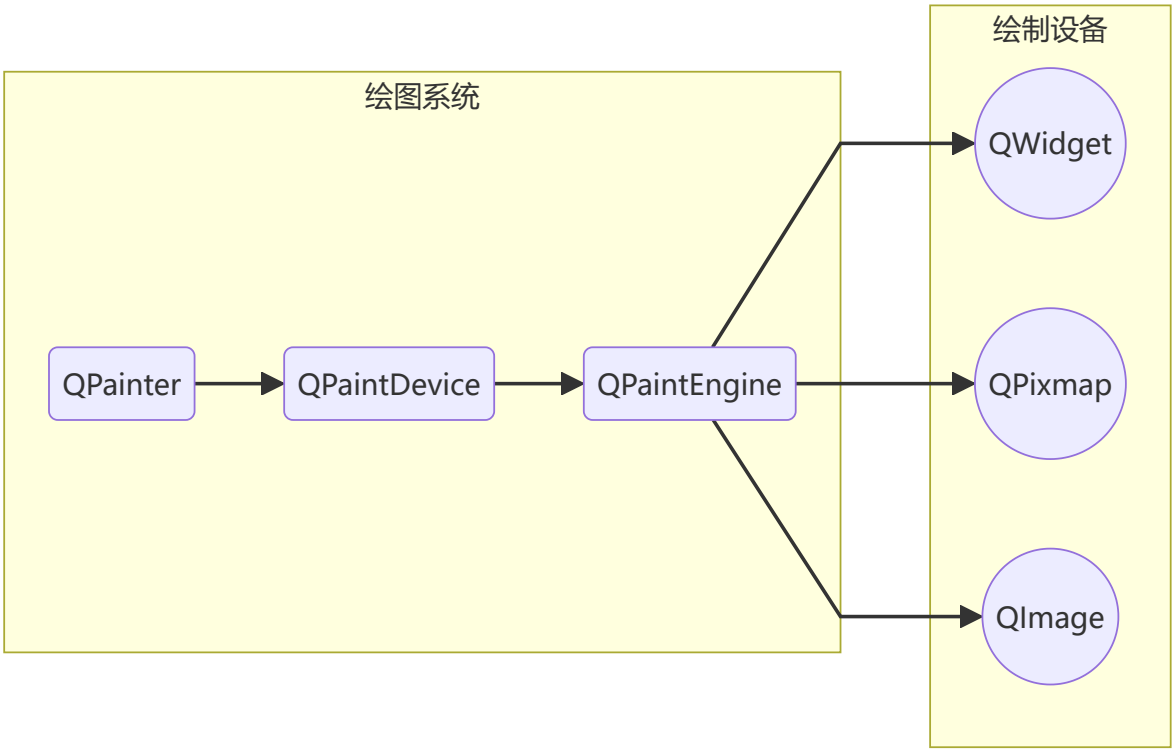
26 QPainter基本绘图

借助QPainter类的实例化对象，可以在绘制设备上绘制各种二维图形。绘制设备包括QWidget、QPixmap、QImage等。所绘制的点、线、面等几何图形，是不能与用户发生交互的纯视觉元素。

26.1 绘图系统

26.1.1 QPainter、QPaintDevice和QPaintEngine

Qt的绘图系统使用户可以在屏幕和打印机上，基于同一套APIs完成图形绘制。Qt的绘图系统由QPainter、QPaintDevice和QPaintEngine三个类组成。QPainter类提供表示具体绘制动作的接口。PainterDevice类表示可在其上绘制图形的二维表面。QPaintEngine类提供在不同设备上完成图形渲染的接口，仅在QPainter类和QPaintDevice类内部使用。应用程序一般无需与QPaintEngine打交道，除非要自己创建某种特殊的设备类型。Qt支持的绘制设备包括QWidget、QPixmap、QImage等，这些绘制设备在QPainter看来，就是一张张“画布”。



26.1.2 paintEvent事件与绘图区

QWidget类及其子类是最常用的绘制设备。QWidget类的所有子类都可以通过覆盖其基类的paintEvent虚函数，响应绘制事件，执行绘制动作。在该虚函数的实现中，只需实例化一个QPainter类的对象，并将作为“画布”的绘制设备交给该对象即可。之后通过该对象完成的所有绘制动作，都会呈现在指定的绘制设备上。

QWidget的绘图区就是其窗口内部的矩形区域。绘图区中的坐标以像素为单位，坐标系的原点位于绘图区的左上角，X轴向右为正，Y轴向下为正。绘图区的宽度和高度分别由QWidget类的width和height方法获得。在这个坐标系中的坐标，称为视口（Viewport）坐标。相应地，还有另一套坐标系，其中的坐标称为窗口（Window）坐标。通过QPainter对象完成的所有绘制，都不会超出绘图区的范围。

26.1.3 QPainter的主要属性

借助QPainter对象，可以实现一些基本图形的绘制，如点、直线、圆形、矩形、曲线、文字等。控制这些图形元素的特性主要来自QPainter的以下三个属性：

- 画笔：用于绘制线条。QPen类的对象，包括颜色、线宽、线型、线端样式、连接样式等属性；
- 画刷：用于填充区域。QBrush类的对象，包括颜色、填充样式、纹理图片等属性；
- 字体：用于绘制文本。QFont类的对象，包括字体名、大小、粗细、加粗、斜体、下划线等属性。

这三个属性决定了所绘图形的基本特征。此外还有一些辅助属性，如叠加模式、旋转缩放等。

26.1.4 案例

26.1.4.1 创建项目

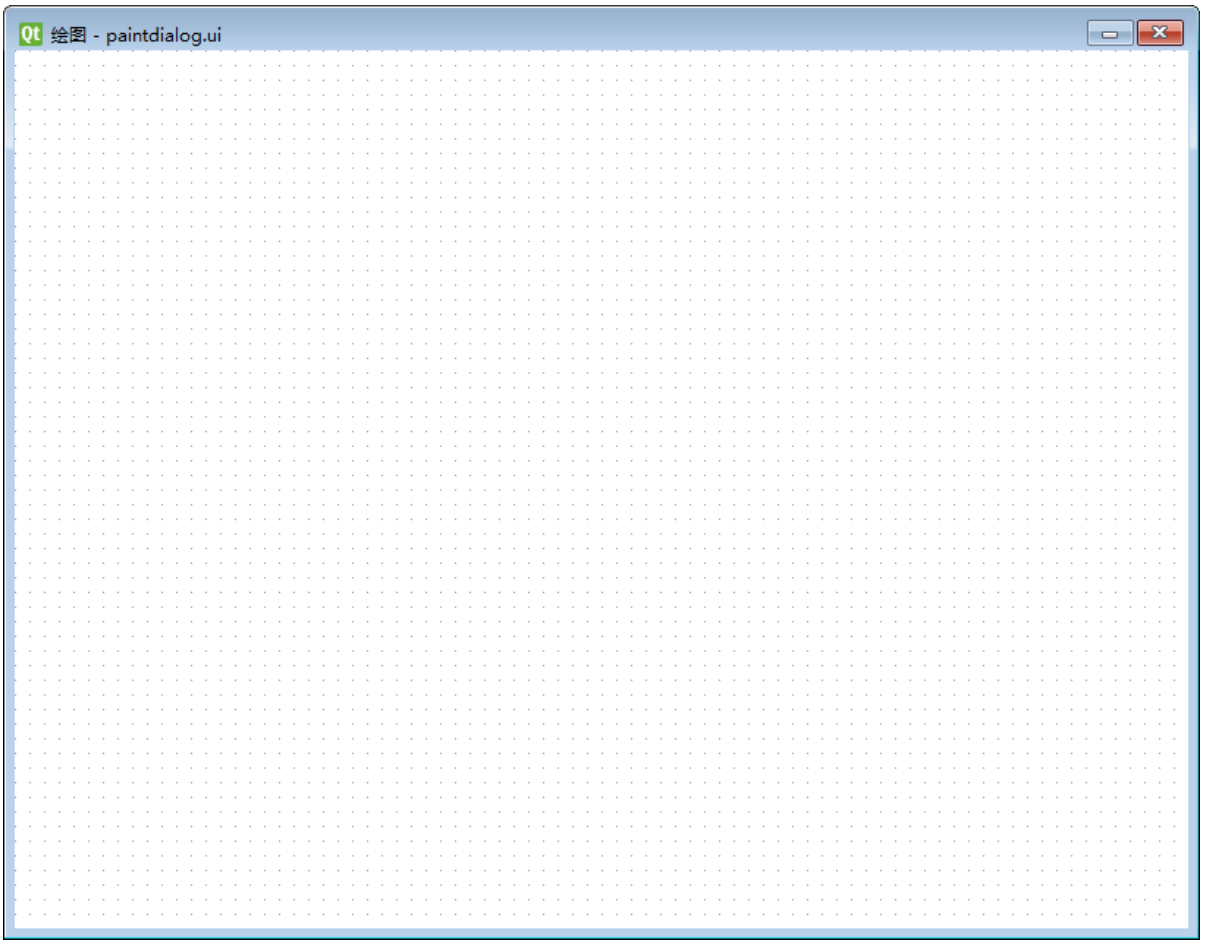
通过QtCreator，在C:\Users\Minwei\Projects\Qt路径下，创建名为Paint的项目。

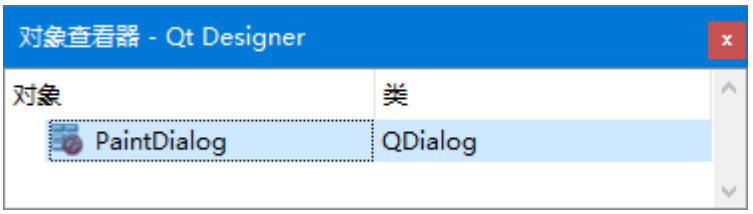
26.1.4.2 添加资源

C:\Users\Minwei\Projects\Qt\Paint\Paint.qrc：

```
<RCC>
    <qresource prefix="/">
        <file>images/1.png</file>
    </qresource>
</RCC>
```

