

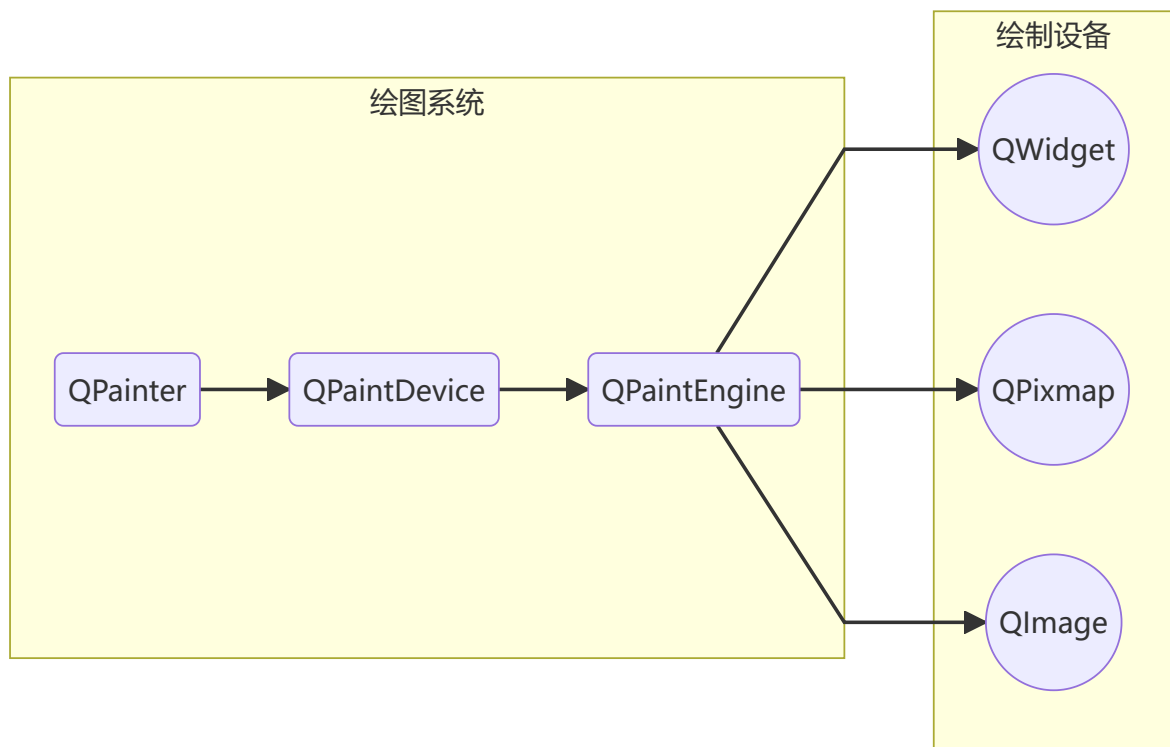
26 QPainter基本绘图

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

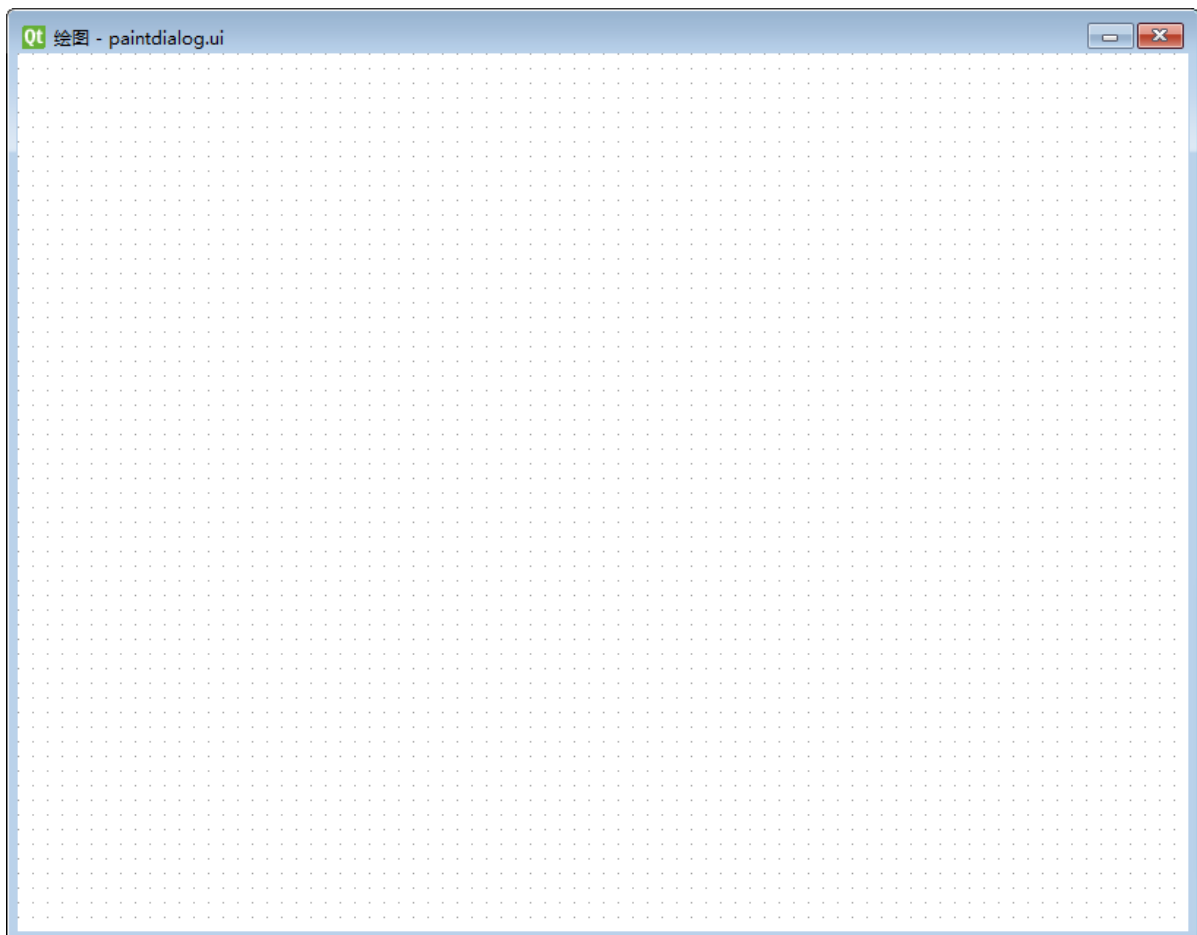
通过QtCreator，在C:\Users\Minwei\Projects\Qt路径下，创建名为Paint的项目。

26.1.4.2 添加资源

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

```
1 <RCC>
2   <qresource prefix="/">
3     <file>images/1.png</file>
4   </qresource>
5 </RCC>
```

26.1.4.3 设计界面





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

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <ui version="4.0">
3 <class>PaintDialog</class>
4 <widget class="QDialog" name="PaintDialog">
5 <property name="geometry">
6 <rect>
7 <x>0</x>
8 <y>0</y>
9 <width>800</width>
10 <height>600</height>
11 </rect>
12 </property>
13 <property name="palette">
14 <palette>
15 <active>
16 <colorrole role="Base">
17 <brush brushstyle="SolidPattern">
18 <color alpha="255">
19 <red>255</red>
20 <green>255</green>
21 <blue>255</blue>
22 </color>
23 </brush>
24 </colorrole>
25 <colorrole role="window">
26 <brush brushstyle="SolidPattern">
27 <color alpha="255">
28 <red>255</red>
29 <green>255</green>
30 <blue>255</blue>
31 </color>
32 </brush>
33 </colorrole>
34 </active>
35 <inactive>
36 <colorrole role="Base">
37 <brush brushstyle="SolidPattern">
38 <color alpha="255">
39 <red>255</red>
40 <green>255</green>
41 <blue>255</blue>
42 </color>
43 </brush>
44 </colorrole>
45 <colorrole role="window">
46 <brush brushstyle="SolidPattern">
47 <color alpha="255">
```

```

48     <red>255</red>
49     <green>255</green>
50     <blue>255</blue>
51     </color>
52     </brush>
53     </colorrole>
54 </inactive>
55 <disabled>
56     <colorrole role="Base">
57     <brush brushstyle="SolidPattern">
58     <color alpha="255">
59     <red>255</red>
60     <green>255</green>
61     <blue>255</blue>
62     </color>
63     </brush>
64     </colorrole>
65     <colorrole role="window">
66     <brush brushstyle="SolidPattern">
67     <color alpha="255">
68     <red>255</red>
69     <green>255</green>
70     <blue>255</blue>
71     </color>
72     </brush>
73     </colorrole>
74 </disabled>
75 </palette>
76 </property>
77 <property name="windowTitle">
78     <string>绘图</string>
79 </property>
80 </widget>
81 <resources/>
82 <connections/>
83 </ui>

```

26.1.4.4 实现功能

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

```

1  #ifndef PAINTDIALOG_H
2  #define PAINTDIALOG_H
3
4  #include <QDialog>
5
6  QT_BEGIN_NAMESPACE
7  namespace Ui { class PaintDialog; }
8  QT_END_NAMESPACE
9
10 class PaintDialog : public QDialog
11 {
12     Q_OBJECT
13
14 public:

```

