目录

[2018 - 2019 学年第1学期 0](#_Toc534974623)

[图形编辑程序设计 2](#_Toc534974624)

[**一、 作业内容** 2](#_Toc534974625)

[**二、 文件与执行环境** 2](#_Toc534974626)

[**2.1电子文档打包文件名及文件列表** 2](#_Toc534974627)

[2.2编译执行环境与命令 3](#_Toc534974628)

[**三、主要功能** 3](#_Toc534974629)

[**四、系统的设计** 4](#_Toc534974630)

[4.1使用流程图 4](#_Toc534974631)

[4.2功能模块图及说明 5](#_Toc534974632)

[**五、程序关键类的实现** 10](#_Toc534974633)

[**六、软件的主要界面截图** 13](#_Toc534974634)

[**七、任务分工与联系方式：** 22](#_Toc534974635)

[7.1 核心绘图代码 23](#_Toc534974636)

[7.2 GUI和I/O代码 38](#_Toc534974637)

[7.3 配置和图形操作代码 54](#_Toc534974638)

[**八、Java课程学习心得与改进意见** 81](#_Toc534974639)

# 图形编辑程序设计

1. **作业内容**

* 图画（参考Microsoft Word ，实现部分功能）
  + 对图形进行编辑、存取
* 基本要求
  + 新建一个空白图形文件
  + 打开或保存一个图形文件
  + 绘制基本图形
    - 直线、矩形、圆、椭圆
  + 任意线
  + 添加文字
  + 设定颜色与文字风格（种类数>=3）
  + 鼠标拖动画图添加文字
* 选做
  + 设定画笔、图像添加、图像修改（橡皮擦）
  + 图形填充、线型设置
  + 操作撤销
  + 菜单
  + 其它

1. **文件与执行环境**

**2.1电子文档打包文件名及文件列表**

打包文件名：paint.zip

源代码放在src文件夹里面，图标资源在文件夹icon里面，外部命令在文件夹external里面，可执行文件jar包+图标资源和exe文件在文件夹bin里面。

**bin**文件夹下为可执行文件

**paint src**文件夹下为源码和资源

**doc**文件夹下为大作业报告和说明文档

com.glede.paint文件夹的文件列表及工程目录：

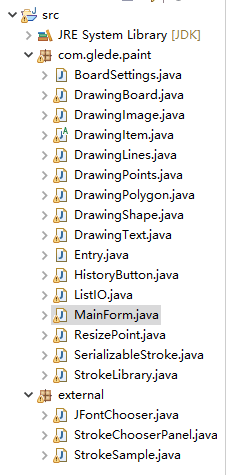


图2.1 com.glede.paint文件夹的文件列表及工程目录

### 2.2编译执行环境与命令

版本信息：

Java JDK版本： 1.8.0\_191

Java 版本： 1.8.0\_191

编译方式：

Eclipse 下打开src文件夹，执行Entry即可。

**三、主要功能**

（1）能够读取、保存、编辑图形文件；

（2）能够满足简单的画图需要

1、可以选择圆形(椭圆)、多边形、矩形、直线和自由作画的方式绘图；

2、可以设置绘图时线条的粗细、样式以及颜色；

3、在绘制圆形(椭圆)、多边形、矩形，可以选择是否填充，填充颜色可选；

4、可以在指定位置插入文字，文字样式、大小、颜色可选；

5、可以保存绘制的文件、读取已有的文件；

6、可以移动、删除指定的图形；

7、可以清空画板，可以选定所有的图形执行操作；

8、可以打开多个图片，实现拼图功能；

9、可以进行亮度调节；

10、保存文件有两种形式，一种pnt，一种jpg，用pnt保存，则可以再次编辑该文件，如果用jpg保存，则只能生成一个图片文件。

**四、系统的设计**

### 4.1使用流程图



图4.1 使用流程图

### 4.2功能模块图及说明

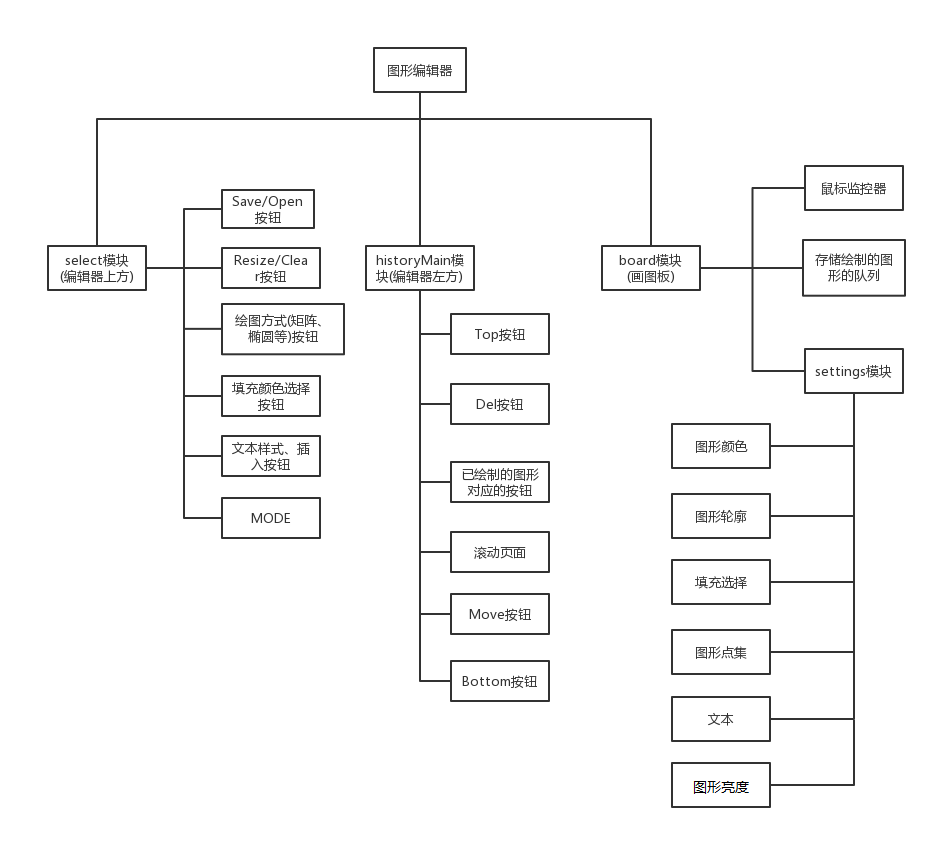


图4.2 功能模块图

图形编辑器是一个继承JFrame的MainForm类的对象mainForm。MainForm类下面有一个Jpanel类的mainPanel对象，在这个对象上，主要划分成三个模块。

第一个模块是JToolBar类的对象select，它里面存放着许多JButton类型的对象，如图2所示，有Save按钮，对象名为buttonSave，还有矩形按钮，对象名为buttonRect等。所有按钮的，以及它们的功能都如图2所示。另外图2还有一个MODE，是一个JLabel类的对象，对象名为labelModeNow，标记着此时图形编辑器进行的操作，例如画圆、画直线、删除、移动等。整个select对象位于图形编辑器的上方。

第二个模块是一个JPanel类的对象historyMain，它里面也存放着一些按钮，正如图所示，有Del、Move、Top、Bottom等，并且每绘制完一个图形，该图形对应的按钮也会存放在historyMain中，例如绘制一个矩形，便会有一个名为Rect的按钮加入其中。historyMain中还有一个JPanel类的对象history，用来实现滚动页面，用以显示所有的按钮。整个historyMain位于图形编辑器的左方。