26.1.4.3 设计界面





C:\Users\Minwei\Projects\Qt\Paint\paintdialog.ui：

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>PaintDialog</class>
 <widget class="QDialog" name="PaintDialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>800</width>
    <height>600</height>
   </rect>
  </property>
  <property name="palette">
   <palette>
    <active>
     <colorrole role="Base">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
        <red>255</red>
        <green>255</green>
        <blue>255</blue>
       </color>
      </brush>
     </colorrole>
     <colorrole role="Window">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
        <red>255</red>
        <green>255</green>
        <blue>255</blue>
       </color>
      </brush>
     </colorrole>
    </active>
    <inactive>
     <colorrole role="Base">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
        <red>255</red>
        <green>255</green>
        <blue>255</blue>
       </color>
      </brush>
     </colorrole>
     <colorrole role="Window">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
```

```
         <red>255</red>
         <green>255</green>
         <blue>255</blue>
        </color>
       </brush>
      </colorrole>
     </inactive>
     <disabled>
      <colorrole role="Base">
       <brush brushstyle="SolidPattern">
        <color alpha="255">
         <red>255</red>
         <green>255</green>
         <blue>255</blue>
        </color>
       </brush>
      </colorrole>
      <colorrole role="Window">
       <brush brushstyle="SolidPattern">
        <color alpha="255">
         <red>255</red>
         <green>255</green>
         <blue>255</blue>
        </color>
       </brush>
      </colorrole>
     </disabled>
    </palette>
   </property>
   <property name="windowTitle">
    <string>绘图</string>
   </property>
  </widget>
  <resources/>
  <connections/>
</ui>
```

26.1.4.4 实现功能

C:\Users\Minwei\Projects\Qt\Paint\paintdialog.h：

```
#ifndef PAINTDIALOG_H
#define PAINTDIALOG_H

#include <QDialog>

QT_BEGIN_NAMESPACE
namespace Ui { class PaintDialog; }
QT_END_NAMESPACE

class PaintDialog : public QDialog
{
    Q_OBJECT

public:
```

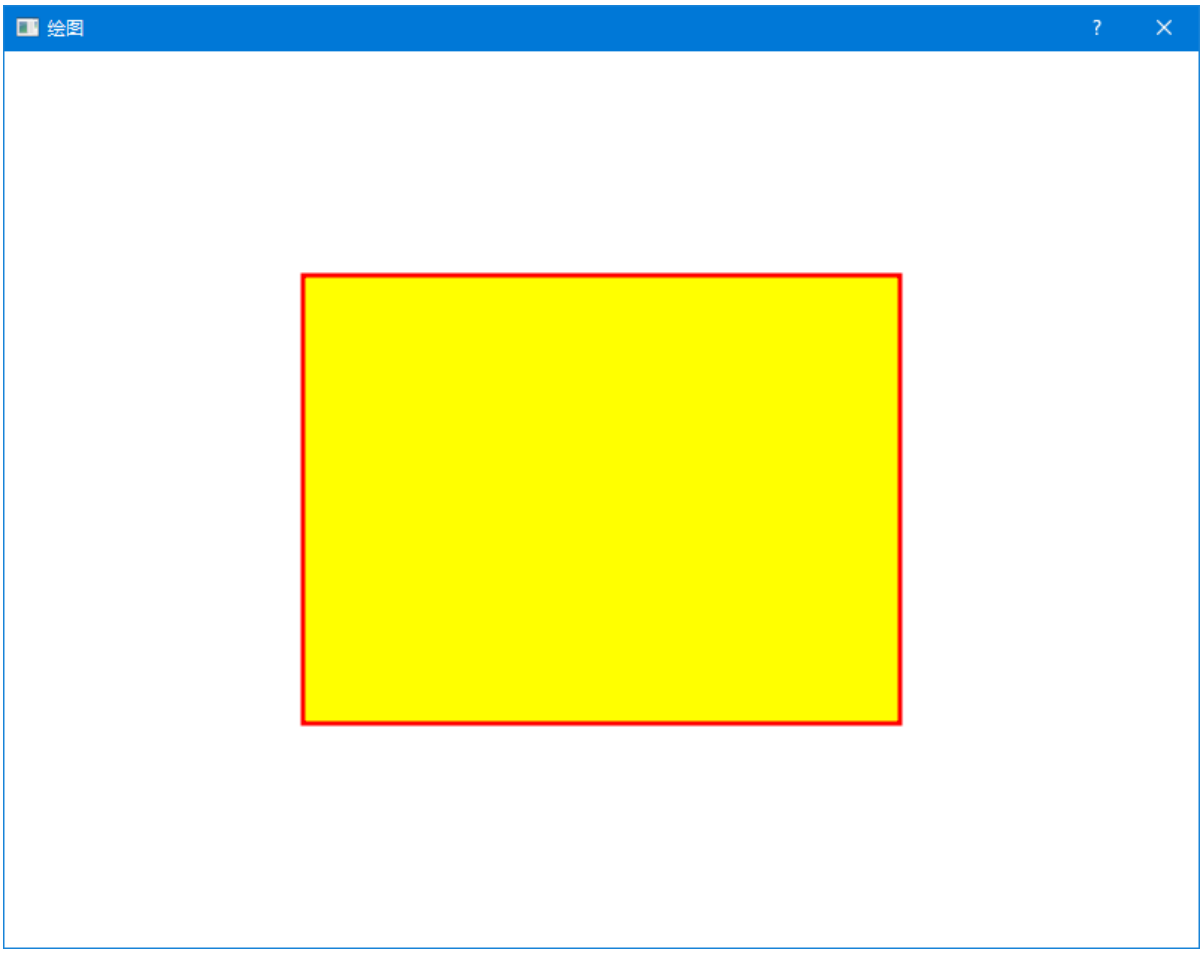
```
    PaintDialog(QWidget *parent = nullptr);
    ~PaintDialog();

protected:
    void paintEvent(QPaintEvent*);

private:
    Ui::PaintDialog *ui;
};

#endif // PAINTDIALOG_H
```

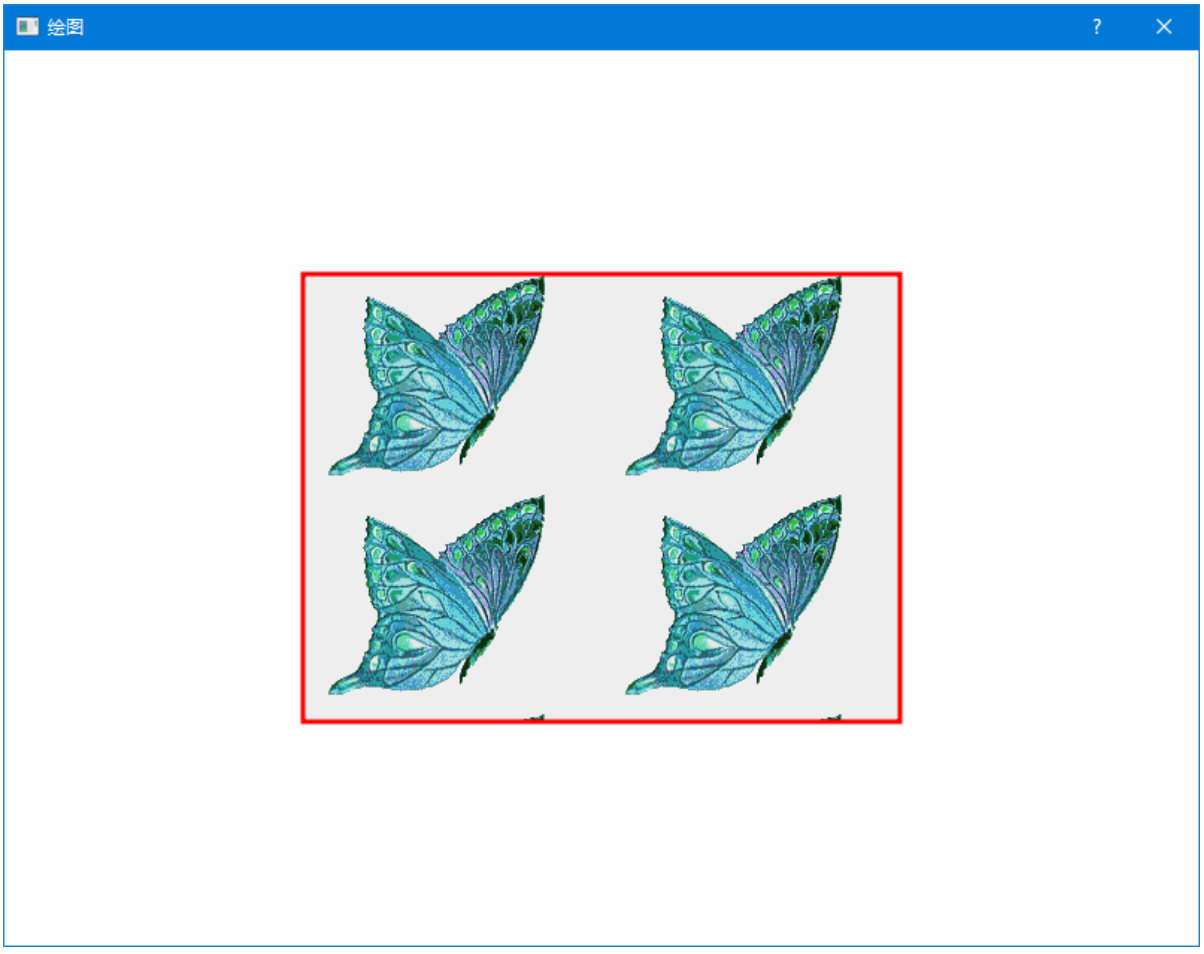
C:\Users\Minwei\Projects\Qt\Paint\paintdialog.cpp:

```
#include <QPainter>

#include "paintdialog.h"
#include "ui_paintdialog.h"

PaintDialog::PaintDialog(QWidget *parent)
    : QDialog(parent)
    , ui(new Ui::PaintDialog)
{
    ui->setupUi(this);
}

PaintDialog::~PaintDialog()
{
    delete ui;
}

void PaintDialog::paintEvent(QPaintEvent*)
{
    QPainter painter(this);
    painter.setRenderHint(QPainter::Antialiasing);

    int w = width(), h = height();

    QPen pen;
    pen.setColor(Qt::red);
    pen.setWidth(3);
    pen.setStyle(Qt::SolidLine);
    pen.setJoinStyle(Qt::MiterJoin);
    painter.setPen(pen);
    QBrush brush;
    //brush.setStyle(Qt::SolidPattern);
    //brush.setColor(Qt::yellow);
    brush.setStyle(Qt::TexturePattern);
    brush.setTextureImage(QImage(":/images/1.png"));
    painter.setBrush(brush);

    painter.drawRect(QRect(QPoint(w/4, h/4), QPoint(w*3/4, h*3/4)));
}
```

26.1.4.5 测试验证

运行效果如图所示：





26.2 基本图形

26.2.1 绘制基本图形的方法

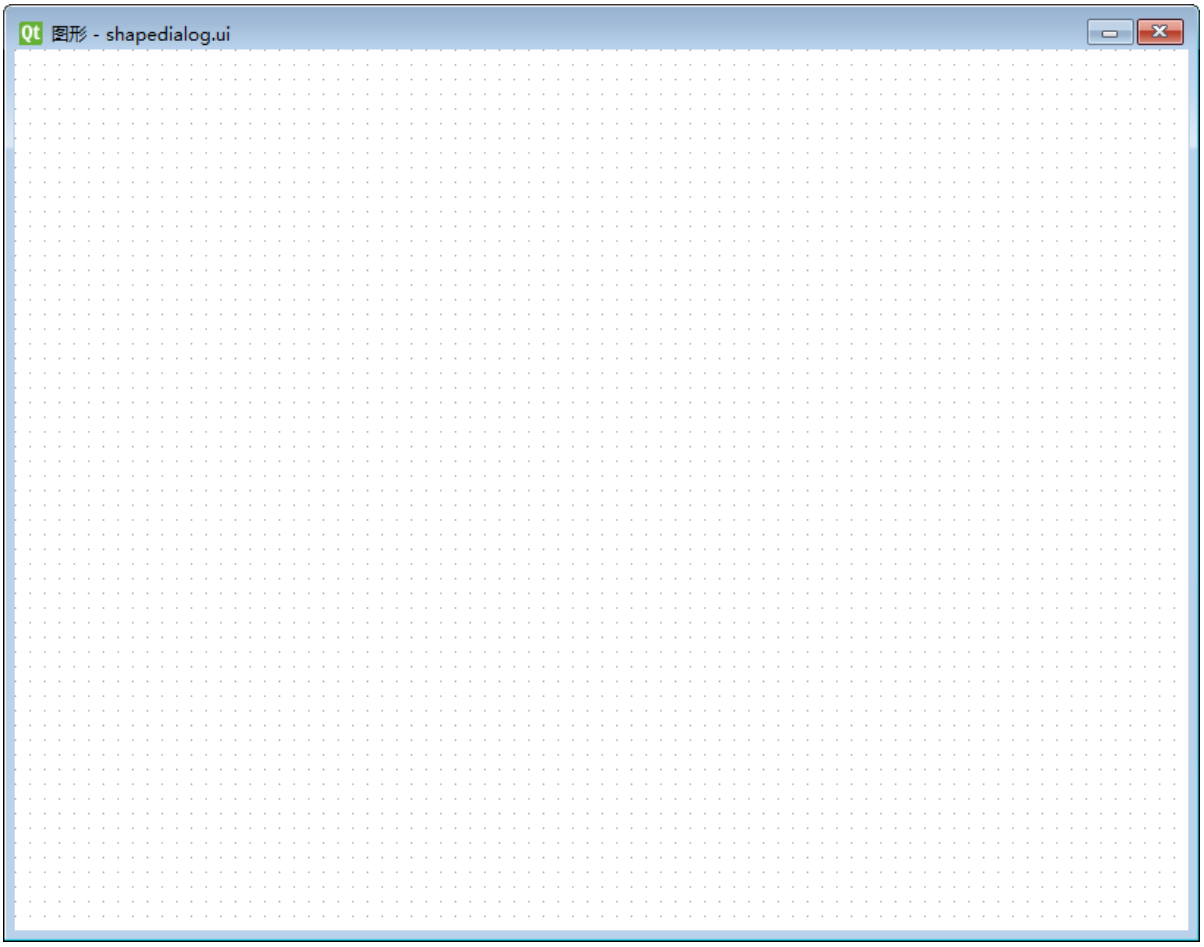
```
1  void QPainter::drawPoint(const QPoint& position);
                                    // 绘制点
2  void QPainter::drawLine(const QLine& line);
                                    // 绘制直线
3  void QPainter::drawRect(const QRect& rectangle);
                                    // 绘制矩形
4  void QPainter::drawEllipse(const QRect& rectangle);
                                    // 绘制椭圆
5  void QPainter::drawArc(const QRect& rectangle, int startAngle, int
   spanAngle);                                    // 绘制椭圆弧
6  void QPainter::drawChord(const QRect& rectangle, int startAngle, int
   spanAngle);                                  // 绘制弦
7  void QPainter::drawPie(const QRect& rectangle, int startAngle, int
   spanAngle);                                    // 绘制扇形
8  void QPainter::drawPolyline(const QPolygon& points);
                                    // 绘制折线
9  void QPainter::drawPolygon(const QPolygon& points, Qt::FillRule fillRule =
   Qt::OddEvenFill);                    // 绘制多边形
10 void QPainter::drawText(const QRect& rectangle, int flags, const QString&
   text, QRect* boundingRect = nullptr); // 绘制文本
11 ...
```

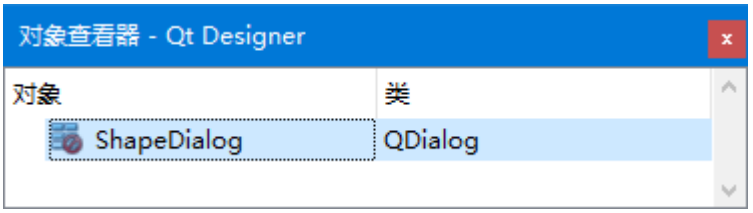
26.2.2 案例

26.2.2.1 创建项目

通过QtCreator，在C:\Users\Minwei\Projects\Qt路径下，创建名为Shape的项目。

26.2.2.2 设计界面





C:\Users\Minwei\Projects\Qt\Shape\shapedialog.ui：

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>ShapeDialog</class>
 <widget class="QDialog" name="ShapeDialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>800</width>
    <height>600</height>
   </rect>
  </property>
  <property name="palette">
   <palette>
    <active>
     <colorrole role="Base">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
        <red>255</red>
        <green>255</green>
```

```
          <blue>255</blue>
         </color>
        </brush>
       </colorrole>
       <colorrole role="Window">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>255</red>
          <green>255</green>
          <blue>255</blue>
         </color>
        </brush>
       </colorrole>
      </active>
      <inactive>
       <colorrole role="Base">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>255</red>
          <green>255</green>
          <blue>255</blue>
         </color>
        </brush>
       </colorrole>
       <colorrole role="Window">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>255</red>
          <green>255</green>
          <blue>255</blue>
         </color>
        </brush>
       </colorrole>
      </inactive>
      <disabled>
       <colorrole role="Base">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>255</red>
          <green>255</green>
          <blue>255</blue>
         </color>
        </brush>
       </colorrole>
       <colorrole role="Window">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>255</red>
          <green>255</green>
          <blue>255</blue>
         </color>
        </brush>
       </colorrole>
      </disabled>
     </palette>
    </property>
```

```
  <property name="windowTitle">
   <string>图形</string>
  </property>
 </widget>
 <resources/>
 <connections/>
</ui>
```

26.2.2.3 实现功能

C:\Users\Minwei\Projects\Qt\Shape\shapedialog.h：

```
#ifndef SHAPEDIALOG_H
#define SHAPEDIALOG_H

#include <QDialog>

QT_BEGIN_NAMESPACE
namespace Ui { class ShapeDialog; }
QT_END_NAMESPACE

class ShapeDialog : public QDialog
{
    Q_OBJECT

public:
    ShapeDialog(QWidget *parent = nullptr);
    ~ShapeDialog();

protected:
    void paintEvent(QPaintEvent*);

private:
    void drawPoint(void);
    void drawLine(void);
    void drawRect(void);
    void drawEllipse(void);
    void drawArc(void);
    void drawChord(void);
    void drawPie(void);
    void drawPolyline(void);
    void drawPolygon(void);
    void drawText(void);

private:
    Ui::ShapeDialog *ui;
};

#endif // SHAPEDIALOG_H
```

