9. Qt – grafika

B2B99PPC – Praktické programování v C/C++

Stanislav Vítek

Katedra radioelektroniky
Fakulta elektrotechnická
České vysoké učení v Praze

Low-level grafika – QPainter 1/2

- QPainter umožňuje kreslit do QPaintDevice
  - QImage
  - QPicture
  - QPixmap
  - QWidget
- Obsahuje celou řadu funkcí pro kreslení grafických primitiv
- Nejběžnější je využití v handleru paintEvent

```
1  void SimpleExampleWidget::paintEvent(QPaintEvent *)
2  {
3    QPainter painter(this);
4    painter.setPen(Qt::blue);
5    painter.setFont(QFont("Arial", 30));
6    painter.drawText(rect(), Qt::AlignCenter, "Qt");
7  }
```

Low-level grafika – `QPainter` 2/2

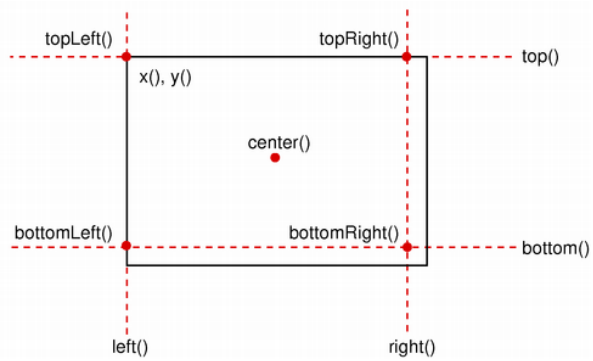
- Je třeba se postarat o destrukci instance
- V jeden okamžik může existovat jen jedna instance
- Pokud je nutné mít instanci `QPainter` sdílenou v třídě, lze využít funkce QPainter::begin() a QPainter::end

Ve většině případů vyřeší konstruktor a destruktor.

- Kromě implicitního volání lze metodu `paintEvent` explicitně volat
  - `update` – naplánuje paint event
  - `update` – vynutí překreslení

Kreslení – pomocné třídy, orientace

- QSize
  - scale, transpose
- QPoint
- QLine
  - translate, dx, dy
- QRect
  - adjust, move
  - translate, scale, center




```
QSize size(100,100);
QPoint point(0,0);
QRect rect(point, size);
rect.adjust(10,10,-10,-10);
QPoint center = rect.center();
```

Kreslení – barvy

- Různé barevné modely (vždy s ohledem na barevný prostor zobrazovače)

```
QColor(255,0,0);
QColor::fromHsv(h,s,v);
QColor::fromCmyk(c,m,y,k);
```

- Definice barvy

```
QColor(255,0,0); // red in RGB
QColor(255,0,0, 63); // red 25% opaque (75% transparent)
QColor("#FF0000"); // red in web-notation
QColor("red"); // by svg-name
Qt::red; // predefined Qt global colors

```

Kreslení – nastavení pera QPen

- barva nebo vzor
- šířka
- style (NoPen / SolidLine)
- konce čar
- způsob spojování čar

```
QPainter painter(this);
QPen pen = painter.pen();
pen.setBrush(Qt::red);
pen.setWidth(3);
painter.setPen(pen);
// draw a rectangle with 3 pixel width red outline
painter.drawRect(0,0,100,100);
```

Dedikovaná kreslicí plocha – QGraphicsScene

- QGraphicsScene je komponenta vhodná pro tvorbu 2D grafiky
- Obsahuje pokročilou funkcionalitu, jako je např. detekce kolizí objektů
- Scéna se zobrazuje v rámci instance QGraphicsView

```
MyWidget (QWidget *parent = 0) : QMainWindow (parent) {
  this->resize(320, 240);
  QGraphicsScene* scene = new QGraphicsScene();
  scene->addLine(20, 50.0, 50, 200);
  scene->addRect(100, 50, 60, 80);
  // scene->addRect(100, 50, 60, 80, QPen(), QBrush(Qt::red));
  scene->addEllipse(200, 100, 80, 80);
  QGraphicsView* view = new QGraphicsView(scene);
  setCentralWidget(view);
}
```

Dedikovaná kreslicí plocha – QGraphicsScene

```
QGraphicsScene* scene = new QGraphicsScene();
QPolygonF polygon;
polygon <<
  QPointF( 50,  50)  << // start here
  QPointF( 50,  70)  << // going down
  QPointF(100,  70)  << // going right
  QPointF(100, 180)  << // going down
  QPointF(120, 180)  << // going right
  QPointF(120,  70)  << // going up
  QPointF(170,  70)  << // going right
  QPointF(170,  50)  << // going up
  QPointF( 50,  50);    // going left (back to start)
scene->addPolygon(polygon);
```