```

15     PaintDialog(QWidget *parent = nullptr);
16     ~PaintDialog();
17
18 protected:
19     void paintEvent(QPaintEvent*);
20
21 private:
22     Ui::PaintDialog *ui;
23 };
24
25 #endif // PAINTDIALOG_H

```

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

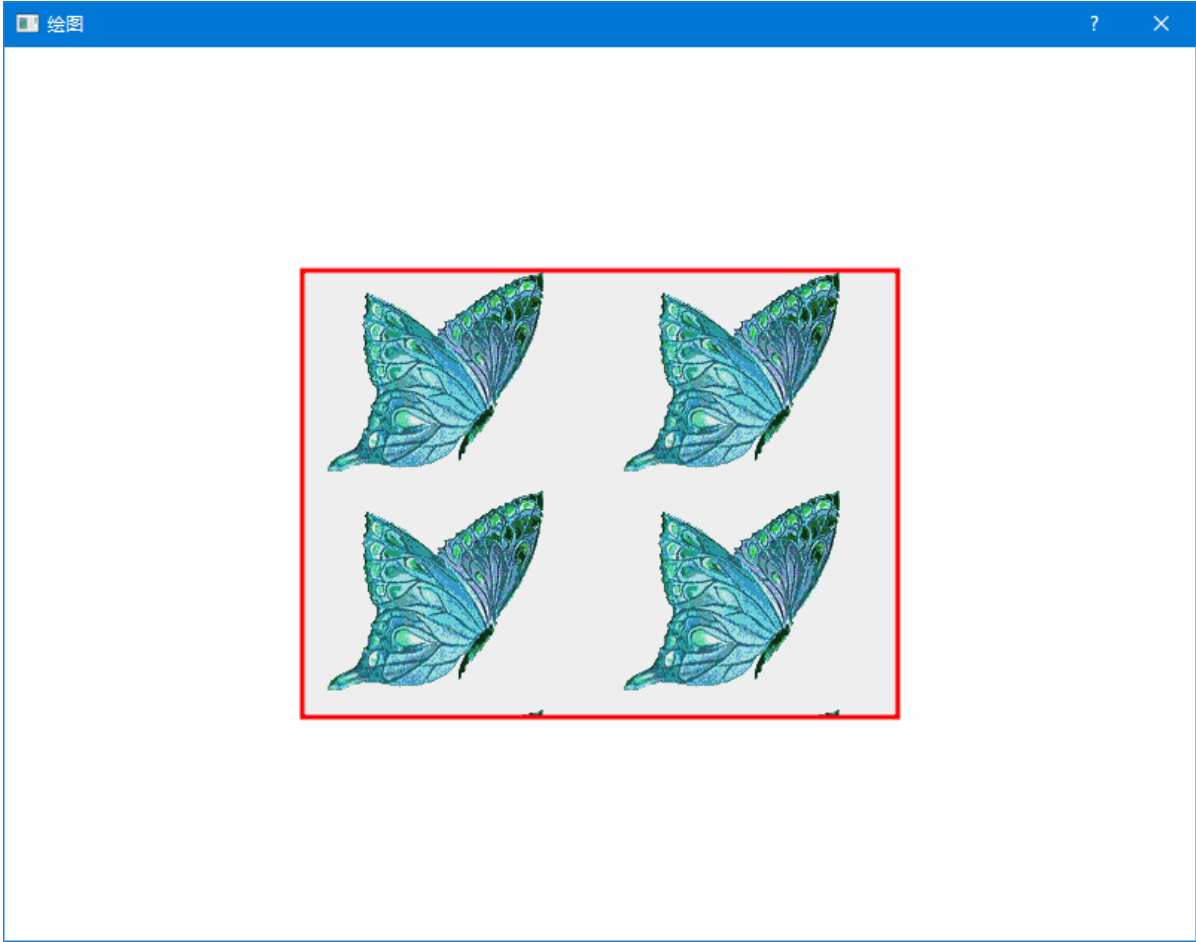
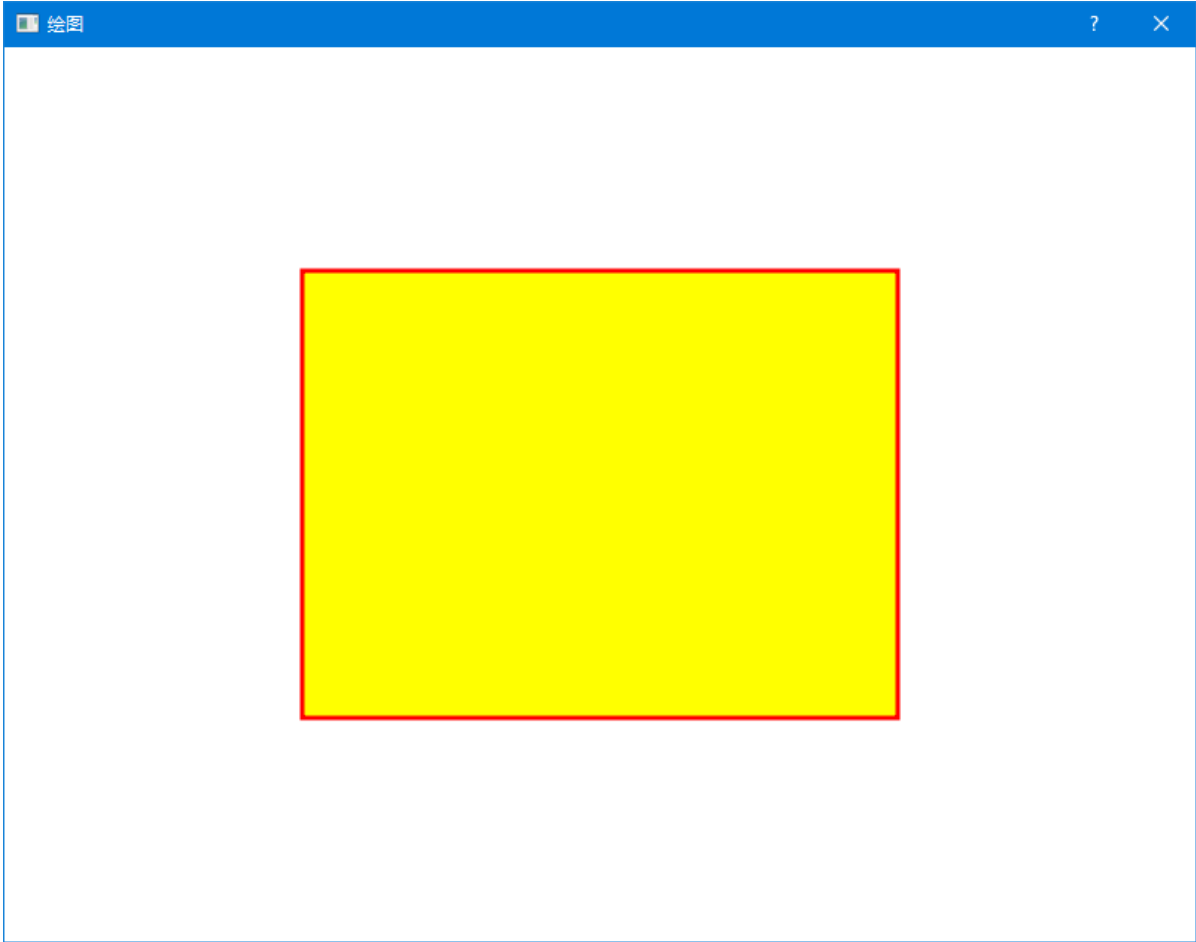
```