C:\Users\Minwei\Projects\Qt\Shape\shapedialog.cpp：

```
#include <QPainter>

#include "shapedialog.h"
```

```cpp
#include "ui_shapedialog.h"

ShapeDialog::ShapeDialog(QWidget *parent)
    : QDialog(parent)
    , ui(new Ui::ShapeDialog)
{
    ui->setupUi(this);
}

ShapeDialog::~ShapeDialog()
{
    delete ui;
}

void ShapeDialog::paintEvent(QPaintEvent*)
{
    drawPoint();
    //drawLine();
    //drawRect();
    //drawEllipse();
    //drawArc();
    //drawChord();
    //drawPie();
    //drawPolyline();
    //drawPolygon();
    //drawText();
}

void ShapeDialog::drawPoint(void)
{
    QPainter painter(this);
    painter.setRenderHint(QPainter::Antialiasing);

    int w = width(), h = height();
    int xo = (w - 256) / 2, yo = (h - 256) / 2;

    for (int x = 0; x < 256; ++x)
        for (int y = 0; y < 256; ++y)
        {
            QPen pen;
            pen.setColor(QColor(x, y, 0));
            painter.setPen(pen);
            painter.drawPoint(QPoint(xo + x, yo + y));
        }
}

void ShapeDialog::drawLine(void)
{
    QPainter painter(this);
    painter.setRenderHint(QPainter::Antialiasing);

    int w = width(), h = height();

    QPen pen;
    pen.setColor(Qt::red);
    pen.setWidth(3);
```

```
    pen.setStyle(Qt::SolidLine);
    pen.setJoinStyle(Qt::MiterJoin);
    painter.setPen(pen);

    painter.drawLine(QLine(QPoint(0, 0), QPoint(w, h)));
    painter.drawLine(QLine(QPoint(w, 0), QPoint(0, h)));
}

void ShapeDialog::drawRect(void)
{
    QPainter painter(this);
    painter.setRenderHint(QPainter::Antialiasing);

    int w = width(), h = height();

    QPen pen;
    pen.setColor(Qt::red);
    pen.setWidth(3);
    pen.setStyle(Qt::SolidLine);
    pen.setJoinStyle(Qt::MiterJoin);
    painter.setPen(pen);
    QBrush brush;
    brush.setColor(Qt::yellow);
    brush.setStyle(Qt::SolidPattern);
    painter.setBrush(brush);

    painter.drawRect(QRect(QPoint(w/4, h/4), QPoint(w*3/4, h*3/4)));
}

void ShapeDialog::drawEllipse(void)
{
    QPainter painter(this);
    painter.setRenderHint(QPainter::Antialiasing);

    int w = width(), h = height();

    QPen pen;
    pen.setColor(Qt::red);
    pen.setWidth(3);
    pen.setStyle(Qt::SolidLine);
    pen.setJoinStyle(Qt::MiterJoin);
    painter.setPen(pen);
    QBrush brush;
    brush.setColor(Qt::yellow);
    brush.setStyle(Qt::SolidPattern);
    painter.setBrush(brush);

    painter.drawEllipse(QRect(QPoint(w/4, h/4), QPoint(w*3/4, h*3/4)));
}

void ShapeDialog::drawArc(void)
{
    QPainter painter(this);
    painter.setRenderHint(QPainter::Antialiasing);

    int w = width(), h = height();
```

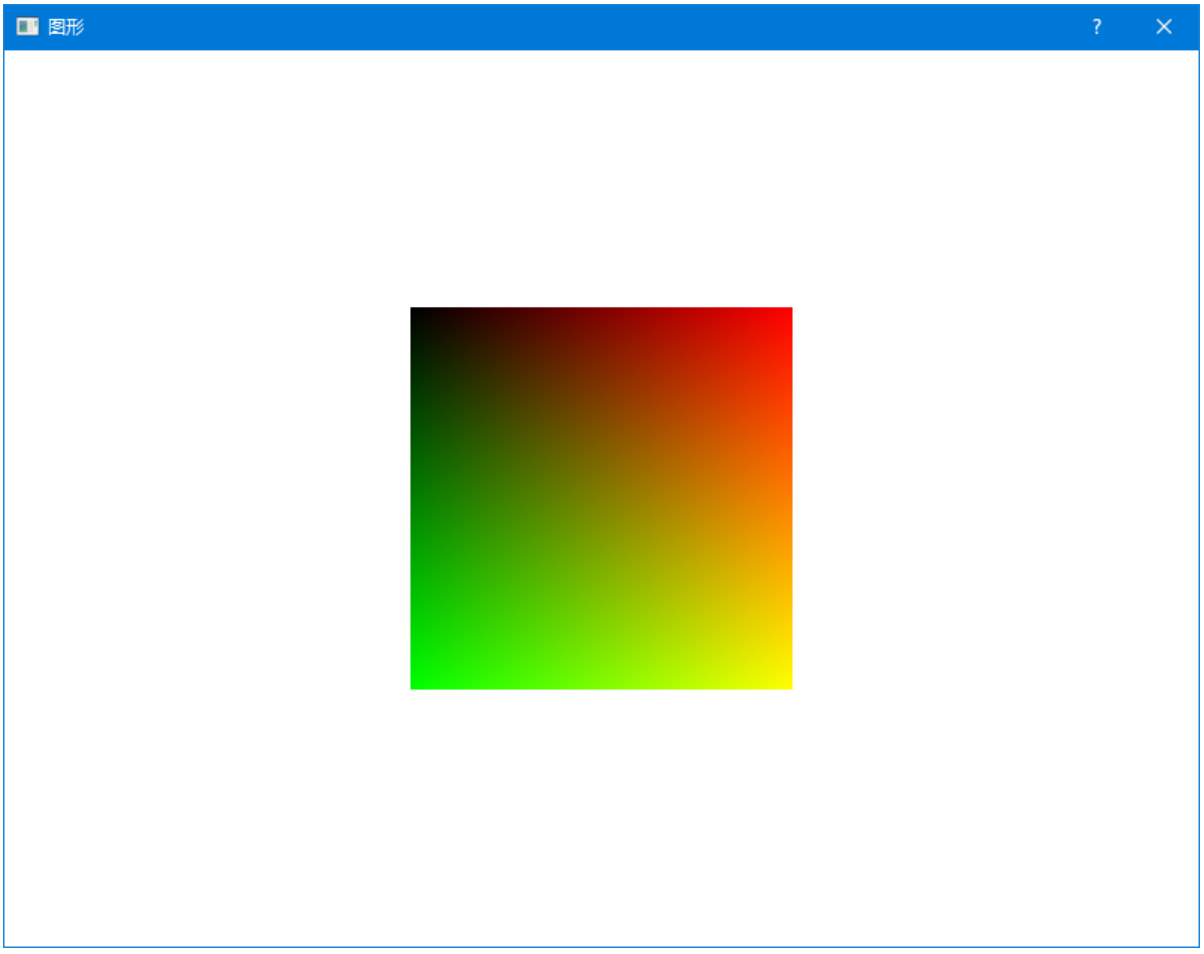
```
116
117     QPen pen;
118     pen.setColor(Qt::red);
119     pen.setWidth(3);
120     pen.setStyle(Qt::SolidLine);
121     pen.setJoinStyle(Qt::MiterJoin);
122     painter.setPen(pen);
123
124     painter.drawArc(QRect(QPoint(w/4, h/4), QPoint(w*3/4, h*3/4)),
125         30 * 16, 120 * 16);
126 }
127
128 void ShapeDialog::drawChord(void)
129 {
130     QPainter painter(this);
131     painter.setRenderHint(QPainter::Antialiasing);
132
133     int w = width(), h = height();
134
135     QPen pen;
136     pen.setColor(Qt::red);
137     pen.setWidth(3);
138     pen.setStyle(Qt::SolidLine);
139     pen.setJoinStyle(Qt::MiterJoin);
140     painter.setPen(pen);
141     QBrush brush;
142     brush.setColor(Qt::yellow);
143     brush.setStyle(Qt::SolidPattern);
144     painter.setBrush(brush);
145
146     painter.drawChord(QRect(QPoint(w/4, h/4), QPoint(w*3/4, h*3/4)),
147         30 * 16, 120 * 16);
148 }
149
150 void ShapeDialog::drawPie(void)
151 {
152     QPainter painter(this);
153     painter.setRenderHint(QPainter::Antialiasing);
154
155     int w = width(), h = height();
156
157     QPen pen;
158     pen.setColor(Qt::red);
159     pen.setWidth(3);
160     pen.setStyle(Qt::SolidLine);
161     pen.setJoinStyle(Qt::MiterJoin);
162     painter.setPen(pen);
163     QBrush brush;
164     brush.setColor(Qt::yellow);
165     brush.setStyle(Qt::SolidPattern);
166     painter.setBrush(brush);
167
168     painter.drawPie(QRect(QPoint(w/4, h/4), QPoint(w*3/4, h*3/4)),
169         30 * 16, 120 * 16);
170 }
171
```

