# 项目报告

## 1.小组成员及分工情况：

3150104316 付强：负责窗口类搭建和交互，完成曲线类的具体实现，完成颜色选择、保存、打开文件的操作。

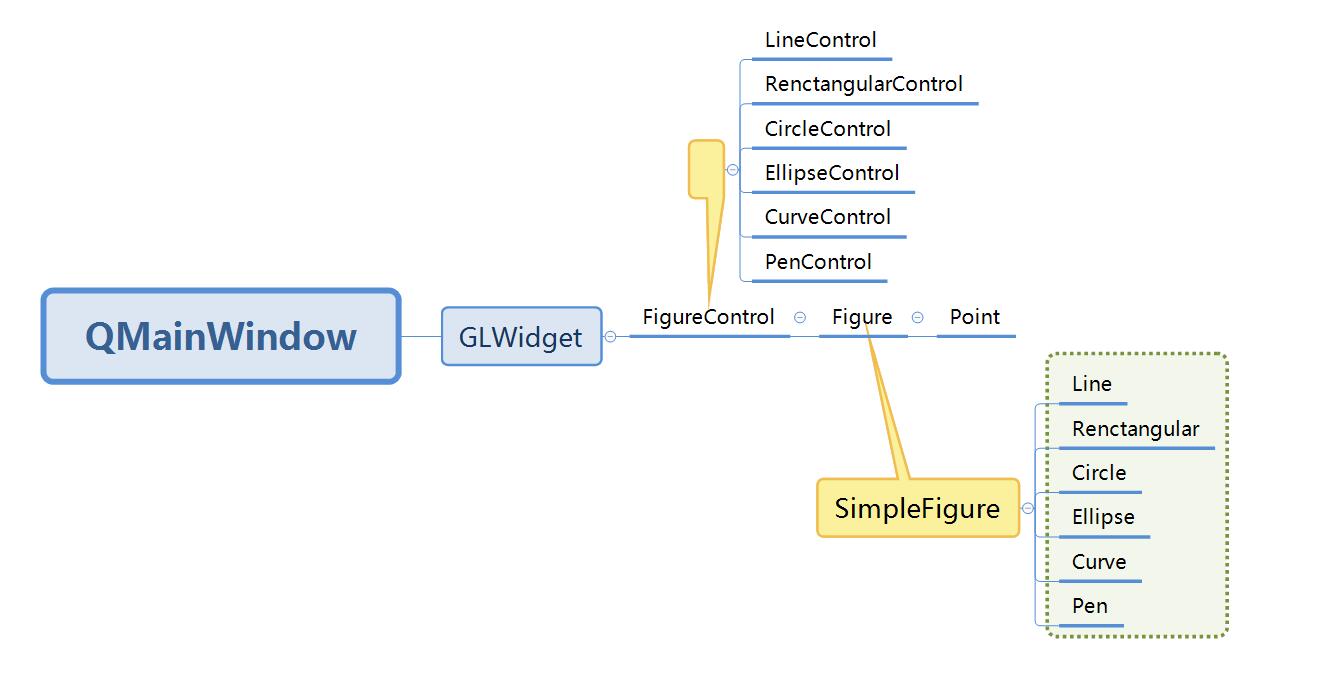
21612013 付蕾：负责Point类的构建，完成对椭圆和画笔两个功能。完成所有图形设置线宽的操作。

21612011 梁育玮：完成Circle类和CircleControl类的具体实现。完成填充，删除，撤销的具体实现。

3160101255 吉治宇：负责直线绘制、SimpleFigure类的构建、完成Rectangle类和RectangleControl类的具体实现。

## 2.项目：图形编辑器

## 3.架构图：



开发环境：

|  |  |
| --- | --- |
| 操作系统 | Windows 10 |
| 编程语言 | C++ 11 |
| 外部库 | Qt 5.8，OpenGL |
| 编译器 | MinGW |
| IDE | Qt Creator 4.2.1 |

