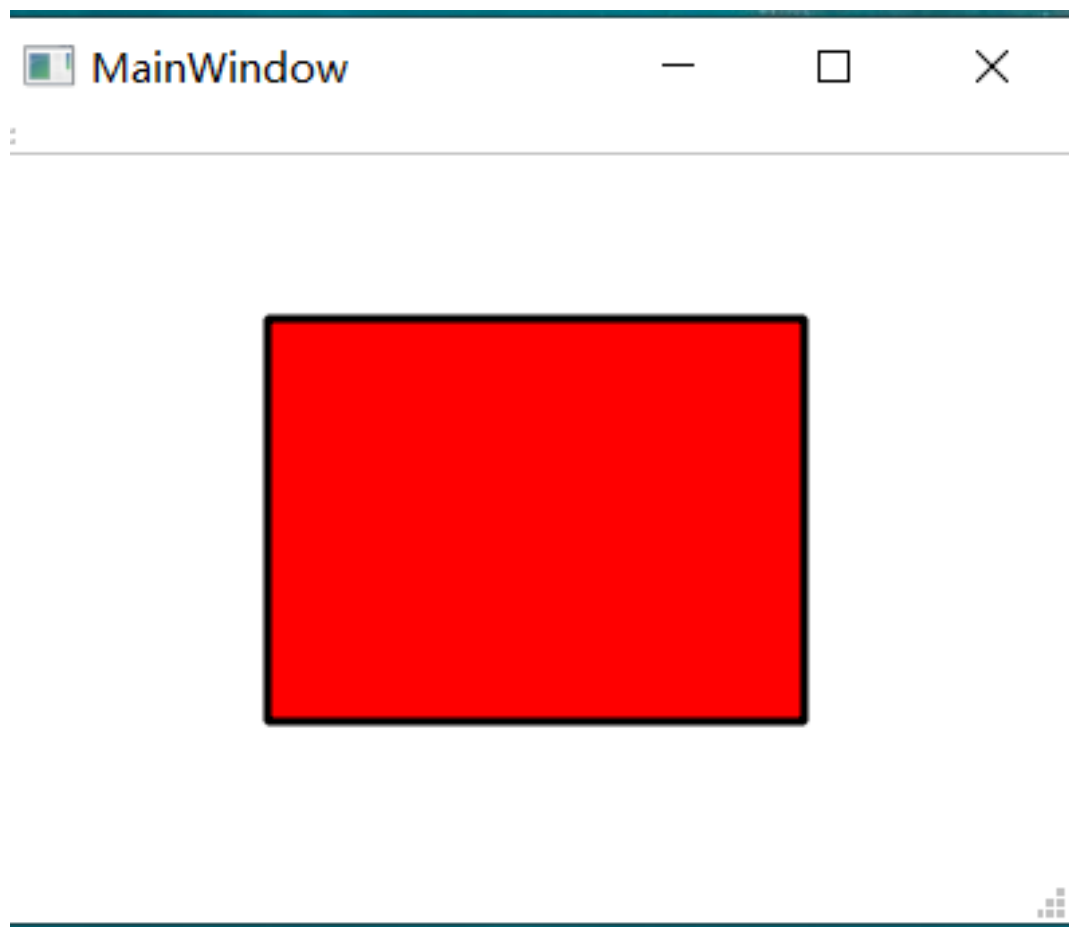


qt实战之画布和画笔

画布和画笔案例



案例知识点

- 1. QPainter绘图系统
- 2. QPen画笔
- 3. QBrush画刷
- 4. QRect矩形绘图

QPainter

1. QPainter 与 QPaintDevice

Qt 的绘图系统使用户可以在屏幕或打印设备上用相同的 API 绘图，绘图系统基于 QPainter、QPaintDevice 和 QPaintEngine 类。QPainter 是用来进行绘图操作的类，QPaintDevice 是一个可以使用 QPainter 进行绘图的抽象的二维界面，QPaintEngine 给 QPainter 提供在不同设备上绘图的接口。QPaintEngine 类由 QPainter 和 QPaintDevice 内部使用，应用程序一般无需和 QPaintEngine 打交道，除非要创建自己的设备类型。

一般的绘图设备包括 QWidget、QPixmap、QImage 等，这些绘图设备为 QPainter 提供一个“画布”。

paintEvent定义

2. paintEvent 事件和绘图区

QWidget 类及其子类是最常用的绘图设备，从 QWidget 类继承的类都有 paintEvent() 事件，要在设备上绘图，只需重定义此事件并编写响应代码。创建一个 QPainter 对象获取绘图设备的接口，然后就可以在绘图设备的“画布”上绘图了。