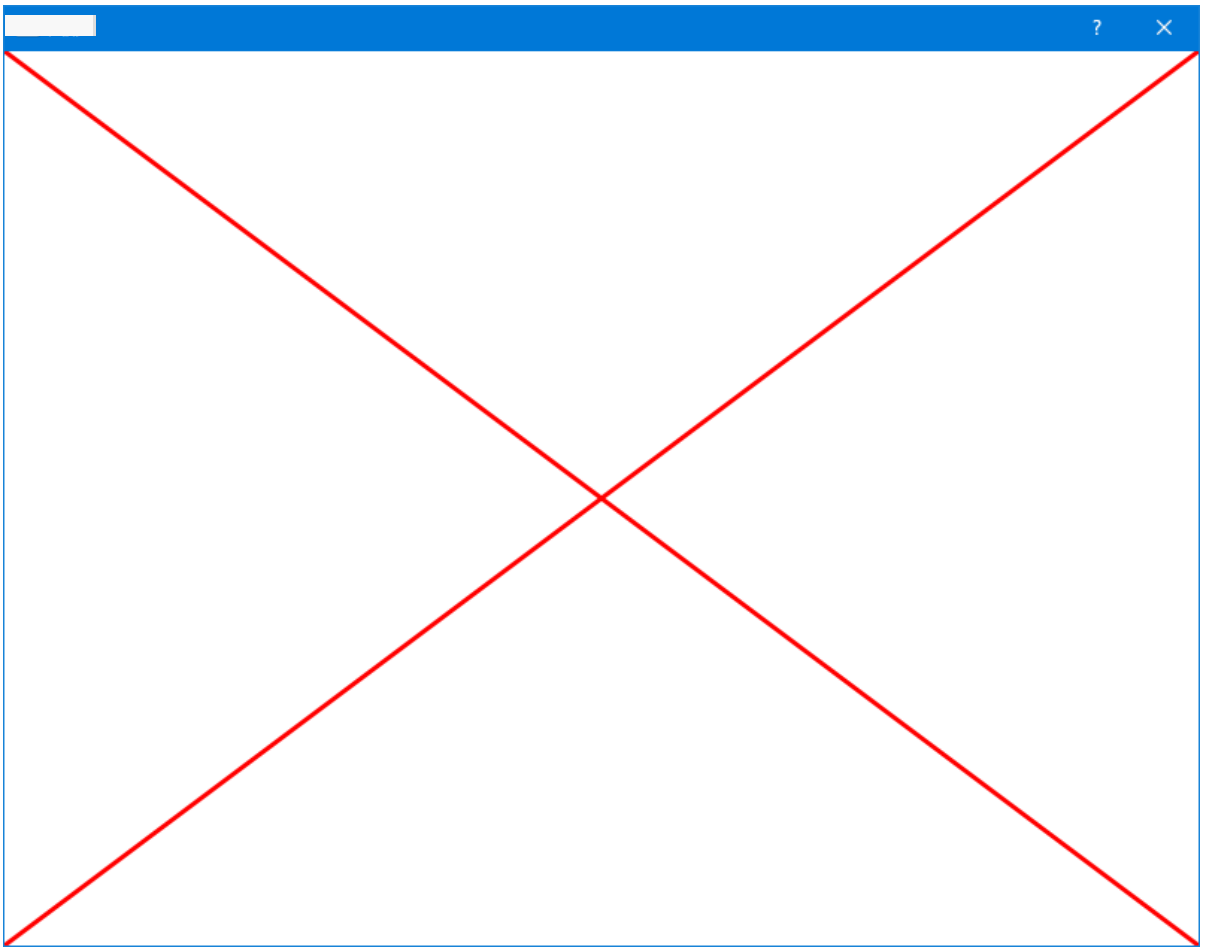
```
172 void ShapeDialog::drawPolyline(void)
173 {
174     QPainter painter(this);
175     painter.setRenderHint(QPainter::Antialiasing);
176 
177     int w = width(), h = height();
178 
179     QPen pen;
180     pen.setColor(Qt::red);
181     pen.setWidth(3);
182     pen.setStyle(Qt::SolidLine);
183     pen.setJoinStyle(Qt::MiterJoin);
184     painter.setPen(pen);
185 
186     painter.drawPolyline(QPolygon()
187         << QPoint(w/4, h*3/4) << QPoint(w/2, h/4) << QPoint(w*3/4, h*3/4));
188 }
189 
190 void ShapeDialog::drawPolygon(void)
191 {
192     QPainter painter(this);
193     painter.setRenderHint(QPainter::Antialiasing);
194 
195     int w = width(), h = height();
196 
197     QPen pen;
198     pen.setColor(Qt::red);
199     pen.setWidth(3);
200     pen.setStyle(Qt::SolidLine);
201     pen.setJoinStyle(Qt::MiterJoin);
202     painter.setPen(pen);
203     QBrush brush;
204     brush.setColor(Qt::yellow);
205     brush.setStyle(Qt::SolidPattern);
206     painter.setBrush(brush);
207 
208     painter.drawPolygon(QPolygon()
209         << QPoint(w/4, h*3/4) << QPoint(w/2, h/4) << QPoint(w*3/4, h*3/4));
210 }
211 
212 void ShapeDialog::drawText(void)
213 {
214     QPainter painter(this);
215     painter.setRenderHint(QPainter::TextAntialiasing);
216 
217     int w = width(), h = height();
218 
219     QPen pen;
220     pen.setColor(Qt::red);
221     painter.setPen(pen);
222     QFont font;
223     font.setFamily("Calibri");
224     font.setPointSize(h/12);
225     font.setBold(true);
226     painter.setFont(font);
227 
```

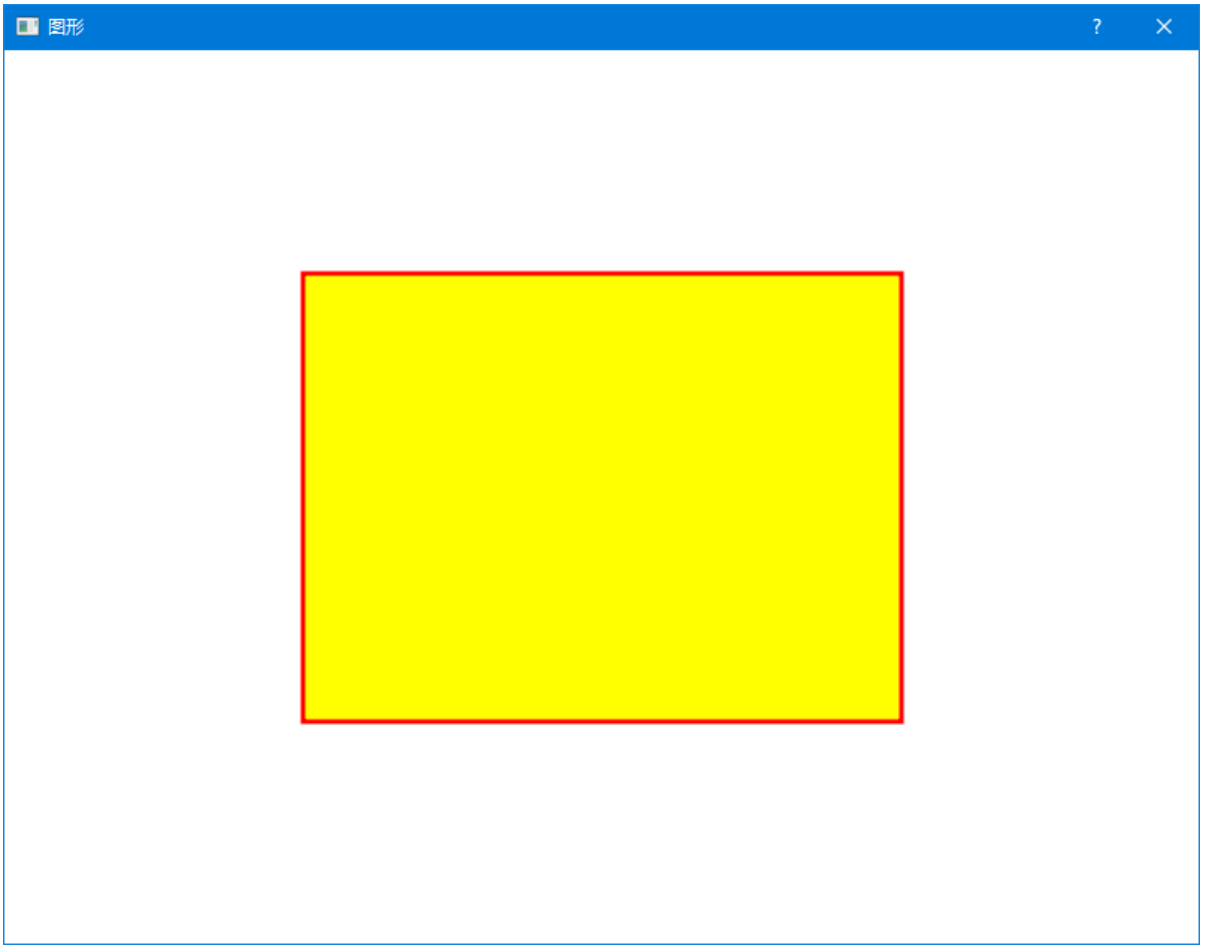
```
228     painter.drawText(QRect(QPoint(0, 0), QPoint(w, h)),
229         Qt::AlignCenter | Qt::TextWordWrap,
230         "The quick brown fox jumps over the lazy dog");
231 }
```

