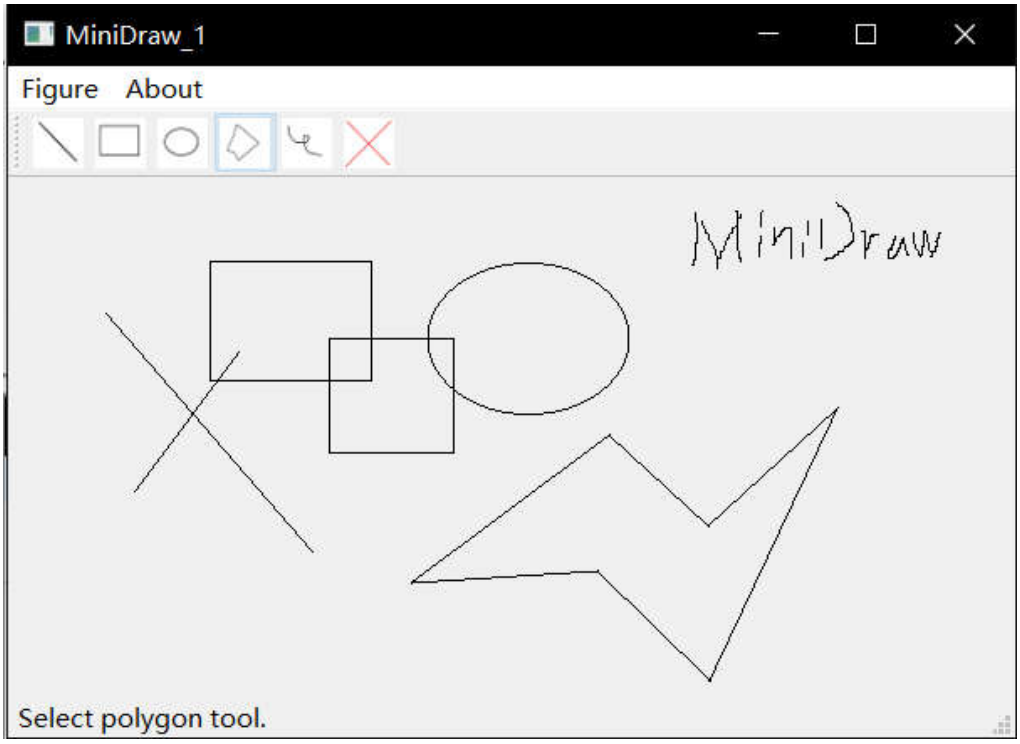
作业 01 MiniDraw

庄涛 PB15111679，ID：85
计算机科学与技术系，215 院 011 系 01 班
2018/03/04

一、内容

写一个画图小程序 MiniDraw，要求画直线(Line)，椭圆(Ellipse)，矩形(Rectangle)，多边形(Polygon)等图形元素(图元)。
每种图元需用一个类（对象）来封装，如 Line, Ellipse, Rect, Polygon, Freehand；
各种图元可从一个父类 Figure 来继承；
每种图元的绘制不一样，但是绘制接口在父类中是一样，因此可学习和使用类的多态性。