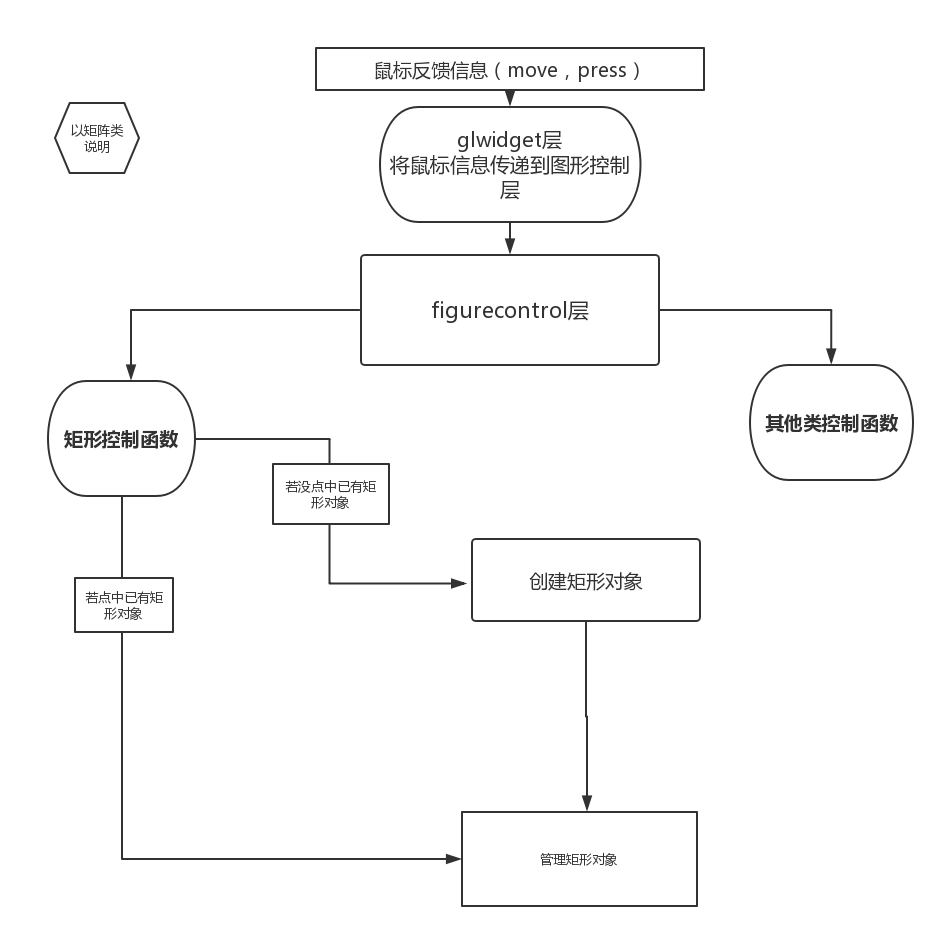
## 4.功能：

1. 新建文件，保存文件，打开文件
2. 基本图形的绘制——直线、矩形、圆、椭圆、曲线、涂鸦笔
3. 对图形的变换——平移、旋转、缩放
4. 对图形的设置——颜色选择、线条粗细设置
5. 对图形的修改——填充、删除、清除、撤销

6.文字编辑的插入（在代码内，但实验并不理想）

## 5.设计说明

### 5.1 矩形

****

**矩形的类**

矩形是Rectangular类和RectangularControl类来实现的，其中Rectangular类是继承SimpleFigure类，RectangularContro是继承FigureControl类。Rectangular类主要包含点的存储和矩形的底层绘图，RectangularContro类主要是存储所有的Rectangular类对象，并与qt进行交互，调用绘图界面。

**矩形的生成算法**

矩形的生成主要是两步：第一步是记录矩形的两个对角线角，第二步是通过两个对角线角点，来计算四条边上的像素点，再将所有像素点显示出来。

### 5.2 圆

Circle类继承SimpleFigure类和Area类。Circle类包含了圆的一些基本变量，圆心，半径，标记点，把手点，继承了SimpleFigure的轮廓点和Area类的填充点。Circle类提供了一些成员函数，能够对Circle对象进行一些操作，如平移，缩放，判断是否在圆上，删除，填充等。CircleControl类为操作Circle类提供了接口，实现鼠标，窗口工具栏对圆的操作，如在画布上画出图形，删除，清除，撤销等一系列操作功能。

**绘制算法**

**圆轮廓的绘制：**

圆心位于原点的圆有四条对称轴x = 0、y = 0、x = y和x = -y，若已知圆弧上一点P(x，y)，就可以得到其关于四条对称轴的七个对称点：（x, -y）、（-x, y）、（-x, -y）、（y, x）、（y, -x）、（-y, x）、（-y, -x），这种性质称为八分对称性。因此只要能画出八分之一的圆弧，就可以利用对称性的原理得到整个圆。这八分之一圆弧则由下述中点圆算法得到。

中点圆轮廓点算法：

考虑圆心在原点，半径为R的圆在第一象限内的八分之一圆弧，从点（0, R）到点（R' , R' ）顺时针方向确定这段圆弧。假定某点Pi(xi, yi)已经是该圆弧上最接近实际圆弧的点，那么Pi的下一个点只可能是正右方的P1或右下方的P2两者之一。

构造判别函数： **F(x,y）= x2+y2-R2** 。当F(x, y）= 0，表示点在圆上，当F(x, y）> 0，表示点在圆外，当F(x, y）< 0，表示点在圆内。如果M是P1和P2的中点，则M的坐标是（xi+1, yi–0.5），当F（xi+1, yi–0.5）< 0时，M点在圆内，说明P1点离实际圆弧更近，应该取P1作为圆的下一个点。同理分析，当F（xi+1, yi–0.5）> 0时，P2离实际圆弧更近，应取P2作为下一个点。当F（xi+1, yi–0.5）= 0时，P1和P2都可以作为圆的下一个点，算法约定取P2作为下一个点。将M点坐标带入F（x,y），得到判别式并规约到整数运算：