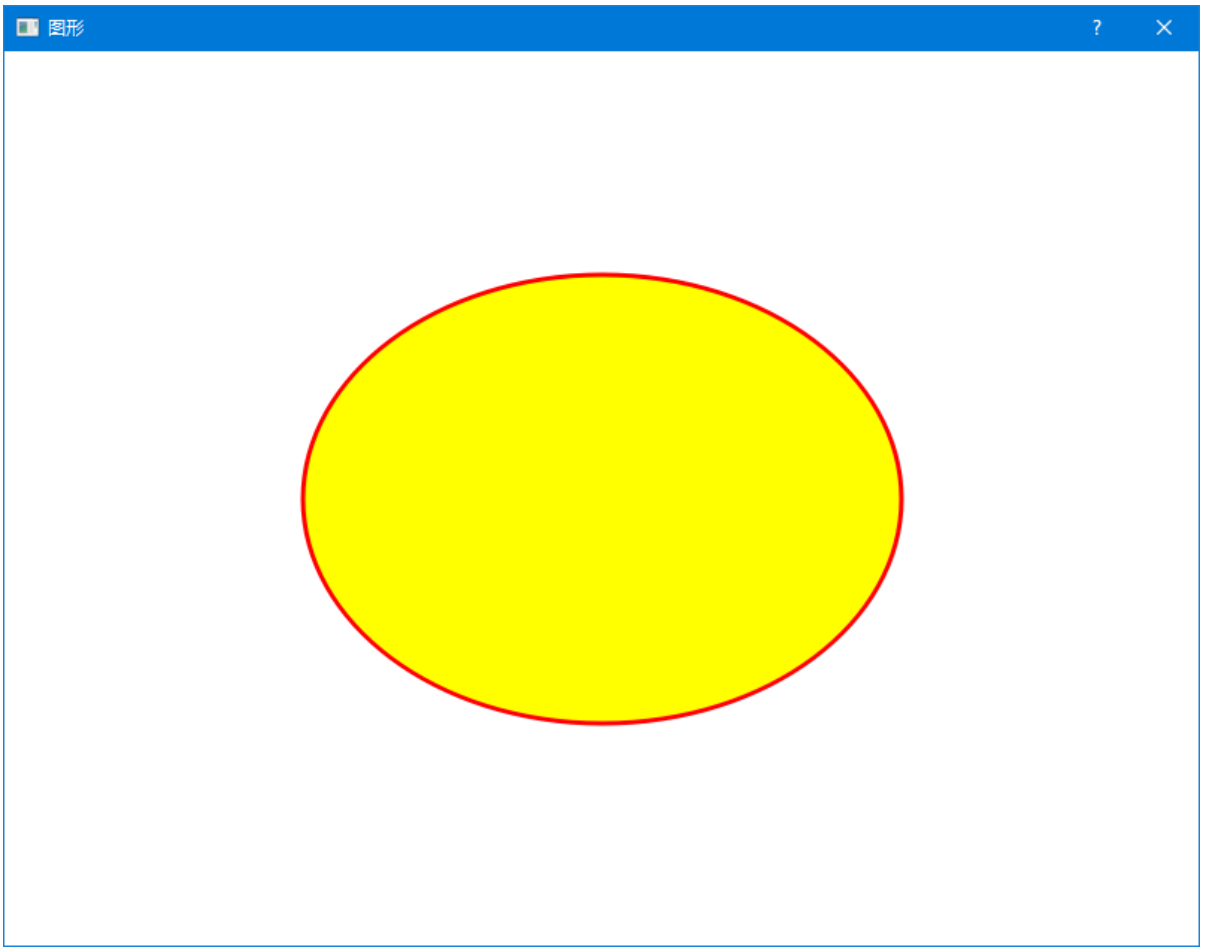
26.2.2.4 测试验证

运行效果如图所示：

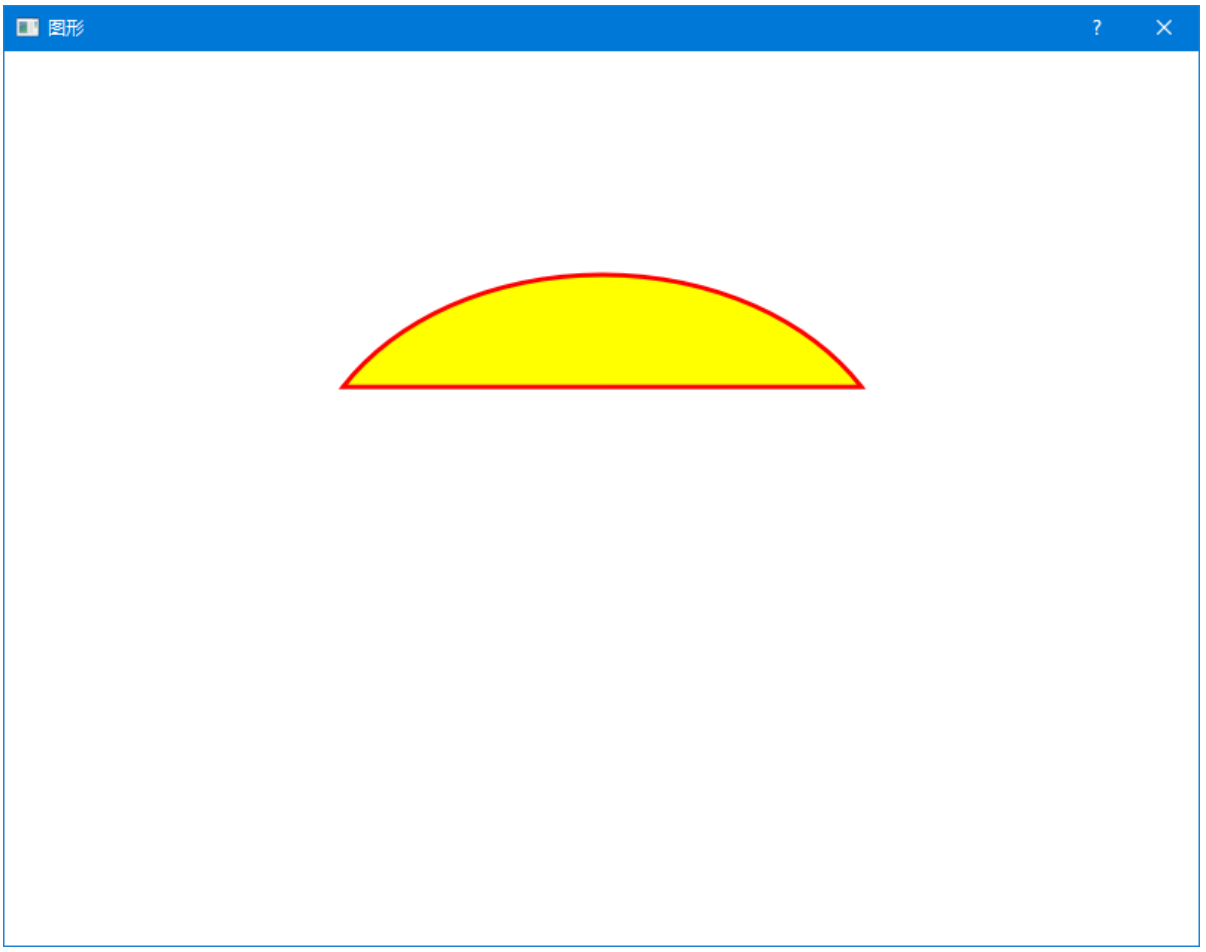


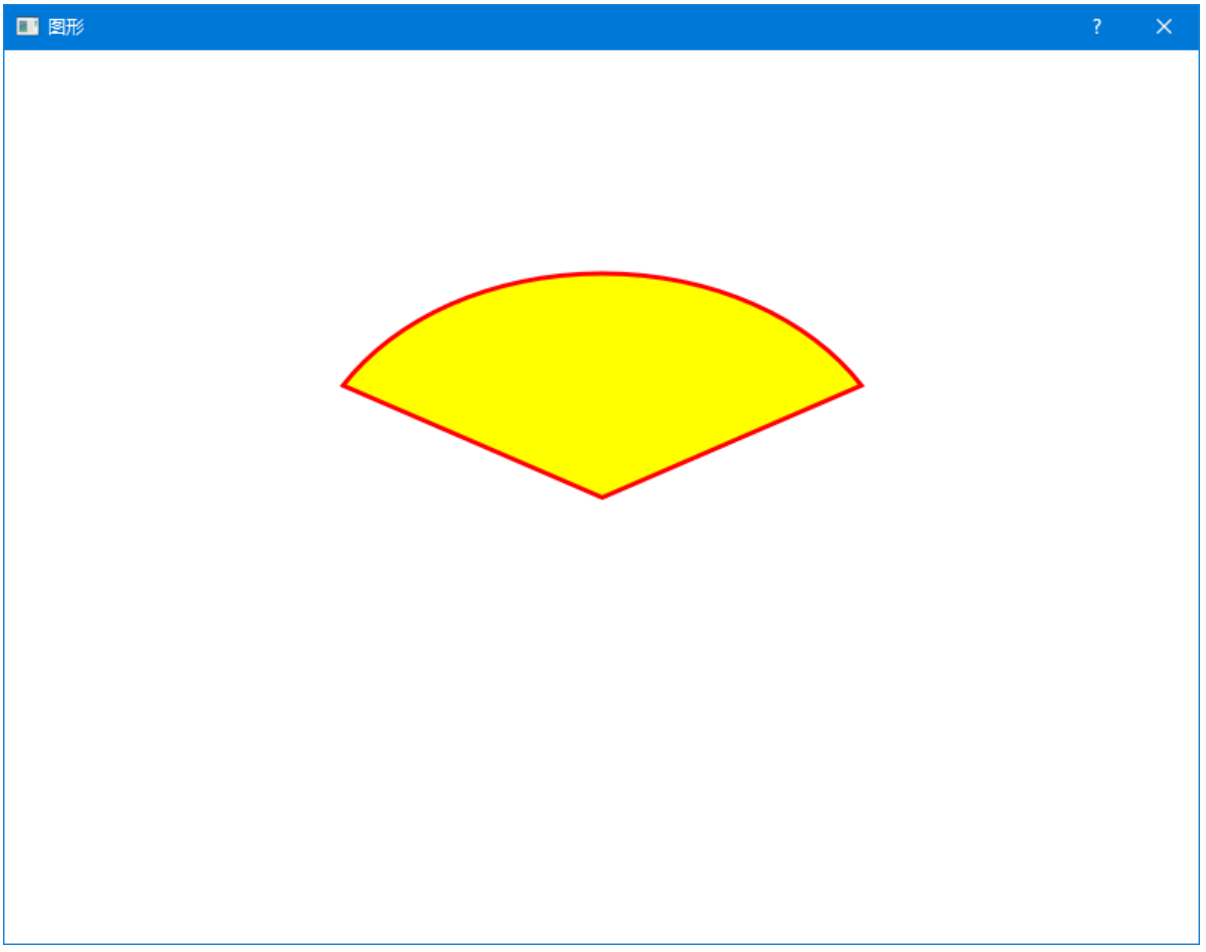


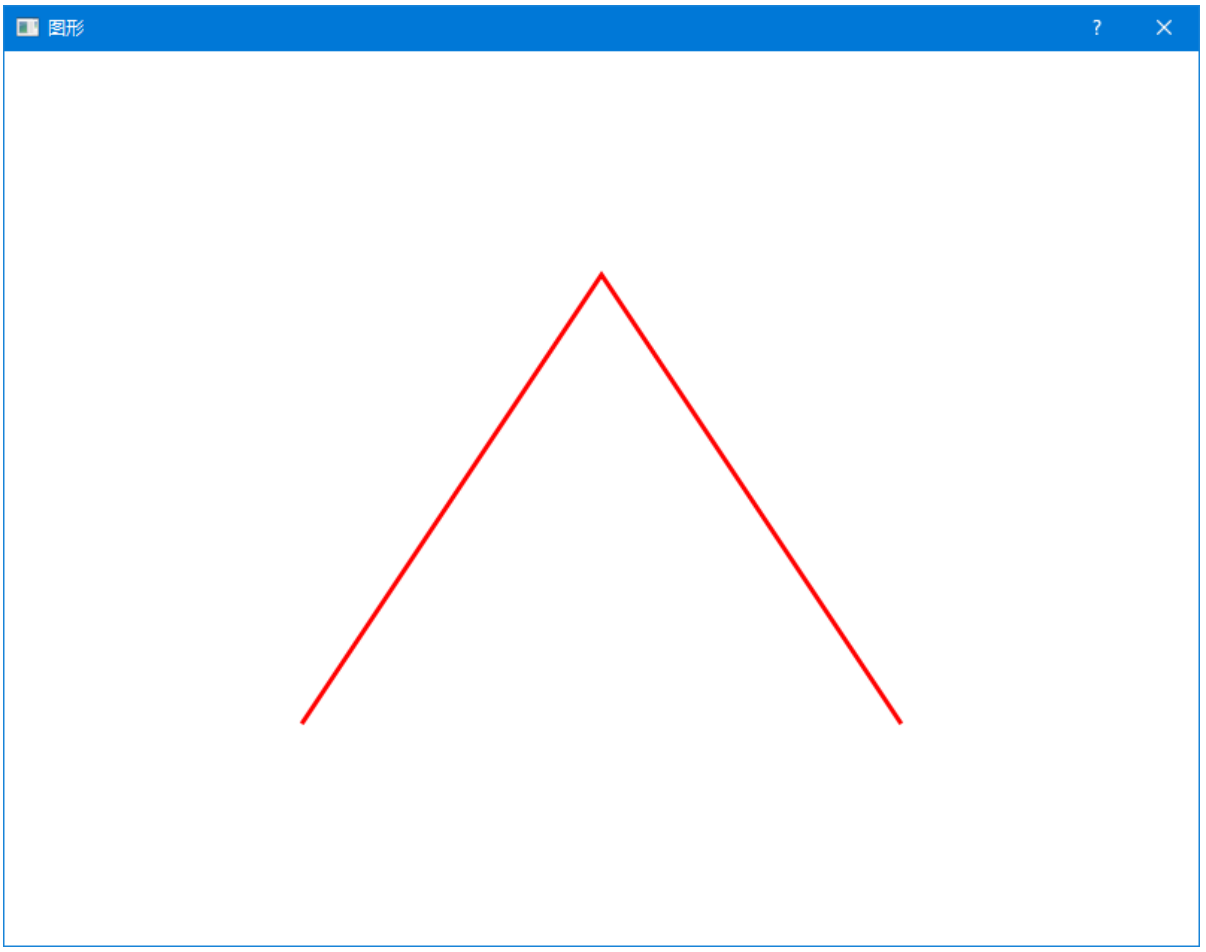

图形

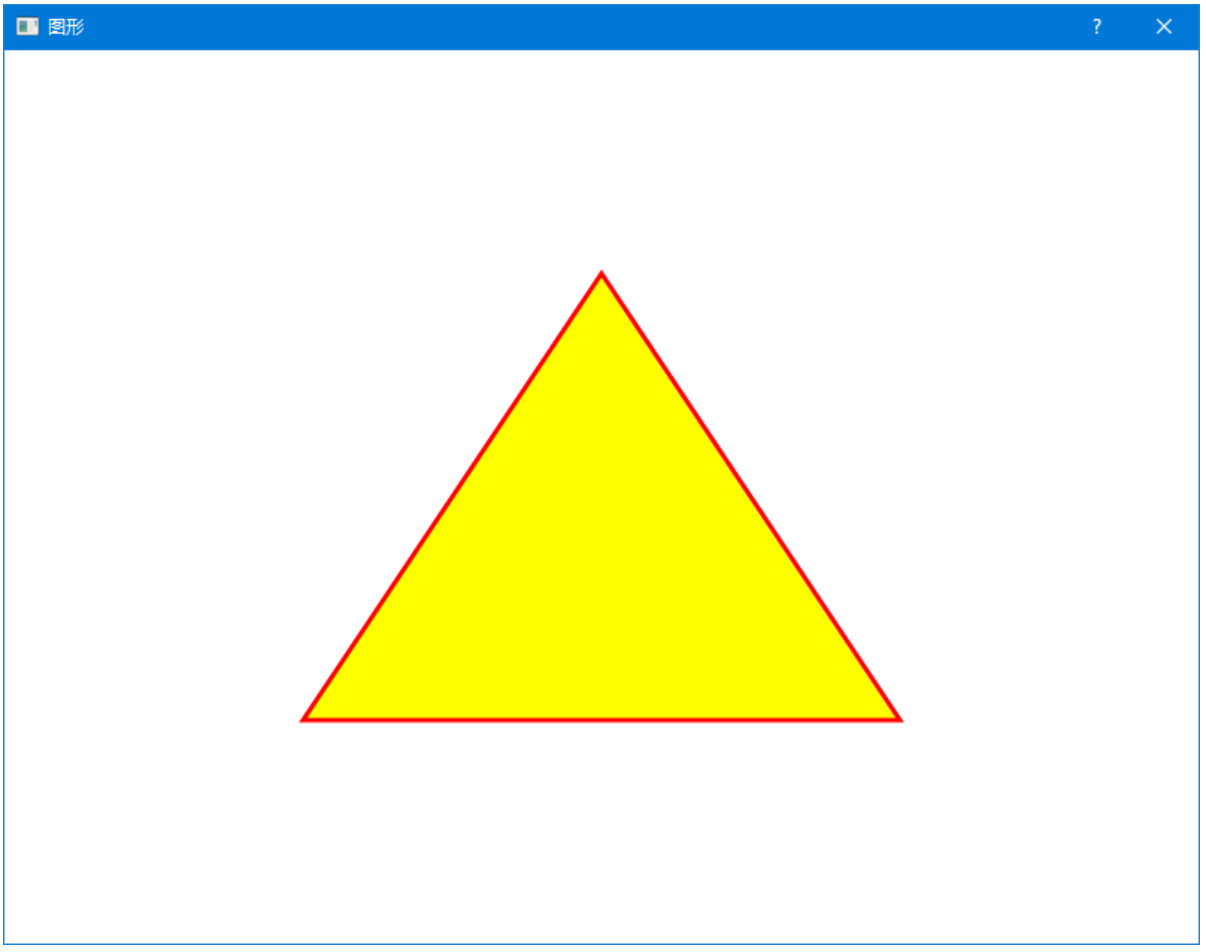

