yg:8<eFFFFFF=&;zzpfwfe",
"FFFF%Alp<**&&uuOs.:FFFM@2:ygg8l%FFFFFFMii<<;,ha+",
"FFFFFF=Dh->11>-Vrg%FFFm2g:yykzeFFFFFFFMi*&iipp;m",
"FFFFFFF%=7w    -f%FM=A<2@::yg,%FFFFFFFM&*&&&zpp4",
"FFFFFFFFF%NNNx%FMl3S,8:@@:8z=FFFFFFFFFe:u**&o$oC",
"FFFFFFFFFFFFFFFF1O1pjg@@@:@mFFFFFFFFFFD8@u*uoj$A",
"FFFFFFFFFFFFFFFF=,1guu@@@@:kAFFFFFFFFFzy@@uui<u=",
"FFFFFFFFFFFFFFFFDho@uuuu@uy2vMFFFFFFF+k::@@u@8;%",
"FFFFFFFFFFFFFFFF%Cpq*u*uuu8n5X+MFFF%C8y::@:ipzAF",
"FFFFFFFFFFFFFFFFFF=lo&uu**:ovv$3Ae+a88::@@@&<z%F",
"FFFFFFFFFFFFFFFFFFF%A$;*u**uq$,fwh$yy:::@@ioqAFF",
"FFFFFFFFFFFFFFFFFFFFFM4p;uu*u:*;zu8y:::@:*jj,FFF",
"FFFFFFFFFFFFFFFFFFFFFFF=mjquuuu::::::::y:-fbMFFF",
"FFFFFFFFFFFFFFFFFFFFFFFF%el;q@@u@@@::y8*-  xFFFF",
"FFFFFFFFFFFFFFFFFFFFFFFFFFMCljo*:yy8yu>   xFFFFF",
"FFFFFFFFFFFFFFFFFFFFFFFFFFFFM=+f->>>-   7%FFFFFF",
"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF%%N7ww77N%FFFFFFFF",
"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
};

/* XPM */
const char *joyicon[] = {
/* columns rows colors chars-per-pixel */
"48 48 64 1",
"  c #e7e4e5",
". c #8a898f",
"X c #4d4c5c",
"o c #7b7a8a",
"O c #1c1b2b",
"+ c #680a0f",
"@ c #575564",
"# c #cfd8ef",
"$ c #282535",
"% c #6b6a7c",
"& c #383648",
"* c #8c0101",
"= c #dbdbde",
"- c #bdbcc2",
"; c #dca9a8",
": c #c4c8da",
"> c #2b1625",
", c #fefefe",
"< c #c6c7ca",
"1 c #f3f2f4",
"2 c #ececee",
"3 c #0c0916",
"4 c #151322",
"5 c #cc0505",
"6 c #801115",
"7 c #a4a4aa",
"8 c #f8f8f8",
"9 c #a50000",
"0 c #ea1515",
"q c #dee6f8",
"w c #171625",
"e c #ecf2ff",
"r c #ababb2",
"t c #932c2d",
"y c #727180",
"u c #f5e9e9",
"i c #a25757",
"p c #413f4f",
"a c #2e2b3c",
"s c #656372",
"d c #95959e",
"f c #5e5d6e",
"g c #1b1928",
"h c #f62020",
"j c #222030",
"k c #41111c",
"l c #333142",
"z c #161b2b",
"x c #12101e",
"c c #191727",
"v c #ac97a6",
"b c #fbfbfb",
"n c #f9fafe",
"m c #73515a",
"M c #eacac9",
"N c #b1b0bc",
"B c #bea6a8",
"V c #ac8a8a",
"C c #bcc0d4",
"Z c #d0d0d6",
"A c #1f1d2d",
"S c #ff4545",
"D c #b93939",
"F c None",
/* pixels */
"FFFFFFFFFFFFFF i6t;,FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFF *9959;FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFF,i95h555uFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFF1690Sh00;FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFF8t950hh5MFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFB+9555DbFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFF,V+**tuFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFF, Bv=FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFF8:qbFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFF,:q1FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFF=#2FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFF2:q,FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFFbCq8FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFFZ#1FFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFF #2FFFFFFF81bFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFF8Cq,FFb -.paX<,FFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFF,:q=-.X>+***kg7,FFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFb <dqoxcz+999**>4d,FFFFFFFFFFFFFF",
"FFFFFFFFFFF,2<.X$4a#rcOzk*****>OcobFFFFFFFFFFFFF",
"FFFFFFF,2<d@$c4ccw4N#$4cc>+++kAAO4y8FFFFFFFFFFFF",
"FFF,2<dmkk>4www4443oe@3wwwzzwwgOOO4s8FFFFFFFFFFF",
"Fb7Xj>*999*kwcccccxped344cggccccOOgxs1FFFFFFFFFF",
",s3x4k99***+gcwwwwxgZZc3xwcgcggggOOg4X2FFFFFFFFF",
"734w4k*****+zc44xxx3dn&cAOwcccccggOOgxp FFFFFFFF",
"XA4wxxk+66kOOOAgwAg4lfc4gjAwccccccggOAO4a FFFFFFF",
"pAO4cxxx44wwgOAAA$j4c4334l&OccccwccgOAAcaZFFFFFF",
"7xjwcgcc4444wcOOAjAg$l&jcp@lOccwwwccgOAjg$=FFFFF",
"1jAA4OOOOgc444wgOAjApdNo&pfpAOOOgcwccOOAjA$<FFFF",
"Fr3jwcOOOOOgc44wgOAjpNq7@pljAjjjjjAOOOOAj$jj-,FF",
"Fb&cAwOAOOOOgcwwwc$lps%p$AOAAOOOOAAAAjjjj$$$$7,F",
"FF<3jgcAOOOOOggcccl&l&OcOgcwwwcgOAAAAAjj$$$$$aNF",
"FF,@4jwOAOggggggc$p$&&cw4wwcgOOOOOOAAAjjjAgwwlp=",
"FFF=3jOcAOgcccccc&aO&j4cggggggggggOAOOg4xxwA$lpy",
"FFF,oxjcOAOccccwj&Oa&Oggggcwccccggw4x33xgja&p@@X",
"FFFF2wAOcAOcwwwwaaA&$cccwwwwwwww4x333xwA$lpXsyof@",
"FFFFFdxjgOAOwwwgljlacww44444xx33334A$lpXfyooyf$-",
"FFFFF1AOAgAOc4w>>O$>>444xx333334A$lpXfyooosXaxo,",
"FFFFFFr3jOOAO44>>>>>4xx333334AalpXf%ooo%@l44fZ,F",
"FFFFFF8&cjgAAc4444x33333xcja&pXf%oooy@pOxp7 ,FFF",
"FFFFFFF<3jOOAO4x333334O$&pXXf%oooyfpjxa.=bFFFFFF",
"FFFFFFF,@4jOAOx33xg$lpXX@f%yooyfX$x$.Z8FFFFFFFFF",
"FFFFFFFF=3AAO$&&apX@ffs%yyoysXawOs-1FFFFFFFFFFFF",
"FFFFFFFF,yxjj@-Ny%%%%yyy%sXlcxXN2FFFFFFFFFFFFFFF",
"FFFFFFFFF2xgA@:Co%%%%%fX&gxp7 ,FFFFFFFFFFFFFFFFF",
"FFFFFFFFFFd3Olfsf%sfX&j4l.ZbFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFF8jxjjapX&jxAy<8FFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFF=g3cAA4gs-2FFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFF2.l$@72,FFFFFFFFFFFFFFFFFFFFFFFFFFFF"
};

/* XPM */
const char *joyicon_sel[] = {
/* columns rows colors chars-per-pixel */
"48 48 64 1",
"  c #373546",
". c #eaf2ff",
"X c #4a4758",
"o c #2e1725",
"O c #8c0102",
"+ c #afafc0",
"@ c #585567",
"# c #ce0606",
"$ c #f01919",
"% c #71070b",
"& c #d0d7ed",
"* c #aea2b4",
"= c #151322",
"- c #4e0c14",
"; c #c8c8d7",
": c #a60000",
"> c #222132",
", c #020103",
"< c #1a1828",
"1 c #7b7a8c",
"2 c #6b6a7c",
"3 c #181625",
"4 c #1f1d2d",
"5 c #2f090f",
"6 c #737284",
"7 c #3f3d4e",
"8 c #858b99",
"9 c #0a0812",
"0 c #626173",
"q c #12111e",
"w c #9595a5",
"e c #5f5e71",
"r c #312f40",
"t c #1c1826",
"y c #e1e9fd",
"u c #1b1a2a",
"i c #24212e",
"p c #f92f2f",
"a c #0f0d19",
"s c #b7bed0",
"d c #2b293b",
"f c #272434",
"g c #1e1c2b",
"h c #706f81",
"j c #6e6d7f",
"k c #151a2a",
"l c #2d2835",
"z c #1f2030",
"x c #5c5b6c",
"c c #111626",
"v c #69687a",
"b c #1a1c2c",
"n c #282638",
"m c #160000",
"M c #4f4d5f",
"N c #1c1b2b",
"B c #161824",
"V c #181727",
"C c #777688",
"Z c #7e7d8f",
"A c #62616b",
"S c #797889",
"D c #666578",
"F c None",
/* pixels */
"FFFFFFFFFFFFFm-%%5,FFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFmO::##-FFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFF,-:#p###mFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFF,%:$p$$$5FFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFF,%:#$$p#mFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFmO:###O,FFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFF,m%OO%mFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFF,,5*AFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFF,ss,FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFF,0.aFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFi.7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFF9&8,FFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFF,8&,FFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFF7.iFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFFqyxFFFFFFF,,,FFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFF,s+,FF,,9at4B,,FFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFFFF,vyB,aBo%OO%-z9,FFFFFFFFFFFFFFFF",
"FFFFFFFFFFFFFFF,,,XyS=>z%:::OOoz9,FFFFFFFFFFFFFFF",
"FFFFFFFFFFF,,9aB4b &+<ub-OOOOOo4>a,FFFFFFFFFFFFFF",
"FFFFFFF,,9a=g44guV=+&l=VVo-%%ob44>q,FFFFFFFFFFFFF",
"FFF,,,95--okV33===aS.@a33ckkcc<Nggzq,FFFFFFFFFFFF",
"F,9qkoO:::O-cV<<<<q7.89==V<<<VV<ugu4=,FFFFFFFFFFF",
",q33c-::OOO%<V3333qu;;3aq3<<<<<<<<ugu43,FFFFFFFFF",
"9u=3c5OOOOO-kV==qqa9w. <4N3<<VV<<uugu4u,FFFFFFFF",
"=>=3qq5%%%-Nbgg<3g<= x<=tig3<<VVV<uNgNg4,FFFFFFF",
"a>N=Vqqq=ccV<u444n>=V=99=r uVVVV3V<<N444i,FFFFFF",
",N>3Vu<V====3<ug4z4ufr i37@rb<V3333<<N44ii,FFFFF",
",a>4=NNNu<V===3<N4>47w+C Xx74ggN<V33<ug4iin,FFFF",
"F,<>3VggNNu<V==3<Nz>X+y*@7r>zz>>zz4NNuggiifl9,FF",
"F,azz3N4NNuuu<333<n 7Dv7f4N44NNNNgg44zziifflr9,F",
"FF,3><<4guuuu<<<VV 7r bVNuV333<<ugg444iifflnl aF",
"FF,94z3g4N<<<<<<<Vf7f  <3=33<<uuuNNggg4iii4t33 X,",
"FFF,=>N<4N<<VV<<< rN >=V<uuuuu<<<uuggut=qq3gfrMg",
"FFF,,N>Vg4uVVVV3> ud N<<<<V33VV<<<3=qaaqt>d 7@@i",
"FFFF,q>g<4g<3333rd47fV<V3333333=qa9aa34nr7M0hCe=",
"FFFFF,u><N4u333trzrrt33=====qaa99a=4nr7MehSShe ,",
"FFFFF,az4<4gV=3ttgloo===qaa999a=gnr7Meh1ZCDMrt,,",
"FFFFFF,V>Nu4N==ooooo=qaa999a=4d 7Mxj1Z12@ 4a,,,F",
"FFFFFF,94z<44V====qa9999q3id 7Mx2SZ1hx7fq9,,,FFF",
"FFFFFFF,=>gN4N=aa999aqgl 7XMx2CZZ6eXd=9,,,FFFFFF",
"FFFFFFF,9N>N4Na99qtfr7XM@xv6116eXr<9,,,FFFFFFFFFF",
"FFFFFFFF,qz4gn7 r7M@xe0vhCCh0M 4a,,,FFFFFFFFFFFFF",
"FFFFFFFF,,uzz@s+hvvv2j66j0M >q9,,FFFFFFFFFFFFFFFF",
"FFFFFFFFF,a44@;;Z2jjjveM f=9,,,FFFFFFFFFFFFFFFFFF",
"FFFFFFFFFF,=grx0ev0xM7n39,,,FFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFF,9<>>d7X7d<a9,,FFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFF,qV<4>Nq9,,FFFFFFFFFFFFFFFFFFFFFFFFFFF",
"FFFFFFFFFFFF,9==9,,,FFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
};
```

demo/main.cpp

```
#include "frame.h"
#include "graph.h"
#include "display.h"
#include "icons.h"
```

```
#include "textdrawing/textedit.h"
#include "textdrawing/helpwindow.h"
#include "dnd/dnd.h"
#include "i18n/i18n.h"

#include <qmodules.h>

#if defined(QT_MODULE_OPENGL)
#include "opengl/glworkspace.h"
#include "opengl/gllandscapeviewer.h"
#include "opengl/glinfotext.h"
#endif

#if defined(QT_MODULE_CANVAS)
#include "qasteroids/toplevel.h"
#endif

#if defined(QT_MODULE_TABLE)
#include "widgets/widgetsbase.h"
#else
#include "widgets/widgetsbase_pro.h"
#endif

#include <stdlib.h>

#include <qapplication.h>
#include <qimage.h>

#include <qtabwidget.h>
#include <qfont.h>
#include <qworkspace.h>
#include <qwidgetstack.h>

#if defined(QT_MODULE_SQL)
#include <qsqldatabase.h>
#include "sql/sqlex.h"
#endif

#if defined(Q_OS_MACX)
#include <stdlib.h>
#include <qdir.h>
#endif

#include "categoryinterface.h"

static void qdemo_set_caption( CategoryInterface *c, int i )
{
   QWidget *w = qApp->mainWidget();
   if ( !w )
     return;
   QString title = Frame::tr( "Qt Demo Collection" );
   title += " - " + c->categoryName( i - c->categoryOffset() );
   w->setCaption( title );
}
```

```
class WidgetCategory : public CategoryInterface
{
public:
    WidgetCategory( QWidgetStack *s ) : CategoryInterface( s ), created( FALSE ) {}

    QString name() const { return "Widgets"; }
    QIconSet icon() const { return QPixmap( widgeticon ); }
    int numCategories() const { return 2; }
    QString categoryName( int i ) const {
      switch ( i ) {
      case 0:
          return Frame::tr( "Widgets" );
          break;
      case 1:
          return Frame::tr( "Drag and Drop" );
          break;
      }
      return QString::null;
    }
    QIconSet categoryIcon( int ) const { return QIconSet(); }
    void setCurrentCategory( int i ) {
      create();
      stack->raiseWidget( i );
      qdemo_set_caption( this, i );
    }
    void create() {
      if ( created )
          return;
      created = TRUE;
      stack->addWidget( new WidgetsBase( stack ), categoryOffset() + 0 );
      stack->addWidget( new DnDDemo( stack ), categoryOffset() + 1 );
    }

    int categoryOffset() const { return 0; }

private:
    bool created;

};

#if defined(QT_MODULE_SQL)
class DatabaseCategory : public CategoryInterface
{
public:
    DatabaseCategory( QWidgetStack *s ) : CategoryInterface( s ), created( FALSE ) {}

    QString name() const { return "Database"; }
    QIconSet icon() const { return QPixmap( dbicon ); }
    int numCategories() const { return 1; }
    QString categoryName( int i ) const {
      switch ( i ) {
      case 0:
          return Frame::tr( "SQL Explorer" );
```

```
     break;
   }
   return QString::null;
  }
  QIconSet categoryIcon( int ) const { return QIconSet(); }
  void setCurrentCategory( int i ) {
   create();
   stack->raiseWidget( i );
   qdemo_set_caption( this, i );
  }
  void create() {
   if ( created )
     return;
   created = TRUE;
   stack->addWidget( new SqlEx( stack ), categoryOffset() + 0 );
  }

  int categoryOffset() const { return 10; }

private:
  bool created;

};
#endif

#if defined(QT_MODULE_CANVAS)
class CanvasCategory : public CategoryInterface
{
public:
  CanvasCategory( QWidgetStack *s ) : CategoryInterface( s ), created( FALSE ) {}

  QString name() const { return "2D Graphics"; }
  QIconSet icon() const { return QPixmap( twodicon ); }
  int numCategories() const { return 2; }
  QString categoryName( int i ) const {
   switch ( i ) {
   case 0:
     return Frame::tr( "Graph Drawing" );
     break;
   case 1:
     return Frame::tr( "Display" );
     break;
   }
   return QString::null;
  }
  QIconSet categoryIcon( int ) const { return QIconSet(); }
  void setCurrentCategory( int i ) {
   create();
   stack->raiseWidget( i );
   qdemo_set_caption( this, i );
  }
  void create() {
   if ( created )
     return;
     created = TRUE;
     stack->addWidget( new GraphWidget( stack ), categoryOffset() + 0 );
     stack->addWidget( new DisplayWidget( stack ), categoryOffset() + 1 );
    }

    int categoryOffset() const { return 100; }

private:
    bool created;

};
#endif

#if defined(QT_MODULE_OPENGL)
class OpenGLCategory : public CategoryInterface
{
public:
    OpenGLCategory( QWidgetStack *s ) : CategoryInterface( s ), created( FALSE ) {}

    QString name() const { return "3D Graphics"; }
    QIconSet icon() const { return QPixmap( threedicon ); }
    int numCategories() const { return 3; }
    QString categoryName( int i ) const {
     switch ( i ) {
     case 0:
         return Frame::tr( "3D Demo" );
         break;
     case 1:
         return Frame::tr( "Fractal landscape" );
         break;
     case 2:
         return Frame::tr( "OpenGL info" );
         break;
     }
     return QString::null;
    }
    QIconSet categoryIcon( int ) const { return QIconSet(); }
    void setCurrentCategory( int i ) {
     create();
     stack->raiseWidget( i );
     qdemo_set_caption( this, i );
    }
    void create() {
     if ( created )
         return;
     created = TRUE;
     stack->addWidget( new GLWorkspace( stack ), categoryOffset() + 0 );
     stack->addWidget( new GLLandscapeViewer( stack ), categoryOffset() + 1 );
     stack->addWidget( new GLInfoText( stack ), categoryOffset() + 2 );
    }
    int categoryOffset() const { return 1000; }

private:
    bool created;
```

```
};
#endif

class TextCategory : public CategoryInterface
{
public:
    TextCategory( QWidgetStack *s ) : CategoryInterface( s ), created( FALSE ) {}

    QString name() const { return "Text Drawing/Editing"; }
    QIconSet icon() const { return QPixmap( texticon ); }
    int numCategories() const { return 2; }
    QString categoryName( int i ) const {
     switch ( i ) {
     case 0:
         return Frame::tr( "Richtext Editor" );
         break;
     case 1:
         return Frame::tr( "Help Browser" );
         break;
     }
     return QString::null;
    }
    QIconSet categoryIcon( int ) const { return QIconSet(); }
    void setCurrentCategory( int i ) {
     create();
     stack->raiseWidget( i );
     qdemo_set_caption( this, i );
    }
    void create() {
     if ( created )
         return;
     created = TRUE;
     TextEdit *te = new TextEdit( stack );
     te->load( "textdrawing/example.html" );
     stack->addWidget( te, categoryOffset() + 0 );
     QString home = QDir( "../../doc/html/index.html" ).absPath();
     HelpWindow *w = new HelpWindow( home, ".", stack, "helpviewer" );
     stack->addWidget( w, categoryOffset() + 1 );
    }
    int categoryOffset() const { return 10000; }

private:
    bool created;

};

class I18NCategory : public CategoryInterface
{
public:
    I18NCategory( QWidgetStack *s ) : CategoryInterface( s ), created( FALSE ) {}

    QString name() const { return "Internationalization"; }
    QIconSet icon() const { return QPixmap( internicon ); }
    int numCategories() const { return 1; }
    QString categoryName( int i ) const {
     switch ( i ) {
     case 0:
         return Frame::tr( "Internationalization" );
         break;
     }
     return QString::null;
    }
    QIconSet categoryIcon( int ) const { return QIconSet(); }
    void setCurrentCategory( int i ) {
     create();
     stack->raiseWidget( i );
     qdemo_set_caption( this, i );
    }
    void create() {
     if ( created )
         return;
     created = TRUE;
     stack->addWidget( new I18nDemo( stack ), categoryOffset() + 0 );
    }
    int categoryOffset() const { return 100000; }

private:
    bool created;

};

#if defined(QT_MODULE_CANVAS)
class GameCategory : public CategoryInterface
{
public:
    GameCategory( QWidgetStack *s ) : CategoryInterface( s ), created( FALSE ) {}

    QString name() const { return "Game"; }
    QIconSet icon() const { return QPixmap( joyicon ); }
    int numCategories() const { return 1; }
    QString categoryName( int i ) const {
     switch ( i ) {
     case 0:
         return Frame::tr( "Asteroids" );
         break;
     }
     return QString::null;
    }
    QIconSet categoryIcon( int ) const { return QIconSet(); }
    void setCurrentCategory( int i ) {
     create();
     stack->raiseWidget( i );
     qdemo_set_caption( this, i );
    }
    void create() {
     if ( created )
         return;
     created = TRUE;
     stack->addWidget( new KAstTopLevel( stack ), categoryOffset() + 0 );
    }
    int categoryOffset() const { return 1000000; }

private:
    bool created;

};
#endif

int main( int argc, char **argv )
{
    QString category;
    QApplication a( argc, argv );

    Frame::updateTranslators();
    Frame frame;
    a.setMainWidget( &frame );

    QPtrList<CategoryInterface> categories;
    categories.append( new WidgetCategory( frame.widgetStack() ) );
#if defined(QT_MODULE_SQL)
    categories.append( new DatabaseCategory( frame.widgetStack() ) );
#endif
    categories.append( new CanvasCategory( frame.widgetStack() ) );
#if defined(QT_MODULE_OPENGL)
    categories.append( new OpenGLCategory( frame.widgetStack() ) );
#endif
    categories.append( new TextCategory( frame.widgetStack() ) );
    categories.append( new I18NCategory( frame.widgetStack() ) );
    categories.append( new GameCategory( frame.widgetStack() ) );
    frame.setCategories( categories );

    frame.resize( 1000, 700 );
    frame.show();

    return a.exec();
}
```

demo/qthumbwheel.cpp

```
#include "qthumbwheel.h"

#ifndef QT_NO_THUMBWHEEL
#include <qpainter.h>
#include <qdrawutil.h>
#include <qpixmap.h>
#include <math.h>

static const double m_pi = 3.14159265358979323846;
static const double rad_factor = 180.0 / m_pi;

QThumbWheel::QThumbWheel( QWidget *parent, const char *name )
    : QFrame( parent, name )
```

```
{
   orient = Vertical;
   init();
}

/*!
  Destructs the wheel.
*/

QThumbWheel::~QThumbWheel()
{
}

/*!
  \internal
 */

void QThumbWheel::init()
{
   track = TRUE;
   mousePressed = FALSE;
   pressedAt = -1;
   rat = 1.0;
   setFrameStyle( WinPanel | Sunken );
   setMargin( 2 );
   setFocusPolicy( WheelFocus );
}

void QThumbWheel::setOrientation( Orientation orientation )
{
   orient = orientation;
   update();
}

void QThumbWheel::setTracking( bool enable )
{
   track = enable;
}

void QThumbWheel::setTransmissionRatio( double r )
{
   rat = r;
}

/*!
  Makes QRangeControl::setValue() available as a slot.
*/

void QThumbWheel::setValue( int value )
{
   QRangeControl::setValue( value );
}

void QThumbWheel::valueChange()
```

```
{
    repaint( FALSE );
    emit valueChanged(value());
}

void QThumbWheel::rangeChange()
{
}

void QThumbWheel::stepChange()
{
}

/*!
  \reimp
*/

void QThumbWheel::keyPressEvent( QKeyEvent *e )
{
    switch ( e->key() ) {
    case Key_Left:
      if ( orient == Horizontal )
          subtractLine();
      break;
    case Key_Right:
      if ( orient == Horizontal )
          addLine();
      break;
    case Key_Up:
      if ( orient == Vertical )
          subtractLine();
      break;
    case Key_Down:
      if ( orient == Vertical )
          addLine();
      break;
    case Key_PageUp:
      subtractPage();
      break;
    case Key_PageDown:
      addPage();
      break;
    case Key_Home:
      setValue( minValue() );
      break;
    case Key_End:
      setValue( maxValue() );
      break;
    default:
      e->ignore();
      return;
    };
}
```

```
/*!
  \reimp
*/

void QThumbWheel::mousePressEvent( QMouseEvent *e )
{
    if ( e->button() == LeftButton ) {
      mousePressed = TRUE;
      pressedAt = valueFromPosition( e->pos() );
    }
}

/*!
  \reimp
*/

void QThumbWheel::mouseReleaseEvent( QMouseEvent *e )
{
    int movedTo = valueFromPosition( e->pos() );
    setValue( value() + movedTo - pressedAt );
    pressedAt = movedTo;
}

/*!
  \reimp
*/

void QThumbWheel::mouseMoveEvent( QMouseEvent *e )
{
    if ( !mousePressed )
      return;
    if ( track ) {
      int movedTo = valueFromPosition( e->pos() );
      setValue( value() + movedTo - pressedAt );
      pressedAt = movedTo;
    }
}

/*!
  \reimp
*/

void QThumbWheel::wheelEvent( QWheelEvent *e )
{
    int step = ( e->state() & ControlButton ) ? lineStep() : pageStep();
    setValue( value() - e->delta()*step/120 );
    e->accept();
}

/*!\reimp
*/

void QThumbWheel::focusInEvent( QFocusEvent *e )
{
    QWidget::focusInEvent( e );
}

/*!\reimp
*/

void QThumbWheel::focusOutEvent( QFocusEvent *e )
{
    QWidget::focusOutEvent( e );
}

void QThumbWheel::drawContents( QPainter *p )
{
    QRect cr = contentsRect();
    // double buffer
    QPixmap pix( width(), height() );
    QPainter pt( &pix );
    QBrush brush = backgroundPixmap() ?
            QBrush( backgroundColor(), *backgroundPixmap() ) : QBrush( backgroundColor() );
    pt.fillRect( cr, brush );

    const int n = 17;
    const double delta = m_pi / double(n);
    // ### use positionFromValue() with rad*16 or similar
    double alpha = 2*m_pi*double(value()-minValue())/
               double(maxValue()-minValue())*transmissionRatio();
    alpha = fmod(alpha, delta);
    QPen pen0( colorGroup().midlight() );
    QPen pen1( colorGroup().dark() );

    if ( orient == Horizontal ) {
     double r = 0.5*cr.width();
     int y0 = cr.y()+1;
     int y1 = cr.bottom()-1;
     for ( int i = 0; i < n; i++ ) {
         int x = cr.x() + int((1-cos(delta*double(i)+alpha))*r);
         pt.setPen( pen0 );
         pt.drawLine( x, y0, x, y1 );
         pt.setPen( pen1 );
         pt.drawLine( x+1, y0, x+1, y1 );
     }
    } else {
     // vertical orientation
     double r = 0.5*cr.height();
     int x0 = cr.x()+1;
     int x1 = cr.right()-1;
     for ( int i = 0; i < n; i++ ) {
         int y = cr.y() + int((1-cos(delta*double(i)+alpha))*r);
         pt.setPen( pen0 );
         pt.drawLine( x0, y, x1, y );
         pt.setPen( pen1 );
         pt.drawLine( x0, y+1, x1, y+1 );
     }
    }
```