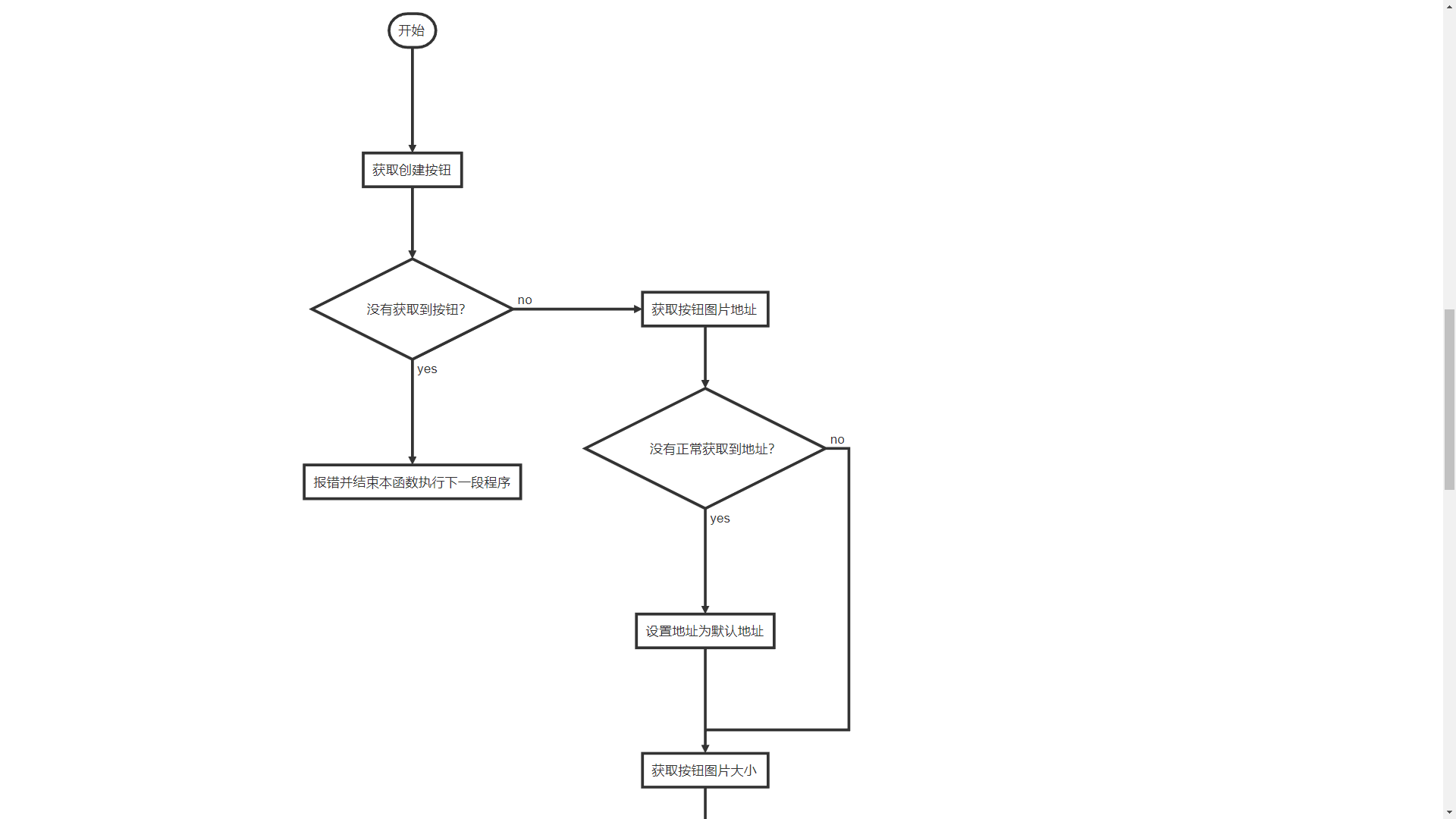
第三个模块是一个DrawingBoard类的对象board，是一个画图板，里面存放着鼠标监控器，还有一个LinkedList<DrawingItem>，用以存放绘制过的图形的信息，再有一个Boardsetting类的对象settings，用于辅助绘图，里面存放着图形的一些基本信息，例如轮廓、点集等，还存放着文本。



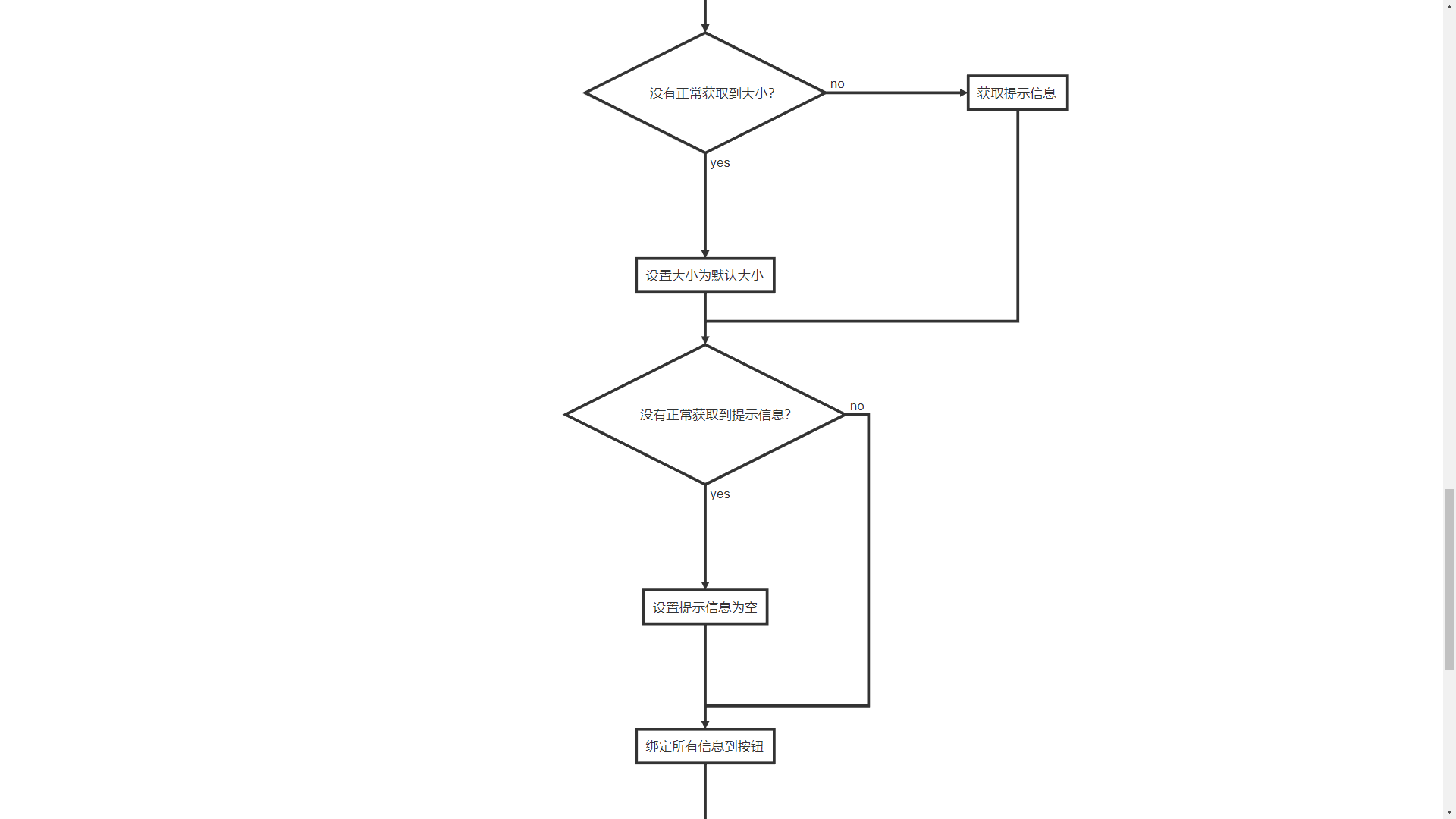
图4.3 亮度调节模块的算法流程图



图4.4 输入框提示信息流程图



（转下页）



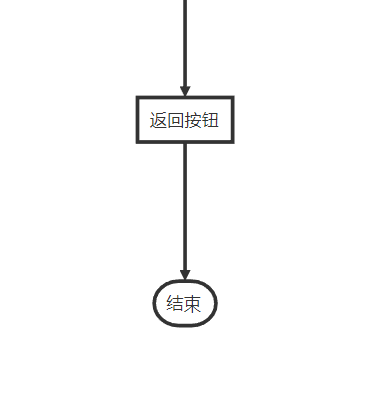


图4.5 按钮绑定事件

**五、程序关键类的实现**

对每个关键类的构成详细描述，包括域成员和成员方法的解释。除了给出成员方法的功能外，还要给出成员方法的执行流程。

JFontChooser类、StrokeChooserPanel类 、StrokeSample类，这三个类为外部导入。

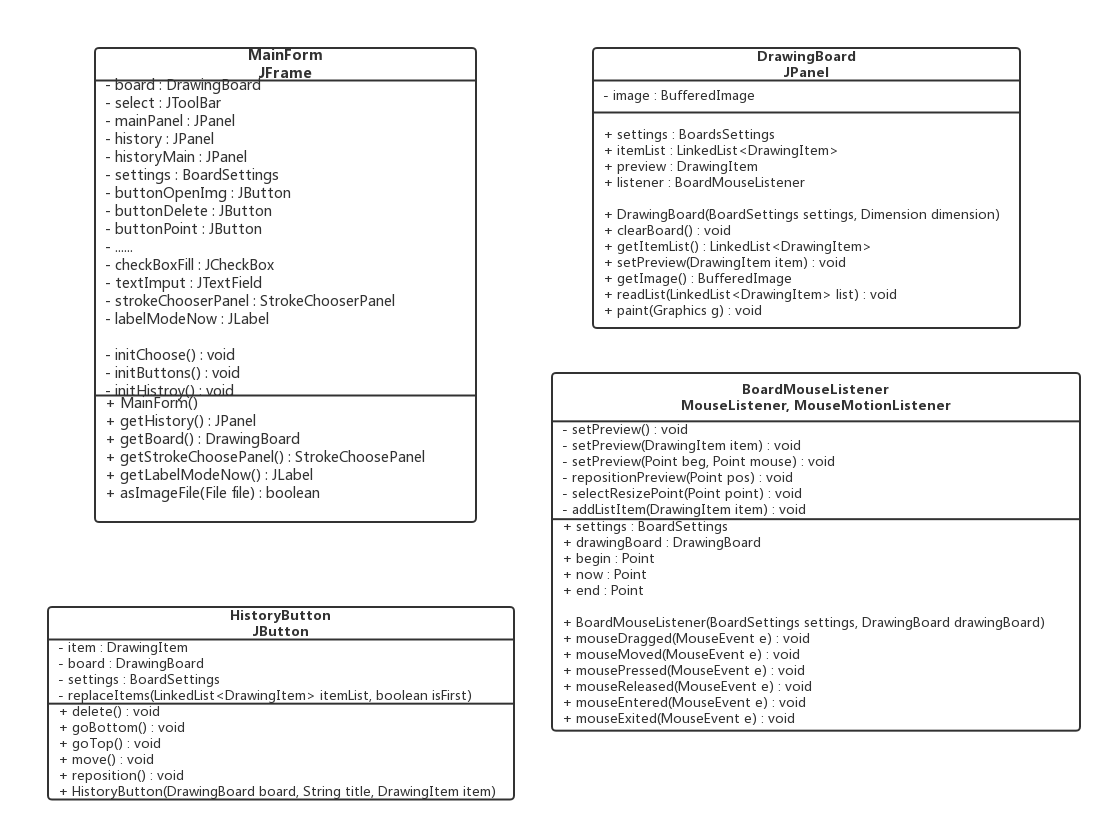


图4.3 部分类图

这部分类图里面有MainForm、DrawingBoard、HistoryButton、BoardMouseListener四个类的成员变量和成员函数的介绍。

**MainForm：**

initChoose()