图形

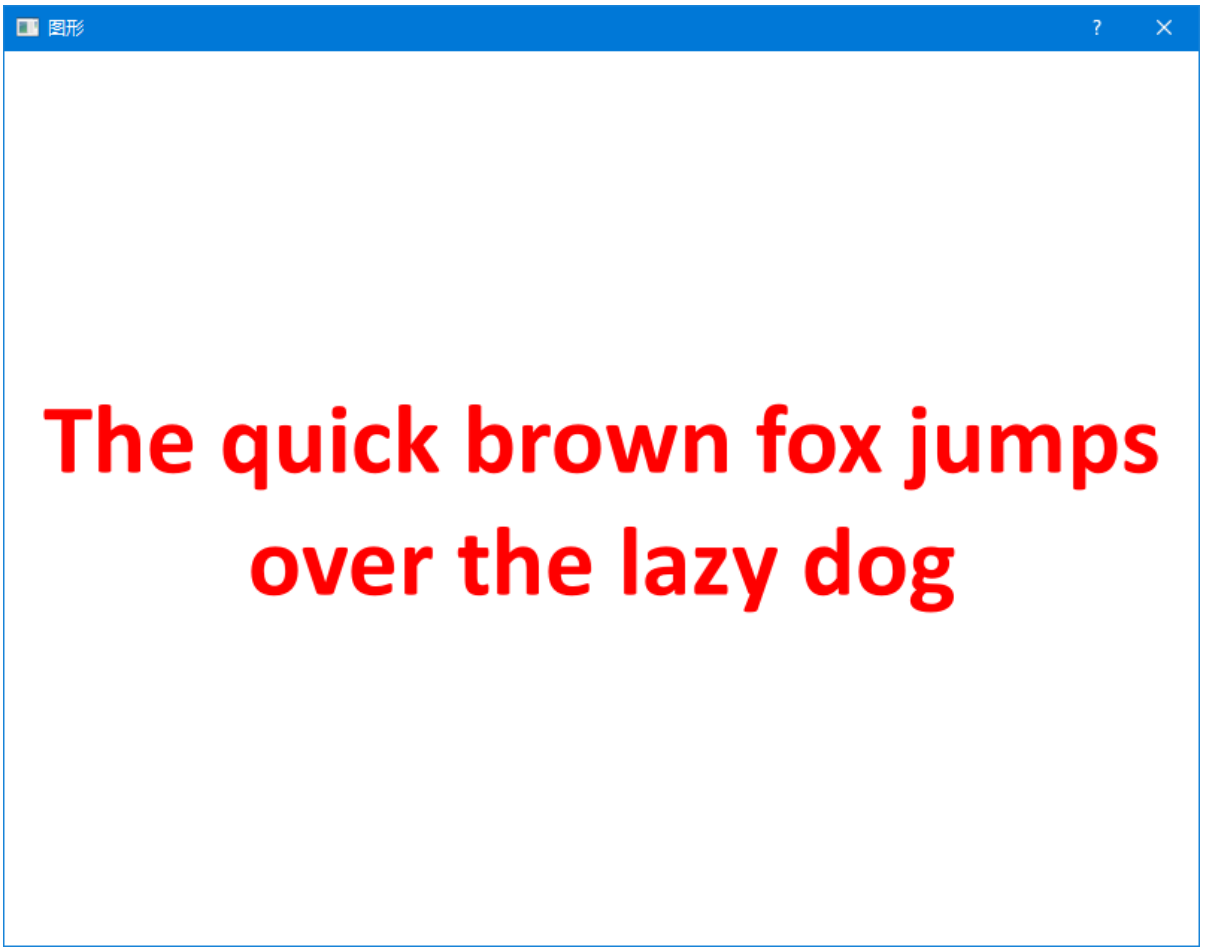

图形


图形


图形


图形


图形



26.3 复杂图形

26.3.1 QPainterPath

QPainter类除了提供上述绘制基本图形的方法外，还提供了一个名为drawPath的方法，用于绘制一些相对复杂的图形：

```
1 void QPainter::drawPath(const QPainterPath& path); // 绘制复杂图形
```