```
   qDrawShadePanel( &pt, cr, colorGroup());

   pt.end();
   p->drawPixmap( 0, 0, pix );
}

int QThumbWheel::valueFromPosition( const QPoint &p )
{
   QRect wrec = contentsRect();
   int pos, min, max;
   if ( orient == Horizontal ) {
    pos = p.x();
    min = wrec.left();
    max = wrec.right();
   } else {
    pos = p.y();
    min = wrec.top();
    max = wrec.bottom();
   }
   double alpha;
   if ( pos < min )
    alpha = 0;
   else if ( pos > max )
    alpha = m_pi;
   else
    alpha = acos( 1.0 - 2.0*double(pos-min)/double(max-min) );// ### taylor
   double deg = alpha*rad_factor/transmissionRatio();
   // ### use valueFromPosition()
   return minValue() + int((maxValue()-minValue())*deg/360.0);
}

#endif
```

demo/qthumbwheel.h

```
#ifndef QTHUMBWHEEL_H
#define QTHUMBWHEEL_H

#ifndef QT_H
#include "qframe.h"
#include "qrangecontrol.h"
#endif // QT_H

#ifndef QT_NO_THUMBWHEEL

class QThumbWheel : public QFrame, public QRangeControl
{
   Q_OBJECT

public:
   QThumbWheel( QWidget *parent=0, const char *name=0 );
   ~QThumbWheel();

   virtual void      setOrientation( Orientation );
   Orientation       orientation() const;
```

```
    virtual voidsetTracking( bool enable );
    bool      tracking() const;
    virtual voidsetTransmissionRatio( double r );
    double        transmissionRatio() const;

public slots:
    virtual void setValue( int );

signals:
    void      valueChanged( int value );

protected:
    void      valueChange();
    void      rangeChange();
    void      stepChange();

    void      keyPressEvent( QKeyEvent * );
    void      mousePressEvent( QMouseEvent * );
    void      mouseReleaseEvent( QMouseEvent * );
    void      mouseMoveEvent( QMouseEvent * );
    void      wheelEvent( QWheelEvent * );
    voidfocusInEvent( QFocusEvent *e );
    voidfocusOutEvent( QFocusEvent *e );

    void      drawContents( QPainter * );

private:
    void      init();
    int       valueFromPosition( const QPoint & );


    double  rat;
    int       pressedAt;
    Orientation orient;
    uint track : 1;
    uint mousePressed : 1;

    class QThumbWheelPrivate;
    QThumbWheelPrivate *d;
};

inline QThumbWheel::Orientation QThumbWheel::orientation() const
{
    return orient;
}

inline bool QThumbWheel::tracking() const
{
    return (bool)track;
}

inline double QThumbWheel::transmissionRatio() const
{
    return rat;
```

```
}

#endif // QT_NO_WHEEL

#endif // QWHEEL_H
```

demo/dnd/dnd.cpp

```
#include <qiconview.h>
#include <qdragobject.h>
#include <qlayout.h>
#include <qmultilineedit.h>

#include "dnd.h"
#include "styledbutton.h"
#include "listview.h"
#include "iconview.h"

DnDDemo::DnDDemo( QWidget* parent, const char* name )
    : DnDDemoBase( parent, name )
{
    buttonPixmap1->setEditor( StyledButton::PixmapEditor );
    buttonPixmap2->setEditor( StyledButton::PixmapEditor );
    buttonPixmap3->setEditor( StyledButton::PixmapEditor );
    buttonPixmap4->setEditor( StyledButton::PixmapEditor );

    multiLine1->setTextFormat( RichText );
    multiLine1->setText( "<p><b>Faust</b> - <i>Goethe</i></p>"
                        "Habe nun, ach! Philosophie,<br>"
                 "Juristerei und Medizin,<br>"
                        "Und leider auch Theologie<br>"
                        "Durchaus studiert, mit heißem Bemühn.<br>"
                        "Da steh ich nun, ich armer Tor!<br>"
                        "Und bin so klug als wie zuvor;<br>"
                        "Heiße Magister, heiße Doktor gar<br>"
                        "Und ziehe schon an die zehen Jahr<br>"
                        "Herauf, herab und quer und krumm<br>"
                        "Meine Schüler an der Nase herum-<br>"
                        "Und sehe, daß wir nichts wissen können!<br>"
                        "Das will mir schier das Herz verbrennen.<br>"
                        "Zwar bin ich gescheiter als all die Laffen,<br>"
                        "Doktoren, Magister, Schreiber und Pfaffen;<br>"
                        "Mich plagen keine Skrupel noch Zweifel,<br>"
                        "Fürchte mich weder vor Hölle noch Teufel-<br>"
                        "Dafür ist mir auch alle Freud entrissen,<br>"
                        "Bilde mir nicht ein, was Rechts zu wissen,<br>"
                        "Bilde mir nicht ein, ich könnte was lehren,<br>"
                        "Die Menschen zu bessern und zu bekehren.<br>"
                        "Auch hab ich weder Gut noch Geld,<br>"
                        "Noch Ehr und Herrlichkeit der Welt;<br>"
                        "Es möchte kein Hund so länger leben!<br>"
                        "Drum hab ich mich der Magie ergeben,<br>"
                        "Ob mir durch Geistes Kraft und Mund<br>"
                        "Nicht manch Geheimnis würde kund;<br>"
                        "Daß ich nicht mehr mit saurem Schweiß<br>"
```

```
                    "Zu sagen brauche, was ich nicht weiß;<br>"
                    "Daß ich erkenne, was die Welt<br>"
                    "Im Innersten zusammenhält,<br>"
                    "Schau alle Wirkenskraft und Samen,<br>"
                    "Und tu nicht mehr in Worten kramen. <br>" );

    multiLine2->setTextFormat( RichText );
    multiLine2->setText( "<p><b>To Milton</b> - <i>Oscar Wilde</i></p>"
                    "Milton!  I think thy spirit hath passed away<br>"
                    "From these white cliffs and high-embattled towers;<br>"
                    "This gorgeous fiery-coloured world of ours<br>"
                    "Seems fallen into ashes dull and grey,<br>"
                    "And the age changed unto a mimic play<br>"
                    "Wherein we waste our else too-crowded hours:<br>"
                    "For all our pomp and pageantry and powers<br>"
                    "We are but fit to delve the common clay,<br>"
                    "Seeing this little isle on which we stand,<br>"
                    "This England, this sea-lion of the sea,<br>"
                    "By ignorant demagogues is held in fee,<br>"
                    "Who love her not:  Dear God! is this the land<br>"
                    "Which bare a triple empire in her hand<br>"
                    "When Cromwell spake the word Democracy!<br>" );


    items.insert( tr("copy"), IconItem( tr("Copy"), "editcopy.png" ) );
    items.insert( tr("cut"), IconItem( tr("Cut"), "editcut.png" ));
    items.insert( tr("paste"), IconItem( tr("Paste"), "editpaste.png" ));
    items.insert( tr("raise"), IconItem( tr("Raise"), "editraise.png" ));
    items.insert( tr("lower"), IconItem( tr("Lower"), "editlower.png" ));
    items.insert( tr("new"), IconItem( tr("New"), "filenew.png" ));
    items.insert( tr("load"), IconItem( tr("Load"), "fileopen.png" ));
    items.insert( tr("save"), IconItem( tr("Save"), "filesave.png" ));
    items.insert( tr("undo"), IconItem( tr("Undo"), "undo.png" ));
    items.insert( tr("redo"), IconItem( tr("Redo"), "redo.png" ));
    items.insert( tr("delete"), IconItem( tr("Delete"), "editdelete.png" ));
    items.insert( tr("help"), IconItem( tr("Help"), "help.png" ));
    items.insert( tr("home"), IconItem( tr("Home"), "home.png" ));

    listView->addColumn( tr("Actions"), 240 );
    listView->setColumnWidthMode( 0, QListView::Maximum );

    QMap<QString,IconItem>::Iterator it;
    for( it = items.begin(); it != items.end(); ++it ) {
        IconItem item = it.data();

        QIconViewItem *iitem = new IconViewItem( iconView, item.name(), *item.pixmap(), it.key() );
        iitem->setRenameEnabled( TRUE );
        QListViewItem *litem = new ListViewItem( listView, item.name(), it.key() );
        litem->setPixmap( 0, *item.pixmap() );
    }
}

DnDDemo::~DnDDemo()
{
}

IconItem::IconItem( const QString& name, const QString& icon )
{
    _name = name;
    _pixmap = loadPixmap( icon );
}

QPixmap IconItem::loadPixmap( const QString& name )
{
    QPixmap pix( "textdrawing/" + name );
    return pix;
}

IconItem DnDDemo::findItem( const QString& tag )
{
    return items[ tag ];
}
```

demo/dnd/dnd.h

```
#include <qpixmap.h>
#include <qmap.h>
#include "dndbase.h"

#ifndef DNDDEMO_H
#define DNDDEMO_H

class IconItem
{
public:
    IconItem( const QString& name = QString::null, const QString& icon = QString::null );

    QString name() { return _name; }
    QPixmap *pixmap() { return &_pixmap; }

    Q_DUMMY_COMPARISON_OPERATOR( IconItem )

protected:
    QPixmap loadPixmap( const QString& name );

private:
    QString _name;
    QPixmap _pixmap;
};

class DnDDemo : public DnDDemoBase
{
    Q_OBJECT

public:
    DnDDemo( QWidget* parent = 0, const char* name = 0 );
    ~DnDDemo();

    IconItem findItem( const QString& tag );
private:
    QMap<QString,IconItem> items;
};

#endif
```

demo/dnd/dndbase.h

```
#ifndef DNDDEMOBASE_H
#define DNDDEMOBASE_H

#include <qvariant.h>
#include <qpixmap.h>
#include <qwidget.h>

class QVBoxLayout;
class QHBoxLayout;
class QGridLayout;
class QSpacerItem;
class StyledButton;
class IconView;
class ListView;
class QTextEdit;

class DnDDemoBase : public QWidget
{
    Q_OBJECT

public:
    DnDDemoBase( QWidget* parent = 0, const char* name = 0, WFlags fl = 0 );
    ~DnDDemoBase();

    StyledButton* buttonColor1;
    StyledButton* buttonColor2;
    StyledButton* buttonColor3;
    StyledButton* buttonColor4;
    StyledButton* buttonPixmap1;
    StyledButton* buttonPixmap2;
    StyledButton* buttonPixmap3;
    StyledButton* buttonPixmap4;
    ListView* listView;
    IconView* iconView;
    QTextEdit* multiLine1;
    QTextEdit* multiLine2;

protected:
    QGridLayout* DnDDemoBaseLayout;
    QHBoxLayout* Layout5;
    QSpacerItem* Spacer1;

protected slots:
    virtual void languageChange();

    virtual void init();
    virtual void destroy();

private:
    QPixmap image0;
    QPixmap image1;
    QPixmap image2;
    QPixmap image3;
    QPixmap image4;
    QPixmap image5;
};

#endif // DNDDEMOBASE_H
```

demo/dnd/iconview.cpp

```
#include <qdragobject.h>

#include "dnd.h"
#include "iconview.h"

IconView::IconView( QWidget* parent, const char* name )
    : QIconView( parent, name )
{
    connect( this, SIGNAL(dropped(QDropEvent*, const QValueList<QIconDragItem>&)),
             SLOT(slotNewItem(QDropEvent*, const QValueList<QIconDragItem>&)));
}

IconView::~IconView()
{
}

QDragObject *IconView::dragObject()
{
    if ( !currentItem() ) return 0;

    QTextDrag * drg = new QTextDrag( ((IconViewItem*)currentItem())->tag(), this );
    drg->setSubtype("dragdemotag");
    drg->setPixmap( *currentItem()->pixmap() );

    return drg;
}

void IconView::slotNewItem( QDropEvent *e, const QValueList<QIconDragItem>& )
{
    QString tag;
    if ( !e->provides( "text/dragdemotag" ) ) return;

    if ( QTextDrag::decode( e, tag ) ) {
        IconItem item = ((DnDDemo*) parentWidget())->findItem( tag );
        IconViewItem *iitem = new IconViewItem( this, item.name(), *item.pixmap(), tag );
        iitem->setRenameEnabled( TRUE );
    }
    e->acceptAction();
}
```

demo/dnd/iconview.h

```
#include <qiconview.h>
#include <qstring.h>

#include "dnd.h"

class IconViewItem : public QIconViewItem
{
public:
   IconViewItem( QIconView * parent, const QString & text, const QPixmap & icon, const QString&
tag )
       : QIconViewItem( parent, text, icon ), _tag( tag ) {}
   virtual ~IconViewItem() {}

   QString tag() { return _tag; }

private:
   QString _tag;
};

class IconView : public QIconView
{
   Q_OBJECT

public:
   IconView( QWidget* parent = 0, const char* name = 0 );
   ~IconView();

   QDragObject *dragObject();

public slots:
   void slotNewItem( QDropEvent *t, const QValueList<QIconDragItem>& );
};
```

demo/dnd/listview.cpp

```
#include <qdragobject.h>
#include <qapplication.h>
#include "listview.h"
#include "dnd.h"

ListView::ListView( QWidget* parent, const char* name )
   : QListView( parent, name )
{
   setAcceptDrops( TRUE );
   setSorting( -1, FALSE );
   dragging = FALSE;
}

ListView::~ListView()
{

}

void ListView::dragEnterEvent( QDragEnterEvent *e )
```

```
{
   if ( e->provides( "text/dragdemotag" ) )
    e->accept();
}

void ListView::dropEvent( QDropEvent *e )
{
   if ( !e->provides( "text/dragdemotag" ) )
       return;

   QString tag;

   if ( QTextDrag::decode( e, tag ) ) {
      IconItem item = ((DnDDemo*) parentWidget())->findItem( tag );
      QListViewItem *after = itemAt( viewport()->mapFromParent( e->pos() ) );
      ListViewItem *litem = new ListViewItem( this, after, item.name(), tag );
      litem->setPixmap( 0, *item.pixmap() );
   }
}

void ListView::contentsMousePressEvent( QMouseEvent *e )
{
   QListView::contentsMousePressEvent( e );
   dragging = TRUE;
   pressPos = e->pos();
}

void ListView::contentsMouseMoveEvent( QMouseEvent *e )
{
   QListView::contentsMouseMoveEvent( e );

   if ( ! dragging ) return;

   if ( !currentItem() ) return;

   if ( ( pressPos - e->pos() ).manhattanLength() > QApplication::startDragDistance() ) {
      QTextDrag *drg = new QTextDrag( ((ListViewItem*)currentItem())->tag(), this );

    const QPixmap *p = ((ListViewItem*)currentItem())->pixmap( 0 );
    if (p)
        drg->setPixmap(*p);
      drg->setSubtype( "dragdemotag" );
      drg->dragCopy();
      dragging = FALSE;
   }
}

void ListView::contentsMouseReleaseEvent( QMouseEvent *e )
{
   QListView::contentsMouseReleaseEvent( e );
   dragging = FALSE;
}
```

demo/dnd/listview.h

```
#include <qlistview.h>

class ListViewItem : public QListViewItem
{
public:
    ListViewItem ( QListView * parent, const QString& name, const QString& tag )
        : QListViewItem( parent, name ), _tag( tag ) {}
    ListViewItem ( QListView * parent, QListViewItem * after, const QString& name, const QString&
tag )
        : QListViewItem( parent, after, name ), _tag( tag ) {}
    virtual ~ListViewItem() {}

    QString tag() { return _tag; }

private:
    QString _tag;
};

class ListView : public QListView
{
    Q_OBJECT

public:
    ListView( QWidget* parent = 0, const char* name = 0 );
    ~ListView();

    void dragEnterEvent( QDragEnterEvent * );
    void dropEvent( QDropEvent * );
    void contentsMousePressEvent( QMouseEvent * );
    void contentsMouseMoveEvent( QMouseEvent * );
    void contentsMouseReleaseEvent( QMouseEvent * );

private:
    QPoint pressPos;
    bool dragging;
};
```

demo/dnd/stylebutton.cpp

```
#include "styledbutton.h"

#include <qcolordialog.h>
#include <qpalette.h>
#include <qlabel.h>
#include <qpainter.h>
#include <qimage.h>
#include <qpixmap.h>
#include <qapplication.h>
#include <qdragobject.h>
#include <qstyle.h>

StyledButton::StyledButton(QWidget* parent, const char* name)
    : QButton( parent, name ), pix( 0 ), spix( 0 ), edit( ColorEditor ), s( 0 ), mousePressed( FALSE )
{
    setMinimumSize( minimumSizeHint() );
    setAcceptDrops( TRUE );

    connect( this, SIGNAL(clicked()), SLOT(onEditor()));
}

StyledButton::StyledButton( const QBrush& b, QWidget* parent, const char* name, WFlags f )
    : QButton( parent, name, f ), spix( 0 ), s( 0 )
{
    col = b.color();
    pix = b.pixmap();
    setMinimumSize( minimumSizeHint() );
}

StyledButton::~StyledButton()
{
    if ( pix ) {
     delete pix;
     pix = 0;
    }
    if ( spix ) {
     delete spix;
     spix = 0;
    }
}

void StyledButton::setEditor( EditorType e )
{
    if ( edit == e )
     return;

    edit = e;
    update();
}

StyledButton::EditorType StyledButton::editor() const
{
    return edit;
}

void StyledButton::setColor( const QColor& c )
{
    col = c;
    update();
}

void StyledButton::setPixmap( const QPixmap & pm )
{
    if ( !pm.isNull() ) {
     delete pix;
     pix = new QPixmap( pm );
    } else {
     delete pix;
     pix = 0;
    }
    scalePixmap();
}

QColor StyledButton::color() const
{
    return col;
}

QPixmap* StyledButton::pixmap() const
{
    return pix;
}

bool StyledButton::scale() const
{
    return s;
}

void StyledButton::setScale( bool on )
{
    if ( s == on )
      return;

    s = on;
    scalePixmap();
}

QSize StyledButton::sizeHint() const
{
    return QSize( 50, 25 );
}

QSize StyledButton::minimumSizeHint() const
{
    return QSize( 50, 25 );
}

void StyledButton::scalePixmap()
{
    delete spix;

    if ( pix ) {
      spix = new QPixmap( 6*width()/8, 6*height()/8 );
      QImage img = pix->convertToImage();

      spix->convertFromImage( s? img.smoothScale( 6*width()/8, 6*height()/8 ) : img );
    } else {
      spix = 0;
    }

    update();
}
```

```
void StyledButton::resizeEvent( QResizeEvent* e )
{
    scalePixmap();
    QButton::resizeEvent( e );
}

void StyledButton::drawButton( QPainter *paint )
{
    style().drawPrimitive(QStyle::PE_ButtonBevel, paint, rect(), colorGroup(),
                     isDown() ? QStyle::Style_Sunken : QStyle::Style_Default);
    drawButtonLabel(paint);

    if (hasFocus())
      style().drawPrimitive(QStyle::PE_FocusRect, paint,
                          style().subRect(QStyle::SR_PushButtonFocusRect, this),
                          colorGroup(), QStyle::Style_Default);
}

void StyledButton::drawButtonLabel( QPainter *paint )
{
    QColor pen = isEnabled() ?
             hasFocus() ? palette().active().buttonText() : palette().inactive().buttonText()
             : palette().disabled().buttonText();
    paint->setPen( pen );

    if(!isEnabled()) {
      paint->setBrush( QBrush( colorGroup().button() ) );
    }
    else if ( edit == PixmapEditor && spix ) {
      paint->setBrush( QBrush( col, *spix ) );
      paint->setBrushOrigin( width()/8, height()/8 );
    } else
      paint->setBrush( QBrush( col ) );

    paint->drawRect( width()/8, height()/8, 6*width()/8, 6*height()/8 );
}

void StyledButton::onEditor()
{
    switch (edit) {
    case ColorEditor: {
      QColor c = QColorDialog::getColor( palette().active().background(), this );
      if ( c.isValid() ) {
          setColor( c );
          emit changed();
      }
    } break;
    case PixmapEditor: {
      QPixmap p;
        /*
        if ( pixmap() )
            p = qChoosePixmap( this,*pixmap() );
        else
            p = qChoosePixmap( this, QPixmap() );
```

```
     if ( !p.isNull() ) {
         setPixmap( p );
         emit changed();
     }
       */
    } break;
    default:
     break;
    }
}

void StyledButton::mousePressEvent(QMouseEvent* e)
{
    QButton::mousePressEvent(e);
    mousePressed = TRUE;
    pressPos = e->pos();
}

void StyledButton::mouseMoveEvent(QMouseEvent* e)
{
    QButton::mouseMoveEvent( e );
#ifndef QT_NO_DRAGANDDROP
    if ( !mousePressed )
     return;
    if ( ( pressPos - e->pos() ).manhattanLength() > QApplication::startDragDistance() ) {
     if ( edit == ColorEditor ) {
         QColorDrag *drg = new QColorDrag( col, this );
         QPixmap pix( 25, 25 );
         pix.fill( col );
         QPainter p( &pix );
         p.drawRect( 0, 0, pix.width(), pix.height() );
         p.end();
         drg->setPixmap( pix );
         mousePressed = FALSE;
         drg->dragCopy();
     }
     else if ( edit == PixmapEditor && pix && !pix->isNull() ) {
         QImage img = pix->convertToImage();
         QImageDrag *drg = new QImageDrag( img, this );
         if(spix)
           drg->setPixmap( *spix );
         mousePressed = FALSE;
         drg->dragCopy();
     }
    }
#endif
}

#ifndef QT_NO_DRAGANDDROP
void StyledButton::dragEnterEvent( QDragEnterEvent *e )
{
    setFocus();
    if ( edit == ColorEditor && QColorDrag::canDecode( e ) )
     e->accept();
```

```
  else if ( edit == PixmapEditor && QImageDrag::canDecode( e ) )
   e->accept();
  else
   e->ignore();
}

void StyledButton::dragLeaveEvent( QDragLeaveEvent * )
{
  if ( hasFocus() )
   parentWidget()->setFocus();
}

void StyledButton::dragMoveEvent( QDragMoveEvent *e )
{
  if ( edit == ColorEditor && QColorDrag::canDecode( e ) )
   e->accept();
  else if ( edit == PixmapEditor && QImageDrag::canDecode( e ) )
   e->accept();
  else
   e->ignore();
}

void StyledButton::dropEvent( QDropEvent *e )
{
  if ( edit == ColorEditor && QColorDrag::canDecode( e ) ) {
   QColor color;
   QColorDrag::decode( e, color );
   setColor(color);
   emit changed();
   e->accept();
  }
  else if ( edit == PixmapEditor && QImageDrag::canDecode( e ) ) {
   QImage img;
   QImageDrag::decode( e, img );
   QPixmap pm;
   pm.convertFromImage(img);
   setPixmap(pm);
   emit changed();
   e->accept();
  } else {
   e->ignore();
  }
}
#endif // QT_NO_DRAGANDDROP
```

demo/dnd/stylebutton.h

```
#ifndef STYLEDBUTTON_H
#define STYLEDBUTTON_H

#include <qbutton.h>
#include <qpixmap.h>

class QColor;
class QBrush;
class StyledButton : public QButton
{
   Q_OBJECT

   Q_PROPERTY( QColor color READ color WRITE setColor )
   Q_PROPERTY( QPixmap pixmap READ pixmap WRITE setPixmap )
   Q_PROPERTY( EditorType editor READ editor WRITE setEditor )
   Q_PROPERTY( bool scale READ scale WRITE setScale )

   Q_ENUMS( EditorType )

public:
   enum EditorType { ColorEditor, PixmapEditor };

   StyledButton( QWidget* parent = 0, const char* name = 0 );
   StyledButton( const QBrush& b, QWidget* parent = 0, const char* name = 0, WFlags f = 0 );
   ~StyledButton();

   void setEditor( EditorType );
   EditorType editor() const;

   void setColor( const QColor& );
   void setPixmap( const QPixmap& );

   QPixmap* pixmap() const;
   QColor color() const;

   void setScale( bool );
   bool scale() const;

   QSize sizeHint() const;
   QSize minimumSizeHint() const;

public slots:
   virtual void onEditor();

signals:
   void changed();

protected:
   void mousePressEvent(QMouseEvent*);
   void mouseMoveEvent(QMouseEvent*);
#ifndef QT_NO_DRAGANDDROP
   void dragEnterEvent ( QDragEnterEvent * );
   void dragMoveEvent ( QDragMoveEvent * );
   void dragLeaveEvent ( QDragLeaveEvent * );
   void dropEvent ( QDropEvent * );
#endif // QT_NO_DRAGANDDROP
   void drawButton( QPainter* );
   void drawButtonLabel( QPainter* );
   void resizeEvent( QResizeEvent* );
   void scalePixmap();
private:
   QPixmap* pix;
   QPixmap* spix;  // the pixmap scaled down to fit into the button
   QColor col;
   EditorType edit;
   bool s;
   QPoint pressPos;
   bool mousePressed;
};

#endif //STYLEDBUTTON_H
```

demo/i18n/i18n.cpp

```
#include "i18n.h"
#include "wrapper.h"
#include "../textdrawing/textedit.h"

#include <qaction.h>
#include <qlayout.h>
#include <qvbox.h>
#include <qworkspace.h>
#include <qpopupmenu.h>
#include <qmenubar.h>
#include <qtoolbar.h>
#include <qtoolbutton.h>
#include <qpixmap.h>
#include <qiconset.h>
#include <qapplication.h>
#include <qwidgetlist.h>
#include <qlabel.h>
#include <qtextedit.h>

static int windowIdNumber = 5000;
static bool firstShow = TRUE;

I18nDemo::I18nDemo(QWidget *parent, const char *name)
    : QMainWindow(parent, name, 0), lastwrapper(0)
{
    initActions();
    initMenuBar();

    QVBox *box = new QVBox(this);
    box->setFrameStyle( QFrame::StyledPanel | QFrame::Sunken );
    box->setMargin( 1 );
    box->setLineWidth( 1 );

    workspace = new QWorkspace(box);
    connect(workspace, SIGNAL(windowActivated(QWidget *)),
        SLOT(windowActivated(QWidget *)));
    workspace->setBackgroundMode(PaletteMid);

    setCentralWidget(box);
}
```

```
I18nDemo::~I18nDemo()
{
}

void I18nDemo::initActions()
{
   actionClose = new QAction(tr("Close the current window."), tr("Close"), CTRL + Key_F4,  this);
   connect(actionClose, SIGNAL(activated()), SLOT(closeSlot()));

   actionCloseAll = new QAction(tr("Close all opened windows."), tr("Close All"), 0, this);
   connect(actionCloseAll, SIGNAL(activated()), SLOT(closeAllSlot()));

   actionTile = new QAction(tr("Tile opened windows."), tr("Tile"), 0,  this);
   connect(actionTile, SIGNAL(activated()), SLOT(tileSlot()));

   actionCascade = new QAction(tr("Cascade opened windows."),
                  tr("Cascade"),
                  0,
                  this);
   connect(actionCascade, SIGNAL(activated()), SLOT(cascadeSlot()));
}

void I18nDemo::initMenuBar()
{
   newMenu = new QPopupMenu(this);
   connect(newMenu, SIGNAL(activated(int)), SLOT(newSlot(int)));

   newMenu->insertItem("&English", 0);
   newMenu->insertItem("&Japanese", 1);
   newMenu->insertItem("&Korean", 2);
   newMenu->insertItem("&Norwegian", 3);

   windowMenu = new QPopupMenu(this);
   connect(windowMenu, SIGNAL(activated(int)), SLOT(windowSlot(int)));

   windowMenu->setCheckable(TRUE);

   actionClose->addTo(windowMenu);
   actionCloseAll->addTo(windowMenu);
   windowMenu->insertSeparator();
   actionTile->addTo(windowMenu);
   actionCascade->addTo(windowMenu);
   windowMenu->insertSeparator();

   menuBar()->insertItem(tr("&New"), newMenu);
   menuBar()->insertItem(tr("&Window"), windowMenu);
}

void I18nDemo::newSlot(int id)
{
   QString qmfile;
   switch (id) {
   default:
   case 0: qmfile = "i18n/en.qm"; break;
```

```
    case 1: qmfile = "i18n/ja.qm"; break;
    case 2: qmfile = "i18n/ko.qm"; break;
    case 3: qmfile = "i18n/no.qm"; break;
    }

    if (lastwrapper) {
     qApp->removeTranslator(&lastwrapper->translator);
     lastwrapper = 0;
    }

    Wrapper *wrapper = new Wrapper(workspace, windowIdNumber);
    wrapper->translator.load(qmfile, ".");

    qApp->installTranslator(&wrapper->translator);

    connect(wrapper, SIGNAL(destroyed()), SLOT(wrapperDead()));
    wrapper->setCaption(tr("--language--"));