**d = d + 4 \*x +6; d0 = 3 - 2 \* r。**

**圆的填充点算法：**

线性扫描种子填充算法：遍历所有的圆的轮廓点，以该点出发，生成与该点同一y值的在圆范围内的点为填充点。

### 5.3 椭圆

**椭圆的类和数据结构**

椭圆的实现是由MyEllipse和EllipseControl两个类来完成的，其中MyEllipse是继承于SimpleFigure类和Area类，ElllipseControl是继承于FigureControl类。MyEllipse类的主要功能在于数学和几何方面：计算椭圆圆周上的点并存在vector中；提供椭圆的缩放、旋转和平移功能。EllipseControl则是关于椭圆的编辑操作和用户交互：用vector保存所有画过的椭圆对象（MyEllipse是其成员变量），并基于用户鼠标操作生成椭圆、平移、旋转、缩放椭圆，删除和清空所有椭圆。

椭圆的几何特征是由其圆周上所有的点和其外接矩形的顶点确定的，这些点的指针均作为成员变量保存在MyEllipse类中的vector中，可以被遍历和查找。点的颜色和大小，决定了椭圆的颜色和椭圆圆周的粗细，由MyEllipse的基类SimpleFigure来设定。

**椭圆生成算法**

椭圆的生成是通过计算椭圆圆周上的像素点来实现的，采用了椭圆的bresenham 算法，该算法是一种生成椭圆的整数型算法，由于整个算法当中，均采用整数运算，生成椭圆的效率比较高。

由于椭圆是中心对称图形，我们只需要计算其1/4的部分，然后根据对称规律即可得到其他三部分的点。首先，将第一象限中的图形分为两部分，区域1是图形切线斜率大于-1的部分，区域2是图形切线斜率小于-1的部分，如图9所示。

|  |  |
| --- | --- |
|  |  |
| （a）区域1 | （b）区域2 |
| 图1 第一象限分区示意图 | |

在区域1中，以x轴方向来增加步长，计算出合适的y值，从而确定合适的像素点。即每次x增加1（像素点），判断y是不变还是减去1，其判别式为：

D= d1 - d2;

其中， ,，d1和d2的几何意义如图1中的（a）图所示，a为椭圆长半轴长度。当判别式D>0，则应该取P1点作为下一个像素点，反之则取P2。以（0，b）作为椭圆上部区域的起点，并将其带入到判别式D中，可以得到递推公式如下：

(1.2-1)

（Di < 0） (1.2-2)

（Di>= 0） (1.2-3)

根据递推公式，可以根据前一个D值，计算后一个D值，就算非常简便。

在区域2中，应以y轴方向来增加步长，计算出合适的x值。即每次y减小1，判断x是不变还是减小1，判别式同区域1中的判别式，当D>0是，则取（b）图中的P1点，反之则取P2点。

(1.2-4)

（Di < 0） (1.2-5)

（Di>= 0） (1.2-6)

有了上述数学分析基础，接着进行下列bresenham算法（伪代码描述）：

（1）输入椭圆的中心点坐标和长短轴长度a,b；

（2）得到中心在原点的椭圆上的第一个点坐标（0，b）；

（3）根据公式（1.2-1）来计算初始的D值；

（4）在区域1中，从X0=0开始，不断递增至Xk，对于每个Xk位置，反复按照公式（1.2-2）或者公式（1.2-3）来计算决策参数Dk，若Dk小于0，则Yk不变，反之Yk要减小1。重复此过程，直到区域1边界点；

（5）在区域2中，对于每个Yk位置，从k=0开始，反复按照公式（1.2-5）或者公式（1.2-6）计算决策参数Dk，若其小于0，则Xk减少1，反之则Xk保持不变，和Xk-1一样；

（6）根据第一象限的点，对称求出其他三个象限的坐标点；

（7）把所有的点存入vector当中。

### 5.4曲线

**曲线类和数据结构**

曲线的实现是由Curve和CurveControl两个类来完成的，Curve类是继承于SimpleFigure类，CurveControl是继承于FigureControl类。Curve类的主要功能在于数学和几何方面：计算曲线上的点并存在vector中；提供缩放、旋转和平移功能。CurveControl则是关于曲线的编辑操作和用户交互：用vector保存所有画过的曲线对象，并基于用户鼠标操作生成曲线、平移、旋转、缩放曲线，删除和清空所有曲线。

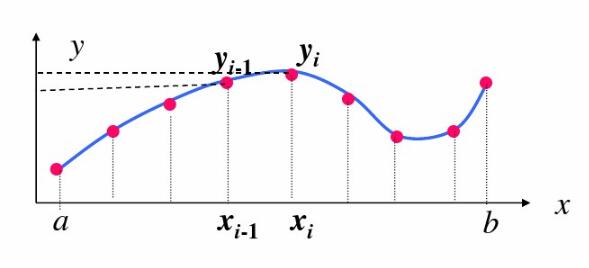
曲线的几何特征是由其4个顶点确定的，这些点的指针均作为成员变量保存在Curve类中的vector中，可以被遍历和查找。点的颜色和大小，决定了曲线的颜色和曲线圆周的粗细，由基类SimpleFigure来设定。