功能：初始化画图板

流程：定义一个图形数组，将其添加进settings中，也就是添加进画图板，将在该类的构造函数中调用

initButtons()

功能：负责初始化编辑器上方的窗口，即Save、Color等按钮

流程：在函数里一一设定好buttonOpenImg、buttonPoint等成员变量，然后把它们添加进select中,将在该类的构造函数中调用

initHistory()

功能：负责初始化编辑器左方的窗口

流程：在函数里一一设定buttonTop、buttonDel等成员变量赋值，添加进historyMain，并设置好history(完成滚动页面功能)，把button类成员和history添加进historyMain中，即把编辑器左方部分的窗口设置好, 将在该类的构造函数中调用

getStrokeChoosePanel()

功能：返回当前正在绘制的图形

流程：返回成员变量strokeChoosePanel

getLabelModeNow()

功能：返回编辑器现在的状态，如删除、移动或者画直线、画矩形等

流程：返回成员变量labelModeNow

asImageFile()

功能：用于保存图像文件用的函数

流程：先定义一个String，用来存储文件名，然后选择存储文件格式

MainForm()

功能：构造函数,整个程序的流程是在该函数里面设置好的

流程：先调用setSize()、setTitle()等函数设置编辑器框架。然后调用addWindowListener，监控编辑器窗口，以判断编辑器关闭的时机。通过调用initHistory()，设置好historyMain；调用initChoose()，设置好settings，把settings放入board中；调用initButtons()设置好select。最后将三者放入mainPanel，mainPanel添加进this，程序总的调用流程完成。

**DrawingBoard:**

clearBoard()

功能：完成清除画图板的功能

流程：调用settings.getHistory().removeAll()，移除所有图形，再调用repaint()，使其能进行重新绘制，该函数将放进buttonClear的侦听器接口中。

getItemList()

功能：得到绘制过的图形存储表，在遍历图形时有用

流程：返回成员变量itemList

DrawingBoard()

功能：构造函数，设置好画图板窗口，以及定义各类成员变量

流程：设置settings、itemList、Listener，初始化好画图板窗口调用addMouseListener()、addMouseMotionListener()

**BoardMouseListener:**

mouseDragged()、mouseMoved()、mouseClicked() 、mouseEntered()、mousePressed()

mouseReleased()、mouseEntered()、mouseExited()

功能：重载上述八个函数，用于捕捉鼠标的行动，辅助画图

流程：主要是通过捕捉鼠标所在点的坐标，以及成员变量settings中存储的图形信息，再调用成员变量drawingBoard中的repaint()，重新绘制图形。当鼠标在画图板上点击移动时，根据画图板所处的模式，并调用这些函数，即可完成画图。

selectResizePoint()

功能：当需要调整图形形状，或者删除图形，或者移动图形会调用该函数

流程：通过for循环，遍历drawingBoard里面存储图形列表，逐个点扫描，调用repaint()重新绘制图形

BoardMouseListener() :

功能：构造函数，给成员变量赋值

流程：给settings和drawingBoard赋值

**HistoryButton:**

replaceItems()

功能：与goTop()、goBottom()协作，更换图形在存储表中的相对位置

流程：移除掉选定的图形，判定传入的boolean类型变量，选择把移除的图形放置在列表开头或者结尾

goBottom() 、goTop()

功能：将选定的图形放在表的最后和放在表的开头

流程：都是调用replaceItems()函数，通过传入不同的boolean值，完成不同的添加功能

HistoryButton()

功能：构造函数，设置成员变量

流程：先后设置item、board、settings等成员变量，然后调用addActionListener，监控按钮的动作



图4.4 DrawingItem类图以及它的派生类

**DramingItem:**

这是一个抽象类，这里介绍一些没有进行重载的函数的功能

drawResizePoint()

功能：对选定的图形重新绘制

流程：传入一个Graphics，调用当前图形的draw()函数，把传入的图形传进draw()

drawBounds()

功能：绘制图形轮廓

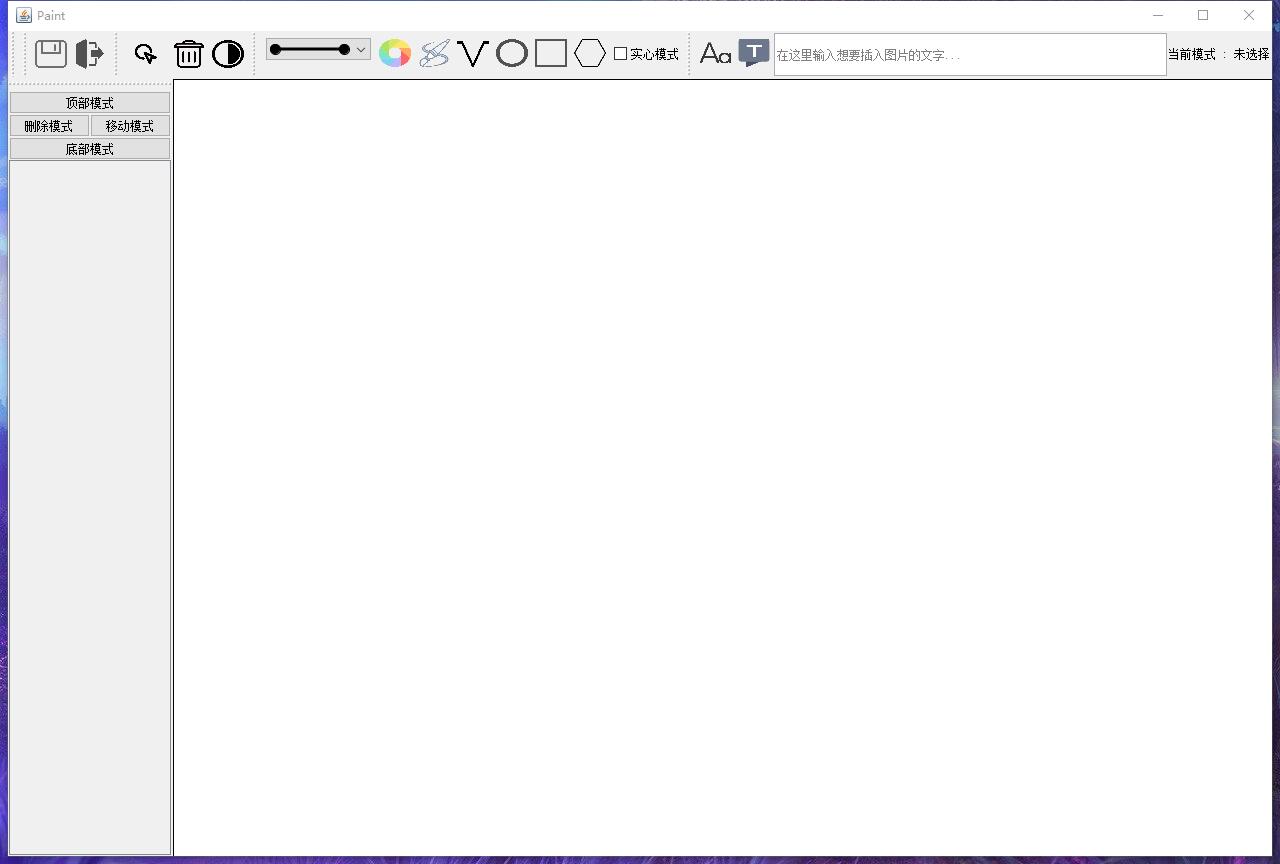
流程：定义一个Graphics2D对象，setStroke()函数，并把调用函数后所得的轮廓画出来