    TextEdit *te = new TextEdit(wrapper);
    te->layout()->setResizeMode( QLayout::FreeResize );
    te->setMinimumSize(500, 400);
    te->fileNew();
    te->currentEditor()->
     setText(tr("<h3>About Qt</h3>"
             "<p>This program uses Qt version %1, a multiplatform C++ "
             "GUI toolkit from Trolltech. Qt provides single-source "
             "portability across Windows 95/98/NT/2000, Mac OS X, Linux, Solaris, "
             "HP-UX and many other versions of Unix with X11.</p>"
             "<p>See <tt>http://www.trolltech.com/qt/</tt> for more "
             "information.</p>").arg(QT_VERSION_STR));

    qApp->removeTranslator(&wrapper->translator);

    te->show();
    wrapper->show();

    windowMenu->insertItem(wrapper->caption(), wrapper->id);
    windowMenu->setItemChecked(wrapper->id, TRUE);
    lastwrapper = wrapper;

    windowIdNumber++;
}

void I18nDemo::windowSlot(int id)
{
    if (id < 5000)
     return;

    QWidgetList list = workspace->windowList();
    Wrapper *wrapper = (Wrapper *) list.first();
    while (wrapper) {
     if (wrapper->id == id) {
         wrapper->setFocus();
         break;
     }

     wrapper = (Wrapper *) list.next();
   }
}

void I18nDemo::windowActivated(QWidget *w)
{
   if (lastwrapper) {
     qApp->removeTranslator(&lastwrapper->translator);
     windowMenu->setItemChecked(lastwrapper->id, FALSE);
   }

   if (! w) {
     lastwrapper = 0;
     return;
   }

   Wrapper *wrapper = (Wrapper *) w;

   windowMenu->setItemChecked(wrapper->id, TRUE);
   lastwrapper = wrapper;
}

void I18nDemo::closeSlot()
{
   QWidget *w = workspace->activeWindow();
   delete w;
}

void I18nDemo::closeAllSlot()
{
   QWidget *w;
   while ((w = workspace->activeWindow()))
     w->close(TRUE);
}

void I18nDemo::tileSlot()
{
   workspace->tile();
}

void I18nDemo::cascadeSlot()
{
   workspace->cascade();
}

void I18nDemo::wrapperDead()
{
   Wrapper *w = (Wrapper *) sender();

   if (w == lastwrapper) {
     qApp->removeTranslator(&w->translator);
     lastwrapper = 0;
```

```
    }

    windowMenu->removeItem(w->id);
}

void I18nDemo::showEvent(QShowEvent *)
{
    if (firstShow) {
      newSlot(1);
      firstShow = FALSE;
      return;
    }

    if (! lastwrapper)
      return;

    qApp->installTranslator(&lastwrapper->translator);
}

void I18nDemo::hideEvent(QHideEvent *)
{
    if (! lastwrapper)
      return;

    qApp->removeTranslator(&lastwrapper->translator);
}
```

demo/i18n/i18n.h

```
#ifndef I18N_H
#define I18N_H

#include <qmainwindow.h>

class QWorkspace;
class QAction;
class QPopupMenu;
class Wrapper;

class I18nDemo : public QMainWindow
{
    Q_OBJECT

public:
    I18nDemo(QWidget *, const char * = 0);
    ~I18nDemo();

    void initActions();
    void initMenuBar();

    void showEvent(QShowEvent *);
    void hideEvent(QHideEvent *);

    QWorkspace *workspace;
    QAction *actionClose, *actionCloseAll, *actionTile, *actionCascade;
    QPopupMenu *windowMenu, *newMenu;
    Wrapper *lastwrapper;

public slots:
    void newSlot(int);
    void windowSlot(int);
    void windowActivated(QWidget *);
    void closeSlot();
    void closeAllSlot();
    void tileSlot();
    void cascadeSlot();
    void wrapperDead();
};

#endif // I18N_H
```

demo/i18n/wrapper.h

```
#ifndef WRAPPER_H
#define WRAPPER_H

#include <qvbox.h>
#include <qtranslator.h>

class Wrapper : public QVBox
{
public:
    Wrapper(QWidget *parent, int i, const char *name = 0)
      : QVBox(parent, name, WDestructiveClose), translator(this), id(i)
    {
    }

    QTranslator translator;
    int id;
};

#endif // WRAPPER_H
```

demo/qasteroids/ledmeter.cpp

```
#include <qpainter.h>
#include "ledmeter.h"

KALedMeter::KALedMeter( QWidget *parent ) : QFrame( parent )
{
    mCRanges.setAutoDelete( TRUE );
    mRange = 100;
    mCount = 20;
    mCurrentCount = 0;
    mValue = 0;
    setMinimumWidth( mCount * 2 + frameWidth() );
}

void KALedMeter::setRange( int r )
{
    mRange = r;
    if ( mRange < 1 )
        mRange = 1;
    setValue( mValue );
    update();
}

void KALedMeter::setCount( int c )
{
    mCount = c;
    if ( mCount < 1 )
        mCount = 1;
    setMinimumWidth( mCount * 2 + frameWidth() );
    calcColorRanges();
    setValue( mValue );
    update();
}

void KALedMeter::setValue( int v )
{
    mValue = v;
    if ( mValue > mRange )
        mValue = mRange;
    else if ( mValue < 0 )
        mValue = 0;
    int c = ( mValue + mRange / mCount - 1 ) * mCount / mRange;
    if ( c != mCurrentCount )
    {
        mCurrentCount = c;
        update();
    }
}

void KALedMeter::addColorRange( int pc, const QColor &c )
{
    ColorRange *cr = new ColorRange;
    cr->mPc = pc;
    cr->mColor = c;
    mCRanges.append( cr );
    calcColorRanges();
}

void KALedMeter::resizeEvent( QResizeEvent *e )
{
    QFrame::resizeEvent( e );
    int w = ( width() - frameWidth() - 2 ) / mCount * mCount;
    w += frameWidth() + 2;
    setFrameRect( QRect( 0, 0, w, height() ) );
}

void KALedMeter::drawContents( QPainter *p )
{
    QRect b = contentsRect();

    unsigned cidx = 0;
    int ncol = mCount;
    QColor col = colorGroup().foreground();

    if ( !mCRanges.isEmpty() )
    {
        col = mCRanges.at( cidx )->mColor;
        ncol = mCRanges.at( cidx )->mValue;
    }
    p->setBrush( col );
    p->setPen( col );

    int lw = b.width() / mCount;
    int lx = b.left() + 1;
    for ( int i = 0; i < mCurrentCount; i++, lx += lw )
    {
        if ( i > ncol )
        {
            if ( ++cidx < mCRanges.count() )
            {
                col = mCRanges.at( cidx )->mColor;
                ncol = mCRanges.at( cidx )->mValue;
                p->setBrush( col );
                p->setPen( col );
            }
        }

        p->drawRect( lx, b.top() + 1, lw - 1, b.height() - 2 );
    }
}

void KALedMeter::calcColorRanges()
{
    int prev = 0;
    ColorRange *cr;
    for ( cr = mCRanges.first(); cr; cr = mCRanges.next() )
    {
        cr->mValue = prev + cr->mPc * mCount / 100;
        prev = cr->mValue;
    }
}
```

demo/qasteroids/ledmeter.h

```
#ifndef __LEDMETER_H__
#define __LEDMETER_H__

#include <qframe.h>
#include <qptrlist.h>


class KALedMeter : public QFrame
{
    Q_OBJECT
public:
    KALedMeter( QWidget *parent );
    int range() const { return mRange; }
    void setRange( int r );

    int count() const { return mCount; }
    void setCount( int c );

    int value () const { return mValue; }

    void addColorRange( int pc, const QColor &c );

public slots:
    void setValue( int v );

protected:
    virtual void resizeEvent( QResizeEvent * );
    virtual void drawContents( QPainter * );
    void calcColorRanges();

protected:
    struct ColorRange
    {
     int mPc;
     int mValue;
     QColor mColor;
    };

    int mRange;
    int mCount;
    int mCurrentCount;
    int mValue;
    QPtrList<ColorRange> mCRanges;
};

#endif
```

demo/qasteroids/sprites.h

```
#ifndef __SPRITES_H__
#define __SPRITES_H__

#include <qcanvas.h>

#define ID_ROCK_LARGE           1024
#define ID_ROCK_MEDIUM          1025
#define ID_ROCK_SMALL           1026

#define ID_MISSILE              1030

#define ID_BIT                  1040
#define ID_EXHAUST              1041

#define ID_ENERGY_POWERUP       1310
#define ID_TELEPORT_POWERUP     1311
#define ID_BRAKE_POWERUP        1312
```

```
#define ID_SHIELD_POWERUP       1313
#define ID_SHOOT_POWERUP        1314

#define ID_SHIP                 1350
#define ID_SHIELD               1351

#define MAX_SHIELD_AGE          350
#define MAX_POWERUP_AGE         500
#define MAX_MISSILE_AGE         40

class KMissile : public QCanvasSprite
{
public:
   KMissile( QCanvasPixmapArray *s, QCanvas *c ) : QCanvasSprite( s, c )
      { myAge = 0; }

   virtual int rtti() const { return ID_MISSILE; }

   void growOlder() { myAge++; }
   bool expired() { return myAge > MAX_MISSILE_AGE; }

private:
   int myAge;
};

class KBit : public QCanvasSprite
{
public:
   KBit( QCanvasPixmapArray *s, QCanvas *c ) : QCanvasSprite( s, c )
     {  death = 7; }

   virtual int rtti() const {  return ID_BIT; }

   void setDeath( int d ) { death = d; }
   void growOlder() { death--; }
   bool expired() { return death <= 0; }

private:
   int death;
};

class KExhaust : public QCanvasSprite
{
public:
   KExhaust( QCanvasPixmapArray *s, QCanvas *c ) : QCanvasSprite( s, c )
     {  death = 1; }

   virtual int rtti() const {  return ID_EXHAUST; }

   void setDeath( int d ) { death = d; }
   void growOlder() { death--; }
   bool expired() { return death <= 0; }

private:
    int death;
};

class KPowerup : public QCanvasSprite
{
public:
  KPowerup( QCanvasPixmapArray *s, QCanvas *c, int t ) : QCanvasSprite( s, c ),
        myAge( 0 ), type(t) { }

  virtual int rtti() const { return type; }

  void growOlder() { myAge++; }
  bool expired() const { return myAge > MAX_POWERUP_AGE; }

protected:
  int myAge;
  int type;
};

class KRock : public QCanvasSprite
{
public:
    KRock (QCanvasPixmapArray *s, QCanvas *c, int t, int sk, int st) : QCanvasSprite( s, c )
        { type = t; skip = cskip = sk; step = st; }

    void nextFrame()
      {
          if (cskip-- <= 0) {
            setFrame( (frame()+step+frameCount())%frameCount() );
            cskip = QABS(skip);
          }
      }

    virtual int rtti() const { return type; }

private:
    int type;
    int skip;
    int cskip;
    int step;
};

class KShield : public QCanvasSprite
{
public:
  KShield( QCanvasPixmapArray *s, QCanvas *c )
      : QCanvasSprite( s, c ) {}

  virtual int rtti() const { return ID_SHIELD; }
};

#endif
```

demo/qasteroids/toplevel.cpp

```
//   --- toplevel.cpp ---
#include <qaccel.h>
#include <qlabel.h>
#include <qlayout.h>
#include <qlcdnumber.h>
#include <qpushbutton.h>

#include <qapplication.h>

#include "toplevel.h"
#include "ledmeter.h"

#define SB_SCORE  1
#define SB_LEVEL  2
#define SB_SHIPS    3

struct SLevel
{
    int    nrocks;
    double rockSpeed;
};

#define MAX_LEVELS 16

SLevel levels[MAX_LEVELS] =
{
    { 1, 0.4 },
    { 1, 0.6 },
    { 2, 0.5 },
    { 2, 0.7 },
    { 2, 0.8 },
    { 3, 0.6 },
    { 3, 0.7 },
    { 3, 0.8 },
    { 4, 0.6 },
    { 4, 0.7 },
    { 4, 0.8 },
    { 5, 0.7 },
    { 5, 0.8 },
    { 5, 0.9 },
    { 5, 1.0 }
};

const char *soundEvents[] =
{
    "ShipDestroyed",
    "RockDestroyed",
    0
};

const char *soundDefaults[] =
{
    "Explosion.wav",
    "ploop.wav",
    0
};


KAstTopLevel::KAstTopLevel( QWidget *parent, const char *name )
    : QMainWindow( parent, name, 0 )
{
    QWidget *border = new QWidget( this );
    border->setBackgroundColor( black );
    setCentralWidget( border );

    QVBoxLayout *borderLayout = new QVBoxLayout( border );
    borderLayout->addStretch( 1 );

    QWidget *mainWin = new QWidget( border );
    mainWin->setFixedSize(640, 480);
    borderLayout->addWidget( mainWin, 0, AlignHCenter );

    borderLayout->addStretch( 1 );

    view = new KAsteroidsView( mainWin );
    view->setFocusPolicy( StrongFocus );
    connect( view, SIGNAL( shipKilled() ), SLOT( slotShipKilled() ) );
    connect( view, SIGNAL( rockHit(int) ), SLOT( slotRockHit(int) ) );
    connect( view, SIGNAL( rocksRemoved() ), SLOT( slotRocksRemoved() ) );
    connect( view, SIGNAL( updateVitals() ), SLOT( slotUpdateVitals() ) );

    QVBoxLayout *vb = new QVBoxLayout( mainWin );
    QHBoxLayout *hb = new QHBoxLayout;
    QHBoxLayout *hbd = new QHBoxLayout;
    vb->addLayout( hb );

    QFont labelFont( "helvetica", 24 );
    QColorGroup grp( darkGreen, black, QColor( 128, 128, 128 ),
        QColor( 64, 64, 64 ), black, darkGreen, black );
    QPalette pal( grp, grp, grp );

    mainWin->setPalette( pal );

    hb->addSpacing( 10 );

    QLabel *label;
    label = new QLabel( tr("Score"), mainWin );
    label->setFont( labelFont );
    label->setPalette( pal );
    label->setFixedWidth( label->sizeHint().width() );
    hb->addWidget( label );

    scoreLCD = new QLCDNumber( 6, mainWin );
    scoreLCD->setFrameStyle( QFrame::NoFrame );
    scoreLCD->setSegmentStyle( QLCDNumber::Flat );
    scoreLCD->setFixedWidth( 150 );
    scoreLCD->setPalette( pal );
    hb->addWidget( scoreLCD );
    hb->addStretch( 10 );

    label = new QLabel( tr("Level"), mainWin );
    label->setFont( labelFont );
    label->setPalette( pal );
    label->setFixedWidth( label->sizeHint().width() );
    hb->addWidget( label );

    levelLCD = new QLCDNumber( 2, mainWin );
    levelLCD->setFrameStyle( QFrame::NoFrame );
    levelLCD->setSegmentStyle( QLCDNumber::Flat );
    levelLCD->setFixedWidth( 70 );
    levelLCD->setPalette( pal );
    hb->addWidget( levelLCD );
    hb->addStretch( 10 );

    label = new QLabel( tr("Ships"), mainWin );
    label->setFont( labelFont );
    label->setFixedWidth( label->sizeHint().width() );
    label->setPalette( pal );
    hb->addWidget( label );

    shipsLCD = new QLCDNumber( 1, mainWin );
    shipsLCD->setFrameStyle( QFrame::NoFrame );
    shipsLCD->setSegmentStyle( QLCDNumber::Flat );
    shipsLCD->setFixedWidth( 40 );
    shipsLCD->setPalette( pal );
    hb->addWidget( shipsLCD );

    hb->addStrut( 30 );

    vb->addWidget( view, 10 );

// -- bottom layout:
    vb->addLayout( hbd );

    QFont smallFont( "helvetica", 14 );
    hbd->addSpacing( 10 );

    QString sprites_prefix = "qasteroids/sprites/";
/*
    label = new QLabel( tr( "T" ), mainWin );
    label->setFont( smallFont );
    label->setFixedWidth( label->sizeHint().width() );
    label->setPalette( pal );
    hbd->addWidget( label );

    teleportsLCD = new QLCDNumber( 1, mainWin );
    teleportsLCD->setFrameStyle( QFrame::NoFrame );
    teleportsLCD->setSegmentStyle( QLCDNumber::Flat );
    teleportsLCD->setPalette( pal );
    teleportsLCD->setFixedHeight( 20 );
    hbd->addWidget( teleportsLCD );
    hbd->addSpacing( 10 );
*/
    QPixmap pm( sprites_prefix + "powerups/brake.png" );
    label = new QLabel( mainWin );
    label->setPixmap( pm );
    label->setFixedWidth( label->sizeHint().width() );
    label->setPalette( pal );
    hbd->addWidget( label );

    brakesLCD = new QLCDNumber( 1, mainWin );
    brakesLCD->setFrameStyle( QFrame::NoFrame );
    brakesLCD->setSegmentStyle( QLCDNumber::Flat );
    brakesLCD->setPalette( pal );
    brakesLCD->setFixedHeight( 20 );
    hbd->addWidget( brakesLCD );

    hbd->addSpacing( 10 );

    pm.load( sprites_prefix + "powerups/shield.png" );
    label = new QLabel( mainWin );
    label->setPixmap( pm );
    label->setFixedWidth( label->sizeHint().width() );
    label->setPalette( pal );
    hbd->addWidget( label );

    shieldLCD = new QLCDNumber( 1, mainWin );
    shieldLCD->setFrameStyle( QFrame::NoFrame );
    shieldLCD->setSegmentStyle( QLCDNumber::Flat );
    shieldLCD->setPalette( pal );
    shieldLCD->setFixedHeight( 20 );
    hbd->addWidget( shieldLCD );

    hbd->addSpacing( 10 );

    pm.load( sprites_prefix + "powerups/shoot.png" );
    label = new QLabel( mainWin );
    label->setPixmap( pm );
    label->setFixedWidth( label->sizeHint().width() );
    label->setPalette( pal );
    hbd->addWidget( label );

    shootLCD = new QLCDNumber( 1, mainWin );
    shootLCD->setFrameStyle( QFrame::NoFrame );
    shootLCD->setSegmentStyle( QLCDNumber::Flat );
    shootLCD->setPalette( pal );
    shootLCD->setFixedHeight( 20 );
    hbd->addWidget( shootLCD );

    hbd->addStretch( 1 );

    label = new QLabel( tr( "Fuel" ), mainWin );
    label->setFont( smallFont );
    label->setFixedWidth( label->sizeHint().width() + 10 );
    label->setPalette( pal );
    hbd->addWidget( label );

    powerMeter = new KALedMeter( mainWin );
    powerMeter->setFrameStyle( QFrame::Box | QFrame::Plain );
    powerMeter->setRange( MAX_POWER_LEVEL );
    powerMeter->addColorRange( 10, darkRed );
    powerMeter->addColorRange( 20, QColor(160, 96, 0) );
    powerMeter->addColorRange( 70, darkGreen );
    powerMeter->setCount( 40 );
    powerMeter->setPalette( pal );
    powerMeter->setFixedSize( 200, 12 );
    hbd->addWidget( powerMeter );

    shipsRemain = 3;
    showHiscores = FALSE;

    actions.insert( Qt::Key_Up, Thrust );
    actions.insert( Qt::Key_Left, RotateLeft );
    actions.insert( Qt::Key_Right, RotateRight );
    actions.insert( Qt::Key_Space, Shoot );
    actions.insert( Qt::Key_Z, Teleport );
    actions.insert( Qt::Key_X, Brake );
    actions.insert( Qt::Key_S, Shield );
    actions.insert( Qt::Key_P, Pause );
    actions.insert( Qt::Key_L, Launch );
    actions.insert( Qt::Key_N, NewGame );

    view->showText( tr( "Press N to start playing" ), yellow );
}

KAstTopLevel::~KAstTopLevel()
{
}

void KAstTopLevel::playSound( const char * )
{
}

void KAstTopLevel::keyPressEvent( QKeyEvent *event )
{
    if ( event->isAutoRepeat() || !actions.contains( event->key() ) )
    {
        event->ignore();
        return;
    }

    Action a = actions[ event->key() ];

    switch ( a )
    {
        case RotateLeft:
            view->rotateLeft( TRUE );
            break;
        case RotateRight:
            view->rotateRight( TRUE );
            break;

        case Thrust:
            view->thrust( TRUE );
            break;

        case Shoot:
            view->shoot( TRUE );
            break;

        case Shield:
            view->setShield( TRUE );
            break;

        case Teleport:
            view->teleport( TRUE );
            break;

        case Brake:
            view->brake( TRUE );
            break;

        default:
            event->ignore();
            return;
    }
    event->accept();
}

void KAstTopLevel::keyReleaseEvent( QKeyEvent *event )
{
    if ( event->isAutoRepeat() || !actions.contains( event->key() ) )
    {
        event->ignore();
        return;
    }

    Action a = actions[ event->key() ];

    switch ( a )
    {
        case RotateLeft:
            view->rotateLeft( FALSE );
            break;

        case RotateRight:
            view->rotateRight( FALSE );
            break;

        case Thrust:
            view->thrust( FALSE );
```

```
         break;

      case Shoot:
         view->shoot( FALSE );
         break;

      case Brake:
         view->brake( FALSE );
         break;

      case Shield:
         view->setShield( FALSE );
         break;

      case Teleport:
         view->teleport( FALSE );
         break;

      case Launch:
         if ( waitShip )
         {
            view->newShip();
            waitShip = FALSE;
            view->hideText();
         }
         else
         {
            event->ignore();
            return;
         }
         break;

    case NewGame:
       slotNewGame();
       break;
/*
      case Pause:
         {
            view->pause( TRUE );
            QMessageBox::information( this,
                               tr("KAsteroids is paused"),
                               tr("Paused") );
            view->pause( FALSE );
         }
         break;
*/
      default:
         event->ignore();
         return;
   }

   event->accept();
}
```

```
void KAstTopLevel::showEvent( QShowEvent *e )
{
   QMainWindow::showEvent( e );
   view->pause( FALSE );
   view->setFocus();
}

void KAstTopLevel::hideEvent( QHideEvent *e )
{
   QMainWindow::hideEvent( e );
   view->pause( TRUE );
}

void KAstTopLevel::slotNewGame()
{
   score = 0;
   shipsRemain = SB_SHIPS;
   scoreLCD->display( 0 );
   level = 0;
   levelLCD->display( level+1 );
   shipsLCD->display( shipsRemain-1 );
   view->newGame();
   view->setRockSpeed( levels[0].rockSpeed );
   view->addRocks( levels[0].nrocks );
//    view->showText( tr( "Press L to launch." ), yellow );
   view->newShip();
   waitShip = FALSE;
   view->hideText();
   isPaused = FALSE;
}

void KAstTopLevel::slotShipKilled()
{
   shipsRemain--;
   shipsLCD->display( shipsRemain-1 );

   playSound( "ShipDestroyed" );

   if ( shipsRemain )
   {
      waitShip = TRUE;
      view->showText( tr( "Ship Destroyed. Press L to launch."), yellow );
   }
   else
   {
      view->showText( tr("Game Over!"), red );
      view->endGame();
     doStats();
//       highscore->addEntry( score, level, showHiscores );
   }
}

void KAstTopLevel::slotRockHit( int size )
{
    switch ( size )
    {
     case 0:
         score += 10;
          break;

     case 1:
         score += 20;
         break;

     default:
         score += 40;
     }

    playSound( "RockDestroyed" );

    scoreLCD->display( score );
}

void KAstTopLevel::slotRocksRemoved()
{
    level++;

    if ( level >= MAX_LEVELS )
     level = MAX_LEVELS - 1;

    view->setRockSpeed( levels[level-1].rockSpeed );
    view->addRocks( levels[level-1].nrocks );

    levelLCD->display( level+1 );
}

void KAstTopLevel::doStats()
{
    QString r( "0.00" );
    if ( view->shots() )
      r = QString::number( (double)view->hits() / view->shots() * 100.0,
                  'g', 2 );

/* multi-line text broken in Qt 3
    QString s = tr( "Game Over\n\nShots fired:\t%1\n  Hit:\t%2\n  Missed:\t%3\nHit ratio:\t%4
%\n\nPress N for a new game" )
      .arg(view->shots()).arg(view->hits())
      .arg(view->shots() - view->hits())
      .arg(r);
*/

    view->showText( "Game Over.   Press N for a new game.", yellow, FALSE );
}

void KAstTopLevel::slotUpdateVitals()
{
    brakesLCD->display( view->brakeCount() );
    shieldLCD->display( view->shieldCount() );
    shootLCD->display( view->shootCount() );
//    teleportsLCD->display( view->teleportCount() );
    powerMeter->setValue( view->power() );
}
```

demo/qasteroids/toplevel.h

```
#ifndef __KAST_TOPLEVEL_H__
#define __KAST_TOPLEVEL_H__

#include <qmainwindow.h>
#include <qdict.h>
#include <qmap.h>

#include "view.h"


class KALedMeter;
class QLCDNumber;

class KAstTopLevel : public QMainWindow
{
    Q_OBJECT
public:
    KAstTopLevel( QWidget *parent=0, const char *name=0 );
    virtual ~KAstTopLevel();

private:
    void playSound( const char *snd );
    void readSoundMapping();
    void doStats();

protected:
    virtual void showEvent( QShowEvent * );
    virtual void hideEvent( QHideEvent * );
    virtual void keyPressEvent( QKeyEvent *event );
    virtual void keyReleaseEvent( QKeyEvent *event );

private slots:
    void slotNewGame();

    void slotShipKilled();
    void slotRockHit( int size );
    void slotRocksRemoved();

    void slotUpdateVitals();

private:
    KAsteroidsView *view;
    QLCDNumber *scoreLCD;
    QLCDNumber *levelLCD;
    QLCDNumber *shipsLCD;

    QLCDNumber *teleportsLCD;
//    QLCDNumber *bombsLCD;
```

```
   QLCDNumber *brakesLCD;
   QLCDNumber *shieldLCD;
   QLCDNumber *shootLCD;
   KALedMeter *powerMeter;

   bool   sound;
   QDict<QString> soundDict;

   // waiting for user to press Enter to launch a ship
   bool waitShip;
   bool isPaused;

   int shipsRemain;
   int score;
   int level;
   bool showHiscores;

   enum Action { Launch, Thrust, RotateLeft, RotateRight, Shoot, Teleport,
                 Brake, Shield, Pause, NewGame  };

   QMap<int,Action> actions;
};

#endif
```

demo/qasteroids/view.cpp

```
#include <stdlib.h>
#include <math.h>
#include <qapplication.h>
#include <qkeycode.h>
#include <qaccel.h>
#include <qmessagebox.h>

#include "view.h"

#define IMG_BACKGROUND "qasteroids/bg.png"

#define REFRESH_DELAY           33
#define SHIP_SPEED              0.3
#define MISSILE_SPEED           10.0
#define SHIP_STEPS              64
#define ROTATE_RATE             2
#define SHIELD_ON_COST          1
#define SHIELD_HIT_COST         30
#define BRAKE_ON_COST           4

#define MAX_ROCK_SPEED          2.5
#define MAX_POWERUP_SPEED       1.5
#define MAX_SHIP_SPEED          12
#define MAX_BRAKES              5
#define MAX_SHIELDS             5
#define MAX_FIREPOWER           5

#define TEXT_SPEED              4
```

```
#define PI_X_2               6.283185307
#ifndef M_PI
#define M_PI 3.141592654
#endif

static struct
{
   int id;
   const char *path;
   int frames;
}
kas_animations [] =
{
   { ID_ROCK_LARGE,       "rock1/rock1%1.png",       32 },
   { ID_ROCK_MEDIUM,      "rock2/rock2%1.png",       32 },
   { ID_ROCK_SMALL,       "rock3/rock3%1.png",       32 },
   { ID_SHIP,             "ship/ship%1.png",         32 },
   { ID_MISSILE,          "missile/missile.png",      1 },
   { ID_BIT,              "bits/bits%1.png",         16 },
   { ID_EXHAUST,          "exhaust/exhaust.png",      1 },
   { ID_ENERGY_POWERUP,   "powerups/energy.png",      1 },
//    { ID_TELEPORT_POWERUP, "powerups/teleport%1.png", 12 },
   { ID_BRAKE_POWERUP,    "powerups/brake.png",       1 },
   { ID_SHIELD_POWERUP,   "powerups/shield.png",      1 },
   { ID_SHOOT_POWERUP,    "powerups/shoot.png",       1 },
   { ID_SHIELD,           "shield/shield%1.png",      6 },
   { 0,                   0,                          0 }
};