**曲线生成算法**

在三阶Bézier曲线中有四个控制点，设第i个控制点为，则和分别为曲线的起点和终点。曲线上点的计算公式为：

|  |  |
| --- | --- |
|  | (1.1.4-1) |

将从0累计到1，从而可以计算得到曲线上从到的所有点。（其中每次增加0.001，以保证曲线的连续）



### 5.5画笔

**画笔的类和数据结构**

画笔功能是通过Pen类和myPenControl类来实现的，其中Pen类是继承SimpleFigure类，myPenControl是继承FigureControl类。Pen类主要包含点的存储和线的底层绘图，myPenControl类主要是存储所有的Pen类对象，并与qt进行交互。

用户每次执行画笔功能，myPenControl便会鼠标的位置保存在Pen类对象当中，并将其显示在画布上，达到实时显示鼠标轨迹的效果。所有的Pen类对象的指针，存储在vector容器中，用于遍历、查找、删除等。

**画笔的生成算法**

画笔功能的算法，原理比较简单，即通过Qt捕捉鼠标移动轨迹上的点，将每个点的坐标存储在点的vector当中，然后，每保存一个点，便将所有的点遍历一遍，每两个点之间，用直线连接起来，这样就生成了一条实时的轨迹曲线。

### 5.6文本框分析

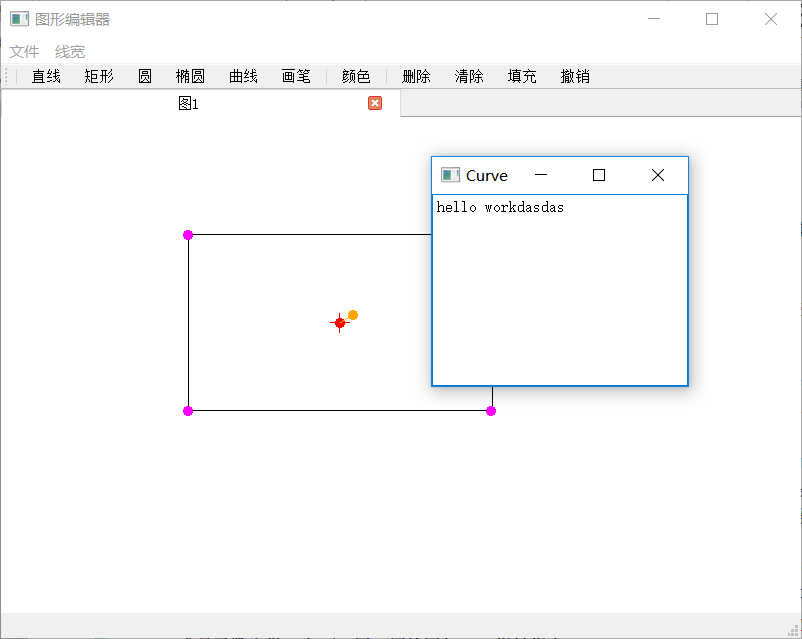
**1 文本框：如何在已有图层上插入文本是一个讨论点，我认为可以把文本框通过继承矩形类，来**

* 1. **数据结构**

文本框的设计是通过继承矩形类，然后修改矩形的构造函数，通过键盘回调函数来输入文本，保存信息到文本框类中。

* 1. **遇到的问题**

**其中思考过几种途径，包括通过模仿画笔，通过opengl库来绘制文本，但与画笔不同的是文本不能简单的通过点来制作；又思考到加载graphics库的文本绘制函数，但注意到了画布的问题。后面又考虑到QTextEdit库，但也不能输出到glwidget上来**



**最后采用的是Qt本身自带的QPainter来绘制，阅读说明，注意到QPainter的类构造函数中可以通过Qwidget的指针进行生成。本次项目的画布是GLWidget类的对象，是QWidget的派生类，但编译的时候一直报错，最后由于缺乏解决的渠道，未能解决这个问题。**

* 1. **体会：**

**首先由于我们的软件架构模式，我发现在矩形类对象生成的时候，根本无法找到画布GLwidget，这个问题在我们绘图，变换图形方面本身不是问题，但涉及到对画布本身进行操作的时候，感觉到了不如面向过程的编程方便，为解决这个问题，我不得不从glwidget的成员函数出发，向下一层一层的添加this指针指向glwidget。**