剩余的DramingLines、DramingPolygon、DramingImages等用于绘制图形，直线、多边形等。主要通过调用draw()实现绘图

**六、软件的主要界面截图**

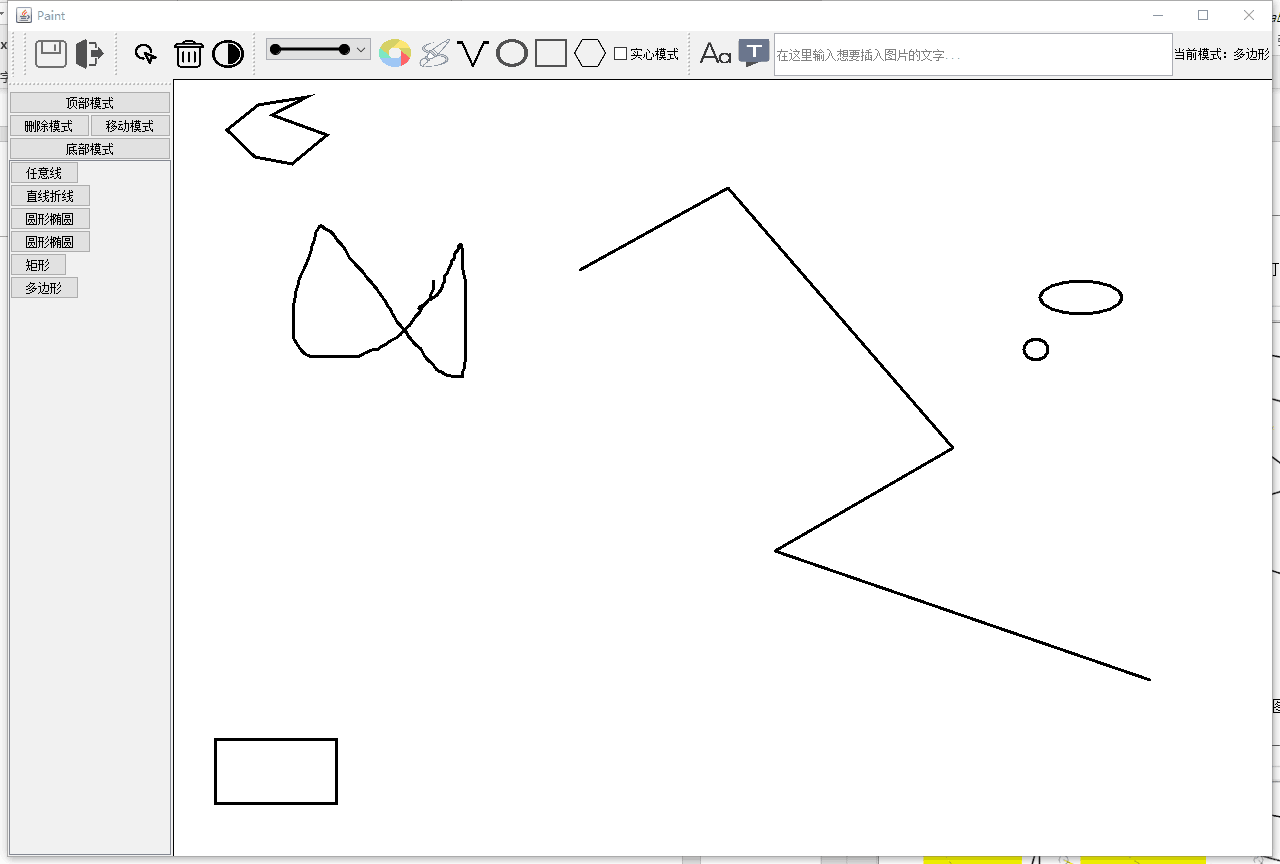
主要的功能界面截图要加文字描述。

1. 打开时的界面

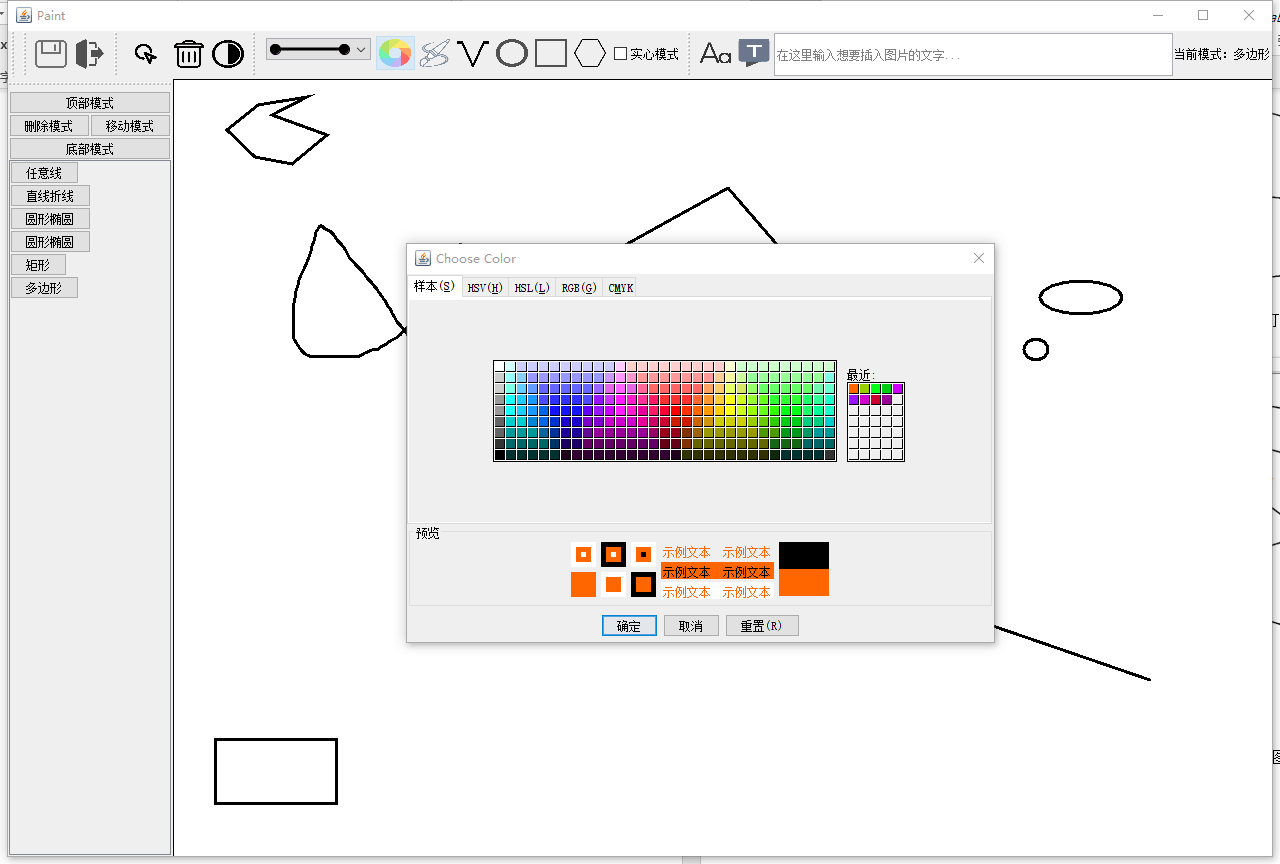


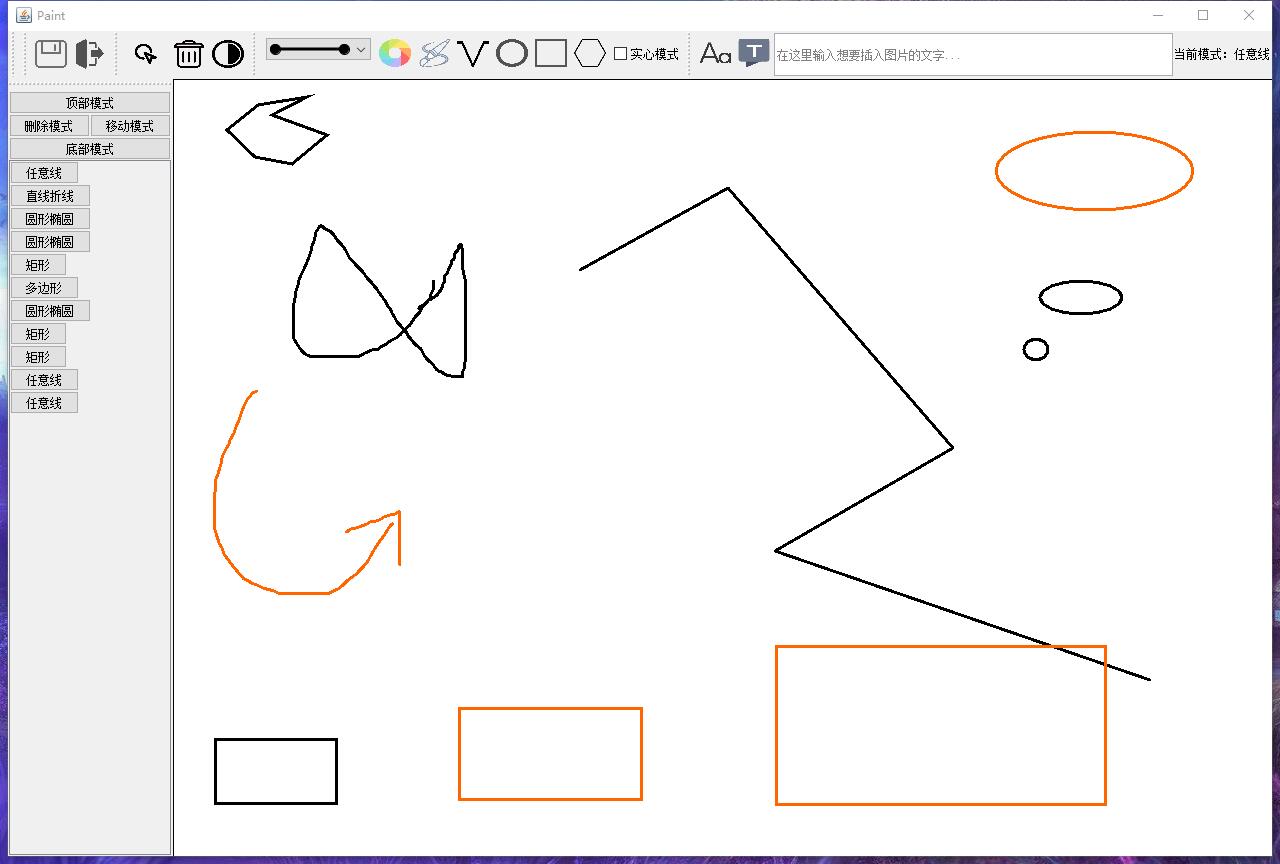
1. 默认线条、颜色下自由作画、画直线、画椭圆、画矩形、画多边形

注意图形编辑器左边的部分，底部模式下面有圆形椭圆、直线折线、任意线等按钮，每个按钮对应一个图形，只要先点击删除模式或者移动模式，再点击对应图形的按钮就可以进行删除或者移动。这是一个历史记录。



