

## 9. Qt – grafika

B2B99PPC – Praktické programování v C/C++

Stanislav Vítek

Katedra radioelektroniky  
Fakulta elektrotechnická  
České vysoké učení v Praze

- 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 }
```

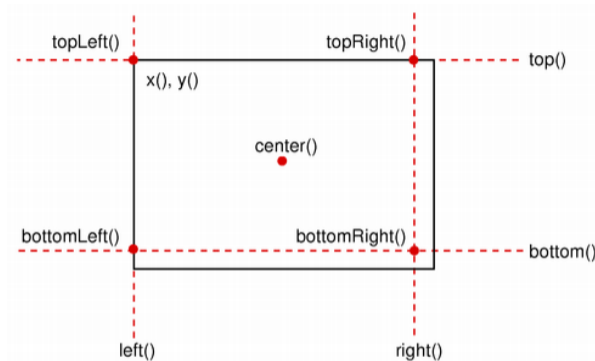
- Je třeba se postarat o destrukci instance
- V jeden okamžik může existovat jen jedna instance
- Pokud je nutné mít instanci `QPainter` sdílenou v třídě, lze využít funkce `QPainter::begin()` ↗ a `QPainter::end` ↗

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



```
1 QSize size(100,100);
2 QPoint point(0,0);
3 QRect rect(point, size);
4 rect.adjust(10,10,-10,-10);
5 QPoint center = rect.center();
```

# Kreslení – barvy

---

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

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

- Definice barvy

```
1 | QColor(255,0,0); // red in RGB  
3 | QColor(255,0,0, 63); // red 25% opaque (75% transparent)  
5 | QColor("#FF0000"); // red in web-notation  
7 | QColor("red"); // by svg-name  
9 | Qt::red; // predefined Qt global colors  
10
```

## Kreslení – nastavení pera QPen

---

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

```
1  QPainter painter(this);
2  QPen pen = painter.pen();
3  pen.setBrush(Qt::red);
4  pen.setWidth(3);
5  painter.setPen(pen);
6  // draw a rectangle with 3 pixel width red outline
7  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 [↗](#)

```
1 MyWidget (QWidget *parent = 0) : QMainWindow (parent) {
2     this->resize(320, 240);
3     QGraphicsScene* scene = new QGraphicsScene();
4     scene->addLine(20, 50.0, 50, 200);
5     scene->addRect(100, 50, 60, 80);
6     // scene->addRect(100, 50, 60, 80, QPen(), QBrush(Qt::red));
7     scene->addEllipse(200, 100, 80, 80);
8     QGraphicsView* view = new QGraphicsView(scene);
9     setCentralWidget(view);
10 }
```

## Dedikovaná kreslicí plocha – QGraphicsScene

---

```
1  QGraphicsScene* scene = new QGraphicsScene();
2  QPolygonF polygon;
3  polygon <<
4      QPointF( 50,  50) << // start here
5      QPointF( 50,  70) << // going down
6      QPointF(100,  70) << // going right
7      QPointF(100, 180) << // going down
8      QPointF(120, 180) << // going right
9      QPointF(120,  70) << // going up
10     QPointF(170,  70) << // going right
11     QPointF(170,  50) << // going up
12     QPointF( 50,  50); // going left (back to start)
13 scene->addPolygon(polygon);
```