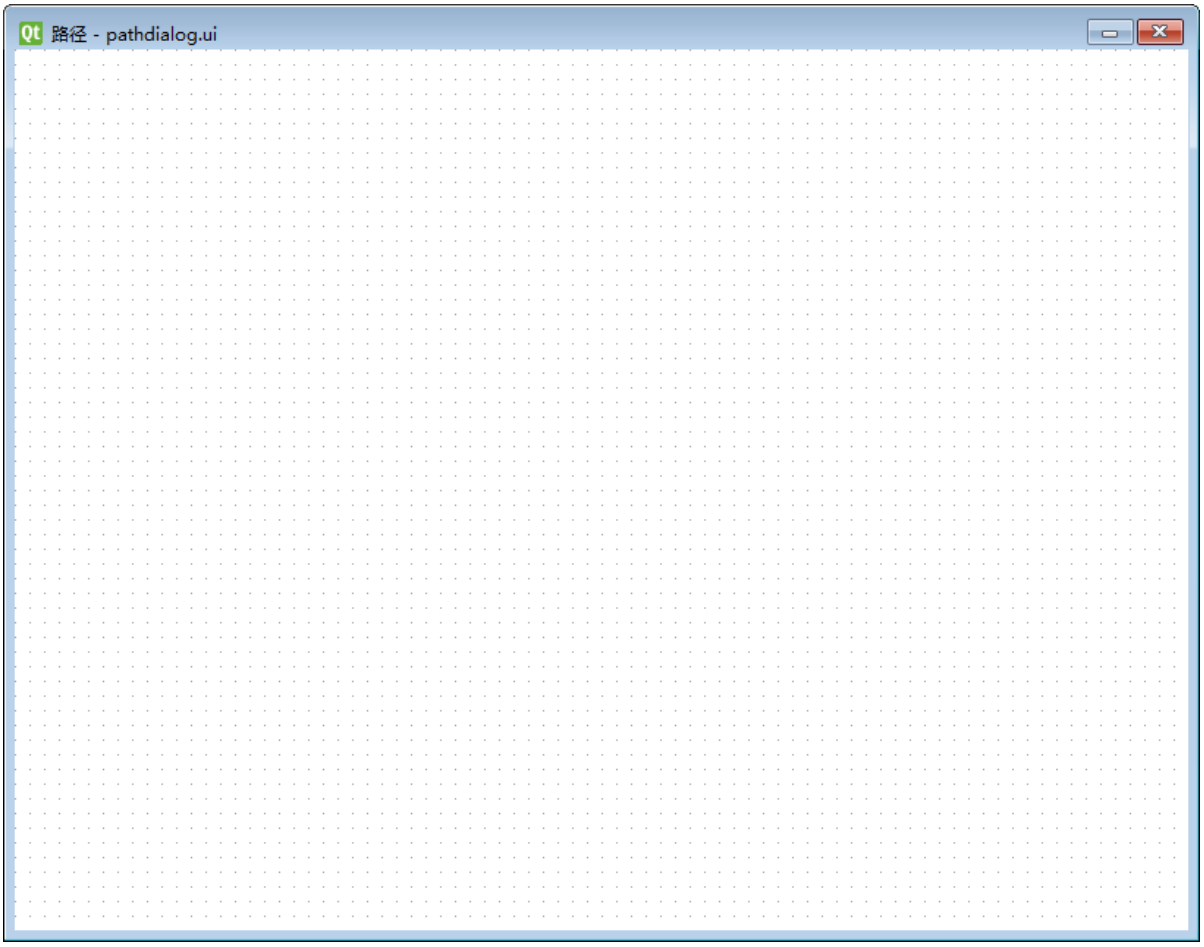
该方法的参数是一个对QPainterPath类型对象的引用。QPainterPath的妙处在于，它可以记录一系列参与复杂图形绘制的绘制动作。当将其作为参数交给QPainter类的drawPath方法时，这些绘制动作将被依次执行，最终完成复杂图形的绘制。此外，记录了绘制过程的QPainterPath对象还可以重复使用，比如在不同的位置，以不同的缩放比例和旋转角度，甚至以不同的扭曲形式，绘制相同的图形。

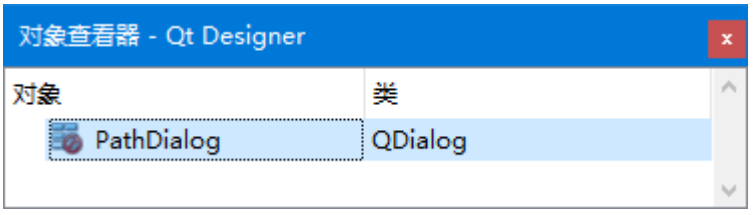
26.3.2 案例

26.3.2.1 创建项目

通过QtCreator，在C:\Users\Minwei\Projects\Qt路径下，创建名为Path的项目。

26.3.2.2 设计界面





C:\Users\Minwei\Projects\Qt\Path\pathdialog.ui：

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>PathDialog</class>
 <widget class="QDialog" name="PathDialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>800</width>
    <height>600</height>
   </rect>
  </property>
  <property name="palette">
   <palette>
    <active>
     <colorrole role="Base">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
        <red>255</red>
        <green>255</green>
```

```
          <blue>255</blue>
         </color>
        </brush>
       </colorrole>
       <colorrole role="Window">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>255</red>
          <green>255</green>
          <blue>255</blue>
         </color>
        </brush>
       </colorrole>
      </active>
      <inactive>
       <colorrole role="Base">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>255</red>
          <green>255</green>
          <blue>255</blue>
         </color>
        </brush>
       </colorrole>
       <colorrole role="Window">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>255</red>
          <green>255</green>
          <blue>255</blue>
         </color>
        </brush>
       </colorrole>
      </inactive>
      <disabled>
       <colorrole role="Base">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>255</red>
          <green>255</green>
          <blue>255</blue>
         </color>
        </brush>
       </colorrole>
       <colorrole role="Window">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>255</red>
          <green>255</green>
          <blue>255</blue>
         </color>
        </brush>
       </colorrole>
      </disabled>
     </palette>
    </property>
```

```
  <property name="windowTitle">
   <string>路径</string>
  </property>
 </widget>
 <resources/>
 <connections/>
</ui>
```

26.3.2.3 实现功能

C:\Users\Minwei\Projects\Qt\Path\pathdialog.h：

```
#ifndef PATHDIALOG_H
#define PATHDIALOG_H

#include <QDialog>

QT_BEGIN_NAMESPACE
namespace Ui { class PathDialog; }
QT_END_NAMESPACE

class PathDialog : public QDialog
{
    Q_OBJECT

public:
    PathDialog(QWidget *parent = nullptr);
    ~PathDialog();

protected:
    void paintEvent(QPaintEvent*);

private:
    Ui::PathDialog *ui;
};

#endif // PATHDIALOG_H
```

C:\Users\Minwei\Projects\Qt\Path\pathdialog.cpp：

```
#include <QPainter>

#include "pathdialog.h"
#include "ui_pathdialog.h"

PathDialog::PathDialog(QWidget *parent)
    : QDialog(parent)
    , ui(new Ui::PathDialog)
{
    ui->setupUi(this);
}

PathDialog::~PathDialog()
{
    delete ui;
```

```
}

void PathDialog::paintEvent(QPaintEvent*)
{
    QPainter painter(this);
    painter.setRenderHint(QPainter::Antialiasing);

    int w = width(), h = height();

    QPen pen;
    pen.setColor(Qt::red);
    pen.setWidth(3);
    pen.setStyle(Qt::SolidLine);
    pen.setJoinStyle(Qt::MiterJoin);
    painter.setPen(pen);
    QBrush brush;
    brush.setColor(Qt::yellow);
    brush.setStyle(Qt::SolidPattern);
    painter.setBrush(brush);

    QPainterPath path;
    path.moveTo(-100, 0);
    path.quadTo(QPoint(-50, -100),QPoint(0, 0));
    path.quadTo(QPoint(50, 100),QPoint(100, 0));
    path.closeSubpath();

    painter.save();
    painter.translate(w/3, h/3);
    painter.drawPath(path);
    painter.restore();

    painter.save();
    painter.translate(w*2/3, h/3);
    painter.scale(0.5,0.5);
    painter.drawPath(path);
    painter.restore();

    painter.save();
    painter.translate(w/3, h*2/3);
    painter.rotate(90);
    painter.drawPath(path);
    painter.restore();

    painter.save();
    painter.translate(w*2/3, h*2/3);
    painter.shear(-1, 0);
    painter.drawPath(path);
    painter.restore();
}
```

26.3.2.4 测试验证

运行效果如图所示：