paintEvent定义

```
< > h/mainwindow.h | Select Symbol
1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QMainWindow>
5
6  namespace Ui {
7  class MainWindow;
8  }
9
10 class MainWindow : public QMainWindow
11 {
12     Q_OBJECT
13
14 protected:
15     void paintEvent(QPaintEvent *event) Q_DECL_OVERRIDE;
16
17 public:
18     explicit MainWindow(QWidget *parent = nullptr);
19     ~MainWindow();
20
21 private:
22     Ui::MainWindow *ui;
23 };
24
25 #endif // MAINWINDOW_H
26
```

```
re 工具(I) 控件(W) 帮助(H)
> C++/mainwindow.cpp | MainWindow::paintEvent(QPaint
1  #include "mainwindow.h"
2  #include "ui_mainwindow.h"
3  #include "qpainter.h"
4
5  MainWindow::MainWindow(QWidget *parent) :
6      QMainWindow(parent),
7      ui(new Ui::MainWindow)
8  {
9      ui->setupUi(this);
10     this->setPalette(QPalette(Qt::white));
11     this->setAutoFillBackground(true);
12 }
13
14 MainWindow::~MainWindow()
15 {
16     delete ui;
17 }
18
19 void MainWindow::paintEvent(QPaintEvent *event) {
20     QPainter painter(this);
21
22     painter.setRenderHint(QPainter::Antialiasing);
23     painter.setRenderHint(QPainter::TextAntialiasing);
24
25     int W=this->width();
26     int H=this->height();
27     QRect rect(W/4, H/4, W/2, H/2);
28
29     QPen pen;
30     pen.setWidth(3);
31     pen.setColor(Qt::black);
32     pen.setStyle(Qt::SolidLine);
33     pen.setCapStyle(Qt::FlatCap);
34     pen.setJoinStyle(Qt::BevelJoin);
35 }
```

代码详情

```
void MainWindow::paintEvent(QPaintEvent *event) {  
    QPainter painter(this);
```

```
    painter.setRenderHint(QPainter::Antialiasing);  
    painter.setRenderHint(QPainter::TextAntialiasing);
```

```
    int W=this->width();  
    int H=this->height();  
    QRect rect(W/4, H/4, W/2, H/2);
```

```
    QPen pen;  
    pen.setWidth(3);  
    pen.setColor(Qt::black);  
    pen.setStyle(Qt::SolidLine);  
    pen.setCapStyle(Qt::FlatCap);  
    pen.setJoinStyle(Qt::BevelJoin);  
    painter.setPen(pen);
```

```
    QBrush brush;  
    brush.setColor(Qt::red);  
    brush.setStyle(Qt::SolidPattern);  
    painter.setBrush(brush);
```

```
    painter.drawRect(rect);
```

```
}
```

图形和文字保真（抗锯齿）

设置绘制的矩形区域

设置画笔

设置画刷

Qpen主要功能

QPen 用于绘图时对线条进行设置，主要包括线宽、颜色、线型等，表 8-1 是 QPen 类的主要接口函数。通常一个设置函数都有一个对应的读取函数，例如 setColor()用于设置画笔颜色，对应的读取画笔颜色的函数为 color()，表 8-1 仅列出设置函数（省略了函数参数中的 const 关键字）。

表 8-1 QPen 的主要函数

函数原型	功能
void setColor(QColor &color)	设置画笔颜色，即线条颜色
void setWidth(int width)	设置线条宽度
void setStyle (Qt::PenStyle style)	设置线条样式，参数为 Qt::PenStyle 枚举类型
void setCapStyle (Qt::PenCapStyle style)	设置线条端点样式，参数为 Qt::PenCapStyle 枚举类型
void setJoinStyle (Qt::PenJoinStyle style)	设置连接样式，参数为 Qt::PenJoinStyle 枚举类型

线条颜色和宽度的设置无需多说，QPen 影响线条特性的另外 3 个主要属性是线条样式(style)、端点样式（capStyle）和连接样式（joinStyle）。

线条样式

1. 线条样式

`setStyle(Qt::PenStyle style)`函数用于设置线条样式，参数是一个枚举类型 `Qt::PenStyle` 的常量，几种典型的线条样式的绘图效果如图 8-2 所示。`Qt::PenStyle` 类型还有一个常量 `Qt::NoPen` 表示不绘制线条。



图 8-2 各种样式的线条（来自 Qt 帮助文件）

线条样式2

2. 线条端点样式

`setCapStyle (Qt::PenCapStyle style)` 函数用于设置线条端点样式，参数是一个枚举类型 `Qt::PenCapStyle` 的常量，该枚举类型的 3 种取值及其绘图效果如图 8-3 所示。



图 8-3 各种线条端点样式（来自 Qt 帮助文件）

3. 线条连接样式

`setJoinStyle (Qt::PenJoinStyle style)` 函数用于设置线条连接样式，参数是一个枚举类型 `Qt::PenJoinStyle` 的常量，该枚举类型的取值及其绘图效果如图 8-4 所示。