KAsteroidsView::KAsteroidsView( QWidget *parent, const char *name )
   : QWidget( parent, name ),
     field(640, 440),
     view(&field,this)
{
   view.setVScrollBarMode( QScrollView::AlwaysOff );
   view.setHScrollBarMode( QScrollView::AlwaysOff );
   view.viewport()->setFocusProxy( this );
   rocks.setAutoDelete( TRUE );
   missiles.setAutoDelete( TRUE );
   bits.setAutoDelete( TRUE );
   powerups.setAutoDelete( TRUE );
   exhaust.setAutoDelete( TRUE );

   field.setBackgroundColor( black );
   QPixmap pm( IMG_BACKGROUND );
   field.setBackgroundPixmap( pm );

   textSprite = new QCanvasText( &field );
   QFont font( "helvetica", 18 );
   textSprite->setFont( font );
```

```
    shield = 0;
    shieldOn = FALSE;
    refreshRate = REFRESH_DELAY;

    initialized = readSprites();

    shieldTimer = new QTimer( this );
    connect( shieldTimer, SIGNAL(timeout()), this, SLOT(hideShield()) );
    mTimerId = -1;

    shipPower = MAX_POWER_LEVEL;
    vitalsChanged = TRUE;
    can_destroy_powerups = FALSE;

    mPaused = TRUE;

    if ( !initialized ) {
      textSprite->setText( tr("Error: Cannot read sprite images") );
      textSprite->setColor( red );
      textSprite->move( (field.width()-textSprite->boundingRect().width()) / 2,
                (field.height()-textSprite->boundingRect().height()) / 2 );
      textSprite->show();
    }
}

// - - -

KAsteroidsView::~KAsteroidsView()
{
}

// - - -

void KAsteroidsView::reset()
{
    if ( !initialized )
      return;
    rocks.clear();
    missiles.clear();
    bits.clear();
    powerups.clear();
    exhaust.clear();

    shotsFired = 0;
    shotsHit = 0;

    rockSpeed = 1.0;
    powerupSpeed = 1.0;
    mFrameNum = 0;
    mPaused = FALSE;

    ship->hide();
    shield->hide();
/*
```

```
   if ( mTimerId >= 0 ) {
    killTimer( mTimerId );
    mTimerId = -1;
   }
*/
}

// - --

void KAsteroidsView::newGame()
{
   if ( !initialized )
    return;
   if ( shieldOn )
   {
    shield->hide();
    shieldOn = FALSE;
   }
   reset();
   if ( mTimerId < 0 )
    mTimerId = startTimer( REFRESH_DELAY );
   emit updateVitals();
}

// - - -

void KAsteroidsView::endGame()
{
}

void KAsteroidsView::pause( bool p )
{
   if ( !initialized )
    return;
   if ( !mPaused && p ) {
    if ( mTimerId >= 0 ) {
       killTimer( mTimerId );
       mTimerId = -1;
    }
   } else if ( mPaused && !p )
    mTimerId = startTimer( REFRESH_DELAY );
   mPaused = p;
}

// - - -

void KAsteroidsView::newShip()
{
   if ( !initialized )
    return;
   ship->move( width()/2, height()/2, 0 );
   shield->move( width()/2, height()/2, 0 );
   ship->setVelocity( 0.0, 0.0 );
   shipDx = 0;
```

```
    shipDy = 0;
    shipAngle = 0;
    rotateL = FALSE;
    rotateR = FALSE;
    thrustShip = FALSE;
    shootShip = FALSE;
    brakeShip = FALSE;
    teleportShip = FALSE;
    shieldOn = TRUE;
    shootDelay = 0;
    shipPower = MAX_POWER_LEVEL;
    rotateRate = ROTATE_RATE;
    rotateSlow = 0;

    mBrakeCount = 0;
    mTeleportCount = 0;
    mShootCount = 0;

    ship->show();
    shield->show();
    mShieldCount = 1;   // just in case the ship appears on a rock.
    shieldTimer->start( 1000, TRUE );
}

void KAsteroidsView::setShield( bool s )
{
    if ( !initialized )
      return;
    if ( shieldTimer->isActive() && !s ) {
      shieldTimer->stop();
      hideShield();
    } else {
      shieldOn = s && mShieldCount;
    }
}

void KAsteroidsView::brake( bool b )
{
    if ( !initialized )
      return;
    if ( mBrakeCount )
    {
      if ( brakeShip && !b )
      {
          rotateL = FALSE;
          rotateR = FALSE;
          thrustShip = FALSE;
          rotateRate = ROTATE_RATE;
      }

      brakeShip = b;
    }
}
```

```
// - - -

bool KAsteroidsView::readSprites()
{
   QString sprites_prefix = "qasteroids/sprites/";

   int i = 0;
   while ( kas_animations[i].id )
   {
    QCanvasPixmapArray *anim =
       new QCanvasPixmapArray( sprites_prefix + kas_animations[i].path,
                    kas_animations[i].frames );
    if ( !anim->isValid() )
       return FALSE;
    animation.insert( kas_animations[i].id, anim );
    i++;
   }

   ship = new QCanvasSprite( animation[ID_SHIP], &field );
   ship->hide();

   shield = new KShield( animation[ID_SHIELD], &field );
   shield->hide();

   return (ship->image(0) && shield->image(0));
}

// - - -

void KAsteroidsView::addRocks( int num )
{
   if ( !initialized )
    return;
   for ( int i = 0; i < num; i++ )
   {
    KRock *rock = new KRock( animation[ID_ROCK_LARGE], &field,
                ID_ROCK_LARGE, randInt(2), randInt(2) ? -1 : 1 );
    double dx = (2.0 - randDouble()*4.0) * rockSpeed;
    double dy = (2.0 - randDouble()*4.0) * rockSpeed;
    rock->setVelocity( dx, dy );
    rock->setFrame( randInt( rock->frameCount() ) );
    if ( dx > 0 )
    {
       if ( dy > 0 )
        rock->move( 5, 5, 0 );
       else
        rock->move( 5, field.height() - 25, 0 );
    }
    else
    {
       if ( dy > 0 )
        rock->move( field.width() - 25, 5, 0 );
       else
        rock->move( field.width() - 25, field.height() - 25, 0 );
```

```
     }
     rock->show( );
     rocks.append( rock );
   }
}

// - - -

void KAsteroidsView::showText( const QString &text, const QColor &color, bool scroll )
{
   if ( !initialized )
     return;
   textSprite->setText( text );
   textSprite->setColor( color );

   if ( scroll ) {
     textSprite->move( (field.width()-textSprite->boundingRect().width()) / 2,
                 -textSprite->boundingRect().height() );
     textDy = TEXT_SPEED;
   } else {
     textSprite->move( (field.width()-textSprite->boundingRect().width()) / 2,
               (field.height()-textSprite->boundingRect().height()) / 2 );
     textDy = 0;
   }
   textSprite->show();
}

// - - -

void KAsteroidsView::hideText()
{
   textDy = -TEXT_SPEED;
}

// - - -

void KAsteroidsView::resizeEvent(QResizeEvent* event)
{
   QWidget::resizeEvent(event);
   field.resize(width()-4, height()-4);
   view.resize(width(),height());
}

// - - -

void KAsteroidsView::timerEvent( QTimerEvent * )
{
   field.advance();

   QCanvasSprite *rock;

   // move rocks forward
   for ( rock = rocks.first(); rock; rock = rocks.next() ) {
     ((KRock *)rock)->nextFrame();
```

```
      wrapSprite( rock );
    }

    wrapSprite( ship );

    // check for missile collision with rocks.
    processMissiles();

    // these are generated when a ship explodes
    for ( KBit *bit = bits.first(); bit; bit = bits.next() )
    {
      if ( bit->expired() )
      {
        bits.removeRef( bit );
      }
      else
      {
        bit->growOlder();
        bit->setFrame( ( bit->frame()+1 ) % bit->frameCount() );
      }
    }

    for ( KExhaust *e = exhaust.first(); e; e = exhaust.next() )
      exhaust.removeRef( e );

    // move / rotate ship.
    // check for collision with a rock.
    processShip();

    // move powerups and check for collision with player and missiles
    processPowerups();

    if ( textSprite->isVisible() )
    {
      if ( textDy < 0 &&
           textSprite->boundingRect().y() <= -textSprite->boundingRect().height() ) {
        textSprite->hide();
      } else {
        textSprite->moveBy( 0, textDy );
      }
      if ( textSprite->boundingRect().y() > (field.height()-textSprite->boundingRect().height())/2 )
        textDy = 0;
    }

    if ( vitalsChanged && !(mFrameNum % 10) ) {
      emit updateVitals();
      vitalsChanged = FALSE;
    }

    mFrameNum++;
}

void KAsteroidsView::wrapSprite( QCanvasItem *s )
{
    int x = int(s->x() + s->boundingRect().width() / 2);
    int y = int(s->y() + s->boundingRect().height() / 2);

    if ( x > field.width() )
     s->move( s->x() - field.width(), s->y() );
    else if ( x < 0 )
     s->move( field.width() + s->x(), s->y() );

    if ( y > field.height() )
     s->move( s->x(), s->y() - field.height() );
    else if ( y < 0 )
     s->move( s->x(), field.height() + s->y() );
}

// - - -

void KAsteroidsView::rockHit( QCanvasItem *hit )
{
    KPowerup *nPup = 0;
    int rnd = int(randDouble()*30.0) % 30;
    switch( rnd )
    {
     case 4:
     case 5:
     nPup = new KPowerup( animation[ID_ENERGY_POWERUP], &field,
                    ID_ENERGY_POWERUP );
     break;
     case 10:
//       nPup = new KPowerup( animation[ID_TELEPORT_POWERUP], &field,
//                            ID_TELEPORT_POWERUP );
     break;
     case 15:
     nPup = new KPowerup( animation[ID_BRAKE_POWERUP], &field,
                     ID_BRAKE_POWERUP );
     break;
     case 20:
     nPup = new KPowerup( animation[ID_SHIELD_POWERUP], &field,
                     ID_SHIELD_POWERUP );
     break;
     case 24:
     case 25:
     nPup = new KPowerup( animation[ID_SHOOT_POWERUP], &field,
                     ID_SHOOT_POWERUP );
     break;
    }
    if ( nPup )
    {
     double r = 0.5 - randDouble();
     nPup->move( hit->x(), hit->y(), 0 );
     nPup->setVelocity( hit->xVelocity() + r, hit->yVelocity() + r );
     nPup->show( );
     powerups.append( nPup );
    }
```

```
  if ( hit->rtti() == ID_ROCK_LARGE || hit->rtti() == ID_ROCK_MEDIUM )
  {
   // break into smaller rocks
   double addx[4] = { 1.0, 1.0, -1.0, -1.0 };
   double addy[4] = { -1.0, 1.0, -1.0, 1.0 };

   double dx = hit->xVelocity();
   double dy = hit->yVelocity();

   double maxRockSpeed = MAX_ROCK_SPEED * rockSpeed;
   if ( dx > maxRockSpeed )
      dx = maxRockSpeed;
   else if ( dx < -maxRockSpeed )
      dx = -maxRockSpeed;
   if ( dy > maxRockSpeed )
      dy = maxRockSpeed;
   else if ( dy < -maxRockSpeed )
      dy = -maxRockSpeed;

   QCanvasSprite *nrock;

   for ( int i = 0; i < 4; i++ )
   {
      double r = rockSpeed/2 - randDouble()*rockSpeed;
      if ( hit->rtti() == ID_ROCK_LARGE )
      {
       nrock = new KRock( animation[ID_ROCK_MEDIUM], &field,
               ID_ROCK_MEDIUM, randInt(2), randInt(2) ? -1 : 1 );
       emit rockHit( 0 );
      }
      else
      {
       nrock = new KRock( animation[ID_ROCK_SMALL], &field,
               ID_ROCK_SMALL, randInt(2), randInt(2) ? -1 : 1 );
       emit rockHit( 1 );
      }

      nrock->move( hit->x(), hit->y(), 0 );
      nrock->setVelocity( dx+addx[i]*rockSpeed+r, dy+addy[i]*rockSpeed+r );
      nrock->setFrame( randInt( nrock->frameCount() ) );
      nrock->show( );
      rocks.append( nrock );
   }
  }
  else if ( hit->rtti() == ID_ROCK_SMALL )
   emit rockHit( 2 );
  rocks.removeRef( (QCanvasSprite *)hit );
  if ( rocks.count() == 0 )
   emit rocksRemoved();
}

void KAsteroidsView::reducePower( int val )
{
  shipPower -= val;
```

```
    if ( shipPower <= 0 )
    {
      shipPower = 0;
      thrustShip = FALSE;
      if ( shieldOn )
      {
          shieldOn = FALSE;
          shield->hide();
      }
    }
    vitalsChanged = TRUE;
}

void KAsteroidsView::addExhaust( double x, double y, double dx,
                        double dy, int count )
{
    for ( int i = 0; i < count; i++ )
    {
      KExhaust *e = new KExhaust( animation[ID_EXHAUST], &field );
      e->move( x + 2 - randDouble()*4, y + 2 - randDouble()*4 );
      e->setVelocity( dx, dy );
      e->show( );
      exhaust.append( e );
    }
}

void KAsteroidsView::processMissiles()
{
    KMissile *missile;

    // if a missile has hit a rock, remove missile and break rock into smaller
    // rocks or remove completely.
    QPtrListIterator<KMissile> it(missiles);

    for ( ; it.current(); ++it )
    {
      missile = it.current();
      missile->growOlder();

      if ( missile->expired() )
      {
          missiles.removeRef( missile );
          continue;
      }

      wrapSprite( missile );

      QCanvasItemList hits = missile->collisions( TRUE );
      QCanvasItemList::Iterator hit;
      for ( hit = hits.begin(); hit != hits.end(); ++hit )
      {
          if ( (*hit)->rtti() >= ID_ROCK_LARGE &&
             (*hit)->rtti() <= ID_ROCK_SMALL )
          {
```

```
      shotsHit++;
      rockHit( *hit );
      missiles.removeRef( missile );
      break;
     }
   }
  }
}

// - - -

void KAsteroidsView::processShip()
{
  if ( ship->isVisible() )
  {
   if ( shieldOn )
   {
     shield->show();
     reducePower( SHIELD_ON_COST );
     static int sf = 0;
     sf++;

     if ( sf % 2 )
      shield->setFrame( (shield->frame()+1) % shield->frameCount() );
     shield->move( ship->x() - 9, ship->y() - 9 );

     QCanvasItemList hits = shield->collisions( TRUE );
     QCanvasItemList::Iterator it;
     for ( it = hits.begin(); it != hits.end(); ++it )
     {
      if ( (*it)->rtti() >= ID_ROCK_LARGE &&
         (*it)->rtti() <= ID_ROCK_SMALL )
      {
        int factor;
        switch ( (*it)->rtti() )
        {
         case ID_ROCK_LARGE:
           factor = 3;
           break;

         case ID_ROCK_MEDIUM:
           factor = 2;
           break;

         default:
           factor = 1;
        }

        if ( factor > mShieldCount )
        {
         // shield not strong enough
         shieldOn = FALSE;
         break;
        }
```

```
          rockHit( *it );
          // the more shields we have the less costly
          reducePower( factor * (SHIELD_HIT_COST - mShieldCount*2) );
        }
      }
    }

    if ( !shieldOn )
    {
      shield->hide();
      QCanvasItemList hits = ship->collisions( TRUE );
      QCanvasItemList::Iterator it;
      for ( it = hits.begin(); it != hits.end(); ++it )
      {
       if ( (*it)->rtti() >= ID_ROCK_LARGE &&
           (*it)->rtti() <= ID_ROCK_SMALL )
       {
          KBit *bit;
          for ( int i = 0; i < 12; i++ )
          {
           bit = new KBit( animation[ID_BIT], &field );
           bit->move( ship->x() + 5 - randDouble() * 10,
                 ship->y() + 5 - randDouble() * 10,
                 randInt(bit->frameCount()) );
           bit->setVelocity( 1-randDouble()*2,
                    1-randDouble()*2 );
           bit->setDeath( 60 + randInt(60) );
           bit->show( );
           bits.append( bit );
          }
          ship->hide();
          shield->hide();
          emit shipKilled();
          break;
       }
      }
    }


    if ( rotateSlow )
      rotateSlow--;

    if ( rotateL )
    {
      shipAngle -= rotateSlow ? 1 : rotateRate;
      if ( shipAngle < 0 )
       shipAngle += SHIP_STEPS;
    }

    if ( rotateR )
    {
      shipAngle += rotateSlow ? 1 : rotateRate;
      if ( shipAngle >= SHIP_STEPS )
       shipAngle -= SHIP_STEPS;
```

```
}

double angle = shipAngle * PI_X_2 / SHIP_STEPS;
double cosangle = cos( angle );
double sinangle = sin( angle );

if ( brakeShip )
{
  thrustShip = FALSE;
  rotateL = FALSE;
  rotateR = FALSE;
  rotateRate = ROTATE_RATE;
  if ( fabs(shipDx) < 2.5 && fabs(shipDy) < 2.5 )
  {
   shipDx = 0.0;
   shipDy = 0.0;
   ship->setVelocity( shipDx, shipDy );
   brakeShip = FALSE;
  }
  else
  {
   double motionAngle = atan2( -shipDy, -shipDx );
   if ( angle > M_PI )
      angle -= PI_X_2;
   double angleDiff = angle - motionAngle;
   if ( angleDiff > M_PI )
      angleDiff = PI_X_2 - angleDiff;
   else if ( angleDiff < -M_PI )
      angleDiff = PI_X_2 + angleDiff;
   double fdiff = fabs( angleDiff );
   if ( fdiff > 0.08 )
   {
      if ( angleDiff > 0 )
       rotateL = TRUE;
      else if ( angleDiff < 0 )
       rotateR = TRUE;
      if ( fdiff > 0.6 )
       rotateRate = mBrakeCount + 1;
      else if ( fdiff > 0.4 )
       rotateRate = 2;
      else
       rotateRate = 1;

      if ( rotateRate > 5 )
       rotateRate = 5;
   }
   else if ( fabs(shipDx) > 1 || fabs(shipDy) > 1 )
   {
      thrustShip = TRUE;
      // we'll make braking a bit faster
      shipDx += cosangle/6 * (mBrakeCount - 1);
      shipDy += sinangle/6 * (mBrakeCount - 1);
      reducePower( BRAKE_ON_COST );
      addExhaust( ship->x() + 20 - cosangle*22,
         ship->y() + 20 - sinangle*22,
         shipDx-cosangle, shipDy-sinangle,
         mBrakeCount+1 );
   }
  }
}

if ( thrustShip )
{
  // The ship has a terminal velocity, but trying to go faster
  // still uses fuel (can go faster diagonally - don't care).
  double thrustx = cosangle/4;
  double thrusty = sinangle/4;
  if ( fabs(shipDx + thrustx) < MAX_SHIP_SPEED )
   shipDx += thrustx;
  if ( fabs(shipDy + thrusty) < MAX_SHIP_SPEED )
   shipDy += thrusty;
  ship->setVelocity( shipDx, shipDy );
  reducePower( 1 );
  addExhaust( ship->x() + 20 - cosangle*20,
      ship->y() + 20 - sinangle*20,
      shipDx-cosangle, shipDy-sinangle, 3 );
}

ship->setFrame( shipAngle >> 1 );

if ( shootShip )
{
  if ( !shootDelay && (int)missiles.count() < mShootCount + 2 )
  {
   KMissile *missile = new KMissile( animation[ID_MISSILE], &field );
   missile->move( 21+ship->x()+cosangle*21,
        21+ship->y()+sinangle*21, 0 );
   missile->setVelocity( shipDx + cosangle*MISSILE_SPEED,
           shipDy + sinangle*MISSILE_SPEED );
   missile->show( );
   missiles.append( missile );
   shotsFired++;
   reducePower( 1 );

   shootDelay = 5;
  }

  if ( shootDelay )
   shootDelay--;
}

if ( teleportShip )
{
  int ra = rand() % 10;
  if( ra == 0 )
  ra += rand() % 20;
  int xra = ra * 60 + ( (rand() % 20) * (rand() % 20) );
  int yra = ra * 50 - ( (rand() % 20) * (rand() % 20) );
```

```
        ship->move( xra, yra );
      }

      vitalsChanged = TRUE;
    }
}

// - - -

void KAsteroidsView::processPowerups()
{
    if ( !powerups.isEmpty() )
    {
      // if player gets the powerup remove it from the screen, if option
      // "Can destroy powerups" is enabled and a missile hits the powerup
      // destroy it

      KPowerup *pup;
      QPtrListIterator<KPowerup> it( powerups );

      for( ; it.current(); ++it )
      {
        pup = it.current();
        pup->growOlder();

        if( pup->expired() )
        {
         powerups.removeRef( pup );
         continue;
        }

        wrapSprite( pup );

        QCanvasItemList hits = pup->collisions( TRUE );
        QCanvasItemList::Iterator it;
        for ( it = hits.begin(); it != hits.end(); ++it )
        {
         if ( (*it) == ship )
         {
            switch( pup->rtti() )
            {
             case ID_ENERGY_POWERUP:
             shipPower += 150;
             if ( shipPower > MAX_POWER_LEVEL )
                shipPower = MAX_POWER_LEVEL;
             break;
             case ID_TELEPORT_POWERUP:
             mTeleportCount++;
             break;
             case ID_BRAKE_POWERUP:
             if ( mBrakeCount < MAX_BRAKES )
                mBrakeCount++;
             break;
             case ID_SHIELD_POWERUP:
```

```
             if ( mShieldCount < MAX_SHIELDS )
                mShieldCount++;
             break;
              case ID_SHOOT_POWERUP:
             if ( mShootCount < MAX_FIREPOWER )
                mShootCount++;
             break;
            }

            powerups.removeRef( pup );
            vitalsChanged = TRUE;
         }
         else if ( (*it) == shield )
         {
            powerups.removeRef( pup );
         }
         else if ( (*it)->rtti() == ID_MISSILE )
         {
            if ( can_destroy_powerups )
            {
              powerups.removeRef( pup );
            }
         }
        }
     }
   }        // -- if( powerups.isEmpty() )
}

// - - -

void KAsteroidsView::hideShield()
{
   shield->hide();
   mShieldCount = 0;
   shieldOn = FALSE;
}

double KAsteroidsView::randDouble()
{
   int v = rand();
   return (double)v / (double)RAND_MAX;
}

int KAsteroidsView::randInt( int range )
{
   return rand() % range;
}

void KAsteroidsView::showEvent( QShowEvent *e )
{
#if defined( QT_LICENSE_PROFESSIONAL )
   static bool wasThere = FALSE;

   if ( !wasThere ) {
     wasThere = TRUE;
     QMessageBox::information( this, tr("QCanvas demo"),
                     tr("This game has been implemented using the QCanvas class.\n"
                        "The QCanvas class is not part of the Professional Edition. Please \n"
                        "contact Trolltech if you want to upgrade to the Enterprise Edition.") );
   }
#endif

   QWidget::showEvent( e );
}
```

demo/qasteroids/view.h

```
#ifndef __AST_VIEW_H__
#define __AST_VIEW_H__

#include <qwidget.h>
#include <qptrlist.h>
#include <qintdict.h>
#include <qtimer.h>
#include <qcanvas.h>
#include "sprites.h"

#define MAX_POWER_LEVEL          1000

class KAsteroidsView : public QWidget
{
  Q_OBJECT
public:
  KAsteroidsView( QWidget *parent = 0, const char *name = 0 );
  virtual ~KAsteroidsView();

  int refreshRate;

  void reset();
  void setRockSpeed( double rs ) { rockSpeed = rs; }
  void addRocks( int num );
  void newGame();
  void endGame();
  void newShip();

  void rotateLeft( bool r ) { rotateL = r; rotateSlow = 5; }
  void rotateRight( bool r ) { rotateR = r; rotateSlow = 5; }
  void thrust( bool t ) { thrustShip = t && shipPower > 0; }
  void shoot( bool s ) { shootShip = s; shootDelay = 0; }
  void setShield( bool s );
  void teleport( bool te) { teleportShip = te && mTeleportCount; }
  void brake( bool b );
  void pause( bool p);

  void showText( const QString &text, const QColor &color, bool scroll=TRUE );
  void hideText();

  int shots() const { return shotsFired; }
  int hits() const { return shotsHit; }
```

```
    int power() const { return shipPower; }

    int teleportCount() const { return mTeleportCount; }
    int brakeCount() const { return mBrakeCount; }
    int shieldCount() const { return mShieldCount; }
    int shootCount() const { return mShootCount; }

signals:
    void shipKilled();
    void rockHit( int size );
    void rocksRemoved();
    void updateVitals();

private slots:
    void hideShield();

protected:
    bool readSprites();
    void wrapSprite( QCanvasItem * );
    void rockHit( QCanvasItem * );
    void reducePower( int val );
    void addExhaust( double x, double y, double dx, double dy, int count );
    void processMissiles();
    void processShip();
    void processPowerups();
    void processShield();
    double randDouble();
    int randInt( int range );

    virtual void resizeEvent( QResizeEvent *event );
    virtual void timerEvent( QTimerEvent * );

    void showEvent( QShowEvent * );

private:
    QCanvas field;
    QCanvasView view;
    QIntDict<QCanvasPixmapArray> animation;
    QPtrList<QCanvasSprite> rocks;
    QPtrList<KMissile> missiles;
    QPtrList<KBit> bits;
    QPtrList<KExhaust> exhaust;
    QPtrList<KPowerup> powerups;
    KShield *shield;
    QCanvasSprite *ship;
    QCanvasText *textSprite;

    bool rotateL;
    bool rotateR;
    bool thrustShip;
    bool shootShip;
    bool teleportShip;
    bool brakeShip;
    bool pauseShip;
```

```
   bool shieldOn;

   bool vitalsChanged;

   int  shipAngle;
   int  rotateSlow;
   int  rotateRate;
   int  shipPower;

   int shotsFired;
   int shotsHit;
   int shootDelay;

   int mBrakeCount;
   int mShieldCount;
   int mTeleportCount;
   int mShootCount;

   double shipDx;
   double shipDy;

   int  textDy;
   int  mFrameNum;
   bool mPaused;
   int  mTimerId;

   double rockSpeed;
   double powerupSpeed;

   bool can_destroy_powerups;

   QTimer *shieldTimer;
   bool initialized;
};

#endif
```

demo/sql/connect.ui
(Qt 실례원천참고)

demo/sql/connect.ui.h

```
#include <qsqldatabase.h>

void ConnectDialog::init()
{
   comboDriver->clear();
   comboDriver->insertStringList( QSqlDatabase::drivers() );
}

void ConnectDialog::destroy()
{
}
```

demo/sql/splex.ui.h

```
#include <qsqldriver.h>
#include <qmessagebox.h>
#include <qsqldatabase.h>
#include <qlineedit.h>
#include <qcombobox.h>
#include <qspinbox.h>
#include <qsqlerror.h>
#include <qsqlcursor.h>
#include <qsqlselectcursor.h>
#include <qdatatable.h>
#include "connect.h"

static void showError( const QSqlError& err, QWidget* parent = 0 )
{
  QString errStr ( "The database reported an error\n" );
   if ( !err.databaseText().isEmpty() )
    errStr += err.databaseText();
   if ( !err.driverText().isEmpty() )
    errStr += err.driverText();
   QMessageBox::warning( parent, "Error", errStr );
}