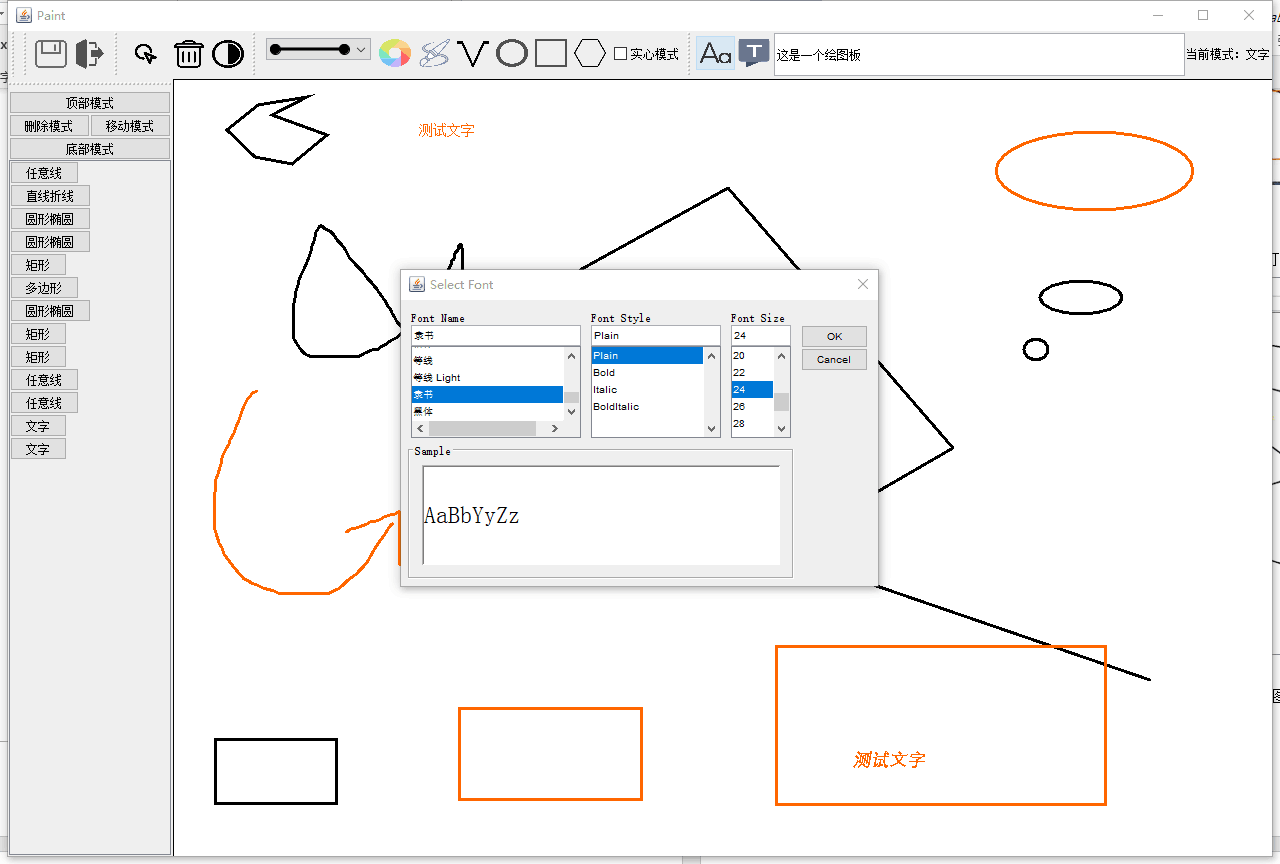
1. 颜色选择，并绘制

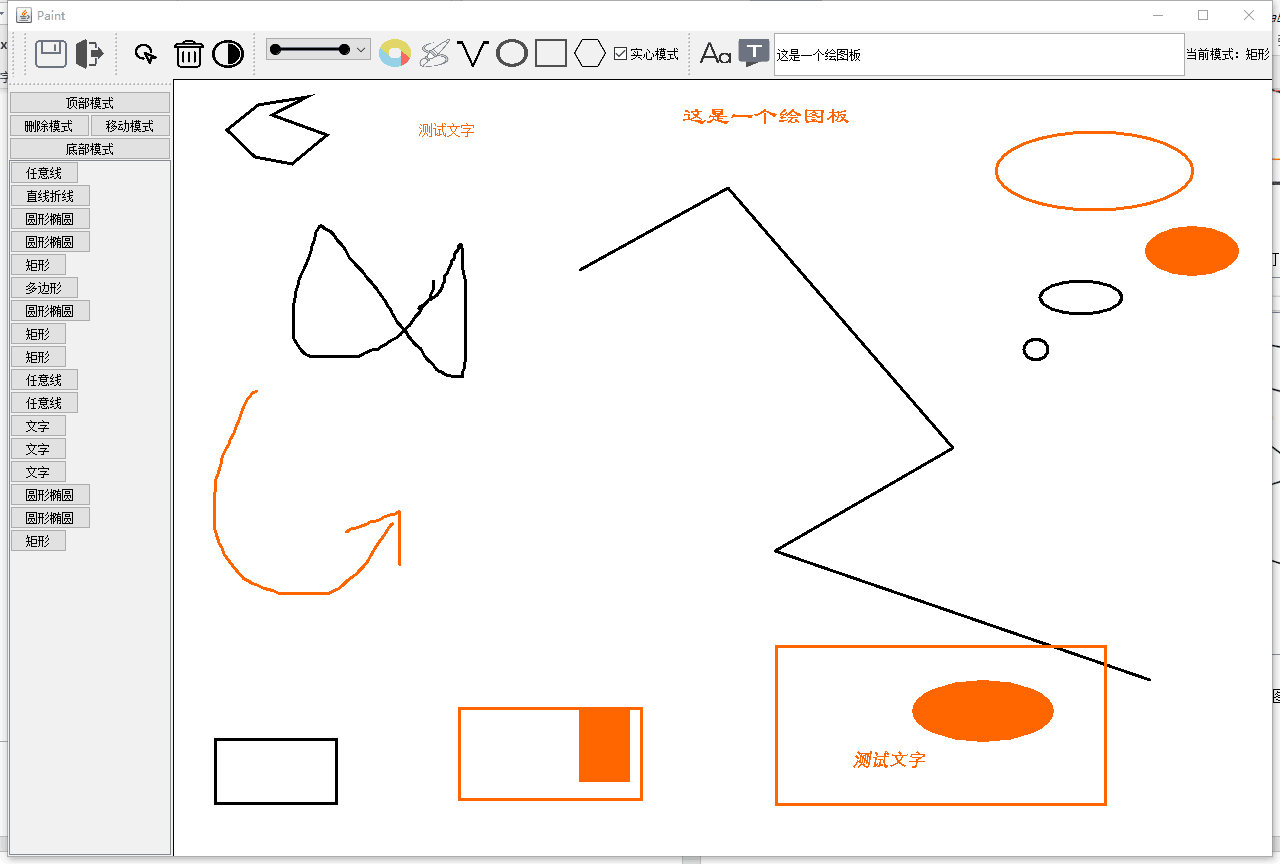




4、绘制有填充色的图形，以及添加文字

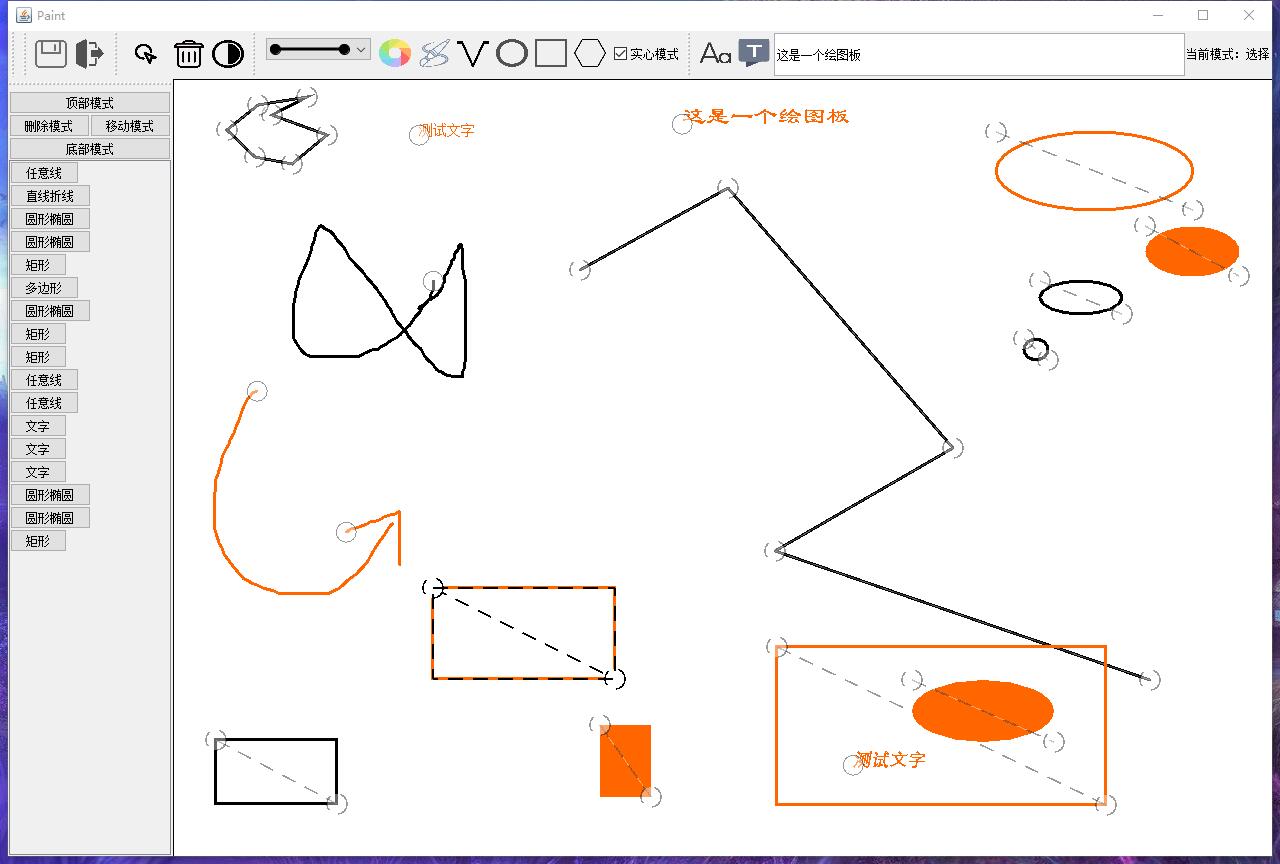
添加的文字在右上角的栏目里面输入，要填充只需要Fill前面的方框打钩即可