1  #include <QPainter>
2
3  #include "paintdialog.h"
4  #include "ui_paintdialog.h"
5
6  PaintDialog::PaintDialog(QWidget *parent)
7      : QDialog(parent)
8      , ui(new Ui::PaintDialog)
9  {
10     ui->setupUi(this);
11 }
12
13 PaintDialog::~PaintDialog()
14 {
15     delete ui;
16 }
17
18 void PaintDialog::paintEvent(QPaintEvent*)
19 {
20     QPainter painter(this);
21     painter.setRenderHint(QPainter::Antialiasing);
22
23     int w = width(), h = height();
24
25     QPen pen;
26     pen.setColor(Qt::red);
27     pen.setWidth(3);
28     pen.setStyle(Qt::SolidLine);
29     pen.setJoinStyle(Qt::MiterJoin);
30     painter.setPen(pen);
31     QBrush brush;
32     brush.setColor(Qt::yellow);
33     brush.setStyle(Qt::SolidPattern);
34     painter.setBrush(brush);
35
36     painter.drawRect(QRect(QPoint(w/4, h/4), QPoint(w*3/4, h*3/4)));
37 }

```

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

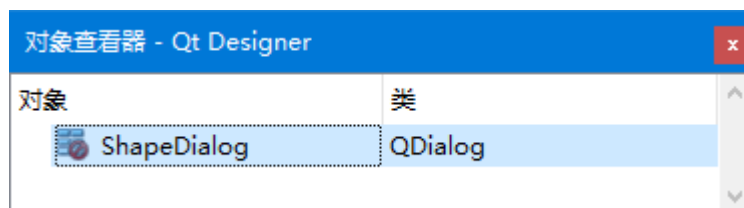
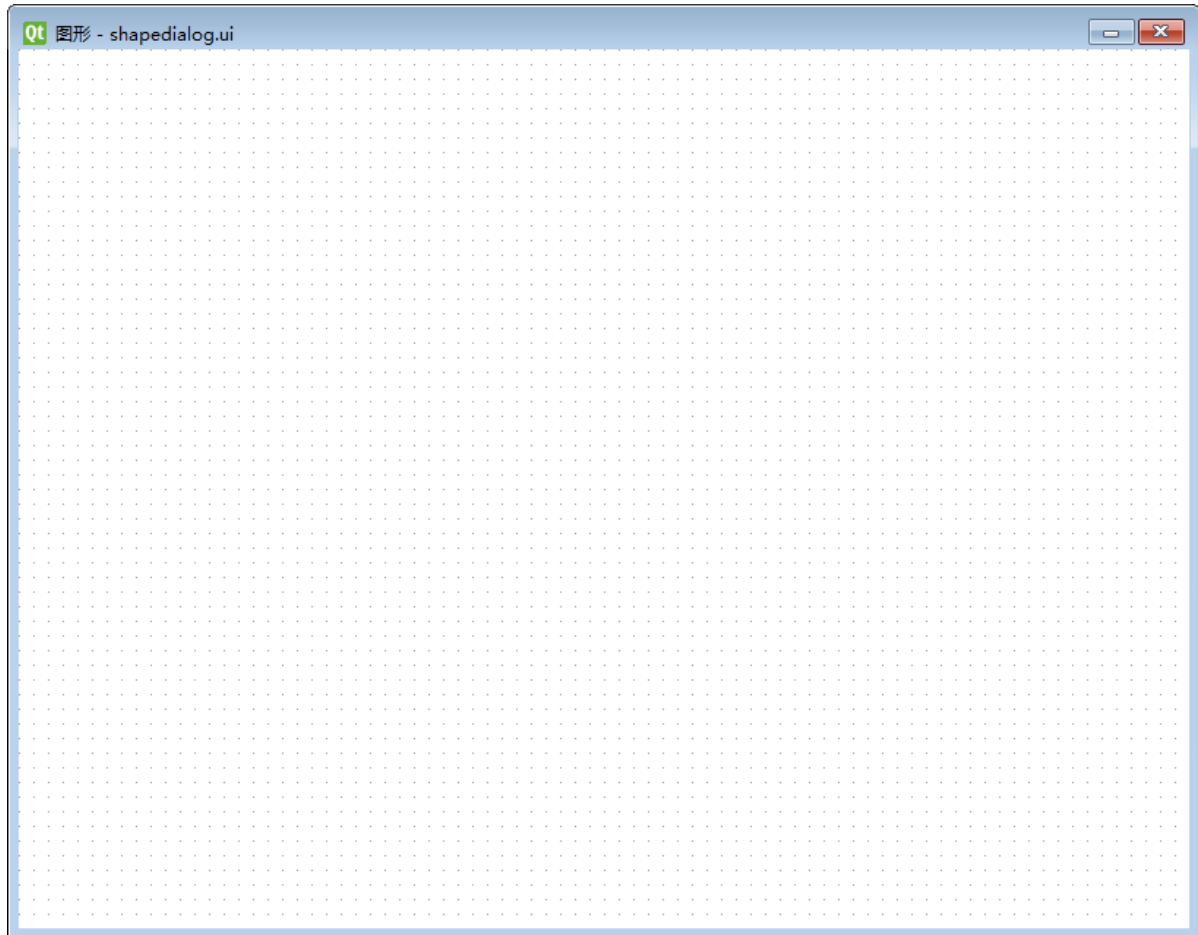
```
1 void QPainter::drawPoint(const QPoint& position);  
    // 绘制点  
2 void QPainter::drawLine(const QLine& line);  
    // 绘制直线  
3 void QPainter::drawRect(const QRect& rectangle);  
    // 绘制矩形  
4 void QPainter::drawEllipse(const QRect& rectangle);  
    // 绘制椭圆  
5 void QPainter::drawArc(const 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:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <ui version="4.0">
3   <class>ShapeDialog</class>
4   <widget class="QDialog" name="ShapeDialog">
5     <property name="geometry">
6       <rect>
7         <x>0</x>
8         <y>0</y>
9         <width>800</width>
10        <height>600</height>
11      </rect>
12    </property>
13    <property name="palette">
14      <palette>
15        <active>
16          <colorrole role="Base">
17            <brush brushstyle="SolidPattern">
18              <color alpha="255">
19                <red>255</red>
20                <green>255</green>
```



```
21     <blue>255</blue>
22     </color>
23     </brush>
24 </colorrole>
25 <colorrole role="window">
26     <brush brushstyle="SolidPattern">
27     <color alpha="255">
28     <red>255</red>
29     <green>255</green>
30     <blue>255</blue>
31     </color>
32     </brush>
33 </colorrole>
34 </active>
35 <inactive>
36     <colorrole role="Base">
37     <brush brushstyle="SolidPattern">
38     <color alpha="255">
39     <red>255</red>
40     <green>255</green>
41     <blue>255</blue>
42     </color>
43     </brush>
44 </colorrole>
45 <colorrole role="window">
46     <brush brushstyle="SolidPattern">
47     <color alpha="255">
48     <red>255</red>
49     <green>255</green>
50     <blue>255</blue>
51     </color>
52     </brush>
53 </colorrole>
54 </inactive>
55 <disabled>
56     <colorrole role="Base">
57     <brush brushstyle="SolidPattern">
58     <color alpha="255">
59     <red>255</red>
60     <green>255</green>
61     <blue>255</blue>
62     </color>
63     </brush>
64 </colorrole>
65 <colorrole role="window">
66     <brush brushstyle="SolidPattern">
67     <color alpha="255">
68     <red>255</red>
69     <green>255</green>
70     <blue>255</blue>
71     </color>
72     </brush>
73 </colorrole>
74 </disabled>
75 </palette>
76 </property>
```

```

77     <property name="windowTitle">
78         <string>图形</string>
79     </property>
80 </widget>
81 <resources/>
82 <connections/>
83 </ui>

```

26.2.2.3 实现功能

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

```

1  #ifndef SHAPEDIALOG_H
2  #define SHAPEDIALOG_H
3
4  #include <QDialog>
5
6  QT_BEGIN_NAMESPACE
7  namespace Ui { class ShapeDialog; }
8  QT_END_NAMESPACE
9
10 class ShapeDialog : public QDialog
11 {
12     Q_OBJECT
13
14 public:
15     ShapeDialog(QWidget *parent = nullptr);
16     ~ShapeDialog();
17
18 protected:
19     void paintEvent(QPaintEvent*);
20
21 private:
22     void drawPoint(void);
23     void drawLine(void);
24     void drawRect(void);
25     void drawEllipse(void);
26     void drawArc(void);
27     void drawChord(void);
28     void drawPie(void);
29     void drawPolyline(void);
30     void drawPolygon(void);
31     void drawText(void);
32
33 private:
34     Ui::ShapeDialog *ui;
35 };
36
37 #endif // SHAPEDIALOG_H

```

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

```

1  #include <QPainter>
2
3  #include "shapedialog.h"

```

```

4  #include "ui_shapedialog.h"
5
6  ShapeDialog::ShapeDialog(QWidget *parent)
7      : QDialog(parent)
8      , ui(new Ui::ShapeDialog)
9  {
10     ui->setupUi(this);
11 }
12
13 ShapeDialog::~ShapeDialog()
14 {
15     delete ui;
16 }
17
18 void ShapeDialog::paintEvent(QPaintEvent*)
19 {
20     drawPoint();
21     //drawLine();
22     //drawRect();
23     //drawEllipse();
24     //drawArc();
25     //drawChord();
26     //drawPie();
27     //drawPolyline();
28     //drawPolygon();
29     //drawText();
30 }
31
32 void ShapeDialog::drawPoint(void)
33 {
34     QPainter painter(this);
35     painter.setRenderHint(QPainter::Antialiasing);
36
37     int w = width(), h = height();
38     int xo = (w - 256) / 2, yo = (h - 256) / 2;
39
40     for (int x = 0; x < 256; ++x)
41         for (int y = 0; y < 256; ++y)
42             {
43                 QPen pen;
44                 pen.setColor(QColor(x, y, 0));
45                 painter.setPen(pen);
46                 painter.drawPoint(QPoint(xo + x, yo + y));
47             }
48 }
49
50 void ShapeDialog::drawLine(void)
51 {
52     QPainter painter(this);
53     painter.setRenderHint(QPainter::Antialiasing);
54
55     int w = width(), h = height();
56
57     QPen pen;
58     pen.setColor(Qt::red);
59     pen.setWidth(3);

```