ConnectDialog* conDiag = 0;

void SqlEx::init()
{
   hsplit->setResizeMode( lv, QSplitter::KeepSize );
   vsplit->setResizeMode( gb, QSplitter::KeepSize );
   submitBtn->setEnabled( FALSE );
   conDiag = new ConnectDialog( this, "Connection Dialog", TRUE );
}

void SqlEx::dbConnect()
{
   if ( conDiag->exec() != QDialog::Accepted )
    return;
   if ( dt->sqlCursor() ) {
    dt->setSqlCursor( 0 );
   }
   // close old connection (if any)
   if ( QSqlDatabase::contains( "SqlEx" ) ) {
    QSqlDatabase* oldDb = QSqlDatabase::database( "SqlEx" );
    oldDb->close();
    QSqlDatabase::removeDatabase( "SqlEx" );
   }
   // open the new connection
   QSqlDatabase* db = QSqlDatabase::addDatabase( conDiag->comboDriver->currentText(), "SqlEx" );
   if ( !db ) {
    QMessageBox::warning( this, "Error", "Could not open database" );
    return;
   }
   db->setHostName( conDiag->editHostname->text() );
   db->setDatabaseName( conDiag->editDatabase->text() );
   db->setPort( conDiag->portSpinBox->value() );
   if ( !db->open( conDiag->editUsername->text(), conDiag->editPassword->text() ) ) {
    showError( db->lastError(), this );
    return;
   }
   lbl->setText( "Double-Click on a table-name to view the contents" );
   lv->clear();

   QStringList tables = db->tables();
   for ( QStringList::Iterator it = tables.begin(); it != tables.end(); ++it ) {
    QListViewItem* lvi = new QListViewItem( lv, *it );
    QSqlRecordInfo ri = db->recordInfo ( *it );
    for ( QSqlRecordInfo::Iterator it = ri.begin(); it != ri.end(); ++it ) {
       QString req;
       if ( (*it).isRequired() > 0 ) {
        req = "Yes";
       } else if ( (*it).isRequired() == 0 ) {
        req = "No";
       } else {
        req = "?";
       }
       QListViewItem* fi = new QListViewItem( lvi, (*it).name(),  +
QVariant::typeToName( (*it).type() ), req );
       lvi->insertItem( fi );
    }
    lv->insertItem( lvi );
   }
   submitBtn->setEnabled( TRUE );
}

void SqlEx::execQuery()
{
   // use a custom cursor to populate the data table
   QSqlSelectCursor* cursor = new QSqlSelectCursor( te->text(), QSqlDatabase::database( "SqlEx",
TRUE ) );
   if ( cursor->isSelect() ) {
    dt->setSqlCursor( cursor, TRUE, TRUE );
    dt->setSort( QStringList() );
    dt->refresh( QDataTable::RefreshAll );
    QString txt( "Query OK" );
    if ( cursor->size() >= 0 )
       txt += ", returned rows: " + QString::number( cursor->size() );
    lbl->setText( txt );
   } else {
    // an error occured if the cursor is not active
    if ( !cursor->isActive() ) {
       showError( cursor->lastError(), this );
    } else {
       lbl->setText( QString("Query OK, affected rows: %1").arg( cursor->numRowsAffected() ) );
    }
   }
}

void SqlEx::showTable( QListViewItem * item )
```

```
{
    // get the table name
    QListViewItem* i = item->parent();
    if ( !i ) {
     i = item;
    }

    // populate the data table
    QSqlCursor* cursor = new QSqlCursor( i->text( 0 ), TRUE, QSqlDatabase::database( "SqlEx",
TRUE ) );
    dt->setSqlCursor( cursor, TRUE, TRUE );
    dt->setSort( cursor->primaryIndex() );
    dt->refresh( QDataTable::RefreshAll );
    lbl->setText( "Displaying table " + i->text( 0 ) );
}
```

demo/textdrawing/helpwindow.cpp

```
#include "helpwindow.h"
#include <qstatusbar.h>
#include <qpixmap.h>
#include <qpopupmenu.h>
#include <qmenubar.h>
#include <qtoolbar.h>
#include <qtoolbutton.h>
#include <qiconset.h>
#include <qfile.h>
#include <qtextstream.h>
#include <qstylesheet.h>
#include <qmessagebox.h>
#include <qfiledialog.h>
#include <qapplication.h>
#include <qcombobox.h>
#include <qevent.h>
#include <qlineedit.h>
#include <qobjectlist.h>
#include <qfileinfo.h>
#include <qfile.h>
#include <qdatastream.h>
#include <qprinter.h>
#include <qsimplerichtext.h>
#include <qpainter.h>
#include <qpaintdevicemetrics.h>

#include <ctype.h>

HelpWindow::HelpWindow( const QString& home_, const QString& _path,
               QWidget* parent, const char *name )
    : QMainWindow( parent, name, WDestructiveClose ),
      pathCombo( 0 ), selectedURL()
{
    readHistory();
    readBookmarks();

    browser = new QTextBrowser( this );
```

```
browser->mimeSourceFactory()->setFilePath( _path );
browser->setFrameStyle( QFrame::Panel | QFrame::Sunken );
connect( browser, SIGNAL( textChanged() ),
      this, SLOT( textChanged() ) );

setCentralWidget( browser );

if ( !home_.isEmpty() )
  browser->setSource( home_ );

connect( browser, SIGNAL( highlighted( const QString&) ),
      statusBar(), SLOT( message( const QString&)) );

resize( 640,700 );

QPopupMenu* file = new QPopupMenu( this );
file->insertItem( tr("&New Window"), this, SLOT( newWindow() ), ALT | Key_N );
file->insertItem( tr("&Open File"), this, SLOT( openFile() ), ALT | Key_O );
file->insertItem( tr("&Print"), this, SLOT( print() ), ALT | Key_P );

// The same three icons are used twice each.
QIconSet icon_back( QPixmap("textdrawing/previous.png") );
QIconSet icon_forward( QPixmap("textdrawing/next.png") );
QIconSet icon_home( QPixmap("textdrawing/home.png") );

QPopupMenu* go = new QPopupMenu( this );
backwardId = go->insertItem( icon_back,
                 tr("&Backward"), browser, SLOT( backward() ),
                 ALT | Key_Left );
forwardId = go->insertItem( icon_forward,
                tr("&Forward"), browser, SLOT( forward() ),
                ALT | Key_Right );
go->insertItem( icon_home, tr("&Home"), browser, SLOT( home() ) );

hist = new QPopupMenu( this );
QStringList::Iterator it = history.begin();
for ( ; it != history.end(); ++it )
  mHistory[ hist->insertItem( *it ) ] = *it;
connect( hist, SIGNAL( activated( int ) ),
      this, SLOT( histChosen( int ) ) );

bookm = new QPopupMenu( this );
bookm->insertItem( tr( "Add Bookmark" ), this, SLOT( addBookmark() ) );
bookm->insertSeparator();

QStringList::Iterator it2 = bookmarks.begin();
for ( ; it2 != bookmarks.end(); ++it2 )
  mBookmarks[ bookm->insertItem( *it2 ) ] = *it2;
connect( bookm, SIGNAL( activated( int ) ),
      this, SLOT( bookmChosen( int ) ) );

menuBar()->insertItem( tr("&File"), file );
menuBar()->insertItem( tr("&Go"), go );
menuBar()->insertItem( tr( "History" ), hist );
```

```
    menuBar()->insertItem( tr( "Bookmarks" ), bookm );

    menuBar()->setItemEnabled( forwardId, FALSE);
    menuBar()->setItemEnabled( backwardId, FALSE);
    connect( browser, SIGNAL( backwardAvailable( bool ) ),
         this, SLOT( setBackwardAvailable( bool ) ) );
    connect( browser, SIGNAL( forwardAvailable( bool ) ),
         this, SLOT( setForwardAvailable( bool ) ) );


    QToolBar* toolbar = new QToolBar( this );
    addToolBar( toolbar, "Toolbar");
    QToolButton* button;

    button = new QToolButton( icon_back, tr("Backward"), "", browser, SLOT(backward()), toolbar );
    connect( browser, SIGNAL( backwardAvailable(bool) ), button, SLOT( setEnabled(bool) ) );
    button->setEnabled( FALSE );
    button = new QToolButton( icon_forward, tr("Forward"), "", browser, SLOT(forward()), toolbar );
    connect( browser, SIGNAL( forwardAvailable(bool) ), button, SLOT( setEnabled(bool) ) );
    button->setEnabled( FALSE );
    button = new QToolButton( icon_home, tr("Home"), "", browser, SLOT(home()), toolbar );

    toolbar->addSeparator();

    pathCombo = new QComboBox( TRUE, toolbar );
    connect( pathCombo, SIGNAL( activated( const QString & ) ),
         this, SLOT( pathSelected( const QString & ) ) );
    toolbar->setStretchableWidget( pathCombo );
    setRightJustification( TRUE );
    setDockEnabled( DockLeft, FALSE );
    setDockEnabled( DockRight, FALSE );

    pathCombo->insertItem( home_ );

    browser->setFocus();
}

void HelpWindow::setBackwardAvailable( bool b)
{
    menuBar()->setItemEnabled( backwardId, b);
}

void HelpWindow::setForwardAvailable( bool b)
{
    menuBar()->setItemEnabled( forwardId, b);
}

void HelpWindow::textChanged()
{
    if ( browser->documentTitle().isNull() )
     setCaption( browser->context() );
    else
     setCaption( browser->documentTitle() ) ;
```

```
  selectedURL = caption();
  if ( !selectedURL.isEmpty() && pathCombo ) {
   bool exists = FALSE;
   int i;
   for ( i = 0; i < pathCombo->count(); ++i ) {
      if ( pathCombo->text( i ) == selectedURL ) {
        exists = TRUE;
        break;
      }
   }
   if ( !exists ) {
      pathCombo->insertItem( selectedURL, 0 );
      pathCombo->setCurrentItem( 0 );
      mHistory[ hist->insertItem( selectedURL ) ] = selectedURL;
   } else
      pathCombo->setCurrentItem( i );
   selectedURL = QString::null;
  }
}

HelpWindow::~HelpWindow()
{
  history.clear();
  QMap<int, QString>::Iterator it = mHistory.begin();
  for ( ; it != mHistory.end(); ++it )
   history.append( *it );

  QFile f( QDir::currentDirPath() + "/.history" );
  f.open( IO_WriteOnly );
  QDataStream s( &f );
  s << history;
  f.close();

  bookmarks.clear();
  QMap<int, QString>::Iterator it2 = mBookmarks.begin();
  for ( ; it2 != mBookmarks.end(); ++it2 )
   bookmarks.append( *it2 );

  QFile f2( QDir::currentDirPath() + "/.bookmarks" );
  f2.open( IO_WriteOnly );
  QDataStream s2( &f2 );
  s2 << bookmarks;
  f2.close();
}

void HelpWindow::about()
{
  QMessageBox::about( this, "HelpViewer Example",
            "<p>This example implements a simple HTML help viewer "
            "using Qt's rich text capabilities</p>"
            "<p>It's just about 100 lines of C++ code, so don't expect too much :-)</p>"
            );
}
```

```
void HelpWindow::aboutQt()
{
  QMessageBox::aboutQt( this, "QBrowser" );
}

void HelpWindow::openFile()
{
#ifndef QT_NO_FILEDIALOG
  QString fn = QFileDialog::getOpenFileName( QString::null, QString::null, this );
  if ( !fn.isEmpty() )
   browser->setSource( fn );
#endif
}

void HelpWindow::newWindow()
{
  ( new HelpWindow(browser->source(), "qbrowser") )->show();
}

void HelpWindow::print()
{
#ifndef QT_NO_PRINTER
  QPrinter printer;
  printer.setFullPage(TRUE);
  if ( printer.setup() ) {
   QPainter p( &printer );
   QPaintDeviceMetrics metrics(p.device());
   int dpix = metrics.logicalDpiX();
   int dpiy = metrics.logicalDpiY();
   const int margin = 72; // pt
   QRect body(margin*dpix/72, margin*dpiy/72,
        metrics.width()-margin*dpix/72*2,
        metrics.height()-margin*dpiy/72*2 );
   QFont font("times", 10);
   QStringList filePaths = browser->mimeSourceFactory()->filePath();
   QString file;
   QStringList::Iterator it = filePaths.begin();
   for ( ; it != filePaths.end(); ++it ) {
     file = QUrl( *it, QUrl( browser->source() ).path() ).path();
     if ( QFile::exists( file ) )
      break;
     else
      file = QString::null;
   }
   if ( file.isEmpty() )
     return;
   QFile f( file );
   if ( !f.open( IO_ReadOnly ) )
     return;
   QTextStream ts( &f );
   QSimpleRichText richText( ts.read(), font, browser->context(), browser->styleSheet(),
             browser->mimeSourceFactory(), body.height() );
   richText.setWidth( &p, body.width() );
   QRect view( body );
   int page = 1;
   do {
     richText.draw( &p, body.left(), body.top(), view, colorGroup() );
     view.moveBy( 0, body.height() );
     p.translate( 0 , -body.height() );
     p.setFont( font );
     p.drawText( view.right() - p.fontMetrics().width( QString::number(page) ),
          view.bottom() + p.fontMetrics().ascent() + 5, QString::number(page) );
     if ( view.top()  >= richText.height() )
      break;
     printer.newPage();
     page++;
   } while (TRUE);
  }
#endif
}

void HelpWindow::pathSelected( const QString &_path )
{
  browser->setSource( _path );
  QMap<int, QString>::Iterator it = mHistory.begin();
  bool exists = FALSE;
  for ( ; it != mHistory.end(); ++it ) {
   if ( *it == _path ) {
     exists = TRUE;
     break;
   }
  }
  if ( !exists )
   mHistory[ hist->insertItem( _path ) ] = _path;
}

void HelpWindow::readHistory()
{
  if ( QFile::exists( QDir::currentDirPath() + "/.history" ) ) {
   QFile f( QDir::currentDirPath() + "/.history" );
   f.open( IO_ReadOnly );
   QDataStream s( &f );
   s >> history;
   f.close();
   while ( history.count() > 20 )
     history.remove( history.begin() );
  }
}

void HelpWindow::readBookmarks()
{
  if ( QFile::exists( QDir::currentDirPath() + "/.bookmarks" ) ) {
   QFile f( QDir::currentDirPath() + "/.bookmarks" );
   f.open( IO_ReadOnly );
   QDataStream s( &f );
   s >> bookmarks;
   f.close();
  }
```

```
}

void HelpWindow::histChosen( int i )
{
   if ( mHistory.contains( i ) )
    browser->setSource( mHistory[ i ] );
}

void HelpWindow::bookmChosen( int i )
{
   if ( mBookmarks.contains( i ) )
    browser->setSource( mBookmarks[ i ] );
}

void HelpWindow::addBookmark()
{
   mBookmarks[ bookm->insertItem( caption() ) ] = caption();
}
```

demo/textdrawing/helpwindow.h

```
#ifndef HELPWINDOW_H
#define HELPWINDOW_H

#include <qmainwindow.h>
#include <qtextbrowser.h>
#include <qstringlist.h>
#include <qmap.h>
#include <qdir.h>

class QComboBox;
class QPopupMenu;

class HelpWindow : public QMainWindow
{
   Q_OBJECT
public:
   HelpWindow( const QString& home_,  const QString& path, QWidget* parent = 0, const char
*name=0 );
   ~HelpWindow();

private slots:
   void setBackwardAvailable( bool );
   void setForwardAvailable( bool );

   void textChanged();
   void about();
   void aboutQt();
   void openFile();
   void newWindow();
   void print();

   void pathSelected( const QString & );
   void histChosen( int );
   void bookmChosen( int );
```

```
    void addBookmark();

private:
    void readHistory();
    void readBookmarks();

    QTextBrowser* browser;
    QComboBox *pathCombo;
    int backwardId, forwardId;
    QString selectedURL;
    QStringList history, bookmarks;
    QMap<int, QString> mHistory, mBookmarks;
    QPopupMenu *hist, *bookm;

};

#endif
```

demo/textdrawing/textedit.cpp

```
#include "textedit.h"

#include <qtextedit.h>
#include <qaction.h>
#include <qmenubar.h>
#include <qpopupmenu.h>
#include <qtoolbar.h>
#include <qtabwidget.h>
#include <qapplication.h>
#include <qfontdatabase.h>
#include <qcombobox.h>
#include <qlineedit.h>
#include <qfileinfo.h>
#include <qfile.h>
#include <qfiledialog.h>
#include <qprinter.h>
#include <qpaintdevicemetrics.h>
#include <qsimplerichtext.h>
#include <qcolordialog.h>
#include <qpainter.h>

TextEdit::TextEdit( QWidget *parent, const char *name )
    : QMainWindow( parent, name, 0 )
{
    setupFileActions();
    setupEditActions();
    setupTextActions();

    tabWidget = new QTabWidget( this );
    connect( tabWidget, SIGNAL( currentChanged( QWidget * ) ),
          this, SLOT( editorChanged( QWidget * ) ) );
    setCentralWidget( tabWidget );
}

void TextEdit::setupFileActions()
```

```
{
   QToolBar *tb = new QToolBar( this );
   QPopupMenu *menu = new QPopupMenu( this );
   menuBar()->insertItem( tr( "&File" ), menu );

   QAction *a;
   a = new QAction( tr( "New" ), QPixmap( "textdrawing/filenew.png" ), tr( "&New..." ), CTRL +
Key_N, this, "fileNew" );
   connect( a, SIGNAL( activated() ), this, SLOT( fileNew() ) );
   a->addTo( tb );
   a->addTo( menu );
   a = new QAction( tr( "Open" ), QPixmap( "textdrawing/fileopen.png" ), tr( "&Open..." ), CTRL +
Key_O, this, "fileOpen" );
   connect( a, SIGNAL( activated() ), this, SLOT( fileOpen() ) );
   a->addTo( tb );
   a->addTo( menu );
   menu->insertSeparator();
   a = new QAction( tr( "Save" ), QPixmap( "textdrawing/filesave.png" ), tr( "&Save..." ), CTRL +
Key_S, this, "fileSave" );
   connect( a, SIGNAL( activated() ), this, SLOT( fileSave() ) );
   a->addTo( tb );
   a->addTo( menu );
   a = new QAction( tr( "Save As" ), QPixmap(), tr( "Save &As..." ), 0, this, "fileSaveAs" );
   connect( a, SIGNAL( activated() ), this, SLOT( fileSaveAs() ) );
   a->addTo( menu );
   menu->insertSeparator();
   a = new QAction( tr( "Print" ), QPixmap( "textdrawing/print.png" ), tr( "&Print..." ), CTRL + Key_P,
this, "filePrint" );
   connect( a, SIGNAL( activated() ), this, SLOT( filePrint() ) );
   a->addTo( tb );
   a->addTo( menu );
   a = new QAction( tr( "Close" ), QPixmap(), tr( "&Close" ), 0, this, "fileClose" );
   connect( a, SIGNAL( activated() ), this, SLOT( fileClose() ) );
   a->addTo( menu );
}

void TextEdit::setupEditActions()
{
   QToolBar *tb = new QToolBar( this );
   QPopupMenu *menu = new QPopupMenu( this );
   menuBar()->insertItem( tr( "&Edit" ), menu );

   QAction *a;
   a = new QAction( tr( "Undo" ), QPixmap( "textdrawing/undo.png" ), tr( "&Undo" ), CTRL + Key_Z,
this, "editUndo" );
   connect( a, SIGNAL( activated() ), this, SLOT( editUndo() ) );
   a->addTo( tb );
   a->addTo( menu );
   a = new QAction( tr( "Redo" ), QPixmap( "textdrawing/redo.png" ), tr( "&Redo" ), CTRL + Key_Y,
this, "editRedo" );
   connect( a, SIGNAL( activated() ), this, SLOT( editRedo() ) );
   a->addTo( tb );
   a->addTo( menu );
   menu->insertSeparator();
   a = new QAction( tr( "Cut" ), QPixmap( "textdrawing/editcut.png" ), tr( "&Cut" ), CTRL + Key_X,
this, "editCut" );
   connect( a, SIGNAL( activated() ), this, SLOT( editCut() ) );
   a->addTo( tb );
   a->addTo( menu );
   a = new QAction( tr( "Copy" ), QPixmap( "textdrawing/editcopy.png" ), tr( "C&opy" ), CTRL +
Key_C, this, "editCopy" );
   connect( a, SIGNAL( activated() ), this, SLOT( editCopy() ) );
   a->addTo( tb );
   a->addTo( menu );
   a = new QAction( tr( "Paste" ), QPixmap( "textdrawing/editpaste.png" ), tr( "&Paste" ), CTRL +
Key_V, this, "editPaste" );
   connect( a, SIGNAL( activated() ), this, SLOT( editPaste() ) );
   a->addTo( tb );
   a->addTo( menu );
}

void TextEdit::setupTextActions()
{
   QToolBar *tb = new QToolBar( this );
   QPopupMenu *menu = new QPopupMenu( this );
   menuBar()->insertItem( tr( "For&mat" ), menu );

   comboStyle = new QComboBox( FALSE, tb );
   comboStyle->insertItem( tr("Standard") );
   comboStyle->insertItem( tr("Bullet List (Disc)") );
   comboStyle->insertItem( tr("Bullet List (Circle)") );
   comboStyle->insertItem( tr("Bullet List (Square)") );
   comboStyle->insertItem( tr("Ordered List (Decimal)") );
   comboStyle->insertItem( tr("Ordered List (Alpha lower)") );
   comboStyle->insertItem( tr("Ordered List (Alpha upper)") );
   connect( comboStyle, SIGNAL( activated( int ) ),
        this, SLOT( textStyle( int ) ) );

   comboFont = new QComboBox( TRUE, tb );
   QFontDatabase db;
   comboFont->insertStringList( db.families() );
   connect( comboFont, SIGNAL( activated( const QString & ) ),
        this, SLOT( textFamily( const QString & ) ) );
   comboFont->lineEdit()->setText( QApplication::font().family() );

   comboSize = new QComboBox( TRUE, tb );
   QValueList<int> sizes = db.standardSizes();
   QValueList<int>::Iterator it = sizes.begin();
   for ( ; it != sizes.end(); ++it )
    comboSize->insertItem( QString::number( *it ) );
   connect( comboSize, SIGNAL( activated( const QString & ) ),
        this, SLOT( textSize( const QString & ) ) );
   comboSize->lineEdit()->setText( QString::number( QApplication::font().pointSize() ) );

   actionTextBold = new QAction( tr( "Bold" ), QPixmap( "textdrawing/textbold.png" ), tr( "&Bold" ),
CTRL + Key_B, this, "textBold" );
   connect( actionTextBold, SIGNAL( activated() ), this, SLOT( textBold() ) );
   actionTextBold->addTo( tb );
```

```
    actionTextBold->addTo( menu );
    actionTextBold->setToggleAction( TRUE );
    actionTextItalic = new QAction( tr( "Italic" ), QPixmap( "textdrawing/textitalic.png" ), tr( "&Italic" ),
CTRL + Key_I, this, "textItalic" );
    connect( actionTextItalic, SIGNAL( activated() ), this, SLOT( textItalic() ) );
    actionTextItalic->addTo( tb );
    actionTextItalic->addTo( menu );
    actionTextItalic->setToggleAction( TRUE );
    actionTextUnderline = new QAction( tr( "Underline" ), QPixmap( "textdrawing/textunderline.png" ),
tr( "&Underline" ), CTRL + Key_U, this, "textUnderline" );
    connect( actionTextUnderline, SIGNAL( activated() ), this, SLOT( textUnderline() ) );
    actionTextUnderline->addTo( tb );
    actionTextUnderline->addTo( menu );
    actionTextUnderline->setToggleAction( TRUE );
    menu->insertSeparator();

    QActionGroup *grp = new QActionGroup( this );
    grp->setExclusive( TRUE );
    connect( grp, SIGNAL( selected( QAction* ) ), this, SLOT( textAlign( QAction* ) ) );

    actionAlignLeft = new QAction( tr( "Left" ), QPixmap( "textdrawing/textleft.png" ), tr( "&Left" ),
CTRL + Key_L, grp, "textLeft" );
    actionAlignLeft->addTo( tb );
    actionAlignLeft->addTo( menu );
    actionAlignLeft->setToggleAction( TRUE );
    actionAlignCenter = new QAction( tr( "Center" ), QPixmap( "textdrawing/textcenter.png" ),
tr( "C&enter" ), CTRL + Key_M, grp, "textCenter" );
    actionAlignCenter->addTo( tb );
    actionAlignCenter->addTo( menu );
    actionAlignCenter->setToggleAction( TRUE );
    actionAlignRight = new QAction( tr( "Right" ), QPixmap( "textdrawing/textright.png" ),
tr( "&Right" ), CTRL + Key_R, grp, "textRight" );
    actionAlignRight->addTo( tb );
    actionAlignRight->addTo( menu );
    actionAlignRight->setToggleAction( TRUE );
    actionAlignJustify = new QAction( tr( "Justify" ), QPixmap( "textdrawing/textjustify.png" ),
tr( "&Justify" ), CTRL + Key_J, grp, "textjustify" );
    actionAlignJustify->addTo( tb );
    actionAlignJustify->addTo( menu );
    actionAlignJustify->setToggleAction( TRUE );

    menu->insertSeparator();

    QPixmap pix( 16, 16 );
    pix.fill( black );
    actionTextColor = new QAction( tr( "Color" ), pix, tr( "&Color..." ), 0, this, "textColor" );
    connect( actionTextColor, SIGNAL( activated() ), this, SLOT( textColor() ) );
    actionTextColor->addTo( tb );
    actionTextColor->addTo( menu );
}

void TextEdit::load( const QString &f )
{
    if ( !QFile::exists( f ) )
```

```
      return;
    QTextEdit *edit = new QTextEdit( tabWidget );
    doConnections( edit );
    tabWidget->addTab( edit, QFileInfo( f ).fileName() );

    QFile fl( f );
    fl.open( IO_ReadOnly );
    QByteArray array = fl.readAll();
    array.resize( array.size() +1 );
    array[ (int)array.size() - 1 ] = '\0';
    QString text = ( f.find( "bidi.txt" ) != -1 ? QString::fromUtf8( array.data() ) :
QString::fromLatin1( array.data() ) );
    edit->setText( text );

    edit->viewport()->setFocus();
    edit->setTextFormat( Qt::RichText );
}

QTextEdit *TextEdit::currentEditor() const
{
    if ( tabWidget->currentPage() &&
       tabWidget->currentPage()->inherits( "QTextEdit" ) )
      return (QTextEdit*)tabWidget->currentPage();
    return 0;
}

void TextEdit::doConnections( QTextEdit *e )
{
    connect( e, SIGNAL( currentFontChanged( const QFont & ) ),
          this, SLOT( fontChanged( const QFont & ) ) );
    connect( e, SIGNAL( currentColorChanged( const QColor & ) ),
          this, SLOT( colorChanged( const QColor & ) ) );
    connect( e, SIGNAL( currentAlignmentChanged( int ) ),
          this, SLOT( alignmentChanged( int ) ) );
}

void TextEdit::fileNew()
{
    QTextEdit *edit = new QTextEdit( tabWidget );
    doConnections( edit );
    tabWidget->addTab( edit, tr( "noname" ) );
    tabWidget->showPage( edit );
    edit->viewport()->setFocus();
}

void TextEdit::fileOpen()
{
    QString fn = QFileDialog::getOpenFileName( QString::null, tr( "HTML-Files (*.htm *.html);;All
Files (*)" ), this );
    if ( !fn.isEmpty() )
      load( fn );
}

void TextEdit::fileSave()
```

```
{
   if ( !currentEditor() )
    return;
   QString fn;
   if ( filenames.find( currentEditor() ) == filenames.end() ) {
    fileSaveAs();
   } else {
    QFile file( *filenames.find( currentEditor() ) );
    if ( !file.open( IO_WriteOnly ) )
        return;
    QTextStream ts( &file );
    ts << currentEditor()->text();
   }
}

void TextEdit::fileSaveAs()
{
   if ( !currentEditor() )
    return;
   QString fn = QFileDialog::getSaveFileName( QString::null, tr( "HTML-Files (*.htm *.html);;All
Files (*)" ), this );
   if ( !fn.isEmpty() ) {
    filenames.replace( currentEditor(), fn );
    fileSave();
    tabWidget->setTabLabel( currentEditor(), QFileInfo( fn ).fileName() );
   }
}

void TextEdit::filePrint()
{
   if ( !currentEditor() )
    return;
#ifndef QT_NO_PRINTER
   QPrinter printer;
   printer.setFullPage(TRUE);
   QPaintDeviceMetrics screen( this );
   printer.setResolution( screen.logicalDpiY() );
   if ( printer.setup( this ) ) {
    QPainter p( &printer );
    QPaintDeviceMetrics metrics( p.device() );
    int dpix = metrics.logicalDpiX();
    int dpiy = metrics.logicalDpiY();
    const int margin = 72; // pt
    QRect body( margin * dpix / 72, margin * dpiy / 72,
            metrics.width() - margin * dpix / 72 * 2,
            metrics.height() - margin * dpiy / 72 * 2 );
    QFont font( "times", 10 );
    QSimpleRichText richText( currentEditor()->text(), font, currentEditor()->context(), currentEditor()-
>styleSheet(),
                  currentEditor()->mimeSourceFactory(), body.height() );
    richText.setWidth( &p, body.width() );
    QRect view( body );
    int page = 1;
    do {
      richText.draw( &p, body.left(), body.top(), view, colorGroup() );
      view.moveBy( 0, body.height() );
      p.translate( 0 , -body.height() );
      p.setFont( font );
      p.drawText( view.right() - p.fontMetrics().width( QString::number( page ) ),
          view.bottom() + p.fontMetrics().ascent() + 5, QString::number( page ) );
      if ( view.top()  >= richText.height() )
       break;
      printer.newPage();
      page++;
   } while (TRUE);
  }
#endif
}

void TextEdit::fileClose()
{
  delete currentEditor();
  if ( currentEditor() )
   currentEditor()->viewport()->setFocus();
}

void TextEdit::fileExit()
{
  qApp->quit();
}

void TextEdit::editUndo()
{
  if ( !currentEditor() )
   return;
  currentEditor()->undo();
}

void TextEdit::editRedo()
{
  if ( !currentEditor() )
   return;
  currentEditor()->redo();
}

void TextEdit::editCut()
{
  if ( !currentEditor() )
   return;
  currentEditor()->cut();
}

void TextEdit::editCopy()
{
  if ( !currentEditor() )
   return;
  currentEditor()->copy();
}
```