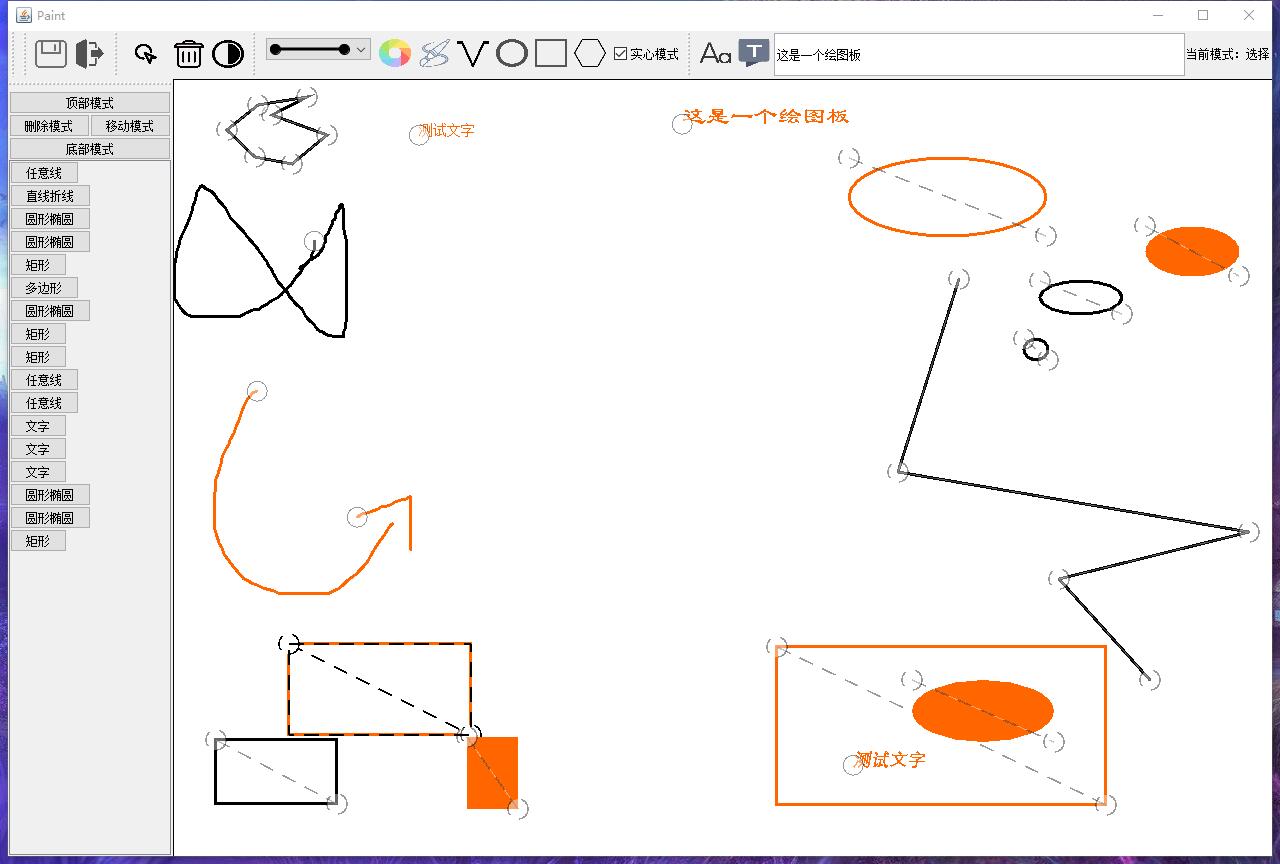




5、在上图的基础上，改变左边的矩形的形状，移动右下角的椭圆

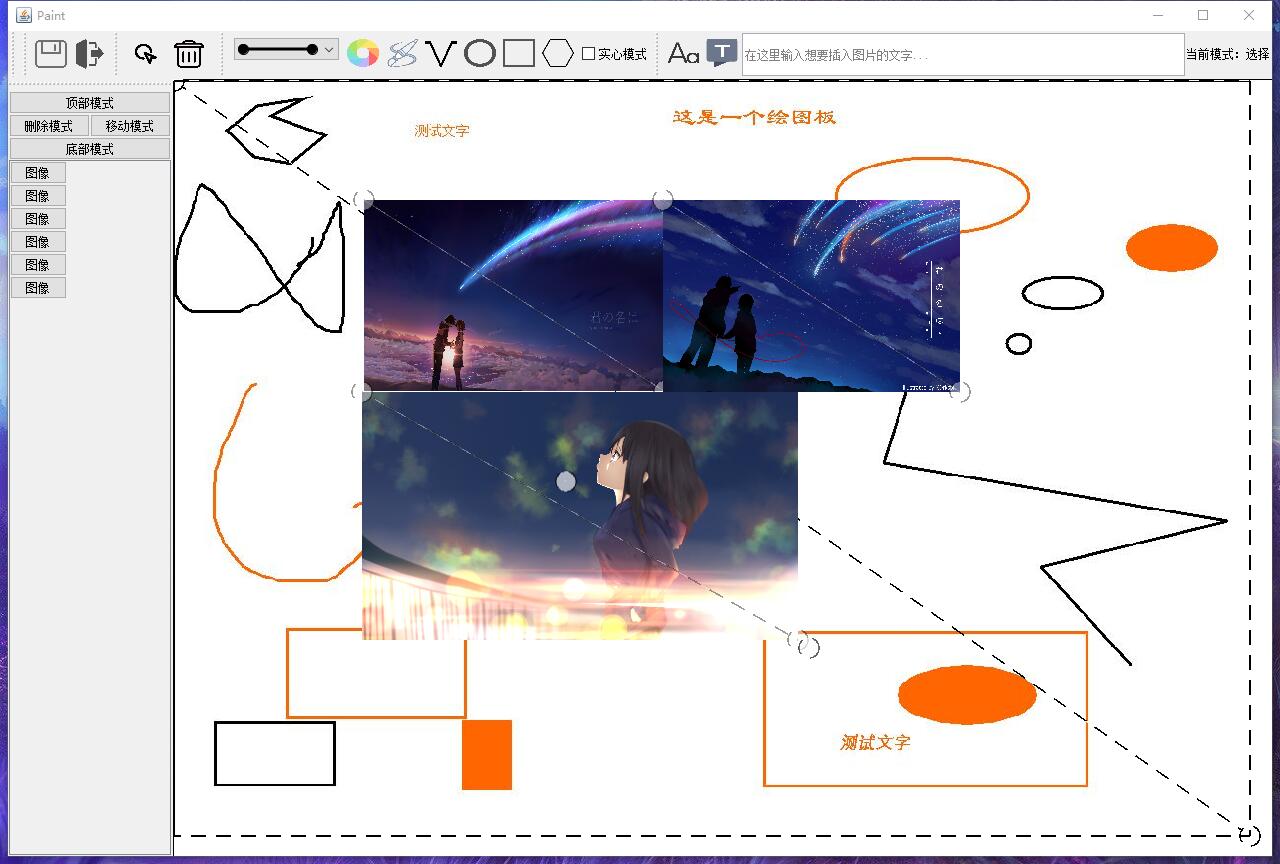
只需要按Resize，就可以操作所有的图形，这里将操作上述说的两个图形,改变后的图形如下所示