```

60     pen.setStyle(Qt::SolidLine);
61     pen.setJoinStyle(Qt::MiterJoin);
62     painter.setPen(pen);
63
64     painter.drawLine(QLine(QPoint(0, 0), QPoint(w, h)));
65     painter.drawLine(QLine(QPoint(w, 0), QPoint(0, h)));
66 }
67
68 void ShapeDialog::drawRect(void)
69 {
70     QPainter painter(this);
71     painter.setRenderHint(QPainter::Antialiasing);
72
73     int w = width(), h = height();
74
75     QPen pen;
76     pen.setColor(Qt::red);
77     pen.setWidth(3);
78     pen.setStyle(Qt::SolidLine);
79     pen.setJoinStyle(Qt::MiterJoin);
80     painter.setPen(pen);
81     QBrush brush;
82     brush.setColor(Qt::yellow);
83     brush.setStyle(Qt::SolidPattern);
84     painter.setBrush(brush);
85
86     painter.drawRect(QRect(QPoint(w/4, h/4), QPoint(w*3/4, h*3/4)));
87 }
88
89 void ShapeDialog::drawEllipse(void)
90 {
91     QPainter painter(this);
92     painter.setRenderHint(QPainter::Antialiasing);
93
94     int w = width(), h = height();
95
96     QPen pen;
97     pen.setColor(Qt::red);
98     pen.setWidth(3);
99     pen.setStyle(Qt::SolidLine);
100    pen.setJoinStyle(Qt::MiterJoin);
101    painter.setPen(pen);
102    QBrush brush;
103    brush.setColor(Qt::yellow);
104    brush.setStyle(Qt::SolidPattern);
105    painter.setBrush(brush);
106
107    painter.drawEllipse(QRect(QPoint(w/4, h/4), QPoint(w*3/4, h*3/4)));
108 }
109
110 void ShapeDialog::drawArc(void)
111 {
112     QPainter painter(this);
113     painter.setRenderHint(QPainter::Antialiasing);
114
115     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

```

```

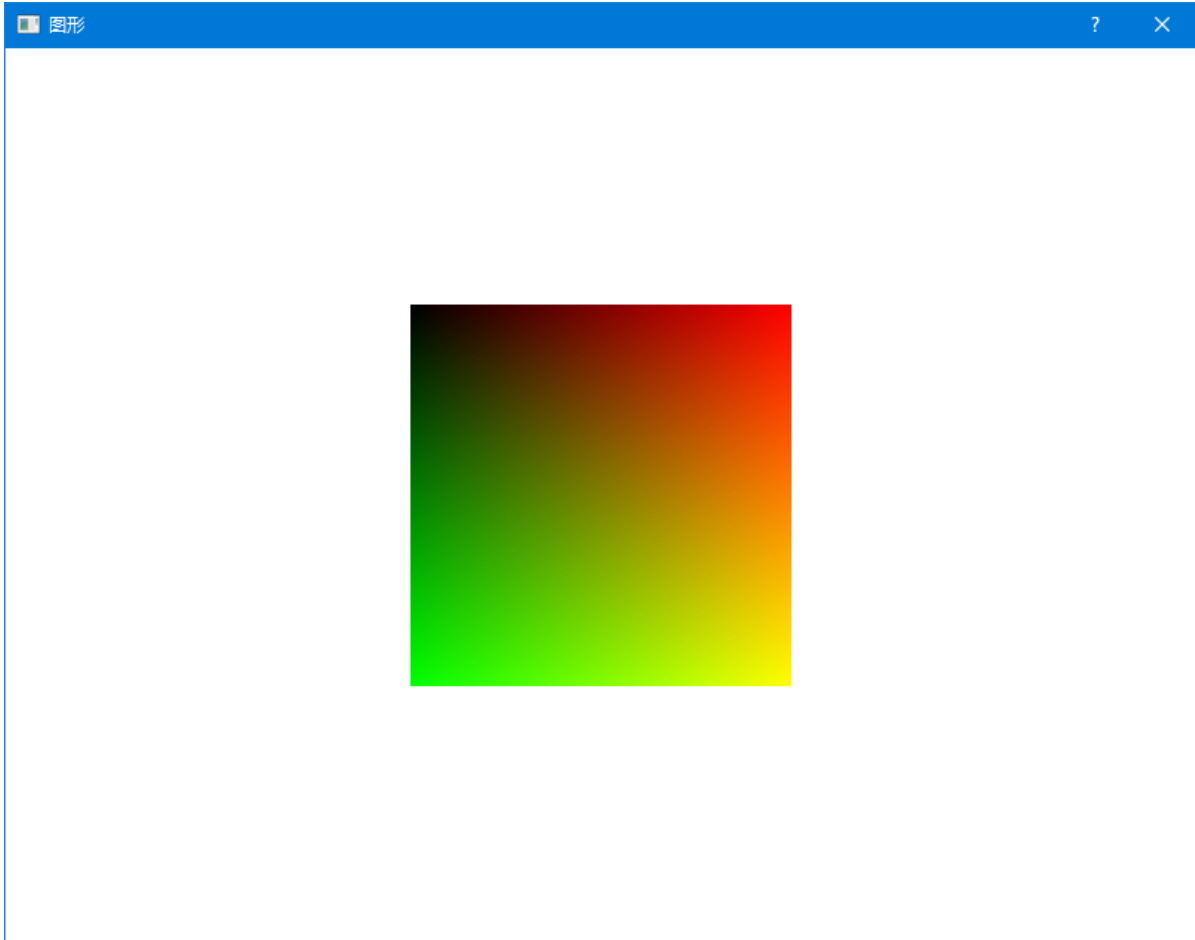
172 void ShapeDialog::drawPolyline(void)
173 {
174     QPainter painter(this);
175     painter.setRenderHint(QPainter::Antialiasing);
176
177     int w = width(), h = height();
178
179     QPen pen;
180     pen.setColor(Qt::red);
181     pen.setWidth(3);
182     pen.setStyle(Qt::SolidLine);
183     pen.setJoinStyle(Qt::MiterJoin);
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