```
void TextEdit::editPaste()
{
   if ( !currentEditor() )
    return;
   currentEditor()->paste();
}

void TextEdit::textBold()
{
   if ( !currentEditor() )
    return;
   currentEditor()->setBold( actionTextBold->isOn() );
}

void TextEdit::textUnderline()
{
   if ( !currentEditor() )
    return;
   currentEditor()->setUnderline( actionTextUnderline->isOn() );
}

void TextEdit::textItalic()
{
   if ( !currentEditor() )
    return;
   currentEditor()->setItalic( actionTextItalic->isOn() );
}

void TextEdit::textFamily( const QString &f )
{
   if ( !currentEditor() )
    return;
   currentEditor()->setFamily( f );
   currentEditor()->viewport()->setFocus();
}

void TextEdit::textSize( const QString &p )
{
   if ( !currentEditor() )
    return;
   currentEditor()->setPointSize( p.toInt() );
   currentEditor()->viewport()->setFocus();
}

void TextEdit::textStyle( int i )
{
   if ( !currentEditor() )
    return;
   if ( i == 0 )
    currentEditor()->setParagType( QStyleSheetItem::DisplayBlock, QStyleSheetItem::ListDisc );
   else if ( i == 1 )
    currentEditor()->setParagType( QStyleSheetItem::DisplayListItem, QStyleSheetItem::ListDisc );
   else if ( i == 2 )
```

```
     currentEditor()->setParagType( QStyleSheetItem::DisplayListItem, QStyleSheetItem::ListCircle );
   else if ( i == 3 )
     currentEditor()->setParagType( QStyleSheetItem::DisplayListItem, QStyleSheetItem::ListSquare );
   else if ( i == 4 )
     currentEditor()->setParagType( QStyleSheetItem::DisplayListItem, QStyleSheetItem::ListDecimal );
   else if ( i == 5 )
     currentEditor()->setParagType( QStyleSheetItem::DisplayListItem,
QStyleSheetItem::ListLowerAlpha );
   else if ( i == 6 )
     currentEditor()->setParagType( QStyleSheetItem::DisplayListItem,
QStyleSheetItem::ListUpperAlpha );
   currentEditor()->viewport()->setFocus();
}

void TextEdit::textColor()
{
   if ( !currentEditor() )
     return;
   QColor col = QColorDialog::getColor( currentEditor()->color(), this );
   if ( !col.isValid() )
     return;
   currentEditor()->setColor( col );
   QPixmap pix( 16, 16 );
   pix.fill( col );
   actionTextColor->setIconSet( pix );
}

void TextEdit::textAlign( QAction *a )
{
   if ( !currentEditor() )
     return;
   if ( a == actionAlignLeft )
     currentEditor()->setAlignment( AlignLeft );
   else if ( a == actionAlignCenter )
     currentEditor()->setAlignment( AlignHCenter );
   else if ( a == actionAlignRight )
     currentEditor()->setAlignment( AlignRight );
   else if ( a == actionAlignJustify )
     currentEditor()->setAlignment( AlignJustify );
}

void TextEdit::fontChanged( const QFont &f )
{
   comboFont->lineEdit()->setText( f.family() );
   comboSize->lineEdit()->setText( QString::number( f.pointSize() ) );
   actionTextBold->setOn( f.bold() );
   actionTextItalic->setOn( f.italic() );
   actionTextUnderline->setOn( f.underline() );
}

void TextEdit::colorChanged( const QColor &c )
{
   QPixmap pix( 16, 16 );
   pix.fill( c );
```

```
    actionTextColor->setIconSet( pix );
}

void TextEdit::alignmentChanged( int a )
{
    if ( ( a == AlignAuto ) || ( a & AlignLeft ))
      actionAlignLeft->setOn( TRUE );
    else if ( ( a & AlignHCenter ) )
      actionAlignCenter->setOn( TRUE );
    else if ( ( a & AlignRight ) )
      actionAlignRight->setOn( TRUE );
    else if ( ( a & AlignJustify ) )
      actionAlignJustify->setOn( TRUE );
}

void TextEdit::editorChanged( QWidget * )
{
    if ( !currentEditor() )
      return;
    fontChanged( currentEditor()->font() );
    colorChanged( currentEditor()->color() );
    alignmentChanged( currentEditor()->alignment() );
}
```

demo/textdrawing/textedit.h

```
#ifndef TEXTEDIT_H
#define TEXTEDIT_H

#include <qmainwindow.h>
#include <qmap.h>

class QAction;
class QComboBox;
class QTabWidget;
class QTextEdit;

class TextEdit : public QMainWindow
{
    Q_OBJECT

public:
    TextEdit( QWidget *parent = 0, const char *name = 0 );

    QTextEdit *currentEditor() const;
    void load( const QString &f );

public slots:
    void fileNew();
    void fileOpen();
    void fileSave();
    void fileSaveAs();
    void filePrint();
    void fileClose();
    void fileExit();
    void editUndo();
    void editRedo();
    void editCut();
    void editCopy();
    void editPaste();

    void textBold();
    void textUnderline();
    void textItalic();
    void textFamily( const QString &f );
    void textSize( const QString &p );
    void textStyle( int s );
    void textColor();
    void textAlign( QAction *a );

    void fontChanged( const QFont &f );
    void colorChanged( const QColor &c );
    void alignmentChanged( int a );
    void editorChanged( QWidget * );

private:
    void setupFileActions();
    void setupEditActions();
    void setupTextActions();
    void doConnections( QTextEdit *e );

    QAction *actionTextBold,
     *actionTextUnderline,
     *actionTextItalic,
     *actionTextColor,
     *actionAlignLeft,
     *actionAlignCenter,
     *actionAlignRight,
     *actionAlignJustify;
    QComboBox *comboStyle,
     *comboFont,
     *comboSize;
    QTabWidget *tabWidget;
    QMap<QTextEdit*, QString> filenames;

};

#endif
```

demo/widgets/widgetsbase.ui
(Qt 실례원천참고)

demo/widgets/widgetsbase.ui.h

```
#include <qobjectlist.h>

void WidgetsBase::init()
{
    timeEdit->setTime( QTime::currentTime() );
    dateEdit->setDate( QDate::currentDate() );
}

void WidgetsBase::destroy()
{
}

void WidgetsBase::resetColors()
{
    groupBox->setPalette( palette(), FALSE );
    QObjectList *chldn = groupBox->queryList();
    if ( chldn ) {
     for(QObject *obj=chldn->first(); obj; obj = chldn->next()) {
        if(obj->isWidgetType()) {
         QWidget *w = (QWidget *)obj;
         if(!w->isTopLevel())
            w->setPalette(palette(), FALSE);
        }
     }
    }
}

void WidgetsBase::setColor( const QString & color )
{
    groupBox->setPalette( QColor( color ), FALSE );
    QObjectList *chldn = groupBox->queryList();
    if ( chldn ) {
     for(QObject *obj=chldn->first(); obj; obj = chldn->next()) {
        if(obj->isWidgetType()) {
         QWidget *w = (QWidget *)obj;
         if(!w->isTopLevel())
            w->setPalette(QColor(color), FALSE);
        }
     }
    }
}

void WidgetsBase::setColor()
{
    setColor( lineEdit->text() );
}

void WidgetsBase::updateClock()
{
    clock->setTime( timeEdit->time() );
}

void WidgetsBase::updateColorTest( const QString & color )
{
    colorTest->setPalette( QColor( color ), TRUE);
}

void WidgetsBase::updateDateTimeString()
{
    QDateTime dt;
    dt.setDate( dateEdit->date() );
    dt.setTime( timeEdit->time() );
    dateTimeLabel->setText( dt.toString() );
}
```

demo/widgets/widgetsbase_pro.h

```
#ifndef WIDGETSBASE_H
#define WIDGETSBASE_H

#include <qvariant.h>
#include <qpixmap.h>
#include <qwidget.h>

class QVBoxLayout;
class QHBoxLayout;
class QGridLayout;
class QSpacerItem;
class AnalogClock;
class QListBox;
class QListBoxItem;
class QTextEdit;
class QTabWidget;
class QIconView;
class QIconViewItem;
class QListView;
class QListViewItem;
class QGroupBox;
class QLCDNumber;
class QSlider;
class QLabel;
class QPushButton;
class QComboBox;
class QLineEdit;
class QSpinBox;
class QProgressBar;
class QButtonGroup;
class QCheckBox;
class QRadioButton;
class QDateEdit;
class QTimeEdit;

class WidgetsBase : public QWidget
{
    Q_OBJECT

public:
    WidgetsBase( QWidget* parent = 0, const char* name = 0, WFlags fl = 0 );
    ~WidgetsBase();

    QListBox* ListBox3;
    QTextEdit* TextEdit1;
    QTabWidget* TabWidget2;
    QWidget* tab;
    QIconView* IconView1;
    QWidget* tab_2;
    QListView* ListView3;
    QGroupBox* groupBox;
    QLCDNumber* lcdDisplay;
    QSlider* slider;
    QLabel* TextLabel1_2;
    QPushButton* pushButton;
    QComboBox* buttonColorBox;
    QLineEdit* lineEdit;
    QLabel* TextLabel1_2_2;
    QSpinBox* spinBox;
    QProgressBar* progressBar;
    QLabel* colorTest;
    QLabel* PixmapLabel1;
    QButtonGroup* ButtonGroup1;
    QCheckBox* CheckBox1;
    QCheckBox* CheckBox2;
    QCheckBox* CheckBox3;
    QButtonGroup* ButtonGroup2;
    QRadioButton* RadioButton3;
    QRadioButton* RadioButton4;
    QRadioButton* RadioButton2;
    QGroupBox* GroupBox1;
    AnalogClock* clock;
    QDateEdit* dateEdit;
    QTimeEdit* timeEdit;
    QLabel* dateTimeLabel;

public slots:
    virtual void resetColors();
    virtual void setColor();
    virtual void updateClock();

protected:
    QGridLayout* WidgetsBaseLayout;
    QGridLayout* tabLayout;
    QGridLayout* tabLayout_2;
    QGridLayout* groupBoxLayout;
    QHBoxLayout* Layout9;
    QGridLayout* ButtonGroup1Layout;
    QGridLayout* ButtonGroup2Layout;
    QGridLayout* GroupBox1Layout;
    QHBoxLayout* Layout5;
    QSpacerItem* Spacer2;

protected slots:
    virtual void languageChange();

    virtual void init();
    virtual void destroy();
    virtual void setColor( const QString & color );
    virtual void updateColorTest( const QString & color );
    virtual void updateDateTimeString();
```

```
private:
    QPixmap image0;
    QPixmap image1;
    QPixmap image2;
    QPixmap image3;
    QPixmap image4;
    QPixmap image5;
    QPixmap image6;
    QPixmap image7;
    QPixmap image8;
    QPixmap image9;
    QPixmap image10;
    QPixmap image11;
    QPixmap image12;
    QPixmap image13;
    QPixmap image14;
    QPixmap image15;
    QPixmap image16;
    QPixmap image17;
    QPixmap image18;
    QPixmap image19;
    QPixmap image20;
    QPixmap image21;
    QPixmap image22;
    QPixmap image23;
    QPixmap image24;
    QPixmap image25;
    QPixmap image26;
    QPixmap image27;
    QPixmap image28;
    QPixmap image29;
    QPixmap image30;
    QPixmap image31;
    QPixmap image32;
    QPixmap image33;
    QPixmap image34;
    QPixmap image35;
    QPixmap image36;
    QPixmap image37;
    QPixmap image38;
    QPixmap image39;
    QPixmap image40;
};

#endif // WIDGETSBASE_H
```

demo/widgets/widgetsbase_pro.ui
(Qt 실례원천참고)

demo/widgets/widgetsbase_pro.ui.h

```
#include <qobjectlist.h>

void WidgetsBase::init()
```

```
{
    timeEdit->setTime( QTime::currentTime() );
    dateEdit->setDate( QDate::currentDate() );
}

void WidgetsBase::destroy()
{
}

void WidgetsBase::resetColors()
{
   groupBox->setPalette( palette(), FALSE );
   if(QObjectList *chldn = groupBox->queryList()) {
    for(QObject *obj=chldn->first(); obj; obj = chldn->next()) {
       if(obj->isWidgetType()) {
        QWidget *w = (QWidget *)obj;
        if(!w->isTopLevel())
           w->setPalette(palette(), FALSE);
       }
    }
   }
}

void WidgetsBase::setColor( const QString & color )
{
   groupBox->setPalette( QColor( color ), FALSE );
   if(QObjectList *chldn = groupBox->queryList()) {
    for(QObject *obj=chldn->first(); obj; obj = chldn->next()) {
       if(obj->isWidgetType()) {
        QWidget *w = (QWidget *)obj;
        if(!w->isTopLevel())
           w->setPalette(QColor(color), FALSE);
       }
    }
   }
}

void WidgetsBase::setColor()
{
    setColor( lineEdit->text() );
}

void WidgetsBase::updateClock()
{
    clock->setTime( timeEdit->time() );
}

void WidgetsBase::updateColorTest( const QString & color )
{
    colorTest->setPalette( QColor( color ) );
}

void WidgetsBase::updateDateTimeString()
{
    QDateTime dt;
    dt.setDate( dateEdit->date() );
    dt.setTime( timeEdit->time() );
    dateTimeLabel->setText( dt.toString() );
}
```

77. SQL 프로그람

sql/sql.pro

```
TEMPLATE  = subdirs
CONFIG    += ordered
SUBDIRS   = overview \
        sqltable \
        blob
```

1) blob

sql/blob/blob.pro

```
TEMPLATE  = app
TARGET      = blob

CONFIG     += qt warn_on release
win32:CONFIG  += console

HEADERS      =
SOURCES      = main.cpp
INTERFACES   =
```

sql/blob/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlquery.h>
#include <qsqlcursor.h>
#include <qfile.h>

#define DRIVER       "QPSQL7" /* see the Qt SQL documentation for a list of available drivers */
#define DATABASE     "" /* the name of your database */
#define USER         "" /* user name with appropriate rights */
#define PASSWORD     ""  /* password for USER */
#define HOST         "" /* host on which the database is running */

int main( int argc, char ** argv )
{

   QApplication a( argc, argv, FALSE );
   QSqlDatabase * db = QSqlDatabase::addDatabase( DRIVER );
   db->setDatabaseName( DATABASE );
   db->setUserName( USER );
   db->setPassword( PASSWORD );
   db->setHostName( HOST );
   if ( !db->open() ) {
    qWarning( db->lastError().databaseText() );
    return 1;
```

```
  }

  if ( argc < 2 ) {
   qWarning( "Usage: %s <filename>", argv[0] );
   return 1;
  }

  // read a file which we want to insert into the database
  QFile f( argv[1] );
  if ( !f.open( IO_ReadOnly ) ) {
   qWarning( "Unable to open data file '%s' - exiting", argv[1] );
   return 1;
  }
  QByteArray binaryData = f.readAll();
  qWarning( "Data size: %d", binaryData.size() );

  // create a table with a binary field
  QSqlQuery q;
  if ( !q.exec( "CREATE TABLE blobexample ( id INT PRIMARY KEY, binfield LONGBLOB )" ) )
{
   qWarning( "Unable to create table - exiting" );
   return 1;
  }

  // insert a BLOB into the table
  if ( !q.prepare( "INSERT INTO blobexample ( id, binfield ) VALUES ( ?, ? )" ) ) {
   qWarning( "Unable to prepare query - exiting" );
   return 1;
  }
  q.bindValue( 0, 1 );
  q.bindValue( 1, binaryData );
  if ( !q.exec() ) {
   qWarning( "Unable to execute prepared query - exiting" );
   return 1;
  }

  // read the BLOB back from the database
  if ( !q.exec( "SELECT id, binfield FROM blobexample" ) ) {
   qWarning( "Unable to execute query - exiting" );
   return 1;
  }
  qWarning( "\nQSqlQuery:" );
  while ( q.next() ) {
   qWarning( "BLOB id: %d", q.value( 0 ).toInt() );
   qWarning( "BLOB size: %d", q.value( 1 ).toByteArray().size() );
  }

  // write another BLOB using QSqlCursor
  QSqlCursor cur( "blobexample" );
  QSqlRecord * r = cur.primeInsert();
  r->setValue( "id", 2 );
  r->setValue( "binfield", binaryData );
  if ( !cur.insert() ) {
   qWarning( "Unable to insert BLOB using QSqlCursor - exiting" );
```

```
     return 1;
    }

    // read the BLOBs back using QSqlCursor
    if ( !cur.select() ) {
     qWarning( "Unable retrieve blobexample table using QSqlCursor - exiting" );
     return 1;
    }
    qWarning( "\nQSqlCursor:" );
    while ( cur.next() ) {
     qWarning( "BLOB id: %d", cur.value( "id" ).toInt() );
     qWarning( "BLOB size: %d", cur.value( "binfield" ).toByteArray().size() );
    }

    if ( !q.exec( "DROP TABLE blobexample" ) ) {
     qWarning( "Unable to drop table - exiting" );
     return 1;
    }
    return 0;
}
```

2) overview

sql/overview/overview.pro

```
TEMPLATE = subdirs

CONFIG  += ordered

SUBDIRS = basicbrowsing \
        basicbrowsing2 \
        basicdatamanip \
        connect1 \
        create_connections \
        custom1 \
        delete \
        extract \
        form1 \
        form2 \
        insert \
        insert2 \
        navigating \
        order1 \
        order2 \
        retrieve1 \
        retrieve2 \
        subclass1 \
        subclass2 \
        subclass3 \
        subclass4 \
        subclass5 \
        table1 \
        table2 \
        table3 \
        table4 \
        update
```

overview/connection.cpp

```
#include <qsqldatabase.h>
#include "connection.h"

bool createConnections()
{

    QSqlDatabase *defaultDB = QSqlDatabase::addDatabase( DB_SALES_DRIVER );
    defaultDB->setDatabaseName( DB_SALES_DBNAME );
    defaultDB->setUserName( DB_SALES_USER );
    defaultDB->setPassword( DB_SALES_PASSWD );
    defaultDB->setHostName( DB_SALES_HOST );
    if ( ! defaultDB->open() ) {
      qWarning( "Failed to open sales database: " + defaultDB->lastError().text() );
      return FALSE;
    }

    QSqlDatabase *oracle = QSqlDatabase::addDatabase( DB_ORDERS_DRIVER, "ORACLE" );
    oracle->setDatabaseName( DB_ORDERS_DBNAME );
    oracle->setUserName( DB_ORDERS_USER );
    oracle->setPassword( DB_ORDERS_PASSWD );
    oracle->setHostName( DB_ORDERS_HOST );
    if ( ! oracle->open() ) {
      qWarning( "Failed to open orders database: " + oracle->lastError().text() );
      return FALSE;
    }

    QSqlQuery q(QString::null, defaultDB);
    q.exec("create table people (id integer primary key, name char(40))");
    q.exec("create table staff (id integer primary key, forename char(40), "
           "surname char(40), salary float, statusid integer)");
    q.exec("create table status (id integer primary key, name char(30))");
    q.exec("create table creditors (id integer primary key, forename char(40), "
           "surname char(40), city char(30))");
    q.exec("create table prices (id integer primary key, name char(40), price float)");
    q.exec("create table invoiceitem (id integer primary key, "
           "pricesid integer, quantity integer, paiddate date)");

    QSqlQuery q2(QString::null, oracle);
    q2.exec("create table people (id integer primary key, name char(40))");

    return TRUE;
}
```

overview/connection.h

```
// Enter your connection info here

#define DB_SALES_DRIVER      "QSQLITE"
#define DB_SALES_DBNAME          ":memory:"
#define DB_SALES_USER     ""
#define DB_SALES_PASSWD     ""
#define DB_SALES_HOST     ""
```

```
#define DB_ORDERS_DRIVER    "QSQLITE"
#define DB_ORDERS_DBNAME    ":memory:"
#define DB_ORDERS_USER      ""
#define DB_ORDERS_PASSWD    ""
#define DB_ORDERS_HOST      ""

bool createConnections();
```

(1) basicbrowsing
sql/overview/basicbrowsing/basicbrowsing.pro

```
TEMPLATE  = app
CONFIG    += qt warn_on release
HEADERS   =
SOURCES   = main.cpp ../connection.cpp
```

sql/overview/basicbrowsing/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlquery.h>
#include "../connection.h"

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

    if ( createConnections() ) {
      QSqlDatabase *oracledb = QSqlDatabase::database( "ORACLE" );
      // Copy data from the oracle database to the ODBC (default)
      // database
      QSqlQuery target;
      QSqlQuery query( "SELECT id, name FROM people", oracledb );
      if ( query.isActive() ) {
          while ( query.next() ) {
            target.exec( "INSERT INTO people ( id, name ) VALUES ( " +
                         query.value(0).toString() +
                         ", '" + query.value(1).toString() +  "' )" );
          }
      }
    }

    return 0;
}
```

(2) basicbrowsing2
sql/overview/basicbrowsing2/basicbrowsing2.pro

```
TEMPLATE  = app
CONFIG    += qt warn_on release
HEADERS   =
SOURCES   = main.cpp ../connection.cpp
```

sql/overview/basicbrowsing2/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
```

```
#include <qsqlquery.h>
#include "../connection.h"

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

   if ( createConnections() ) {
     QSqlDatabase *oracledb = QSqlDatabase::database( "ORACLE" );
     // Copy data from the oracle database to the ODBC (default)
     // database
     QSqlQuery target;
     QSqlQuery query( "SELECT id, name FROM people", oracledb );
     int count = 0;
       if ( query.isActive() ) {
          while ( query.next() ) {
             target.exec( "INSERT INTO people ( id, name ) VALUES ( " +
                        query.value(0).toString() +
                        ", '" + query.value(1).toString() +  "' )" );
             if ( target.isActive() )
                count += target.numRowsAffected();
          }
       }
   }

   return 0;
}
```

(3) basicdatamanip

sql/overview/basicdatamanip/basicdatamanip.pro

```
TEMPLATE  = app
CONFIG      += qt warn_on release
HEADERS          =
SOURCES          = main.cpp ../connection.cpp
```

sql/overview/basicdatamanip/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlquery.h>
#include "../connection.h"

bool createConnections();

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

   int rows = 0;

   if ( createConnections() ) {
     QSqlQuery query( "INSERT INTO staff ( id, forename, surname, salary ) "
              "VALUES ( 1155, 'Ginger', 'Davis', 50000 )" );
     if ( query.isActive() ) rows += query.numRowsAffected() ;
```

```
      query.exec( "UPDATE staff SET salary=60000 WHERE id=1155" );
      if ( query.isActive() ) rows += query.numRowsAffected() ;

      query.exec( "DELETE FROM staff WHERE id=1155" );
      if ( query.isActive() ) rows += query.numRowsAffected() ;
    }

    return ( rows == 3 ) ? 0 : 1;
}
```

(4) connect1
sql/overview/connect1/connect1.pro

```
TEMPLATE  = app
CONFIG    += qt warn_on release
HEADERS      =
SOURCES      = main.cpp ../connection.cpp
```

sql/overview/connect1/connect1.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include "../connection.h"

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

    QSqlDatabase *defaultDB = QSqlDatabase::addDatabase( DB_SALES_DRIVER );
    defaultDB->setDatabaseName( DB_SALES_DBNAME );
    defaultDB->setUserName( DB_SALES_USER );
    defaultDB->setPassword( DB_SALES_PASSWD );
    defaultDB->setHostName( DB_SALES_HOST );

    if ( defaultDB->open() ) {
      // Database successfully opened; we can now issue SQL commands.
    }

    return 0;
}
```

(5) create_connections
sql/overview/create_connections/create_connections.pro

```
TEMPLATE  = app
CONFIG    += qt warn_on release
HEADERS      =
SOURCES      = main.cpp ../connection.cpp
```

sql/overview/create_connections/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include "../connection.h"

int main( int argc, char *argv[] )
{
    QApplication app( argc, argv, FALSE );
    if ( createConnections() ) {
     // Databases successfully opened; get pointers to them:
     QSqlDatabase *oracledb = QSqlDatabase::database( "ORACLE" );
     // Now we can now issue SQL commands to the oracle connection
     // or to the default connection
    }

    return 0;
}
```

(6) custom1

sql/overview/custom1/custom1.pro

```
TEMPLATE = app
CONFIG  += qt warn_on release
HEADERS     = main.h
SOURCES     = main.cpp ../connection.cpp
```

sql/overview/custom1/main.cpp

```
#include "main.h"

CustomEdit::CustomEdit( QWidget *parent, const char *name ) :   QLineEdit( parent, name )
{
    connect( this, SIGNAL(textChanged(const QString &)),  this, SLOT(changed(const QString &)) );
}

void CustomEdit::changed( const QString &line )
{
    setUpperLine( line );
}

void CustomEdit::setUpperLine( const QString &line )
{
    upperLineText = line.upper();
    setText( upperLineText );
}

QString CustomEdit::upperLine() const
{
    return upperLineText;
}

FormDialog::FormDialog()
{
    QLabel *forenameLabel  = new QLabel( "Forename:", this );
    CustomEdit      *forenameEdit     = new CustomEdit( this );
    QLabel *surnameLabel   = new QLabel( "Surname:", this );
    CustomEdit      *surnameEdit= new CustomEdit( this );
    QLabel *salaryLabel  = new QLabel( "Salary:", this );
    QLineEdit  *salaryEdit     = new QLineEdit( this );
    salaryEdit->setAlignment( Qt::AlignRight );
    QPushButton *saveButton     = new QPushButton( "&Save", this );
    connect( saveButton, SIGNAL(clicked()), this, SLOT(save()) );
    QGridLayout *grid = new QGridLayout( this );
    grid->addWidget( forenameLabel, 0, 0 );
    grid->addWidget( forenameEdit,  0, 1 );
    grid->addWidget( surnameLabel,  1, 0 );
    grid->addWidget( surnameEdit,   1, 1 );
    grid->addWidget( salaryLabel,   2, 0 );
    grid->addWidget( salaryEdit,    2, 1 );
    grid->addWidget( saveButton,    3, 0 );
    grid->activate();

    staffCursor = new QSqlCursor( "staff" );
    staffCursor->setTrimmed( "forename", TRUE );
    staffCursor->setTrimmed( "surname",  TRUE );
    idIndex = staffCursor->index( "id" );
    staffCursor->select( idIndex );
    staffCursor->first();

    propMap = new QSqlPropertyMap;
    propMap->insert( forenameEdit->className(), "upperLine" );

    sqlForm = new QSqlForm( this );
    sqlForm->setRecord( staffCursor->primeUpdate() );
    sqlForm->installPropertyMap( propMap );
    sqlForm->insert( forenameEdit, "forename" );
    sqlForm->insert( surnameEdit, "surname" );
    sqlForm->insert( salaryEdit, "salary" );
    sqlForm->readFields();
}

FormDialog::~FormDialog()
{
    delete staffCursor;
}

void FormDialog::save()
{
    sqlForm->writeFields();
    staffCursor->update();
    staffCursor->select( idIndex );
    staffCursor->first();
}