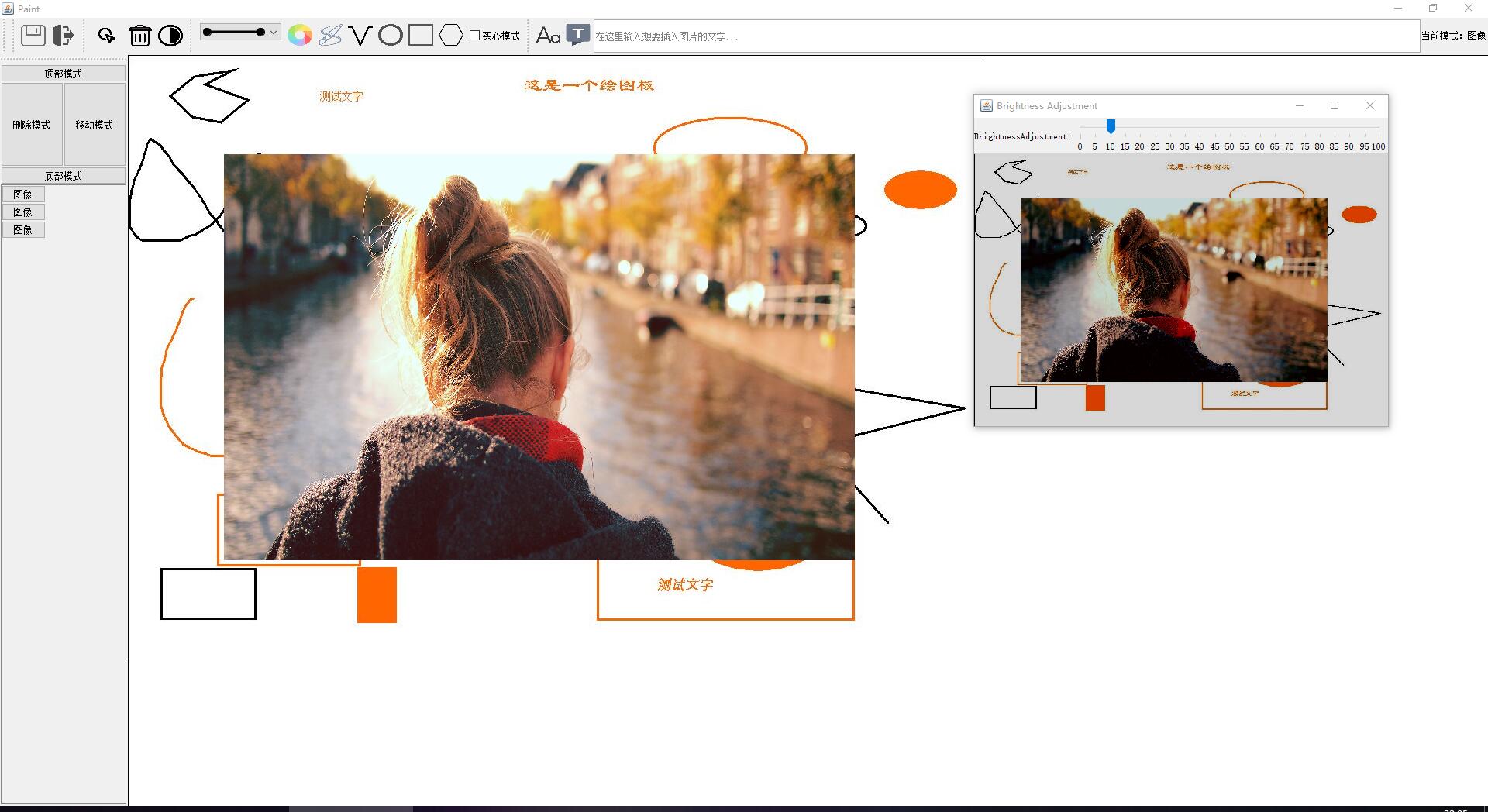


6、其他操作：

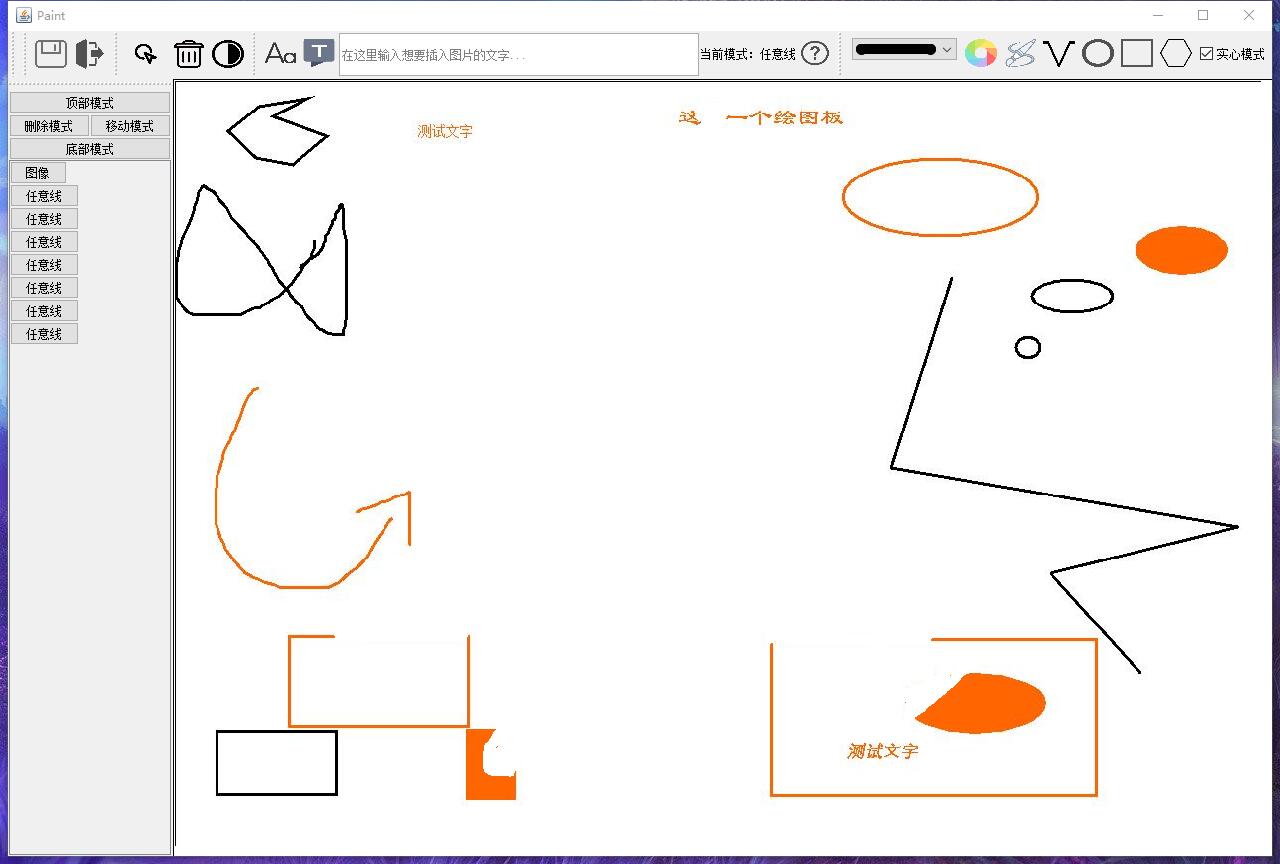
还可以进行拼图等操作：



它还可以调节图片亮度：

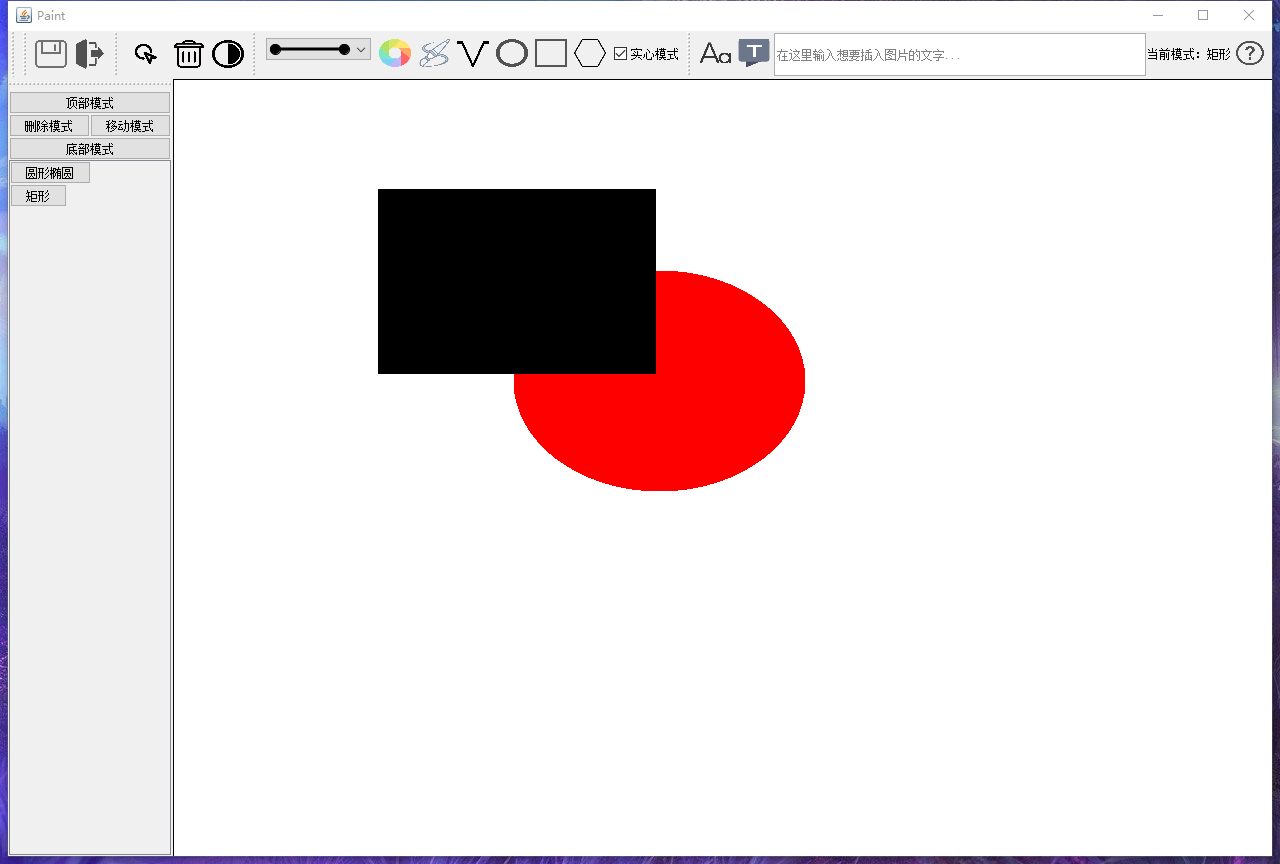