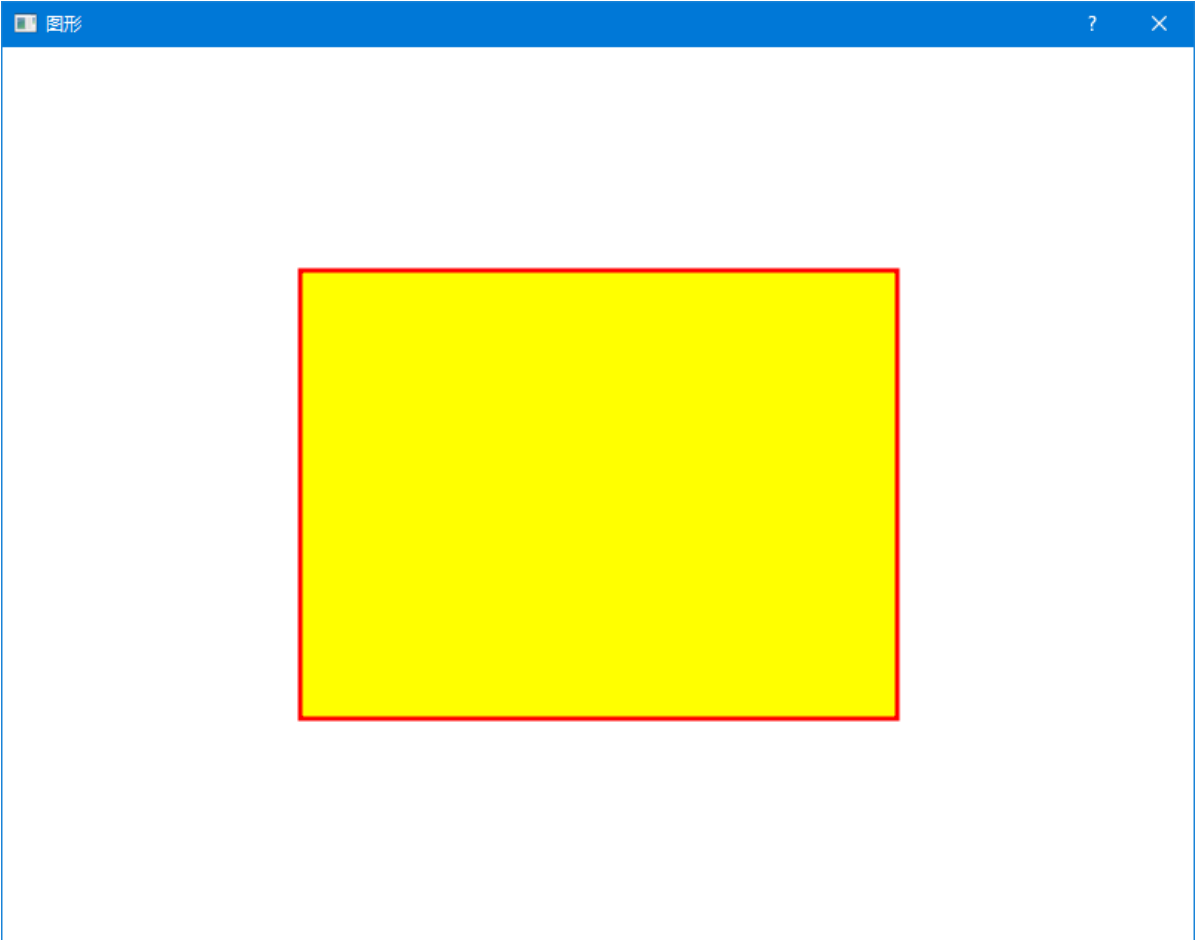
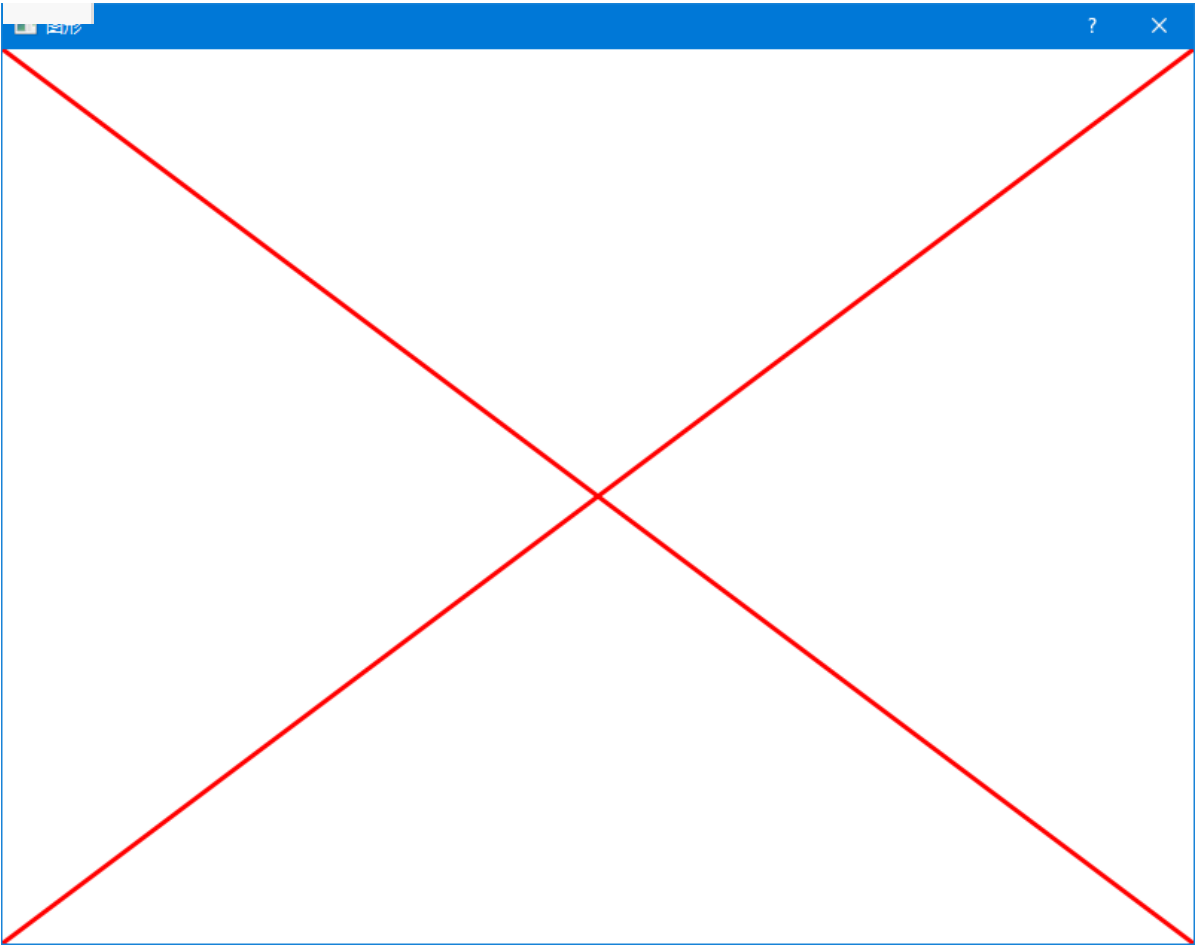
```

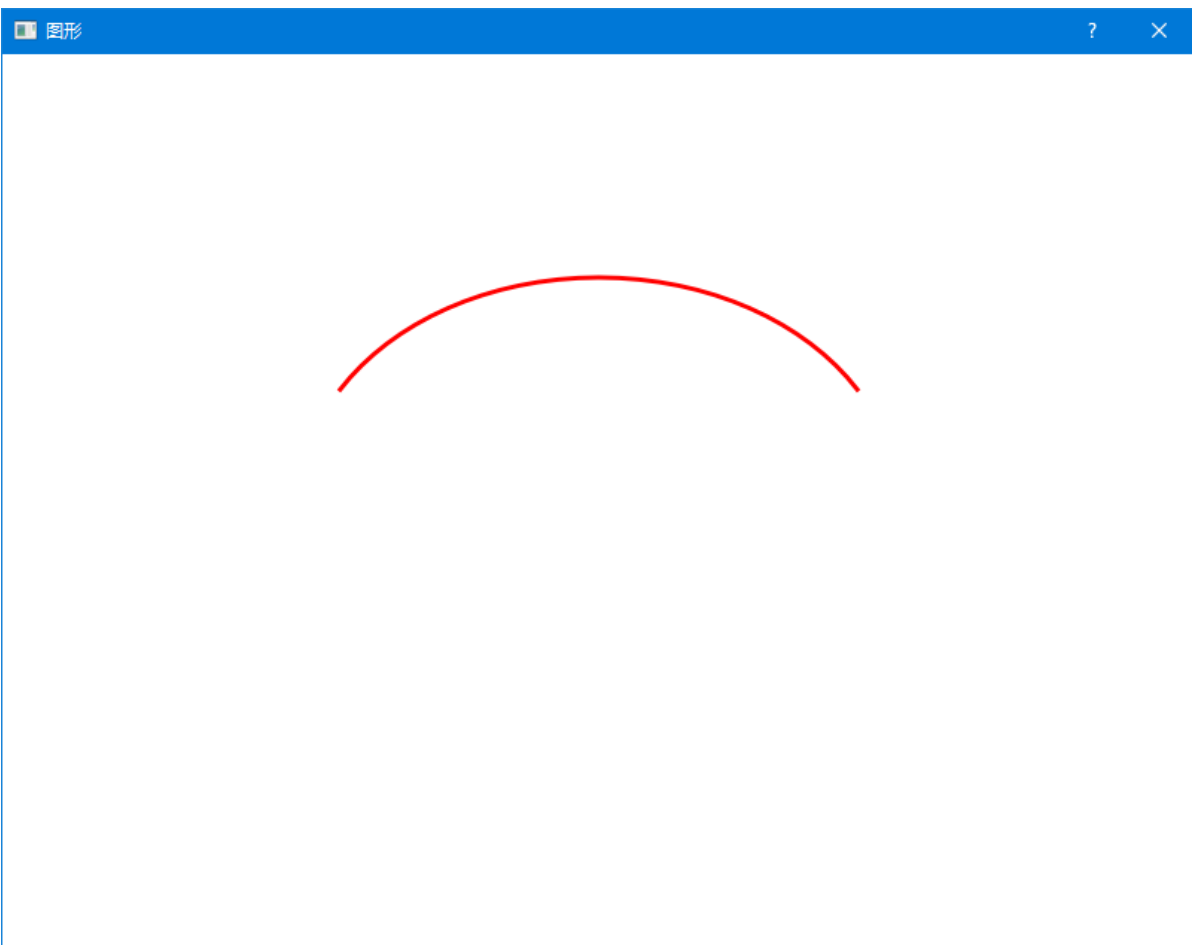
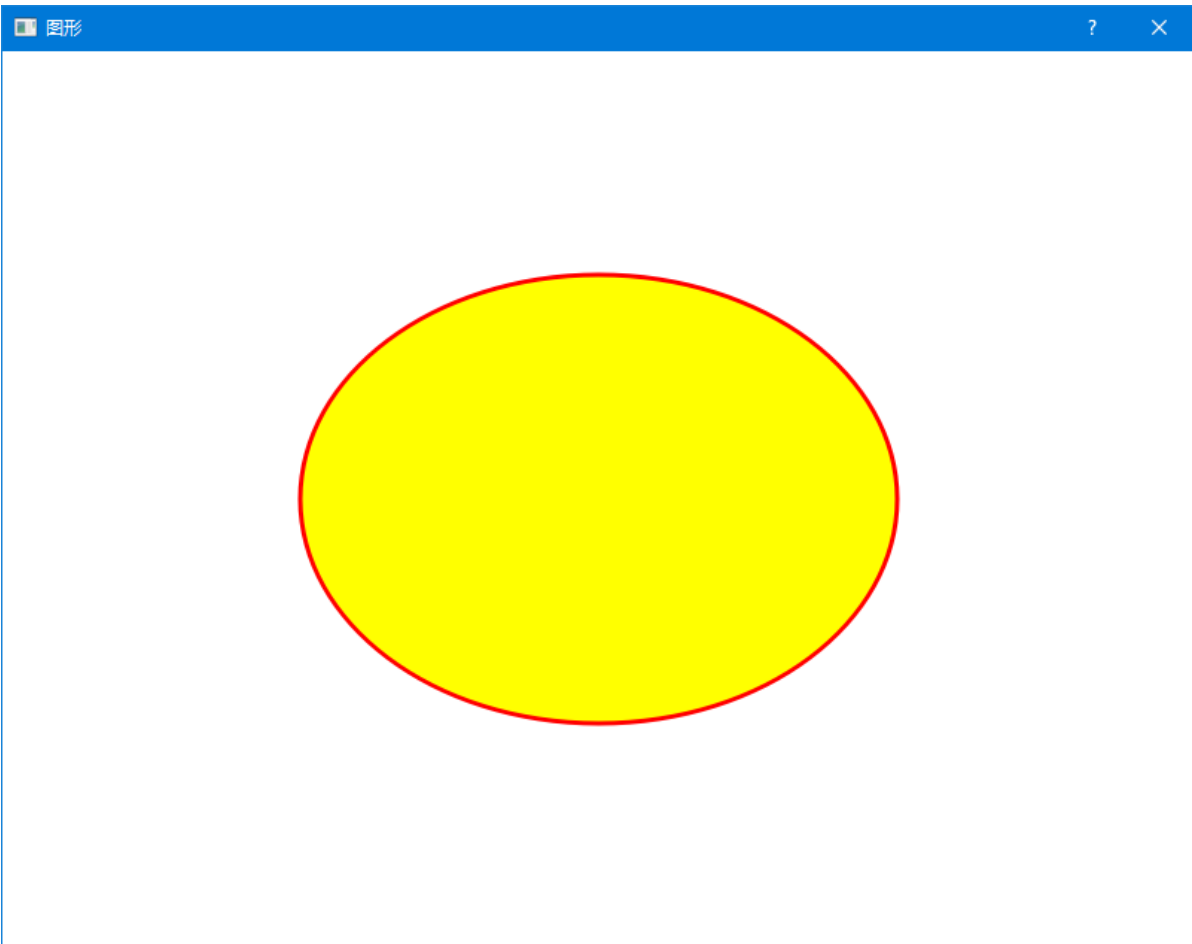
```
228 painter.drawText(QRect(QPoint(0, 0), QPoint(w, h)),
229                 Qt::AlignCenter | Qt::TextWordWrap,
230                 "The quick brown fox jumps over the lazy dog");