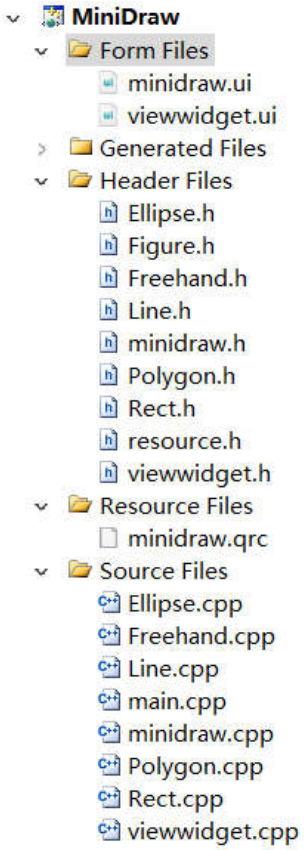
二、环境

IDE：Microsoft Vistual Studio 2010
QT：5.5

三、过程

模仿 Demo-1 实现了 MiniDraw

3.1 目录结构



3.2 动态更新

QT 教程中所写的 MiniDraw 有个问题是必须整个绘制操作完毕后才显示图形，为了能够动态更新，我在 Figure 中添加了函数 update，派生类再去各自实现

```
class Figure{
public:
    virtual ~Figure(){};
    virtual void Draw(QPainter &paint)=0;

    // 提供动态更新的接口
    virtual void update(const QPoint _end_point){};

    // 提供给 Polygon 的接口
    virtual void update(int mode){};
};
```

3.3 使用 QT 的类

在实验过程中，可能会考虑使用 struct Point 来存储点的 x 值和 y 值，也可能考虑使用 vector<Point *>来存储点序列。但使用一些 QT 的接口时会发现这些数据结构不能直接传入。查了文档后发现 QT 有类 QPoint，QPolygon 等，故改成了使用这些类。

3.4 绘制多边形引入的问题

多边形绘制过程中运行释放鼠标按键，如果此时尝试去更换绘制图形类型，demo-1 和 demo-2 发生了不一样的程序行为。Demo-1 允许这样的操作，并且多边形消失（但已保存），然后绘制新图形，再切换回多边形时，多边形重新出现。而 demo-2 不允许上述的切换。显然，demo-1 应该是忘了在切换图像类型时检测当前绘制状态了。相关的代码如下

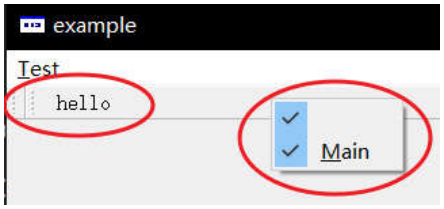
```
void ViewWidget::set_figure_type_to_rectangle(){
    if(draw_status == false)
        figure_type = kRectangle;
}
```

3.5 两个工具栏问题

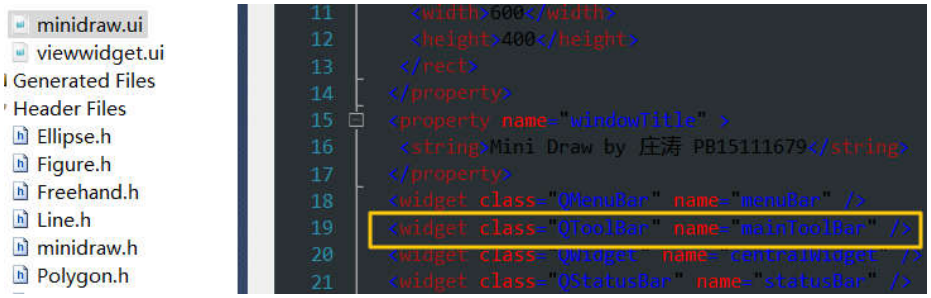
教程中添加工具栏的方式为

```
toolBar = addToolBar(tr("tool"));
```

这样会导致产生两个 toolBar，如下



原因在于窗口中已经有一个 toolBar 了，这可在 MiniDraw.ui 中看到



因此正确的做法是获取该 toolBar，做法如下

```
toolBar = findChild<QToolBar*>(tr("mainToolBar"));
```

这个问题在 demo-1 中解决了，但 demo-2 中存在

3.6 添加图形 button

demo-1 将文字按键改成了图形按键，实现的方法也很简单，接口如下

```
QAction::QAction(const QIcon &icon, const QString &text, QObject *parent = nullptr)
```

首先将图片资源放到 minidraw.qrc 的相同目录下，我将图片放到了文件夹 images 中，并且将 images 放到 minidraw.qrc 的相同目录下，如下