## 使用说明

### 6.1 矩形

**绘制**

本程序绘制矩形的功能包括：根据用户鼠标拖动，生成对应的矩形，并可以设置矩形的外边框颜色、边框线条粗细。首先在已有画布的情况下，点击矩形功能，然后按下鼠标左键，并拖动，就可以得到相应的矩形，其中，鼠标移动的起点和终点是矩形的对角线角点，两点决定了矩形的位置和大小。

|  |
| --- |
|  |
| 图x 矩形绘制实例 |

**选中**

只要用鼠标左键点击已有矩形的外边框，便可以选中矩形，选中的矩形，会显示四个角点，并用粉红色标记；显示中心点，用红色标记；显示把手点，用橘色标记。

**平移**

首先选中某个目标矩形，然后鼠标点击中心点，并拖动，则矩形将会沿着鼠标方向平移。

### 6.2圆：

**绘制圆**

本程序可以绘制圆，在建立好画布的条件下，点击工具栏的“圆”，进入到圆绘制模式，通过点击鼠标左键，然后保持按下左键的同时拖动鼠标，松开鼠标左键后，即可画出一个圆。当拖动鼠标绘图时，鼠标拖动起始点为圆的圆心，同时生成圆的外接矩形，决定了圆的位置和大小，如图1所示。

该圆刚画出时，其外接矩形将会通过虚线标出，并显示其标记点（四个顶点、中心点和把手点），中心点和把手点之间有直线相连。

|  |
| --- |
|  |
| 图1 绘制圆示意图 |

**选中圆**

当前刚绘制完的圆默认是被选中的状态，也可以选择其他已经画过的圆作为当前标记圆。选中的方法是用户的鼠标必须点击在某个圆的轮廓。

当某个圆被选中后，该圆的把手点用橘黄色标记，中心点用红色原点标记、外接矩形用蓝色虚线框显示、外接矩形四个顶点用紫色圆点显示，如图1所示。该圆会被置于所有图形的顶层。选中圆后，可以对其进行编辑操作。

**圆编辑**

选中某个圆后，可以对该圆进行编辑操作，包括：平移、旋转和缩放。

**平移**

当选中圆后，鼠标点击并拖动圆中心点，即可整体平移圆。鼠标移动的矢量即为圆的平移矢量。

**缩放**

用户选中目标圆后，点击圆外接矩形标记点，并移动鼠标，可以对当前圆进行放大或缩小，图2展示了缩放的示意图。

|  |  |
| --- | --- |
|  |  |
| （a）选中的圆 | （b）缩放之后的圆 |
| 图2 圆缩放示意图 | |

**填充：**

选择想要填充的图形，（目前，圆，椭圆，矩形可以被填充），点击工具栏中的“填充”按钮，便能对图形填充颜色。颜色默认为绿色，若想改变颜色，则需在填充之前选择颜色。图3是填充功能的展示。

|  |  |
| --- | --- |
|  |  |
| （a）选中的圆 | （b）填充之后的圆 |
| 图3 填充示意图 | |

### 6.3椭圆

**绘制椭圆**

本程序可以绘制椭圆，在建立好画布的条件下，进入到椭圆绘制模式，通过点击鼠标左键，然后保持按下左键的同时拖动鼠标，松开鼠标左键后，即可画出一个椭圆。当拖动鼠标绘图时，鼠标拖动起始点和终止点构成了一个矩形的对角线顶点，该矩形为椭圆的外接矩形，决定了椭圆的位置和大小，如图1所示。

该椭圆刚画出时，其外接矩形将会通过虚线标出，并显示其标记点（四个顶点、中心点和把手点），中心点和把手点之间有直线相连。

|  |
| --- |
|  |
| 图1 绘制椭圆示意图 |

**选中椭圆**

当前刚绘制完的椭圆默认是被选中的状态，也可以选择其他已经画过的椭圆作为当前标记椭圆。选中的方法是用户的鼠标必须点击在某个椭圆图形内部，或者是其标记点。

当某个椭圆被选中后，该椭圆的把手点用橘黄色标记，中心点用红色原点标记、外接矩形用蓝色虚线框显示、外接矩形四个顶点用紫色圆点显示，如图1所示。该椭圆会被置于所有图形的顶层。选中椭圆后，可以对其进行编辑操作。

**椭圆编辑**

选中某个椭圆后，可以对该椭圆进行编辑操作，包括：平移、旋转和缩放。

**平移**

当选中椭圆后，鼠标点击并拖动椭圆中心点，即可整体平移椭圆。鼠标移动的矢量即为椭圆的平移矢量。

**旋转**

选中目标椭圆，可以看到椭圆的把手点。点击并拖动这个把手点，就可以绕着椭圆中心旋转该椭圆。由于椭圆旋转后会变形，旋转功能只是实现90°旋转，即用户拖动把手点旋转，若拖动的角度小于90°则不旋转，大于90°则整个椭圆旋转90°。椭圆旋转操作示意图如图2所示。

|  |
| --- |
|  |
| （a）选中目标椭圆 |
|  |
| （b）旋转后的椭圆 |
| 图2 椭圆旋转操作示意图 |

**缩放**

用户选中目标椭圆后，点击椭圆外接矩形标记点，并移动鼠标，可以对当前椭圆进行放大或缩小，图3展示了缩放的示意图。

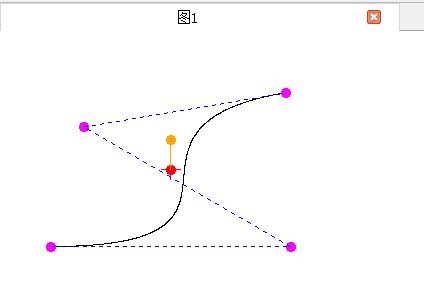
|  |  |
| --- | --- |
|  |  |
| （a）选中的椭圆 | （b）缩放之后的椭圆 |
| 图3 椭圆缩放示意图 | |