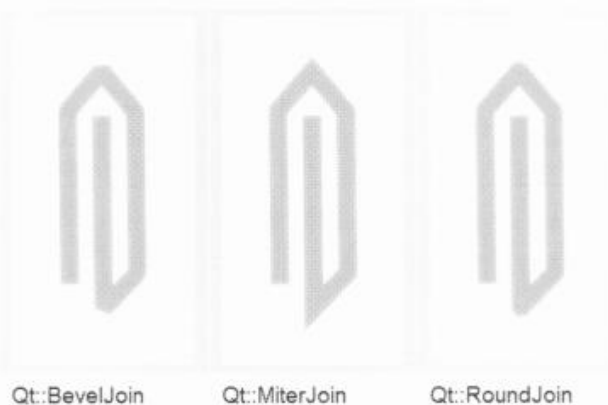


图 8-4 各种线条连接样式（来自 Qt 帮助文件）

QBrush

QBrush 定义了 QPainter 绘图时的填充特性，包括填充颜色、填充样式、材质填充时的材质图片等，其主要函数见表 8-2（省略了函数参数中的 const 关键字）。

表 8-2 QBrush 的主要函数

函数原型	功能
void setColorr(QColor &color)	设置画刷颜色，实体填充时即为填充颜色
void setStyle(Qt::BrushStyle style)	设置画刷样式，参数为 Qt::BrushStyle 枚举类型
void setTexture(QPixmap &pixmap)	设置一个 QPixmap 类型的图片作为画刷的图片，画刷样式自动设置为 Qt::TexturePattern
void setTextureImage(QImage &image)	设置一个 QImage 类型的图片作为画刷的图片，画刷样式自动设置为 Qt::TexturePattern

Qbrush填充样式

表 8-3 枚举类型 Qt:: BrushStyle 几个主要常量及其意义

枚举常量	描述
Qt:: NoBrush	不填充
Qt:: SolidPattern	单一颜色填充
Qt:: HorPattern	水平线填充
Qt:: VerPattern	垂直线填充
Qt:: LinearGradientPattern	线性渐变，需要使用 QLinearGradient 类对象作为 Brush
Qt:: RadialGradientPattern	辐射渐变，需要使用 QRadialGradient 类对象作为 Brush
Qt:: ConicalGradientPattern	圆锥型渐变，需要使用 QConicalGradient 类对象作为 Brush
Qt::TexturePattern	材质填充，需要指定 texture 或 textureImage 图片

渐变填充需要使用专门的类作为 Brush 赋值给 QPainter，这部分在后面详细介绍。其他各种线型填充只需设置类型参数即可，使用材质需要设置材质图片。

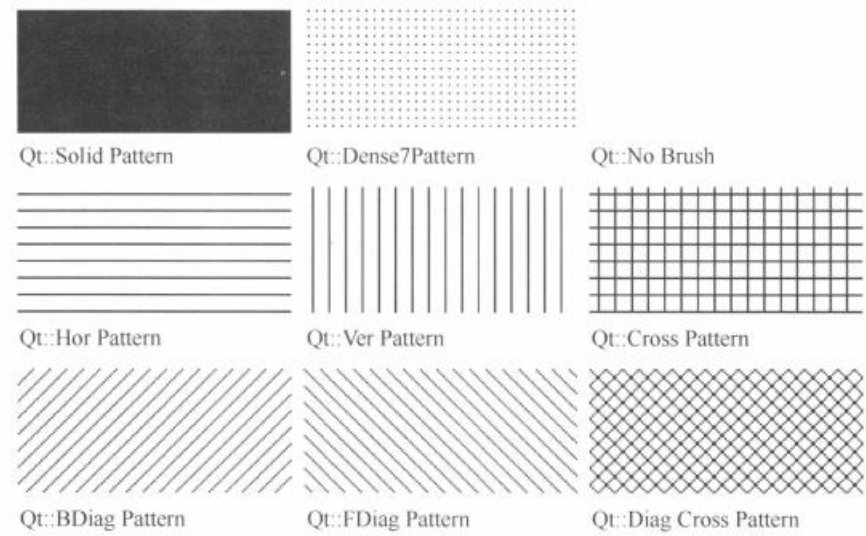






图 8-5 Qt::BrushStyle 几种填充样式（来自 Qt 帮助文件）

绘制各种图形

函数名	功能和示例代码	示例图形
drawArc	画弧线, 例如 <pre>QRect rect(W/4,H/4,W/2,H/2); int startAngle = 90 * 16; //起始 90° int spanAngle = 90 * 16; //旋转 90° painter.drawArc(rect, startAngle, spanAngle);</pre>	
drawChord	画一段弦, 例如 <pre>QRect rect(W/4,H/4,W/2,H/2); int startAngle = 90 * 16; //起始 90° int spanAngle = 90 * 16; //旋转 90° painter.drawChord(rect, startAngle, spanAngle);</pre>	
drawEllipse	画椭圆 <pre>QRect rect(W/4,H/4,W/2,H/2); painter.drawEllipse(rect);</pre>	
drawLine	画直线 <pre>QLine Line(W/4,H/4,W/2,H/2); painter.drawLine(Line);</pre>	
drawLines	画一批直线 <pre>QRect rect(W/4,H/4,W/2,H/2); QVector<QLine> Lines; Lines.append(QLine(rect.topLeft(),rect.bottomRight())); Lines.append(QLine(rect.topRight(),rect.bottomLeft())); Lines.append(QLine(rect.topLeft(),rect.bottomLeft())); Lines.append(QLine(rect.topRight(),rect.bottomRight())); painter.drawLines(Lines);</pre>	