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

    if ( ! createConnections() )
      return 1;

    FormDialog *formDialog = new FormDialog();
    formDialog->show();
    app.setMainWidget( formDialog );

    return app.exec();
```

```
}
```

sql/overview/custom1/main.h

```
#include <qapplication.h>
#include <qdialog.h>
#include <qlabel.h>
#include <qlayout.h>
#include <qlineedit.h>
#include <qpushbutton.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include <qsqlform.h>
#include <qsqlpropertymap.h>
#include "../connection.h"

class CustomEdit : public QLineEdit
{
  Q_OBJECT
  Q_PROPERTY( QString upperLine READ upperLine WRITE setUpperLine )
  public:
   CustomEdit( QWidget *parent=0, const char *name=0 );
   QString upperLine() const;
   void setUpperLine( const QString &line );
  public slots:
   void changed( const QString &line );
  private:
   QString upperLineText;
};

class FormDialog : public QDialog
{
  Q_OBJECT
  public:
   FormDialog();
   ~FormDialog();
  public slots:
   void save();
  private:
   QSqlCursor *staffCursor;
   QSqlForm *sqlForm;
   QSqlPropertyMap *propMap;
   QSqlIndex idIndex;
};
```

sql/overview/custom1/moc_main.cpp

```
#undef QT_NO_COMPAT
#include "main.h"
#include <qmetaobject.h>
#include <qapplication.h>

#include <private/qucomextra_p.h>
#if !defined(Q_MOC_OUTPUT_REVISION) || (Q_MOC_OUTPUT_REVISION != 26)
#error "This file was generated using the moc from 3.3.6. It"
#error "cannot be used with the include files from this version of Qt."
```

```
#error "(The moc has changed too much.)"
#endif

#include <qvariant.h>
const char *CustomEdit::className() const
{
   return "CustomEdit";
}

QMetaObject *CustomEdit::metaObj = 0;
static QMetaObjectCleanUp cleanUp_CustomEdit( "CustomEdit", &CustomEdit::staticMetaObject );

#ifndef QT_NO_TRANSLATION
QString CustomEdit::tr( const char *s, const char *c )
{
   if ( qApp )
    return qApp->translate( "CustomEdit", s, c, QApplication::DefaultCodec );
   else
    return QString::fromLatin1( s );
}
#ifndef QT_NO_TRANSLATION_UTF8
QString CustomEdit::trUtf8( const char *s, const char *c )
{
   if ( qApp )
    return qApp->translate( "CustomEdit", s, c, QApplication::UnicodeUTF8 );
   else
    return QString::fromUtf8( s );
}
#endif // QT_NO_TRANSLATION_UTF8

#endif // QT_NO_TRANSLATION

QMetaObject* CustomEdit::staticMetaObject()
{
   if ( metaObj )
    return metaObj;
   QMetaObject* parentObject = QLineEdit::staticMetaObject();
   static const QUParameter param_slot_0[] = {
    { "line", &static_QUType_QString, 0, QUParameter::In }
   };
   static const QUMethod slot_0 = {"changed", 1, param_slot_0 };
   static const QMetaData slot_tbl[] = {
    { "changed(const QString&)", &slot_0, QMetaData::Public }
   };
#ifndef QT_NO_PROPERTIES
   static const QMetaProperty props_tbl[1] = {
    { "QString","upperLine", 0x3000103, &CustomEdit::metaObj, 0, -1 }
   };
#endif // QT_NO_PROPERTIES
   metaObj = QMetaObject::new_metaobject(
    "CustomEdit", parentObject,
    slot_tbl, 1,
    0, 0,
#ifndef QT_NO_PROPERTIES
```

```
    props_tbl, 1,
    0, 0,
#endif // QT_NO_PROPERTIES
    0, 0 );
   cleanUp_CustomEdit.setMetaObject( metaObj );
   return metaObj;
}

void* CustomEdit::qt_cast( const char* clname )
{
   if ( !qstrcmp( clname, "CustomEdit" ) )
    return this;
   return QLineEdit::qt_cast( clname );
}

bool CustomEdit::qt_invoke( int _id, QUObject* _o )
{
   switch ( _id - staticMetaObject()->slotOffset() ) {
   case 0: changed((const QString&)static_QUType_QString.get(_o+1)); break;
   default:
    return QLineEdit::qt_invoke( _id, _o );
   }
   return TRUE;
}

bool CustomEdit::qt_emit( int _id, QUObject* _o )
{
   return QLineEdit::qt_emit(_id,_o);
}
#ifndef QT_NO_PROPERTIES

bool CustomEdit::qt_property( int id, int f, QVariant* v)
{
   switch ( id - staticMetaObject()->propertyOffset() ) {
   case 0: switch( f ) {
    case 0: setUpperLine(v->asString()); break;
    case 1: *v = QVariant( this->upperLine() ); break;
    case 3: case 4: case 5: break;
    default: return FALSE;
   } break;
   default:
    return QLineEdit::qt_property( id, f, v );
   }
   return TRUE;
}

bool CustomEdit::qt_static_property( QObject* , int , int , QVariant* ){ return FALSE; }
#endif // QT_NO_PROPERTIES

const char *FormDialog::className() const
{
   return "FormDialog";
}
```

```
QMetaObject *FormDialog::metaObj = 0;
static QMetaObjectCleanUp cleanUp_FormDialog( "FormDialog", &FormDialog::staticMetaObject );

#ifndef QT_NO_TRANSLATION
QString FormDialog::tr( const char *s, const char *c )
{
   if ( qApp )
    return qApp->translate( "FormDialog", s, c, QApplication::DefaultCodec );
   else
    return QString::fromLatin1( s );
}
#ifndef QT_NO_TRANSLATION_UTF8
QString FormDialog::trUtf8( const char *s, const char *c )
{
   if ( qApp )
    return qApp->translate( "FormDialog", s, c, QApplication::UnicodeUTF8 );
   else
    return QString::fromUtf8( s );
}
#endif // QT_NO_TRANSLATION_UTF8

#endif // QT_NO_TRANSLATION

QMetaObject* FormDialog::staticMetaObject()
{
   if ( metaObj )
    return metaObj;
   QMetaObject* parentObject = QDialog::staticMetaObject();
   static const QUMethod slot_0 = {"save", 0, 0 };
   static const QMetaData slot_tbl[] = {
    { "save()", &slot_0, QMetaData::Public }
   };
   metaObj = QMetaObject::new_metaobject(
    "FormDialog", parentObject,
    slot_tbl, 1,
    0, 0,
#ifndef QT_NO_PROPERTIES
    0, 0,
    0, 0,
#endif // QT_NO_PROPERTIES
    0, 0 );
   cleanUp_FormDialog.setMetaObject( metaObj );
   return metaObj;
}

void* FormDialog::qt_cast( const char* clname )
{
   if ( !qstrcmp( clname, "FormDialog" ) )
    return this;
   return QDialog::qt_cast( clname );
}

bool FormDialog::qt_invoke( int _id, QUObject* _o )
{
    switch ( _id - staticMetaObject()->slotOffset() ) {
    case 0: save(); break;
    default:
      return QDialog::qt_invoke( _id, _o );
    }
    return TRUE;
}

bool FormDialog::qt_emit( int _id, QUObject* _o )
{
    return QDialog::qt_emit(_id,_o);
}
#ifndef QT_NO_PROPERTIES

bool FormDialog::qt_property( int id, int f, QVariant* v)
{
    return QDialog::qt_property( id, f, v);
}

bool FormDialog::qt_static_property( QObject* , int , int , QVariant* ){ return FALSE; }
#endif // QT_NO_PROPERTIES
```

(7) delete

sql/overview/delete/delete.pro

```
TEMPLATE  = app
CONFIG       += qt warn_on release
HEADERS         =
SOURCES         = main.cpp ../connection.cpp
```

sql/overview/delete/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include "../connection.h"

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

    if ( createConnections() ) {
      QSqlCursor cur( "prices" );
      cur.select( "id=999" );
      if ( cur.next() ) {
          cur.primeDelete();
          cur.del();
      }
    }

    return 0;
}
```

(8) extract

sql/overview/extract.pro

```
TEMPLATE  = app
```

```
CONFIG       += qt warn_on release
HEADERS       =
SOURCES       = main.cpp ../connection.cpp
```

sql/overview/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include "../connection.h"

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

    if ( createConnections() ) {
      QSqlCursor cur( "creditors" );

      QStringList orderFields = QStringList() << "surname" << "forename";
      QSqlIndex order = cur.index( orderFields );

      QStringList filterFields = QStringList() << "surname" << "city";
      QSqlIndex filter = cur.index( filterFields );
      cur.setValue( "surname", "Chirac" );
      cur.setValue( "city", "Paris" );

      cur.select( filter, order );

      while ( cur.next() ) {
          int id = cur.value( "id" ).toInt();
          QString name = cur.value( "forename" ).toString() + " " +
                    cur.value( "surname" ).toString();
          qDebug( QString::number( id ) + ": " + name );
      }
    }

    return 0;
}
```

(9) form1

sql/overview/form1.pro

```
TEMPLATE = app
CONFIG  += qt warn_on release
HEADERS     =
SOURCES     = main.cpp ../connection.cpp
```

sql/overview/main.cpp

```
#include <qapplication.h>
#include <qdialog.h>
#include <qlabel.h>
#include <qlayout.h>
#include <qlineedit.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include <qsqlform.h>
```

```
#include "../connection.h"

class FormDialog : public QDialog
{
   public:
    FormDialog();
};

FormDialog::FormDialog()
{
   QLabel *forenameLabel   = new QLabel( "Forename:", this );
   QLabel *forenameDisplay = new QLabel( this );
   QLabel *surnameLabel    = new QLabel( "Surname:", this );
   QLabel *surnameDisplay  = new QLabel( this );
   QLabel *salaryLabel      = new QLabel( "Salary:", this );
   QLineEdit *salaryEdit   = new QLineEdit( this );

   QGridLayout *grid = new QGridLayout( this );
   grid->addWidget( forenameLabel,   0, 0 );
   grid->addWidget( forenameDisplay,0, 1 );
   grid->addWidget( surnameLabel,     1, 0 );
   grid->addWidget( surnameDisplay,  1, 1 );
   grid->addWidget( salaryLabel,   2, 0 );
   grid->addWidget( salaryEdit,      2, 1 );
   grid->activate();

   QSqlCursor staffCursor( "staff" );
   staffCursor.select();
   staffCursor.next();

   QSqlForm sqlForm( this );
   sqlForm.setRecord( staffCursor.primeUpdate() );
   sqlForm.insert( forenameDisplay, "forename" );
   sqlForm.insert( surnameDisplay, "surname" );
   sqlForm.insert( salaryEdit, "salary" );
   sqlForm.readFields();
}

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

   if ( ! createConnections() ) return 1;

   FormDialog *formDialog = new FormDialog();
   formDialog->show();
   app.setMainWidget( formDialog );

   return app.exec();
}
```

(10) form2

sql/overview/form2.pro

```
TEMPLATE = app
```

```
CONFIG  += qt warn_on release
HEADERS     = main.h
SOURCES     = main.cpp ../connection.cpp
```

sql/overview/main.cpp

```
#include "main.h"

FormDialog::FormDialog()
    : staffCursor( "staff" )
{
    staffCursor.setTrimmed( "forename", TRUE );
    staffCursor.setTrimmed( "surname",  TRUE );

    QLabel *forenameLabel = new QLabel( "Forename:", this );
    QLineEdit  *forenameEdit  = new QLineEdit( this );
    QLabel *surnameLabel  = new QLabel( "Surname:", this );
    QLineEdit  *surnameEdit   = new QLineEdit( this );
    QLabel *salaryLabel   = new QLabel( "Salary:", this );
    QLineEdit  *salaryEdit    = new QLineEdit( this );
    QPushButton *saveButton    = new QPushButton( "&Save", this );
    connect( saveButton, SIGNAL(clicked()), this, SLOT(save()) );

    QGridLayout *grid = new QGridLayout( this );
    grid->addWidget( forenameLabel, 0, 0 );
    grid->addWidget( forenameEdit,  0, 1 );
    grid->addWidget( surnameLabel,  1, 0 );
    grid->addWidget( surnameEdit,   1, 1 );
    grid->addWidget( salaryLabel,   2, 0 );
    grid->addWidget( salaryEdit,    2, 1 );
    grid->addWidget( saveButton,    3, 0 );
    grid->activate();

    idIndex = staffCursor.index( "id" );
    staffCursor.select( idIndex );
    staffCursor.first();

    sqlForm = new QSqlForm( this );
    sqlForm->setRecord( staffCursor.primeUpdate() );
    sqlForm->insert( forenameEdit, "forename" );
    sqlForm->insert( surnameEdit, "surname" );
    sqlForm->insert( salaryEdit, "salary" );
    sqlForm->readFields();
}

FormDialog::~FormDialog()
{
}

void FormDialog::save()
{
    sqlForm->writeFields();
    staffCursor.update();
    staffCursor.select( idIndex );
    staffCursor.first();
}

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

    if ( ! createConnections() )
     return 1;

    FormDialog *formDialog = new FormDialog();
    formDialog->show();
    app.setMainWidget( formDialog );

    return app.exec();
}
```

sql/overview/main.h

```
#include <qapplication.h>
#include <qdialog.h>
#include <qlabel.h>
#include <qlayout.h>
#include <qlineedit.h>
#include <qpushbutton.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include <qsqlform.h>
#include "../connection.h"

class FormDialog : public QDialog
{
   Q_OBJECT
   public:
    FormDialog();
    ~FormDialog();
   public slots:
    void save();
   private:
    QSqlCursor staffCursor;
    QSqlForm *sqlForm;
    QSqlIndex idIndex;
};
```

sql/overview/moc_main.cpp

```
#undef QT_NO_COMPAT
#include "main.h"
#include <qmetaobject.h>
#include <qapplication.h>

#include <private/qucomextra_p.h>
#if !defined(Q_MOC_OUTPUT_REVISION) || (Q_MOC_OUTPUT_REVISION != 26)
#error "This file was generated using the moc from 3.3.6. It"
#error "cannot be used with the include files from this version of Qt."
#error "(The moc has changed too much.)"
#endif
```

```
const char *FormDialog::className() const
{
  return "FormDialog";
}

QMetaObject *FormDialog::metaObj = 0;
static QMetaObjectCleanUp cleanUp_FormDialog( "FormDialog", &FormDialog::staticMetaObject );

#ifndef QT_NO_TRANSLATION
QString FormDialog::tr( const char *s, const char *c )
{
  if ( qApp )
   return qApp->translate( "FormDialog", s, c, QApplication::DefaultCodec );
  else
   return QString::fromLatin1( s );
}
#ifndef QT_NO_TRANSLATION_UTF8
QString FormDialog::trUtf8( const char *s, const char *c )
{
  if ( qApp )
   return qApp->translate( "FormDialog", s, c, QApplication::UnicodeUTF8 );
  else
   return QString::fromUtf8( s );
}
#endif // QT_NO_TRANSLATION_UTF8

#endif // QT_NO_TRANSLATION

QMetaObject* FormDialog::staticMetaObject()
{
  if ( metaObj )
   return metaObj;
  QMetaObject* parentObject = QDialog::staticMetaObject();
  static const QUMethod slot_0 = {"save", 0, 0 };
  static const QMetaData slot_tbl[] = {
   { "save()", &slot_0, QMetaData::Public }
  };
  metaObj = QMetaObject::new_metaobject(
   "FormDialog", parentObject,
   slot_tbl, 1,
   0, 0,
#ifndef QT_NO_PROPERTIES
   0, 0,
   0, 0,
#endif // QT_NO_PROPERTIES
   0, 0 );
  cleanUp_FormDialog.setMetaObject( metaObj );
  return metaObj;
}

void* FormDialog::qt_cast( const char* clname )
{
  if ( !qstrcmp( clname, "FormDialog" ) )
```

```
    return this;
  return QDialog::qt_cast( clname );
}

bool FormDialog::qt_invoke( int _id, QUObject* _o )
{
  switch ( _id - staticMetaObject()->slotOffset() ) {
  case 0: save(); break;
  default:
    return QDialog::qt_invoke( _id, _o );
  }
  return TRUE;
}

bool FormDialog::qt_emit( int _id, QUObject* _o )
{
  return QDialog::qt_emit(_id,_o);
}
#ifndef QT_NO_PROPERTIES

bool FormDialog::qt_property( int id, int f, QVariant* v)
{
  return QDialog::qt_property( id, f, v);
}

bool FormDialog::qt_static_property( QObject* , int , int , QVariant* ){ return FALSE; }
#endif // QT_NO_PROPERTIES
```

(11) insert

sql/overview/insert.pro

```
TEMPLATE  = app
CONFIG      += qt warn_on release
HEADERS       =
SOURCES       = main.cpp ../connection.cpp
```

sql/overview/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include "../connection.h"

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

  if ( createConnections() ) {
    int count = 0;
    QSqlCursor cur( "prices" );
    QStringList names = QStringList() <<
       "Screwdriver" << "Hammer" << "Wrench" << "Saw";
    int id = 20;
    for ( QStringList::Iterator name = names.begin();
          name != names.end(); ++name ) {
       QSqlRecord *buffer = cur.primeInsert();
```

```
     buffer->setValue( "id", id );
     buffer->setValue( "name", *name );
     buffer->setValue( "price", 100.0 + (double)id );
     count += cur.insert();
     id++;
   }
  }
  return 0;
}
```

(12) insert2
sql/overview/insert2.pro

```
TEMPLATE  = app
CONFIG      += qt warn_on release
HEADERS        =
SOURCES        = main.cpp ../connection.cpp
```

sql/overview/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include "../connection.h"

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

  if ( createConnections() ) {
   QSqlCursor cur( "prices" );
   QSqlRecord *buffer = cur.primeInsert();
   buffer->setValue( "id",     53981 );
   buffer->setValue( "name",  "Thingy" );
   buffer->setValue( "price", 105.75 );
   cur.insert();
  }

  return 0;
}
```

(13) naigating
sql/overview/naigating.pro

```
TEMPLATE  = app
CONFIG      += qt warn_on release
HEADERS        =
SOURCES        = main.cpp ../connection.cpp
```

sql/overview/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlquery.h>
#include "../connection.h"

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

    if ( createConnections() ) {
      QSqlQuery query( "SELECT id, name FROM people ORDER BY name" );
      if ( ! query.isActive() ) return 1; // Query failed
      int i;
      i = query.size();        // In this example we have 9 records; i == 9.
      query.first();           // Moves to the first record.
      i = query.at();          // i == 0
      query.last();            // Moves to the last record.
      i = query.at();          // i == 8
      query.seek( query.size() / 2 ); // Moves to the middle record.
      i = query.at();          // i == 4
    }

    return 0;
}
```

(14) order1

sql/overview/order1.pro

```
TEMPLATE  = app
CONFIG      += qt warn_on release
HEADERS       =
SOURCES       = main.cpp ../connection.cpp
```

sql/overview/order1/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include "../connection.h"

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

    if ( createConnections() ) {
        QSqlCursor cur( "staff" );
        QStringList fields = QStringList() << "surname" << "forename";
        QSqlIndex order = cur.index( fields );
        cur.select( order );
        while ( cur.next() ) {
          qDebug( cur.value( "id" ).toString() + ": " +
              cur.value( "surname" ).toString() + " " +
              cur.value( "forename" ).toString() );
        }
    }

    return 0;
}
```

(15) sql/overview/order2

sql/overview/order2/order2.pro

```
TEMPLATE  = app
CONFIG      += qt warn_on release
```

```
HEADERS         =
SOURCES         = main.cpp ../connection.cpp
```

sql/overview/order2/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include "../connection.h"

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

   if ( createConnections() ) {
    QSqlCursor cur( "staff" );
    QStringList fields = QStringList() << "id" << "forename";
    QSqlIndex order = cur.index( fields );
    QSqlIndex filter = cur.index( "surname" );
    cur.setValue( "surname", "Bloggs" );
    cur.select( filter, order );
    while ( cur.next() ) {
        qDebug( cur.value( "id" ).toString() + ": " +
              cur.value( "surname" ).toString() + " " +
              cur.value( "forename" ).toString() );
      }
   }

   return 0;
}
```

(16) retrieve1
sql/overview/retrieve1/retrieve1.pro

```
TEMPLATE  = app
CONFIG       += qt warn_on release
HEADERS         =
SOURCES         = main.cpp ../connection.cpp
```

sql/overview/retrieve1/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlquery.h>
#include "../connection.h"

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

   if ( createConnections() ) {
    QSqlQuery query( "SELECT id, surname FROM staff" );
    if ( query.isActive() ) {
        while ( query.next() ) {
         qDebug( query.value(0).toString() + ": " +
                query.value(1).toString() );
        }
```

```
      }
    }

    return 0;
}
```

(17) retrieve2

sql/overview/retrieve2/retrieve2.pro

```
TEMPLATE  = app
CONFIG       += qt warn_on release
HEADERS          =
SOURCES          = main.cpp ../connection.cpp
```

sql/overview/retrieve2/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include "../connection.h"

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

    if ( createConnections() ) {
      QSqlCursor cur( "staff" ); // Specify the table/view name
      cur.select(); // We'll retrieve every record
      while ( cur.next() ) {
          qDebug( cur.value( "id" ).toString() + ": " +
                cur.value( "surname" ).toString() + " " +
                cur.value( "salary" ).toString() );
      }
    }

    return 0;
}
```

(18) subclass1

sql/overview/subclass1/subclass1.pro

```
TEMPLATE = app
CONFIG  += qt warn_on release
HEADERS     =
SOURCES     = main.cpp ../connection.cpp
```

sql/overview/subclass1/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include <qdatatable.h>
#include "../connection.h"

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

```
   if ( createConnections() ) {
     QSqlCursor invoiceItemCursor( "invoiceitem" );

     QDataTable *invoiceItemTable = new QDataTable( &invoiceItemCursor );

     app.setMainWidget( invoiceItemTable );

     invoiceItemTable->addColumn( "pricesid", "PriceID" );
     invoiceItemTable->addColumn( "quantity", "Quantity" );
     invoiceItemTable->addColumn( "paiddate", "Paid" );

     invoiceItemTable->refresh();
     invoiceItemTable->show();

     return app.exec();
   }

   return 1;
}
```

(19) subclass2
sql/overview/subclass2/subclass2.pro

```
TEMPLATE = app
CONFIG  += qt warn_on release
HEADERS     =
SOURCES     = main.cpp ../connection.cpp
```

sql/overview/subclass2/main.cpp

```
#include "main.h"
#include <qdatatable.h>

InvoiceItemCursor::InvoiceItemCursor() :
   QSqlCursor( "invoiceitem" )
{
   // NOOP
}

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

   if ( createConnections() ) {
     InvoiceItemCursor invoiceItemCursor;

     QDataTable *invoiceItemTable = new QDataTable( &invoiceItemCursor );

     app.setMainWidget( invoiceItemTable );

     invoiceItemTable->addColumn( "pricesid", "PriceID" );
     invoiceItemTable->addColumn( "quantity", "Quantity" );
     invoiceItemTable->addColumn( "paiddate", "Paid" );

     invoiceItemTable->refresh();
     invoiceItemTable->show();
   return app.exec();
  }

  return 1;
}
```

sql/overview/subclass2/main.h

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include "../connection.h"

class QSqlRecord;

class InvoiceItemCursor : public QSqlCursor
{
  public:
   InvoiceItemCursor();
};
```

(20) subclass3

sql/overview/subclass3/subclass3.pro

```
TEMPLATE = app
CONFIG  += qt warn_on release
HEADERS     =
SOURCES     = main.cpp ../connection.cpp
```

sql/overview/subclass3/main.cpp

```
#include "main.h"
#include <qdatatable.h>

InvoiceItemCursor::InvoiceItemCursor() :
  QSqlCursor( "invoiceitem" )
{
  QSqlFieldInfo productName( "productname", QVariant::String );
  append( productName );
  setCalculated( productName.name(), TRUE );
}

QVariant InvoiceItemCursor::calculateField( const QString & name )
{
  if ( name == "productname" ) {
   QSqlQuery query( "SELECT name FROM prices WHERE id=" +
         field( "pricesid" )->value().toString() );
   if ( query.next() )
     return query.value( 0 );
  }

  return QVariant( QString::null );
}

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

  if ( createConnections() ) {
   InvoiceItemCursor invoiceItemCursor;

   QDataTable *invoiceItemTable = new QDataTable( &invoiceItemCursor );

   app.setMainWidget( invoiceItemTable );

   invoiceItemTable->addColumn( "productname", "Product" );
   invoiceItemTable->addColumn( "quantity",    "Quantity" );
   invoiceItemTable->addColumn( "paiddate",    "Paid" );

   invoiceItemTable->refresh();
   invoiceItemTable->show();

   return app.exec();
  }

  return 1;
}
```

sql/overview/subclass3/main.h

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include "../connection.h"

class QSqlRecord;

class InvoiceItemCursor : public QSqlCursor
{
   public:
    InvoiceItemCursor();
   protected:
    QVariant calculateField( const QString & name );
};
```

(21) subclass4

sql/overview/subclass4/subclass4.pro

```
TEMPLATE = app
CONFIG  += qt warn_on release
HEADERS     =
SOURCES     = main.cpp ../connection.cpp
```

sql/overview/subclass4/main.cpp

```
#include "main.h"
#include <qdatatable.h>

InvoiceItemCursor::InvoiceItemCursor() :
   QSqlCursor( "invoiceitem" )
{
   QSqlFieldInfo productName( "productname", QVariant::String );
   append( productName );
```

```
    setCalculated( productName.name(), TRUE );

    QSqlFieldInfo productPrice( "price", QVariant::Double );
    append( productPrice );
    setCalculated( productPrice.name(), TRUE );

    QSqlFieldInfo productCost( "cost", QVariant::Double );
    append( productCost );
    setCalculated( productCost.name(), TRUE );
}

QVariant InvoiceItemCursor::calculateField( const QString & name )
{

    if ( name == "productname" ) {
      QSqlQuery query( "SELECT name FROM prices WHERE id=" +
              field( "pricesid" )->value().toString() );
      if ( query.next() )
         return query.value( 0 );
    }
    else if ( name == "price" ) {
      QSqlQuery query( "SELECT price FROM prices WHERE id=" +
              field( "pricesid" )->value().toString() );
      if ( query.next() )
         return query.value( 0 );
    }
    else if ( name == "cost" ) {
      QSqlQuery query( "SELECT price FROM prices WHERE id=" +
              field( "pricesid" )->value().toString() );
      if ( query.next() )
         return QVariant( query.value( 0 ).toDouble() *
                 value( "quantity").toDouble() );
    }

    return QVariant( QString::null );
}

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

    if ( createConnections() ) {
      InvoiceItemCursor invoiceItemCursor;

      QDataTable *invoiceItemTable = new QDataTable( &invoiceItemCursor );

      app.setMainWidget( invoiceItemTable );

      invoiceItemTable->addColumn( "productname", "Product" );
      invoiceItemTable->addColumn( "price",       "Price" );
      invoiceItemTable->addColumn( "quantity",    "Quantity" );
      invoiceItemTable->addColumn( "cost",        "Cost" );
      invoiceItemTable->addColumn( "paiddate",    "Paid" );
      invoiceItemTable->refresh();
      invoiceItemTable->show();

      return app.exec();
    }

    return 1;
}
```

sql/overview/subclass4/main.h

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include "../connection.h"

class QSqlRecord;

class InvoiceItemCursor : public QSqlCursor
{
    public:
      InvoiceItemCursor();
    protected:
      QVariant calculateField( const QString & name );
};
```

(22) subclass5

sql/overview/subclass5/subclass5.pro

```
TEMPLATE = app
CONFIG  += qt warn_on release
HEADERS     =
SOURCES     = main.cpp ../connection.cpp
```

sql/overview/subclass5/main.cpp

```
#include "main.h"
#include <qdatatable.h>

InvoiceItemCursor::InvoiceItemCursor() :
    QSqlCursor( "invoiceitem" )
{
    QSqlFieldInfo productName( "productname", QVariant::String );
    append( productName );
    setCalculated( productName.name(), TRUE );

    QSqlFieldInfo productPrice( "price", QVariant::Double );
    append( productPrice );
    setCalculated( productPrice.name(), TRUE );