### 6.4曲线

**绘制曲线**

本程序可以绘曲线，在建立好画布的条件下，点击工具栏的“曲线”，进入到曲线绘制模式，通过点击鼠标左键，然后在任意位置画下四个点，即可绘制出曲线。

该曲线刚画出时，其外接矩形将会通过虚线标出，并显示其标记点（四个顶点、中心点和把手点），中心点和把手点之间有直线相连。



**选中曲线**

当某个曲线被选中后，该曲线的把手点用橘黄色标记，中心点用红色原点标记、四个顶点用紫色圆点显示，如图1所示。该圆会被置于所有图形的顶层。选中圆后，可以对其进行编辑操作。

**曲线编辑**

选中某个曲线后，可以对该曲线进行编辑操作，包括：平移、旋转。

**平移**

当选中曲线后，鼠标点击并拖动中心点，即可整体平移。鼠标移动的矢量即为曲线的平移矢量。

### 6.5画笔

本程序的画笔功能是：由鼠标操作，实现任意曲线的绘制。结合设定颜色和线宽，用户可以根据自己需要画出任意的曲线和曲线组合图案。

首先，在一块画布上，点击“画笔”工具，便开启了画笔模式，此后，按下鼠标左键，并拖动，就可以在画布上实时画出曲线。鼠标移动轨迹，即为线条的轨迹。若松开鼠标左键，则画笔功能停止；再次按下鼠标左键，可继续进行绘制轨迹曲线。如图4所示，使用画笔，可以画出鼠标移动轨迹。

由于画笔要实现的是画出任意轨迹曲线的功能，故没有选中、旋转和缩放功能。

|  |
| --- |
|  |
| 图4 画笔使用实例 |

### 6.6删除、清除、撤销：

在软件中，每一个生成的图像都作为一个Figure类保存在vector容器中，每一次删除操作，都把该图像从vector容器中删除，加入到另一个保存历史记录的vector中，以先进后出的原则实现撤销的操作。使用“清除”后，Figure类vector容器中所有的元素都被删除，所以无法被撤销。

**删除：**选中图形后，点击“删除”按钮，即可删除图形。

**清除：**点击“清除”按钮，即可清空画布上的所有图形。

**撤销：**在删除图形后，可点击工具栏中的“撤销”按钮，即可恢复上一次被删除后的图形。

“撤销”功能无法恢复被“清除”的图形。

具体操作可看操作视频。

## 测试分析

### 7.1 矩形

**绘制**

测试了矩形的绘制，不同的鼠标移动轨迹是否可以生成对应的矩形。同时测试了设定不同的颜色和宽度，发现都可以实现。测试表明矩形功能复合设计要求。

|  |
| --- |
|  |
| 图10 矩形测试实例 |

**平移**

对原始矩形进行平移，如图所示。除了图中实例之外，我们还对目标矩形进行了多方向、连续移动，发现均能实现。在有多个矩形的画布中，也可以实现对选中矩形的平移。

|  |  |
| --- | --- |
|  |  |
| （a）原始目标矩形 | （b）平移后的矩形 |
| 图11 矩形平移测试实例 | |

### 7.2圆测试

**绘制**

鼠标点击拖动，绘制不同位置、不同形状、不同颜色和线宽的圆。如图3所示，根据测试，绘制圆功能正常。

|  |
| --- |
|  |
| （a）多个圆、任意位置和大小,不同线宽，颜色 |
| 图3 绘制圆测试 |

**平移**

选中目标圆，对其进行平移操作，测试情况如图4所示。多次平移测试表明，圆可以实现任意平移操作，平移功能正常。

|  |  |
| --- | --- |
|  |  |
| （a）目标圆 | （b）平移后的圆 |
| 图4 平移圆测试 | |

**缩放**

对目标椭圆进行缩放，如图5所示。对目标椭圆，可以进行任意放大和缩小，多次测试表明，缩放功能正常，填充图形也能正常缩放。

|  |  |
| --- | --- |
|  |  |
| （a）目标圆1 | （b）放大目标圆1 |
| 图5 椭圆放大和缩小 | |

### 7.3椭圆测试

**绘制**

鼠标点击拖动，绘制不同位置、不同形状、不同颜色和线宽的椭圆。如图4所示，根据测试，绘制椭圆功能正常。

|  |  |
| --- | --- |
|  |  |
| （a）横向椭圆 | （b）纵向椭圆 |
|  |  |
| （c）多个椭圆、任意位置和大小 | （d）多个椭圆，不同线宽、不同颜色 |
| 图4 绘制椭圆测试 | |