231 }
```

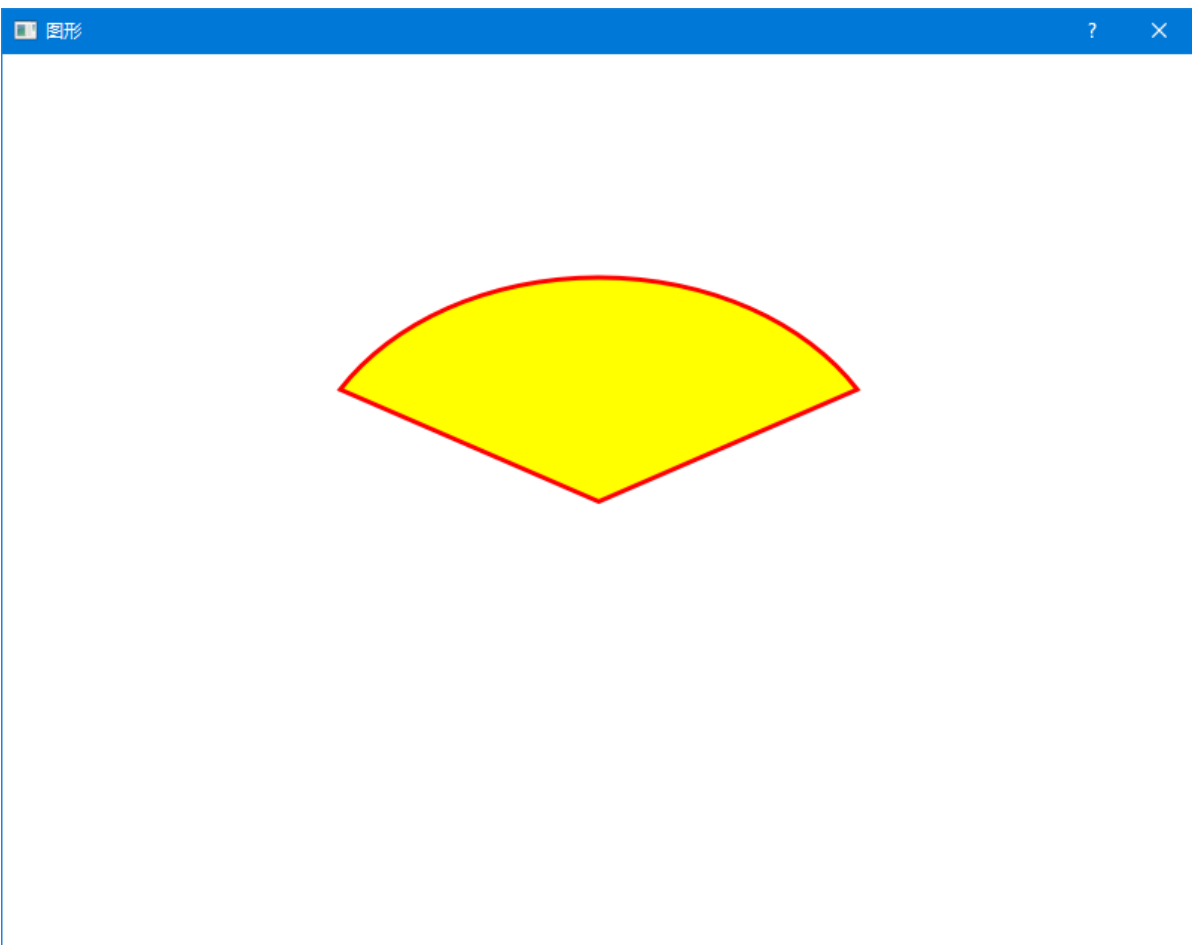
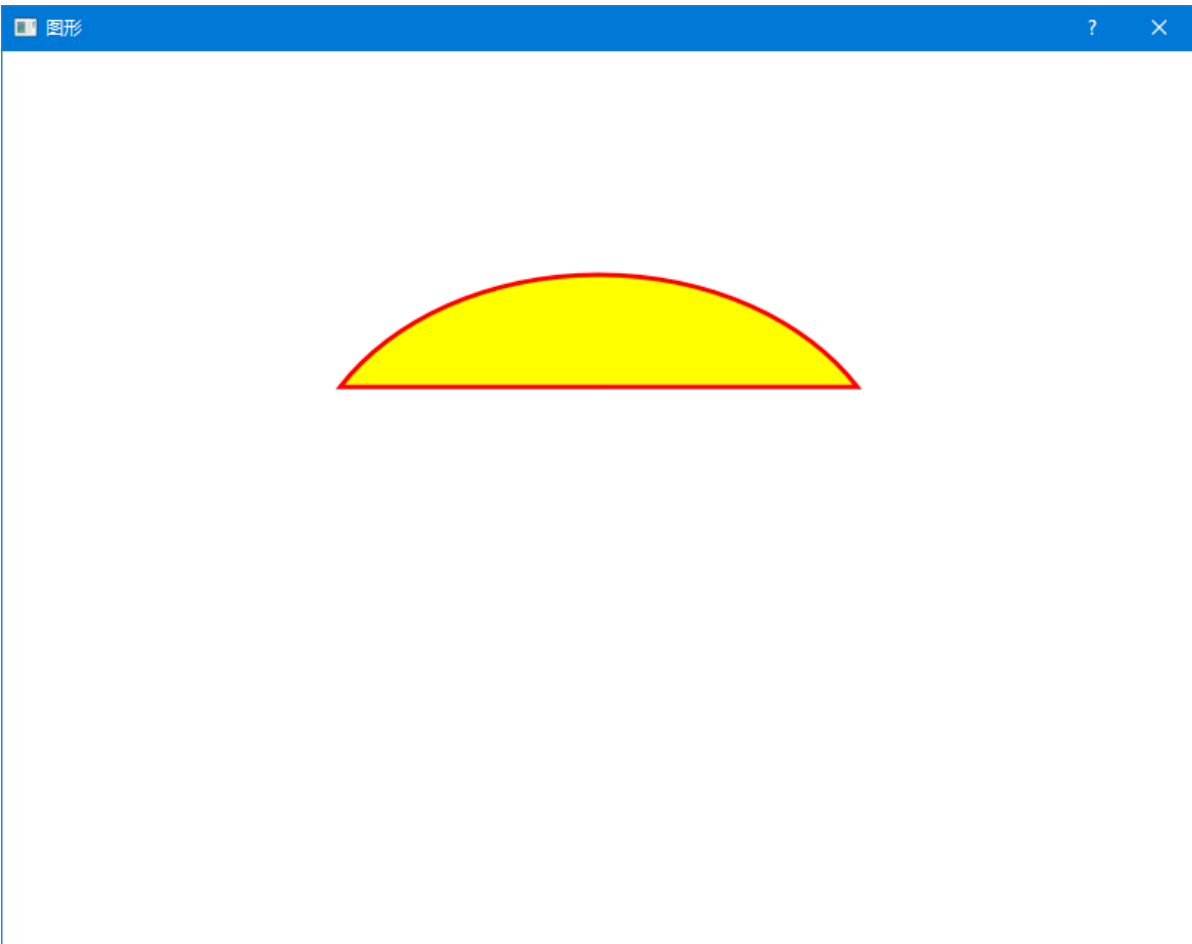
26.2.2.4 测试验证

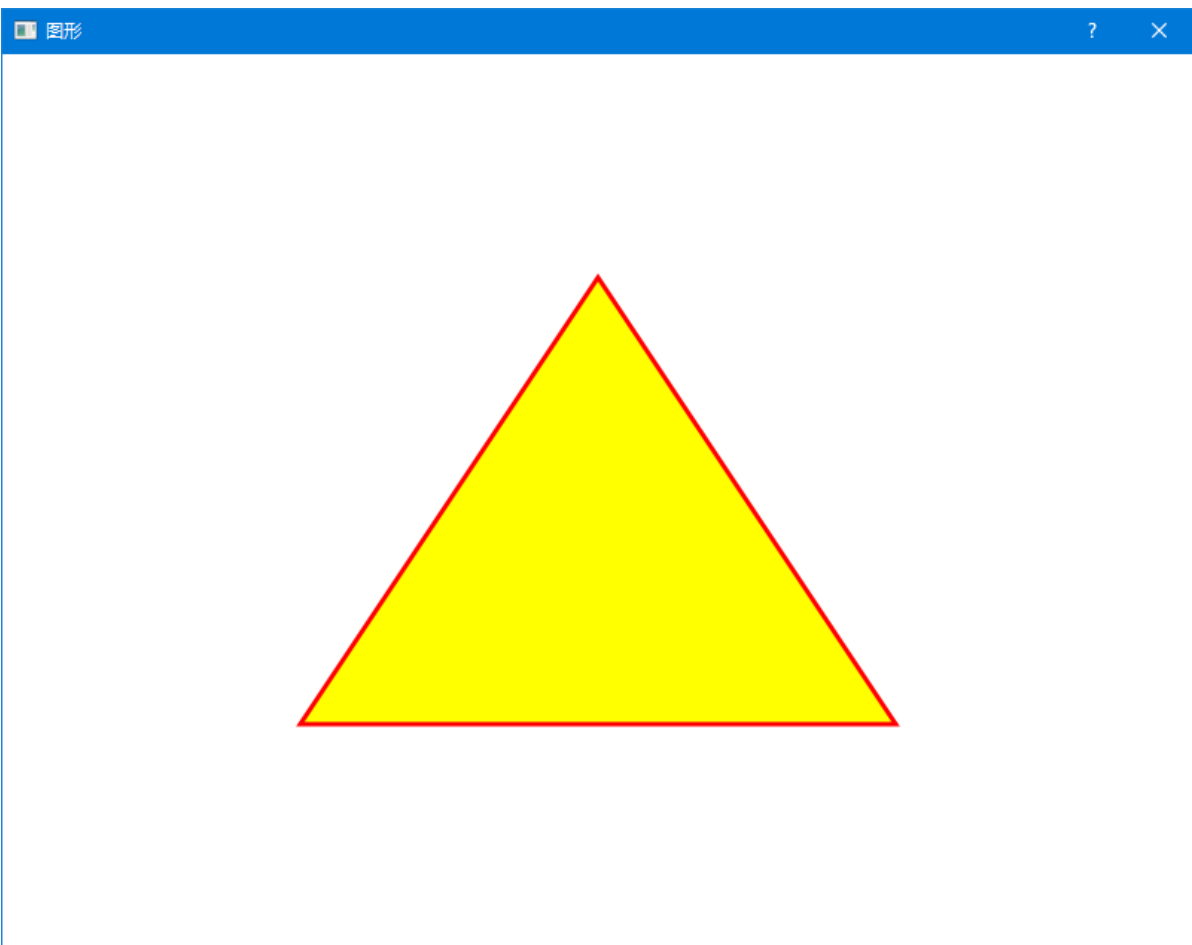
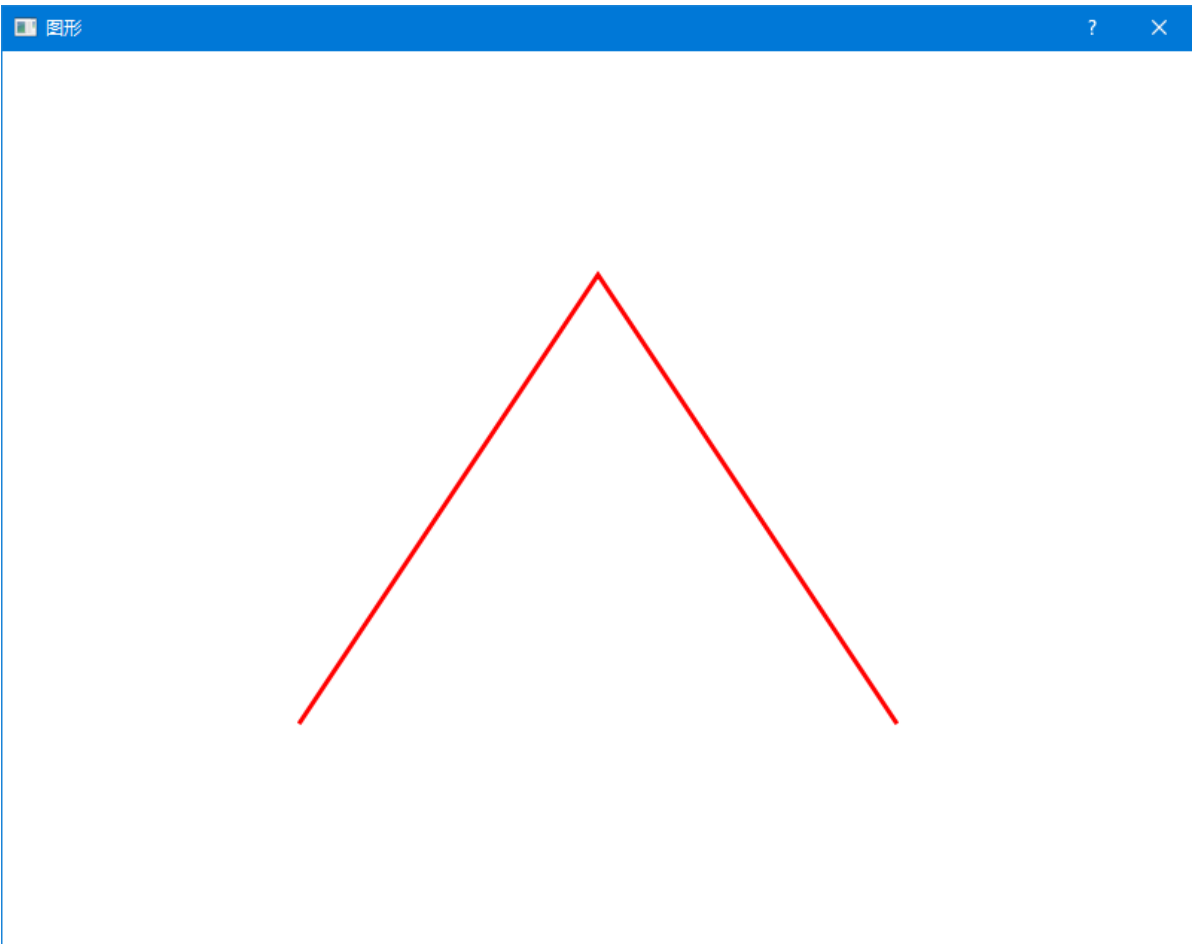
运行效果如图所示：

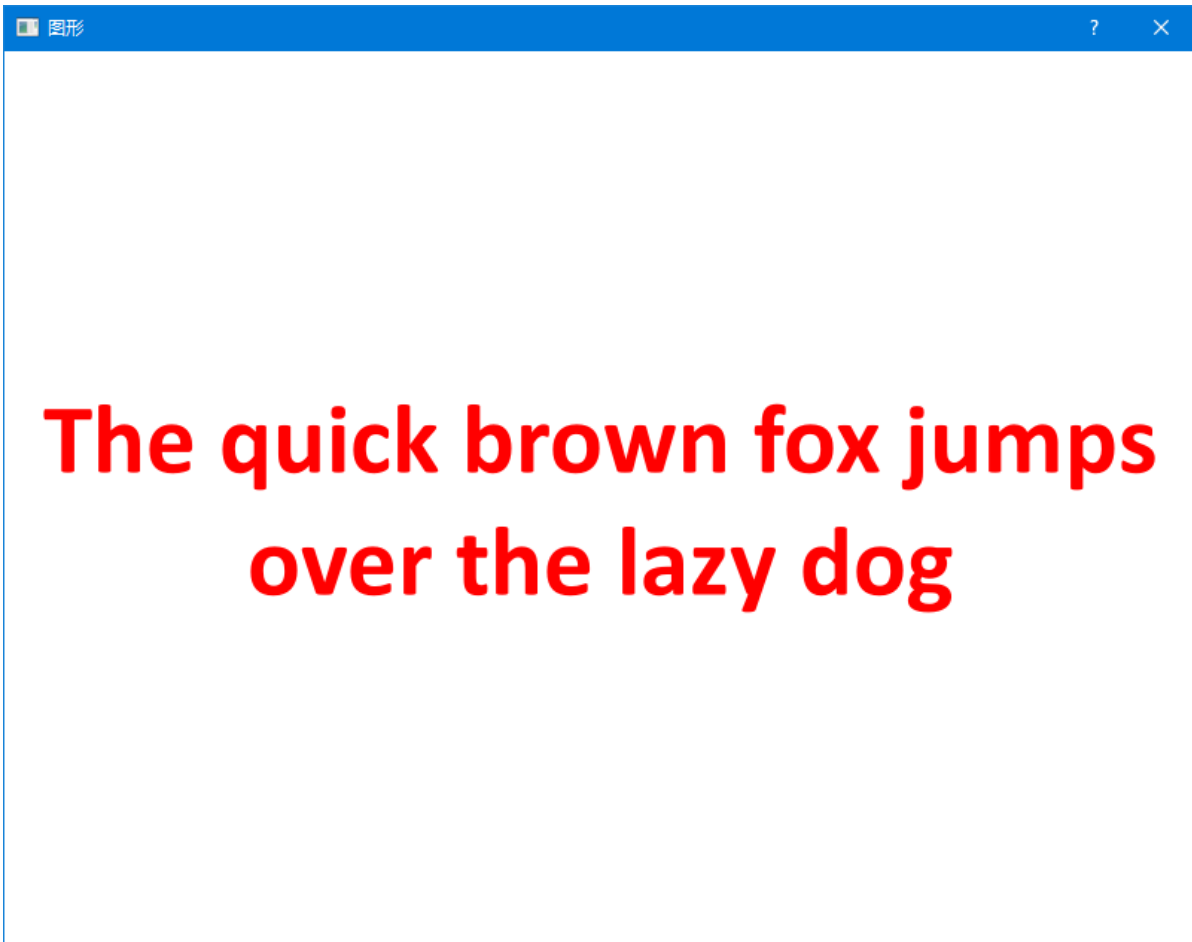












26.3 复杂图形

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
```

