Practical Qt

Real World Solutions to Real World Problems

von
Matthias K Dalheimer, Jesper Pedersen

1. Auflage

Thematische Gliederung:

Grafikprogrammierung

dpunkt.verlag 2004

Verlag C.H. Beck im Internet:
www.beck.de
ISBN 978 3 89864 280 4

Inhaltsverzeichnis: Practical Qt – Dalheimer / Pedersen

```
14   virtual void paintEvent ( QPaintEvent * )
15   {
16     QPainter painter(this);
17     QPixmap map( "jesper.png" );
18
19     QCOORD points[] = { 25,25, 50,50, 0,50 };
20     QRegion region( QPointArray( 3,  points) );
21     region += QRegion( 0, 50, 50, 50, QRegion::Ellipse );
22     region += QRegion( 50, 0, 20, 70 );
23
24     painter.setClipRegion( region );
25
26     painter.drawPixmap( 0,0, map );
27   }
28 };
29
30
31
32 int main( int argc, char** argv )
33 {
34   QApplication app( argc, argv );
35
36   Viewer* viewer = new Viewer(0);
37   app.setMainWidget( viewer );
38   viewer->resize(68,110);
39
40   viewer->show();
41
42   return app.exec();
43 }
```

The interesting code in this example is contained entirely in the `paintEvent()` method. This code first puts together a region that consists of three separate parts: a polygon (more precisely, a triangle), an ellipse (a circle, actually), and a rectangle. The new region is then passed to `QPainter::setClipRegion()` before drawing a pixmap. Note that we have to use `QPainter::drawPixmap()` here instead of the more low-level `bitBlt()`, because the latter function does not take the clipping into account.

Clipping can be very useful, but you should be aware that it can also slow down your program considerably. Clipping to rectangular regions with `QPainter::setClipRect()` can often be done in the graphics hardware and is therefore usually fast, but clipping to anything non-rectangular or non-contiguous is potentially quite slow, particularly if you clip to a bitmap, because this will by definition generate one clipping rectangle for each enabled bit in the bitmap.

10. 2 Merging Images And Text

Sometimes, you want to display both an image and some text at the same location. Some Qt widgets (such as `QToolButton`) support this, but many others (such as

QLabel and QCheckBox) do not. So what do you do when you want to display both a label and an image?

There are two basic strategies: (1) Make the image a text and combine it with the other text to form a new text that is assigned to the widget, or (2) make the text an image and combine it with the other image to form a new image, which is then assigned to the widget. The first strategy is very complicated, maybe close to impossible in some situations, and it is beyond the scope of Qt. (You would have to create a new font out of whose glyphs you would construct the image, a strategy that was used on early home computers before high-resolution graphics were introduced).

We will use the second strategy here, combining the original image and the text to form a new image. This is done in three steps:

- Compute the necessary size to accomodate both the original image and the text, and create a new QPixmap object of this size.
- Render the text into the new pixmap by means of QPainter::drawText()
- Copy the image from the old pixmap into the new one by means of QPainter::drawPixmap() or bitBlt().

The following listing shows these steps twice, once for displaying the text to the right of the image (in the function mergeSideBySide()) and once for displaying the text below the image (in the function mergeOnTop()). The generated pixmaps are then displayed in QLabel objects in main(). Fig. 10-1 and Fig. 10-2 show the output of this program.

```
 1 #include <qpixmap.h>
 2 #include <qpainter.h>
 3 #include <qapplication.h>
 4 #include <qlabel.h>
 5
 6 QPixmap mergeSideBySide( const QPixmap& pix, const QString txt )
 7 {
 8   QPainter p;
 9   int strWidth = p.fontMetrics().width( txt );
10   int strHeight = p.fontMetrics().height();
11
12   int pixWidth = pix.width();
13   int pixHeight = pix.height();
14
15   QPixmap res( strWidth + 3 + pixWidth, QMAX( strHeight, pixHeight ) );
16   res.fill(Qt::white);
17
18   p.begin( &res );
19   p.drawPixmap(0,0, pix );
20   p.drawText( QRect( pixWidth +3, 0, strWidth, strHeight), 0, txt );
21   p.end();
22
23   return res;
24 }
```

```
25
26 QPixmap mergeOnTop( const QPixmap& pix, const QString txt )
27 {
28   QPainter p;
29   int strWidth = p.fontMetrics().width( txt );
30   int strHeight = p.fontMetrics().height();
31
32   int pixWidth = pix.width();
33   int pixHeight = pix.height();
34
35   QPixmap res( QMAX(strWidth, pixWidth), strHeight + pixHeight + 3 );
36   res.fill(Qt::white);
37
38   p.begin( &res );
39   int start = 0;
40   if ( pixWidth < strWidth )
41     start = (strWidth-pixWidth)/2;
42
43   p.drawPixmap( start ,0, pix );
44   p.drawText( QRect( 0, pixHeight+3, strWidth, strHeight), 0, txt );
45   p.end();
46
47   return res;
48 }
49
50 int main( int argc, char** argv )
51 {
52   QApplication app( argc, argv );
53
54   QLabel* label = new QLabel( 0 );
55   label->setPixmap( mergeOnTop( QPixmap("open.xpm"), "Open" ) );
56   label->show();
57
58   QLabel* label2 = new QLabel( 0 );
59   label2->setPixmap( mergeSideBySide( QPixmap("open.xpm"), "Open" ) );
60   label2->show();
61
62   return app.exec();
63 }
```

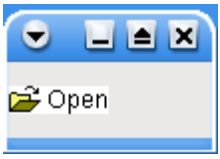

Fig. 10-1: *Displaying an Image and a Text in a QLabel Side By Side*

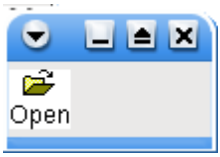

Fig. 10-2: *Displaying an Image and a Text in a QLabel on Top of Each Other*

10. 3 Merging Pixmaps

In Section 10.2 we saw how to merge images and text in order to be able to use both where Qt only supports one or the other. Now we will look at merging several pixmaps so that you can use more than one pixmap in locations where Qt only supports one, as for example in list view or list box items.

There are many possible applications for this technique. For example, the email client KMail, which was written with Qt, displays small icons next to the message headers that show the state of the message. Up to three icons per message can be shown: one for the reading state (read, unread, replies, forwarded, etc.), one for the signature state (signed, unsigned), and one for the encryption state (encrypted, unencrypted). Because the class `QListView` that is used for displaying the message list only supports one pixmap in a column, and it would not be very useful to have three columns just for the icons, it was necessary to merge three icon pixmaps.

The technique is similar to that used for merging an image and text. In the following code, we generalize it even more and allow for merging an arbitrary number of pixmaps:

```
#include <qapplication.h>
#include <qpixmap.h>
#include <qvaluelist.h>
#include <qlabel.h>
#include <qbitmap.h>

typedef QValueList<QPixmap> PixmapList;

QPixmap pixmapMerge( const PixmapList& pixmaps )
{
  int width = 0;
  int height = 0;
  for ( PixmapList::ConstIterator it = pixmaps.begin();
    it != pixmaps.end();
    ++it ) {
    width += (*it).width();
    height = QMAX( height, (*it).height() );
  }

  QPixmap res( width, height );
  QBitmap mask( width, height );

```