再将 minidraw.qrc 改为

```
<RCC>
    <qresource prefix="MiniDraw">
      <file>images\Line.bmp</file>
      <file>images\Rectangle.bmp</file>
      <file>images\Ellipse.bmp</file>
      <file>images\Polygon.bmp</file>
      <file>images\Freehand.bmp</file>
      <file>images\Undo.bmp</file>
    </qresource>
</RCC>
```

然后再生成相应的 QActiion 即可，如下

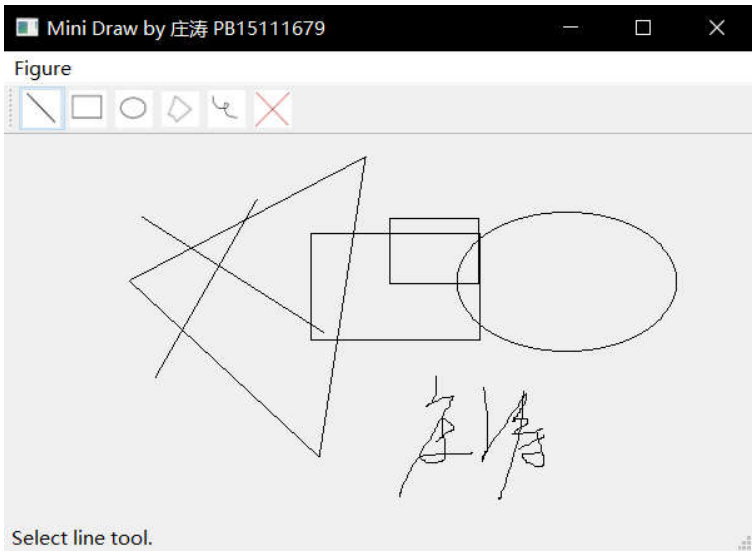
```
QAction(QIcon(tr(":MiniDraw/images/Line.bmp")), tr("Line"), this);
```

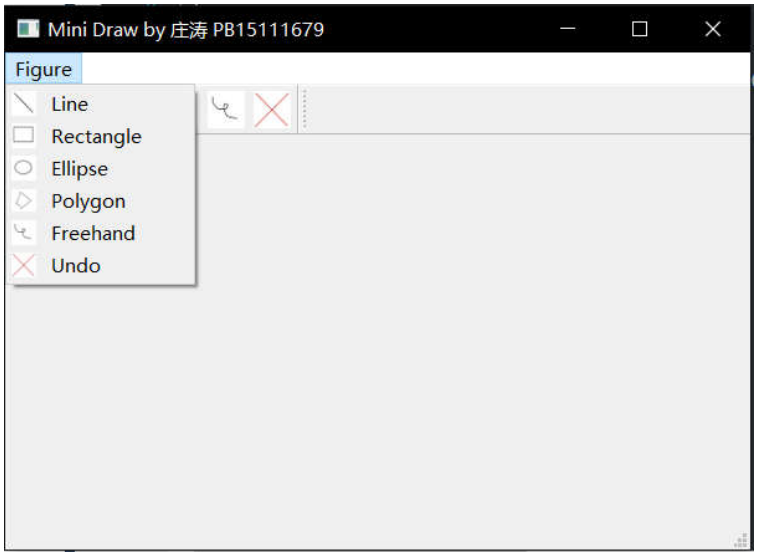
3.7 教程中没提到的功能的实现接口

名称	函数名
Freehand 绘制	QPainter::drawPath
QPath 生成	QPath::moveTo QPath::lineTo
QPolygon 生成	QPolygon::push_back
绘制多段线	QPainter::drawPolyline
绘制多边形	QPainter::drawPolygon
绘制矩形	QPainter::drawRect
设置状态栏提示信息	QAction::setStatusTip

4. 结果

截图如下





5. 总结

总体上来说难度较小，但量较大。能够锻炼 c++的基础编程能力，初步简单了解 QT 的框架。但并没有涉及图形学的内容，很期待接下来的知识学习。