**平移**

选中目标椭圆，对其进行平移操作，测试情况如图5所示。多次平移测试表明，椭圆可以实现任意平移操作，平移功能正常。

|  |  |
| --- | --- |
|  |  |
| （a）目标椭圆 | （b）平移后的椭圆 |
| 图5 平移椭圆测试 | |

**旋转**

对目标椭圆进行旋转操作，使得该椭圆旋转90°。经过对不同椭圆进行旋转测试，发现均可以实现正常的旋转功能。

|  |  |
| --- | --- |
|  |  |
| （a）长轴水平的椭圆 | （b）旋转之后的椭圆 |
|  |  |
| （c）长轴竖直的椭圆 | （d）旋转之后的椭圆 |
| 图6 旋转测试 | |

**缩放**

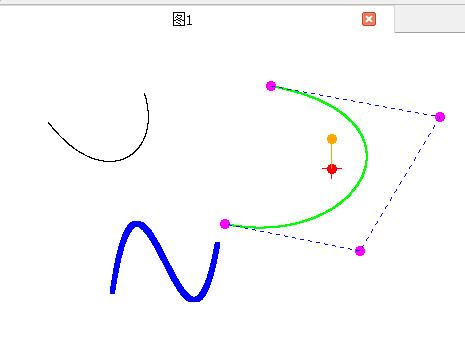
对目标椭圆进行缩放，如图7所示。对目标椭圆，可以进行任意放大和缩小，多次测试表明，缩放功能正常。

|  |  |
| --- | --- |
|  |  |
| （a）目标椭圆 | （b）缩小 |
|  |  |
| （c）目标椭圆2 | （d）放大目标椭圆2 |
| 图7 椭圆放大和缩小 | |

#### 7.4曲线测试

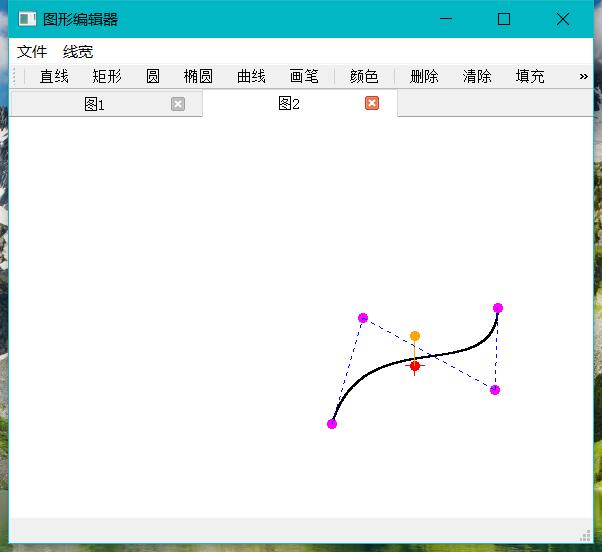
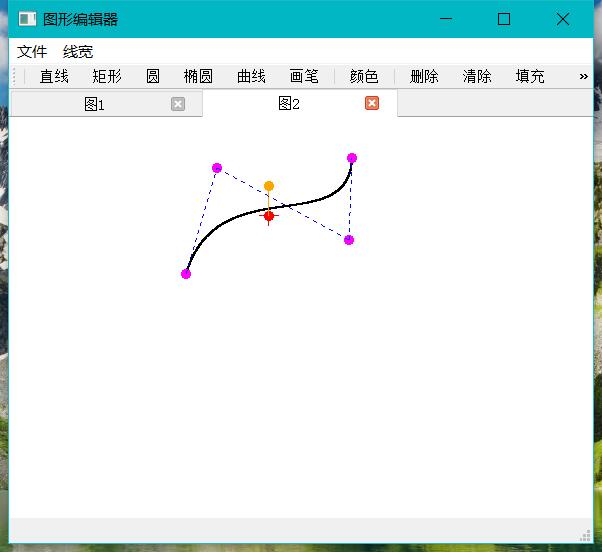
**绘制**