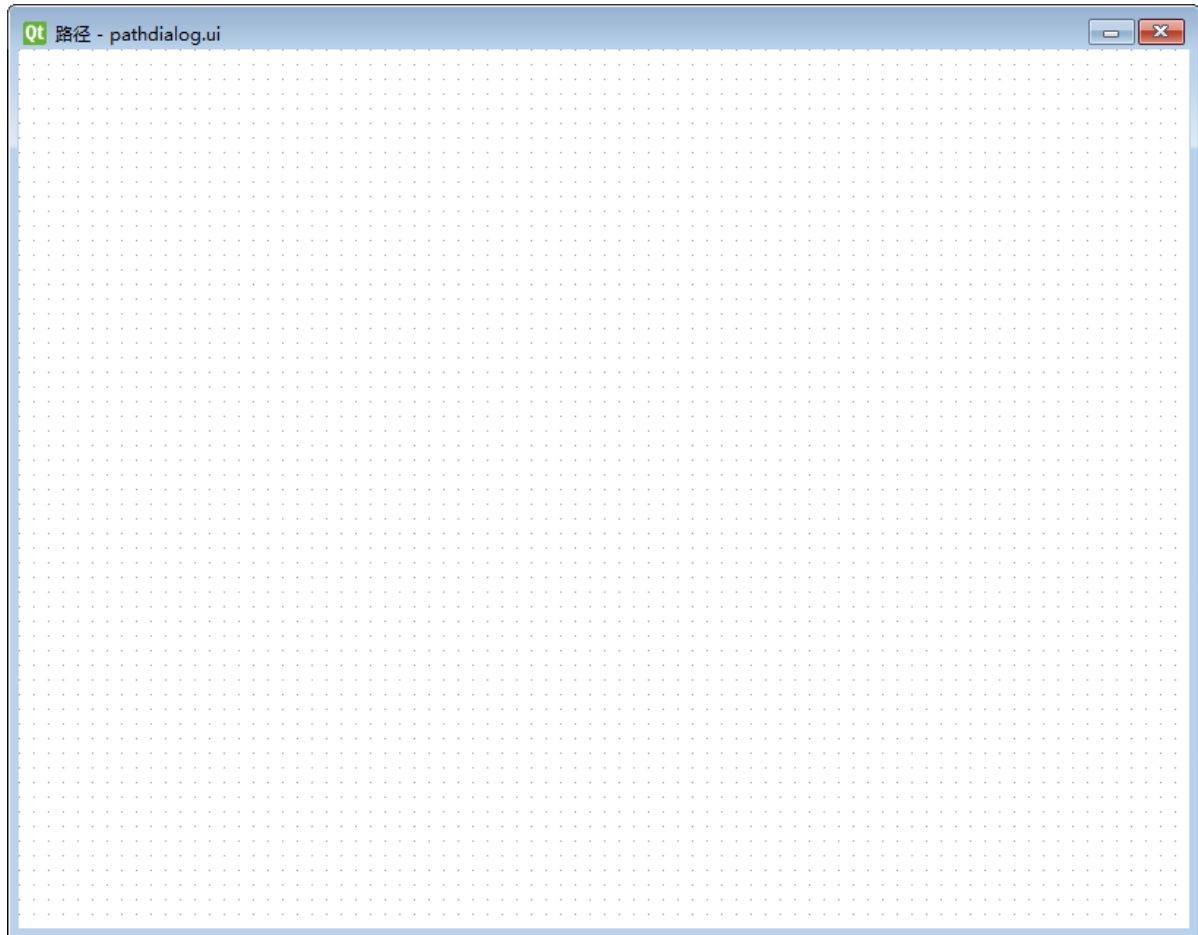
该方法的参数是一个对QPainterPath类型对象的引用。QPainterPath的妙处在于，它可以记录一系列参与复杂图形绘制的绘制动作。当将其作为参数交给QPainter类的drawPath方法时，这些绘制动作将被依次执行，最终完成复杂图形的绘制。此外，记录了绘制过程的QPainterPath对象还可以重复使用，比如在不同的位置，以不同的缩放比例和旋转角度，甚至以不同的扭曲形式，绘制相同的图形。