    QSqlFieldInfo productCost( "cost", QVariant::Double );
    append( productCost );
    setCalculated( productCost.name(), TRUE );
}

QVariant InvoiceItemCursor::calculateField( const QString & name )
{
    if ( name == "productname" ) {
      QSqlQuery query( "SELECT name FROM prices WHERE id=" +
                field( "pricesid" )->value().toString() );
      if ( query.next() )
          return query.value( 0 );
    }
    else if ( name == "price" ) {
      QSqlQuery query( "SELECT price FROM prices WHERE id=" +
                field( "pricesid" )->value().toString() );
      if ( query.next() )
          return query.value( 0 );
    }
    else if ( name == "cost" ) {
      QSqlQuery query( "SELECT price FROM prices WHERE id=" +
                field( "pricesid" )->value().toString() );
      if ( query.next() )
          return QVariant( query.value( 0 ).toDouble() *
                      value( "quantity").toDouble() );
    }

    return QVariant( QString::null );
}

QSqlRecord *InvoiceItemCursor::primeInsert()
{
    QSqlRecord *buffer = editBuffer();
    QSqlQuery query( "SELECT NEXTVAL( 'invoiceitem_seq' )" );
    if ( query.next() )
      buffer->setValue( "id", query.value( 0 ) );
    buffer->setValue( "paiddate", QDate::currentDate() );
    buffer->setValue( "quantity", 1 );

    return buffer;
}

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

    if ( createConnections() ) {
      InvoiceItemCursor invoiceItemCursor;

      QDataTable *invoiceItemTable = new QDataTable( &invoiceItemCursor );

      app.setMainWidget( invoiceItemTable );

      invoiceItemTable->addColumn( "productname", "Product" );
      invoiceItemTable->addColumn( "price",       "Price" );
      invoiceItemTable->addColumn( "quantity",    "Quantity" );
      invoiceItemTable->addColumn( "cost",        "Cost" );
      invoiceItemTable->addColumn( "paiddate",    "Paid" );

      invoiceItemTable->refresh();
```

```
     invoiceItemTable->show();

     return app.exec();
   }

   return 1;
}
```

sql/overview/subclass5/main.h

```
#include <qapplication.h>
#include <qdatetime.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include "../connection.h"

class QSqlRecord;

class InvoiceItemCursor : public QSqlCursor
{
   public:
     InvoiceItemCursor();
     QSqlRecord *primeInsert();
   protected:
     QVariant calculateField( const QString & name );
};
```

(23) table1

sql/overview/table1/table1.pro

```
TEMPLATE = app
CONFIG  += qt warn_on release
HEADERS     =
SOURCES     = main.cpp ../connection.cpp
```

sql/overview/table1/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include <qdatatable.h>
#include "../connection.h"

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

   if ( createConnections() ) {
     QSqlCursor staffCursor( "staff" );
     QDataTable *staffTable = new QDataTable( &staffCursor, TRUE );
     app.setMainWidget( staffTable );
     staffTable->refresh();
     staffTable->show();

     return app.exec();
   }
   return 0;
}
```

(24) table2

sql/overview/table2/table2.pro

```
TEMPLATE = app
CONFIG  += qt warn_on release
HEADERS     =
SOURCES     = main.cpp ../connection.cpp
```

sql/overview/table2/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include <qdatatable.h>
#include "../connection.h"

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

   if ( createConnections() ) {
    QSqlCursor staffCursor( "staff" );

    QDataTable *staffTable = new QDataTable( &staffCursor );

    app.setMainWidget( staffTable );

    staffTable->addColumn( "forename", "Forename" );
    staffTable->addColumn( "surname",  "Surname" );
    staffTable->addColumn( "salary",   "Annual Salary" );

    QStringList order = QStringList() << "surname" << "forename";
    staffTable->setSort( order );

    staffTable->refresh();
    staffTable->show();

    return app.exec();
   }

   return 1;
}
```

(25) table3

sql/overview/table3/table3.pro

```
TEMPLATE = app
CONFIG  += qt warn_on release
HEADERS     = main.h
SOURCES     = main.cpp ../connection.cpp
```

sql/overview/table3/main.cpp

```
#include "main.h"
#include <qdatatable.h>
```

```
StatusPicker::StatusPicker( QWidget *parent, const char *name )
   : QComboBox( parent, name )
{
   QSqlCursor cur( "status" );
   cur.select( cur.index( "name" ) );

   int i = 0;
   while ( cur.next() ) {
    insertItem( cur.value( "name" ).toString(), i );
    index2id[i] = cur.value( "id" ).toInt();
    i++;
   }
}

int StatusPicker::statusId() const
{
   return index2id[ currentItem() ];
}

void StatusPicker::setStatusId( int statusid )
{
   QMap<int,int>::Iterator it;
   for ( it = index2id.begin(); it != index2id.end(); ++it ) {
    if ( it.data() == statusid ) {
       setCurrentItem( it.key() );
       break;
    }
   }
}

QWidget *CustomSqlEditorFactory::createEditor(
   QWidget *parent, const QSqlField *field )
{
   if ( field->name() == "statusid" ) {
    QWidget *editor = new StatusPicker( parent );
    return editor;
   }

   return QSqlEditorFactory::createEditor( parent, field );
}

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

   if ( createConnections() ) {
    QSqlCursor staffCursor( "staff" );

    QDataTable         *staffTable    = new QDataTable( &staffCursor );
    QSqlPropertyMap          *propMap= new QSqlPropertyMap();
    CustomSqlEditorFactory     *editorFactory     = new CustomSqlEditorFactory();
    propMap->insert( "StatusPicker", "statusid" );
    staffTable->installPropertyMap( propMap );
```

```
      staffTable->installEditorFactory( editorFactory );

      app.setMainWidget( staffTable );

      staffTable->addColumn( "forename", "Forename" );
      staffTable->addColumn( "surname",  "Surname" );
      staffTable->addColumn( "salary",   "Annual Salary" );
      staffTable->addColumn( "statusid", "Status" );

      QStringList order = QStringList() << "surname" << "forename";
      staffTable->setSort( order );

      staffTable->refresh();
      staffTable->show();

      return app.exec();
    }

    return 1;
}
```

sql/overview/table3/main.h

```
#include <qapplication.h>
#include <qcombobox.h>
#include <qmap.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include <qsqleditorfactory.h>
#include <qsqlpropertymap.h>
#include "../connection.h"

class StatusPicker : public QComboBox
{
    Q_OBJECT
    Q_PROPERTY( int statusid READ statusId WRITE setStatusId )
    public:
      StatusPicker( QWidget *parent=0, const char *name=0 );
      int statusId() const;
      void setStatusId( int id );
    private:
      QMap< int, int > index2id;
};

class CustomSqlEditorFactory : public QSqlEditorFactory
{
    Q_OBJECT
    public:
      QWidget *createEditor( QWidget *parent, const QSqlField *field );
};
```

sql/overview/table3/moc_main.cpp

```
#undef QT_NO_COMPAT
#include "main.h"
#include <qmetaobject.h>
```

```
#include <qapplication.h>

#include <private/qucomextra_p.h>
#if !defined(Q_MOC_OUTPUT_REVISION) || (Q_MOC_OUTPUT_REVISION != 26)
#error "This file was generated using the moc from 3.3.6. It"
#error "cannot be used with the include files from this version of Qt."
#error "(The moc has changed too much.)"
#endif

#include <qvariant.h>
const char *StatusPicker::className() const
{
   return "StatusPicker";
}

QMetaObject *StatusPicker::metaObj = 0;
static QMetaObjectCleanUp cleanUp_StatusPicker( "StatusPicker", &StatusPicker::staticMetaObject );

#ifndef QT_NO_TRANSLATION
QString StatusPicker::tr( const char *s, const char *c )
{
   if ( qApp )
    return qApp->translate( "StatusPicker", s, c, QApplication::DefaultCodec );
   else
    return QString::fromLatin1( s );
}
#ifndef QT_NO_TRANSLATION_UTF8
QString StatusPicker::trUtf8( const char *s, const char *c )
{
   if ( qApp )
    return qApp->translate( "StatusPicker", s, c, QApplication::UnicodeUTF8 );
   else
    return QString::fromUtf8( s );
}
#endif // QT_NO_TRANSLATION_UTF8

#endif // QT_NO_TRANSLATION

QMetaObject* StatusPicker::staticMetaObject()
{
   if ( metaObj )
    return metaObj;
   QMetaObject* parentObject = QComboBox::staticMetaObject();
#ifndef QT_NO_PROPERTIES
   static const QMetaProperty props_tbl[1] = {
    { "int","statusid", 0x10000003, &StatusPicker::metaObj, 0, -1 }
   };
#endif // QT_NO_PROPERTIES
   metaObj = QMetaObject::new_metaobject(
    "StatusPicker", parentObject,
    0, 0,
    0, 0,
#ifndef QT_NO_PROPERTIES
    props_tbl, 1,
     0, 0,
#endif // QT_NO_PROPERTIES
     0, 0 );
   cleanUp_StatusPicker.setMetaObject( metaObj );
   return metaObj;
}

void* StatusPicker::qt_cast( const char* clname )
{
   if ( !qstrcmp( clname, "StatusPicker" ) )
    return this;
   return QComboBox::qt_cast( clname );
}

bool StatusPicker::qt_invoke( int _id, QUObject* _o )
{
   return QComboBox::qt_invoke(_id,_o);
}

bool StatusPicker::qt_emit( int _id, QUObject* _o )
{
   return QComboBox::qt_emit(_id,_o);
}
#ifndef QT_NO_PROPERTIES

bool StatusPicker::qt_property( int id, int f, QVariant* v)
{
   switch ( id - staticMetaObject()->propertyOffset() ) {
   case 0: switch( f ) {
    case 0: setStatusId(v->asInt()); break;
    case 1: *v = QVariant( this->statusId() ); break;
    case 3: case 4: case 5: break;
    default: return FALSE;
   } break;
   default:
    return QComboBox::qt_property( id, f, v );
   }
   return TRUE;
}

bool StatusPicker::qt_static_property( QObject* , int , int , QVariant* ){ return FALSE; }
#endif // QT_NO_PROPERTIES

const char *CustomSqlEditorFactory::className() const
{
   return "CustomSqlEditorFactory";
}

QMetaObject *CustomSqlEditorFactory::metaObj = 0;
static QMetaObjectCleanUp cleanUp_CustomSqlEditorFactory( "CustomSqlEditorFactory",
&CustomSqlEditorFactory::staticMetaObject );

#ifndef QT_NO_TRANSLATION
QString CustomSqlEditorFactory::tr( const char *s, const char *c )
```

```
{
    if ( qApp )
      return qApp->translate( "CustomSqlEditorFactory", s, c, QApplication::DefaultCodec );
    else
      return QString::fromLatin1( s );
}
#ifndef QT_NO_TRANSLATION_UTF8
QString CustomSqlEditorFactory::trUtf8( const char *s, const char *c )
{
    if ( qApp )
      return qApp->translate( "CustomSqlEditorFactory", s, c, QApplication::UnicodeUTF8 );
    else
      return QString::fromUtf8( s );
}
#endif // QT_NO_TRANSLATION_UTF8

#endif // QT_NO_TRANSLATION

QMetaObject* CustomSqlEditorFactory::staticMetaObject()
{
    if ( metaObj )
      return metaObj;
    QMetaObject* parentObject = QSqlEditorFactory::staticMetaObject();
    metaObj = QMetaObject::new_metaobject(
      "CustomSqlEditorFactory", parentObject,
      0, 0,
      0, 0,
#ifndef QT_NO_PROPERTIES
      0, 0,
      0, 0,
#endif // QT_NO_PROPERTIES
      0, 0 );
    cleanUp_CustomSqlEditorFactory.setMetaObject( metaObj );
    return metaObj;
}

void* CustomSqlEditorFactory::qt_cast( const char* clname )
{
    if ( !qstrcmp( clname, "CustomSqlEditorFactory" ) )
      return this;
    return QSqlEditorFactory::qt_cast( clname );
}

bool CustomSqlEditorFactory::qt_invoke( int _id, QUObject* _o )
{
    return QSqlEditorFactory::qt_invoke(_id,_o);
}

bool CustomSqlEditorFactory::qt_emit( int _id, QUObject* _o )
{
    return QSqlEditorFactory::qt_emit(_id,_o);
}
#ifndef QT_NO_PROPERTIES
```

```
bool CustomSqlEditorFactory::qt_property( int id, int f, QVariant* v)
{
    return QSqlEditorFactory::qt_property( id, f, v);
}

bool CustomSqlEditorFactory::qt_static_property( QObject* , int , int , QVariant* ){ return FALSE; }
#endif // QT_NO_PROPERTIES
```

(26) table4

sql/overview/table4/table4.pro

```
TEMPLATE = app
CONFIG  += qt warn_on release
HEADERS     = main.h
SOURCES     = main.cpp ../connection.cpp
```

sql/overview/table4/main.cpp

```
#include "main.h"

StatusPicker::StatusPicker( QWidget *parent, const char *name )
    : QComboBox( parent, name )
{
    QSqlCursor cur( "status" );
    cur.select( cur.index( "name" ) );

    int i = 0;
    while ( cur.next() ) {
      insertItem( cur.value( "name" ).toString(), i );
      index2id[i] = cur.value( "id" ).toInt();
      i++;
    }
}

int StatusPicker::statusId() const
{
    return index2id[ currentItem() ];
}

void StatusPicker::setStatusId( int statusid )
{
    QMap<int,int>::Iterator it;
    for ( it = index2id.begin(); it != index2id.end(); ++it ) {
      if ( it.data() == statusid ) {
          setCurrentItem( it.key() );
          break;
      }
    }
}

void CustomTable::paintField( QPainter * p, const QSqlField* field,
                        const QRect & cr, bool b)
{
    if ( !field )
      return;
    if ( field->name() == "statusid" ) {
```

```
    QSqlQuery query( "SELECT name FROM status WHERE id=" +
              field->value().toString() );
    QString text;
    if ( query.next() ) {
        text = query.value( 0 ).toString();
    }
    p->drawText( 2,2, cr.width()-4, cr.height()-4, fieldAlignment( field ), text );
  }
  else {
    QDataTable::paintField( p, field, cr, b) ;
  }
}

QWidget *CustomSqlEditorFactory::createEditor( QWidget *parent, const QSqlField *field )
{
  if ( field->name() == "statusid" ) {
    QWidget *editor = new StatusPicker( parent );
    return editor;
  }

  return QSqlEditorFactory::createEditor( parent, field );
}

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

  if ( createConnections() ) {
    QSqlCursor staffCursor( "staff" );

    CustomTable       *staffTable     = new CustomTable( &staffCursor );
    QSqlPropertyMap          *propMap= new QSqlPropertyMap();
    CustomSqlEditorFactory     *editorFactory     = new CustomSqlEditorFactory();
    propMap->insert( "StatusPicker", "statusid" );
    staffTable->installPropertyMap( propMap );
    staffTable->installEditorFactory( editorFactory );

    app.setMainWidget( staffTable );

    staffTable->addColumn( "forename", "Forename" );
    staffTable->addColumn( "surname",  "Surname" );
    staffTable->addColumn( "salary",   "Annual Salary" );
    staffTable->addColumn( "statusid", "Status" );

    QStringList order = QStringList() << "surname" << "forename";
    staffTable->setSort( order );

    staffTable->refresh();
    staffTable->show();

    return app.exec();
  }

  return 1;
```

```
}
```

sql/overview/table4/main.h

```
#include <qapplication.h>
#include <qcombobox.h>
#include <qmap.h>
#include <qpainter.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include <qsqleditorfactory.h>
#include <qsqlpropertymap.h>
#include <qdatatable.h>
#include "../connection.h"

class StatusPicker : public QComboBox
{
   Q_OBJECT
   Q_PROPERTY( int statusid READ statusId WRITE setStatusId )
public:
   StatusPicker( QWidget *parent=0, const char *name=0 );
   int statusId() const;
   void setStatusId( int id );
private:
   QMap< int, int > index2id;
};

class CustomTable : public QDataTable
{
   Q_OBJECT
public:
   CustomTable(
        QSqlCursor *cursor, bool autoPopulate = FALSE,
        QWidget * parent = 0, const char * name = 0 ) :
     QDataTable( cursor, autoPopulate, parent, name ) {}
   void paintField(
        QPainter * p, const QSqlField* field, const QRect & cr, bool );

};

class CustomSqlEditorFactory : public QSqlEditorFactory
{
   Q_OBJECT
public:
   QWidget *createEditor( QWidget *parent, const QSqlField *field );
};
```

sql/overview/table4/moc_main.cpp

```
#undef QT_NO_COMPAT
#include "main.h"
#include <qmetaobject.h>
#include <qapplication.h>

#include <private/qucomextra_p.h>
#if !defined(Q_MOC_OUTPUT_REVISION) || (Q_MOC_OUTPUT_REVISION != 26)
```

```
#error "This file was generated using the moc from 3.3.6. It"
#error "cannot be used with the include files from this version of Qt."
#error "(The moc has changed too much.)"
#endif

#include <qvariant.h>
const char *StatusPicker::className() const
{
   return "StatusPicker";
}

QMetaObject *StatusPicker::metaObj = 0;
static QMetaObjectCleanUp cleanUp_StatusPicker( "StatusPicker", &StatusPicker::staticMetaObject );

#ifndef QT_NO_TRANSLATION
QString StatusPicker::tr( const char *s, const char *c )
{
   if ( qApp )
    return qApp->translate( "StatusPicker", s, c, QApplication::DefaultCodec );
   else
    return QString::fromLatin1( s );
}
#ifndef QT_NO_TRANSLATION_UTF8
QString StatusPicker::trUtf8( const char *s, const char *c )
{
   if ( qApp )
    return qApp->translate( "StatusPicker", s, c, QApplication::UnicodeUTF8 );
   else
    return QString::fromUtf8( s );
}
#endif // QT_NO_TRANSLATION_UTF8

#endif // QT_NO_TRANSLATION

QMetaObject* StatusPicker::staticMetaObject()
{
   if ( metaObj )
    return metaObj;
   QMetaObject* parentObject = QComboBox::staticMetaObject();
#ifndef QT_NO_PROPERTIES
   static const QMetaProperty props_tbl[1] = {
    { "int","statusid", 0x10000003, &StatusPicker::metaObj, 0, -1 }
   };
#endif // QT_NO_PROPERTIES
   metaObj = QMetaObject::new_metaobject(
    "StatusPicker", parentObject,
    0, 0,
    0, 0,
#ifndef QT_NO_PROPERTIES
    props_tbl, 1,
    0, 0,
#endif // QT_NO_PROPERTIES
    0, 0 );
   cleanUp_StatusPicker.setMetaObject( metaObj );
```

```
    return metaObj;
}

void* StatusPicker::qt_cast( const char* clname )
{
    if ( !qstrcmp( clname, "StatusPicker" ) )
      return this;
    return QComboBox::qt_cast( clname );
}

bool StatusPicker::qt_invoke( int _id, QUObject* _o )
{
    return QComboBox::qt_invoke(_id,_o);
}

bool StatusPicker::qt_emit( int _id, QUObject* _o )
{
    return QComboBox::qt_emit(_id,_o);
}
#ifndef QT_NO_PROPERTIES

bool StatusPicker::qt_property( int id, int f, QVariant* v)
{
    switch ( id - staticMetaObject()->propertyOffset() ) {
    case 0: switch( f ) {
      case 0: setStatusId(v->asInt()); break;
      case 1: *v = QVariant( this->statusId() ); break;
      case 3: case 4: case 5: break;
      default: return FALSE;
    } break;
    default:
      return QComboBox::qt_property( id, f, v );
    }
    return TRUE;
}

bool StatusPicker::qt_static_property( QObject* , int , int , QVariant* ){ return FALSE; }
#endif // QT_NO_PROPERTIES

const char *CustomTable::className() const
{
    return "CustomTable";
}

QMetaObject *CustomTable::metaObj = 0;
static QMetaObjectCleanUp cleanUp_CustomTable( "CustomTable",
&CustomTable::staticMetaObject );

#ifndef QT_NO_TRANSLATION
QString CustomTable::tr( const char *s, const char *c )
{
    if ( qApp )
      return qApp->translate( "CustomTable", s, c, QApplication::DefaultCodec );
    else
    return QString::fromLatin1( s );
}
#ifndef QT_NO_TRANSLATION_UTF8
QString CustomTable::trUtf8( const char *s, const char *c )
{
   if ( qApp )
    return qApp->translate( "CustomTable", s, c, QApplication::UnicodeUTF8 );
   else
    return QString::fromUtf8( s );
}
#endif // QT_NO_TRANSLATION_UTF8

#endif // QT_NO_TRANSLATION

QMetaObject* CustomTable::staticMetaObject()
{
   if ( metaObj )
    return metaObj;
   QMetaObject* parentObject = QDataTable::staticMetaObject();
   metaObj = QMetaObject::new_metaobject(
    "CustomTable", parentObject,
    0, 0,
    0, 0,
#ifndef QT_NO_PROPERTIES
    0, 0,
    0, 0,
#endif // QT_NO_PROPERTIES
    0, 0 );
   cleanUp_CustomTable.setMetaObject( metaObj );
   return metaObj;
}

void* CustomTable::qt_cast( const char* clname )
{
   if ( !qstrcmp( clname, "CustomTable" ) )
    return this;
   return QDataTable::qt_cast( clname );
}

bool CustomTable::qt_invoke( int _id, QUObject* _o )
{
   return QDataTable::qt_invoke(_id,_o);
}

bool CustomTable::qt_emit( int _id, QUObject* _o )
{
   return QDataTable::qt_emit(_id,_o);
}
#ifndef QT_NO_PROPERTIES

bool CustomTable::qt_property( int id, int f, QVariant* v)
{
   return QDataTable::qt_property( id, f, v);
}
```

```
bool CustomTable::qt_static_property( QObject* , int , int , QVariant* ){ return FALSE; }
#endif // QT_NO_PROPERTIES

const char *CustomSqlEditorFactory::className() const
{
   return "CustomSqlEditorFactory";
}

QMetaObject *CustomSqlEditorFactory::metaObj = 0;
static QMetaObjectCleanUp cleanUp_CustomSqlEditorFactory( "CustomSqlEditorFactory",
&CustomSqlEditorFactory::staticMetaObject );

#ifndef QT_NO_TRANSLATION
QString CustomSqlEditorFactory::tr( const char *s, const char *c )
{
   if ( qApp )
    return qApp->translate( "CustomSqlEditorFactory", s, c, QApplication::DefaultCodec );
   else
    return QString::fromLatin1( s );
}
#ifndef QT_NO_TRANSLATION_UTF8
QString CustomSqlEditorFactory::trUtf8( const char *s, const char *c )
{
   if ( qApp )
    return qApp->translate( "CustomSqlEditorFactory", s, c, QApplication::UnicodeUTF8 );
   else
    return QString::fromUtf8( s );
}
#endif // QT_NO_TRANSLATION_UTF8

#endif // QT_NO_TRANSLATION

QMetaObject* CustomSqlEditorFactory::staticMetaObject()
{
   if ( metaObj )
    return metaObj;
   QMetaObject* parentObject = QSqlEditorFactory::staticMetaObject();
   metaObj = QMetaObject::new_metaobject(
    "CustomSqlEditorFactory", parentObject,
    0, 0,
    0, 0,
#ifndef QT_NO_PROPERTIES
    0, 0,
    0, 0,
#endif // QT_NO_PROPERTIES
    0, 0 );
   cleanUp_CustomSqlEditorFactory.setMetaObject( metaObj );
   return metaObj;
}

void* CustomSqlEditorFactory::qt_cast( const char* clname )
{
   if ( !qstrcmp( clname, "CustomSqlEditorFactory" ) )
```

```
     return this;
   return QSqlEditorFactory::qt_cast( clname );
}

bool CustomSqlEditorFactory::qt_invoke( int _id, QUObject* _o )
{
   return QSqlEditorFactory::qt_invoke(_id,_o);
}

bool CustomSqlEditorFactory::qt_emit( int _id, QUObject* _o )
{
   return QSqlEditorFactory::qt_emit(_id,_o);
}
#ifndef QT_NO_PROPERTIES

bool CustomSqlEditorFactory::qt_property( int id, int f, QVariant* v)
{
   return QSqlEditorFactory::qt_property( id, f, v);
}

bool CustomSqlEditorFactory::qt_static_property( QObject* , int , int , QVariant* ){ return FALSE; }
#endif // QT_NO_PROPERTIES
```

(27) update

sql/overview/update/update.pro

```
TEMPLATE  = app
CONFIG       += qt warn_on release
HEADERS        =
SOURCES        = main.cpp ../connection.cpp
```

sql/overview/update/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qsqlcursor.h>
#include "../connection.h"

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

   if ( createConnections() ) {
     QSqlCursor cur( "prices" );
     cur.select( "id=202" );
     if ( cur.next() ) {
        QSqlRecord *buffer = cur.primeUpdate();
        double price = buffer->value( "price" ).toDouble();
        double newprice = price * 1.05;
        buffer->setValue( "price", newprice );
        cur.update();
     }
   }

   return 0;
}
```

3) sqltable

이 SQL표실례는 SQL자료기지와의 접속을 요구한다. main.cpp를 수정하여 자기의 자료기지와 련결하여야 한다.

이 실례는 simpletable라는 표가 자료기지에 있을것을 요구한다. 다음의 SQL을 실행하여 이 표를 창조할수 있다.

script (modify to suit your backend, if necessary):

```
drop table simpletable;
create table simpletable
(id number primary key,
name varchar(20),
address varchar(20) );

-- optional, some sample data
insert into simpletable (id, name, address)
values (1, 'Trond', 'Oslo');
insert into simpletable (id, name, address)
values (2, 'Dave', 'Oslo');
```

sql/sqltable/sqltable.pro

```
TEMPLATE  = app
TARGET         = sqltable
CONFIG       += qt warn_on release
HEADERS        =
SOURCES        = main.cpp
INTERFACES    =
```

sql/sqltable/main.cpp

```
#include <qapplication.h>
#include <qsqldatabase.h>
#include <qdatatable.h>
#include <qsqlcursor.h>
#include <qmessagebox.h>

/* Modify the following to match your environment */
#define DRIVER       "QSQLITE"  /* see the Qt SQL documentation for a list of available drivers */
#define DATABASE     ":memory:" /* the name of your database */
#define USER         ""   /* user name with appropriate rights */
#define PASSWORD     ""   /* password for USER */
#define HOST         "" /* host on which the database is running */

class SimpleCursor : public QSqlCursor
{
public:
   SimpleCursor () : QSqlCursor( "simpletable" ) {}
protected:
   QSqlRecord* primeInsert()
   {
    /* a real-world application would use sequences, or the like */
    QSqlRecord* buf = QSqlCursor::primeInsert();
    QSqlQuery q( "select max(id)+1 from simpletable" );
    if ( q.next() )
        buf->setValue( "id", q.value(0) );
    return buf;
   }
};

int main( int argc, char ** argv )
{
   QApplication a( argc, argv );

   QSqlDatabase * db = QSqlDatabase::addDatabase( DRIVER );
   db->setDatabaseName( DATABASE );
   db->setUserName( USER );
   db->setPassword( PASSWORD );
   db->setHostName( HOST );

   if( !db->open() ){
    db->lastError().showMessage( "An error occured. Please read the README file in the sqltable"
                          "dir for more information.\n\n" );
    return 1;
   }

   if (!db->tables().contains("simpletable")) {
    QSqlQuery q("create table simpletable(id int, name varchar(20), address varchar(20))", db);
   }

   SimpleCursor cursor;

   QDataTable table( &cursor ); /* data table uses our cursor */
   table.addColumn( "name", "Name" );
   table.addColumn( "address", "Address" );
   table.setSorting( TRUE );

   a.setMainWidget( &table );
   table.refresh(); /* load data */
   table.show();    /* show widget */

   return a.exec();
}
```

참고문헌

Trolltech 《Qt 3.3.6 Documentation》 Trolltech, 2006.

이 책은 콤퓨터를 전공으로 하는 교원, 연구사, 대학생들을 위한 참고서이다.

Qt실례프로그람

집 필	한영철,홍이철,문홍남	**심 사**	김철남
편 집	차현옥	**교 정**	서금석
장 정	서경애	**콤퓨터편성**	여은정
낸 곳	교육위원회 교육정보쎈터	**인쇄소**	
인 쇄		**발 행**	

교-09-1664　　　　부　　　　값　　원