鼠标点击拖动，绘制不同位置、不同形状、不同颜色和线宽的曲线。根据测试，绘制曲线功能正常。



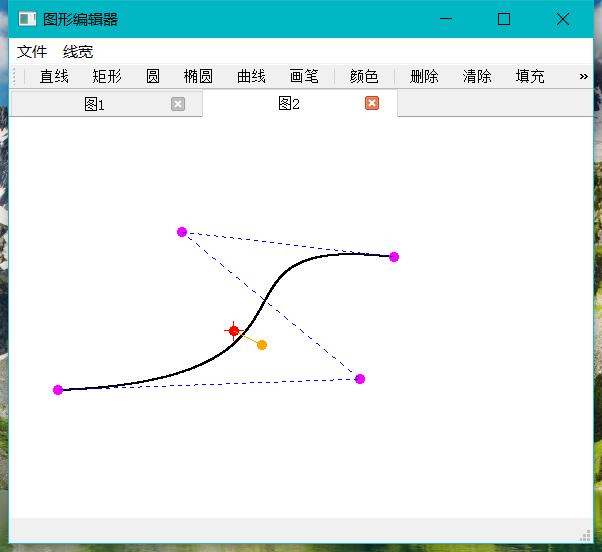
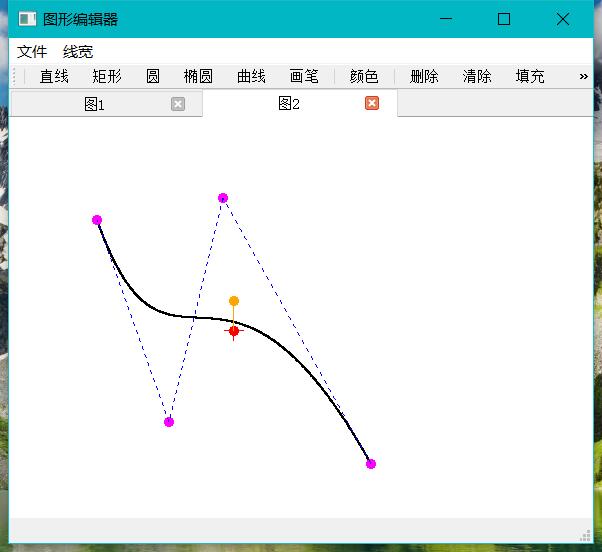
**平移**

选中目标曲线，对其进行平移操作，测试情况如图所示。多次平移测试表明，曲线可以实现任意平移操作，平移功能正常。



**旋转**

对目标曲线进行旋转操作，使得该曲线旋转任意角度，发现均可以实现正常的旋转功能。



#### 7.7画笔

本程序画笔支持用户设定颜色、设定线宽，来画出曲线。如图8所示，可以用画笔功能画出任意曲线，也可以在同一画布当中画出多条曲线，并可以设定不同的颜色和线条宽度。测试表明，画笔可以实现设定功能。

|  |  |
| --- | --- |
|  |  |
| （a）简单曲线 | （b）多条曲线、多种颜色和线宽 |
| 图 8 画笔测试实例 | |

**总结：**

**经验：**

本次大程序设计，能够帮助我们进一步地理解面向对象编程设计。在实现类的过程中，发现了类的设计需要提前精心的设计，在实现类的过程中了解到了类的设计和实现的不易，但在扩展性方面又有很大的便利。本次大程序的作业过程中，也让我们体会到了团队合作的重要性。

**本项目的不足：**

1.填充后无法更改颜色

2.无法选中，删除，撤销画笔

3.无法对拉伸，旋转，平移，放大，填充等步骤进行撤销。

## 8.总结

这个程序经过两个多月的讨论、修改、完善，现在基本达到了我们预先设定的标准，我们从最初对整个架构的不确定、对qt交互的不了解，到不断学习和尝试，在不断学习和犯错中，最终能够完成一个有较为合理的架构的小型软件，是一段很精彩、有很多快乐的编程经历。

对于该程序，我们的核心思路是设定了两种基类SimpleFigure和FigureControl，SimpleFigure关注图形本身，主要是几何特征和基础数据；FigureControl关注与qt的对接以及对所有SimpleFigure类对象的管理和编辑操作。在这样的思想下，我们编写了直线、曲线、椭圆、圆、矩形和画笔等基本图形，让他们在同样的基类上继承和扩展。这样的好处在于，椭圆、圆等基本图形，基于同样的两个基类，相互之间耦合度很低，可以分别编程，提高了合作效率。在这样的架构下，若想要增加一个新的图形，比如多边形，基本可以不对原来已有的程序进行修改。我们后来编写扩展功能如自定义颜色和线宽等，我们都能保证很少修改前面代码的基础上实现扩展功能。上述两点说明我们的程序具有较为良好的扩展性。

当然，该程序还有很多可以改进的地方，很多功能还比较初级，例如撤销操作，我们只能恢复被删除的图形，但是并不能让图形恢复到前一个状态；例如已有图片的读取，我们只能新开一个窗口进行查看，但无法实现在上面新画一些图形。这些地方，我们还需要多些思考，若是能够改进，将使本程序更加优秀。

此次大作业是我们第一个编写的较大的程序，在两个多月的编程实战中，我们充分利用了类的继承和派生关系，使用虚基类，在多态性的使用过程中渐渐体会到其强大功能和它对程序可扩展性的提升，这些都让我们对C++有了更为深入的理解。此外，通过小组合作编写程序，让我们也体会到了合作编程的难处和优势，促使我们不断去思考程序的架构和合作方法，这些都是非常宝贵的收获。