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:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <ui version="4.0">
3   <class>PathDialog</class>
4   <widget class="QDialog" name="PathDialog">
5     <property name="geometry">
6       <rect>
7         <x>0</x>
8         <y>0</y>
9         <width>800</width>
10        <height>600</height>
11      </rect>
12    </property>
13    <property name="palette">
14      <palette>
15        <active>
16          <colorrole role="Base">
17            <brush brushstyle="SolidPattern">
18              <color alpha="255">
19                <red>255</red>
20                <green>255</green>
```

```
21     <blue>255</blue>
22     </color>
23     </brush>
24 </colorrole>
25 <colorrole role="window">
26     <brush brushstyle="SolidPattern">
27     <color alpha="255">
28     <red>255</red>
29     <green>255</green>
30     <blue>255</blue>
31     </color>
32     </brush>
33 </colorrole>
34 </active>
35 <inactive>
36     <colorrole role="Base">
37     <brush brushstyle="SolidPattern">
38     <color alpha="255">
39     <red>255</red>
40     <green>255</green>
41     <blue>255</blue>
42     </color>
43     </brush>
44 </colorrole>
45 <colorrole role="window">
46     <brush brushstyle="SolidPattern">
47     <color alpha="255">
48     <red>255</red>
49     <green>255</green>
50     <blue>255</blue>
51     </color>
52     </brush>
53 </colorrole>
54 </inactive>
55 <disabled>
56     <colorrole role="Base">
57     <brush brushstyle="SolidPattern">
58     <color alpha="255">
59     <red>255</red>
60     <green>255</green>
61     <blue>255</blue>
62     </color>
63     </brush>
64 </colorrole>
65 <colorrole role="window">
66     <brush brushstyle="SolidPattern">
67     <color alpha="255">
68     <red>255</red>
69     <green>255</green>
70     <blue>255</blue>
71     </color>
72     </brush>
73 </colorrole>
74 </disabled>
75 </palette>
76 </property>
```

```

77     <property name="windowTitle">
78         <string>路径</string>
79     </property>
80 </widget>
81 <resources/>
82 <connections/>
83 </ui>

```

26.3.2.3 实现功能

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

```

1  #ifndef PATHDIALOG_H
2  #define PATHDIALOG_H
3
4  #include <QDialog>
5
6  QT_BEGIN_NAMESPACE
7  namespace Ui { class PathDialog; }
8  QT_END_NAMESPACE
9
10 class PathDialog : public QDialog
11 {
12     Q_OBJECT
13
14 public:
15     PathDialog(QWidget *parent = nullptr);
16     ~PathDialog();
17
18 protected:
19     void paintEvent(QPaintEvent*);
20
21 private:
22     Ui::PathDialog *ui;
23 };
24
25 #endif // PATHDIALOG_H

```

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

```

1  #include <QPainter>
2
3  #include "pathdialog.h"
4  #include "ui_pathdialog.h"
5
6  PathDialog::PathDialog(QWidget *parent)
7      : QDialog(parent)
8      , ui(new Ui::PathDialog)
9  {
10     ui->setupUi(this);
11 }
12
13 PathDialog::~PathDialog()
14 {
15     delete ui;

```

```
16 }
17
18 void PathDialog::paintEvent(QPaintEvent*)
19 {
20     QPainter painter(this);
21     painter.setRenderHint(QPainter::Antialiasing);
22
23     int w = width(), h = height();
24
25     QPen pen;
26     pen.setColor(Qt::red);
27     pen.setWidth(3);
28     pen.setStyle(Qt::SolidLine);
29     pen.setJoinStyle(Qt::MiterJoin);
30     painter.setPen(pen);
31     QBrush brush;
32     brush.setColor(Qt::yellow);
33     brush.setStyle(Qt::SolidPattern);
34     painter.setBrush(brush);
35
36     QPainterPath path;
37     path.moveTo(-100, 0);
38     path.quadTo(QPoint(-50, -100),QPoint(0, 0));
39     path.quadTo(QPoint(50, 100),QPoint(100, 0));
40     path.closeSubpath();
41
42     painter.save();
43     painter.translate(w/3, h/3);
44     painter.drawPath(path);
45     painter.restore();
46
47     painter.save();
48     painter.translate(w*2/3, h/3);
49     painter.scale(0.5,0.5);
50     painter.drawPath(path);
51     painter.restore();
52
53     painter.save();
54     painter.translate(w/3, h*2/3);
55     painter.rotate(90);
56     painter.drawPath(path);
57     painter.restore();
58
59     painter.save();
60     painter.translate(w*2/3, h*2/3);
61     painter.shear(-1, 0);
62     painter.drawPath(path);
63     painter.restore();
64 }
```


20.5.2.4 测试验证

运行效果如图所示：