还可以使用橡皮擦：

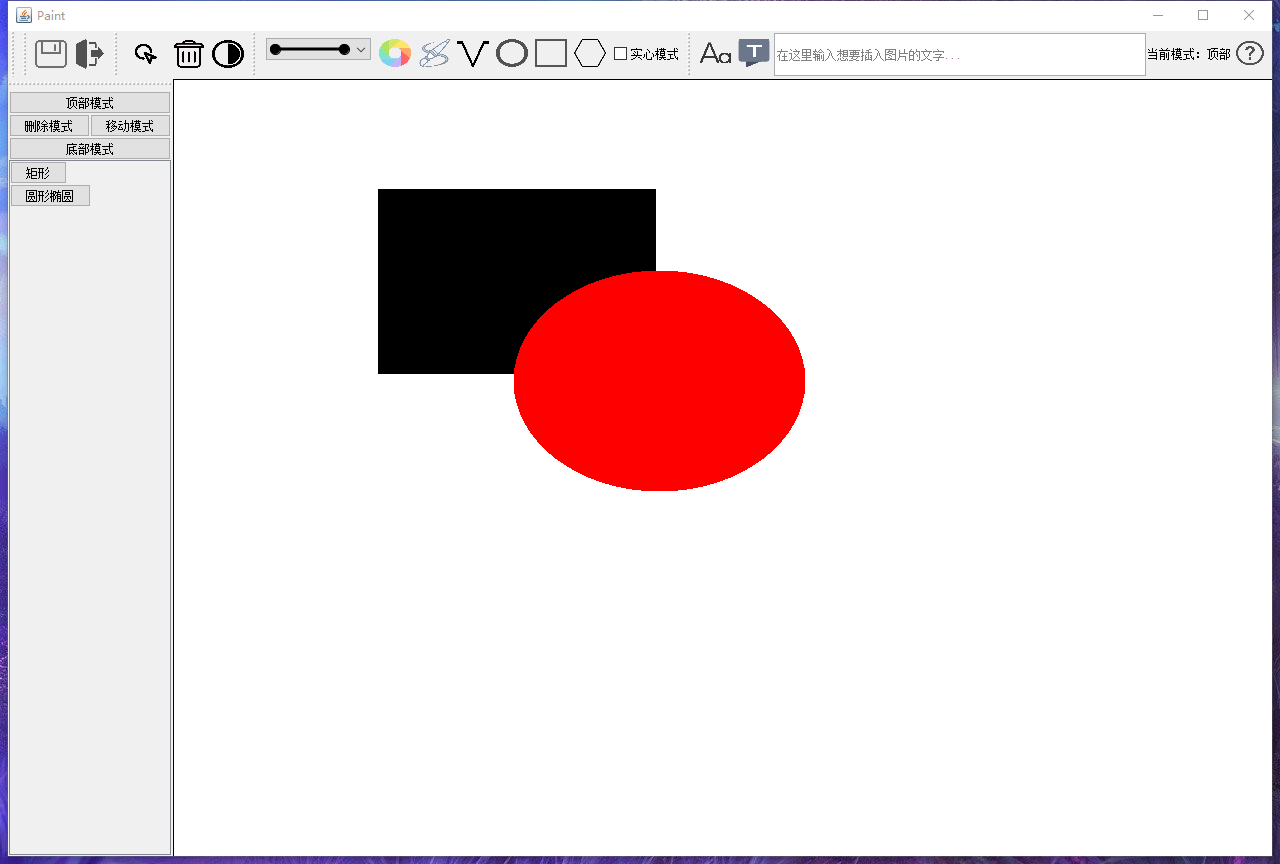


可以进行图层变更：

例如，先画一个红色椭圆，在矩形上画一个黑色矩形：



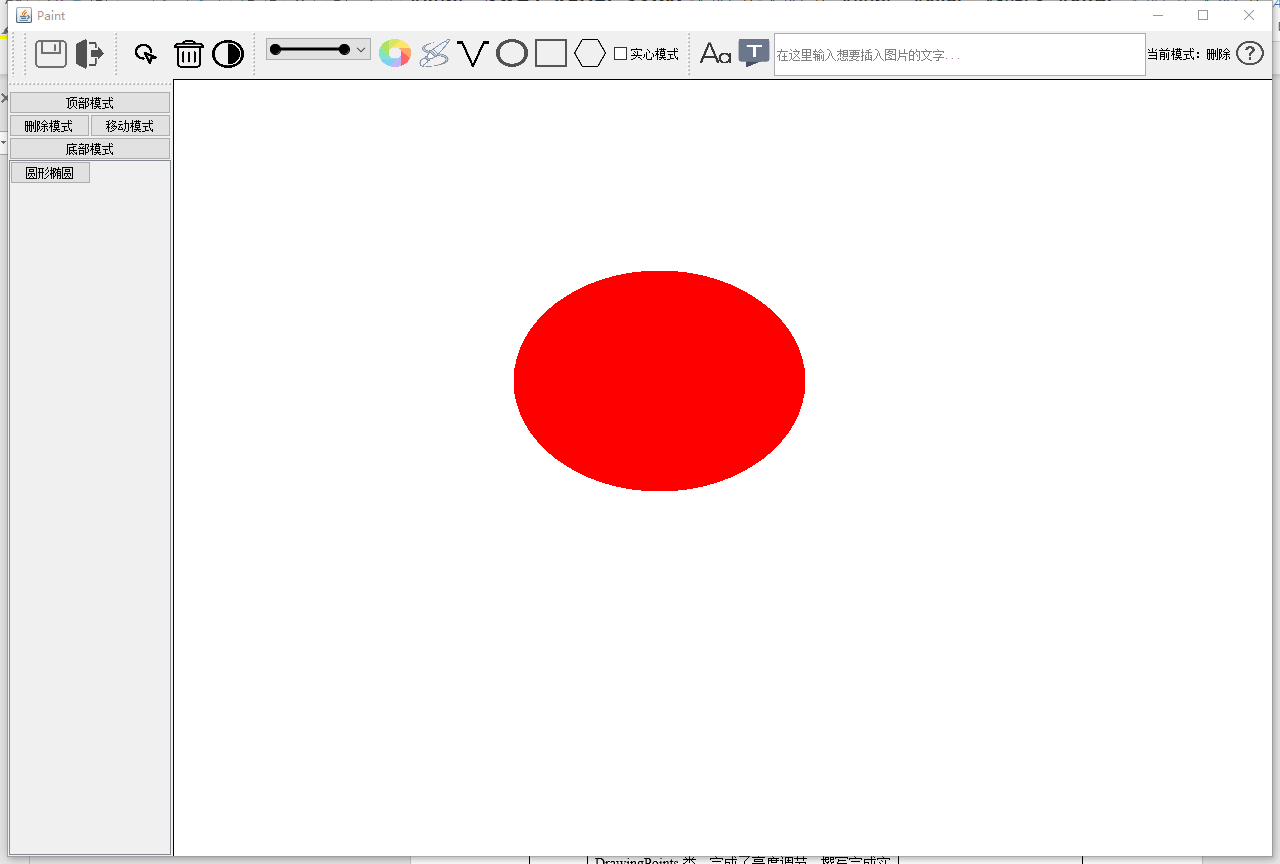
然后点击顶部模式，再点击左侧历史区域的圆形椭圆按钮：



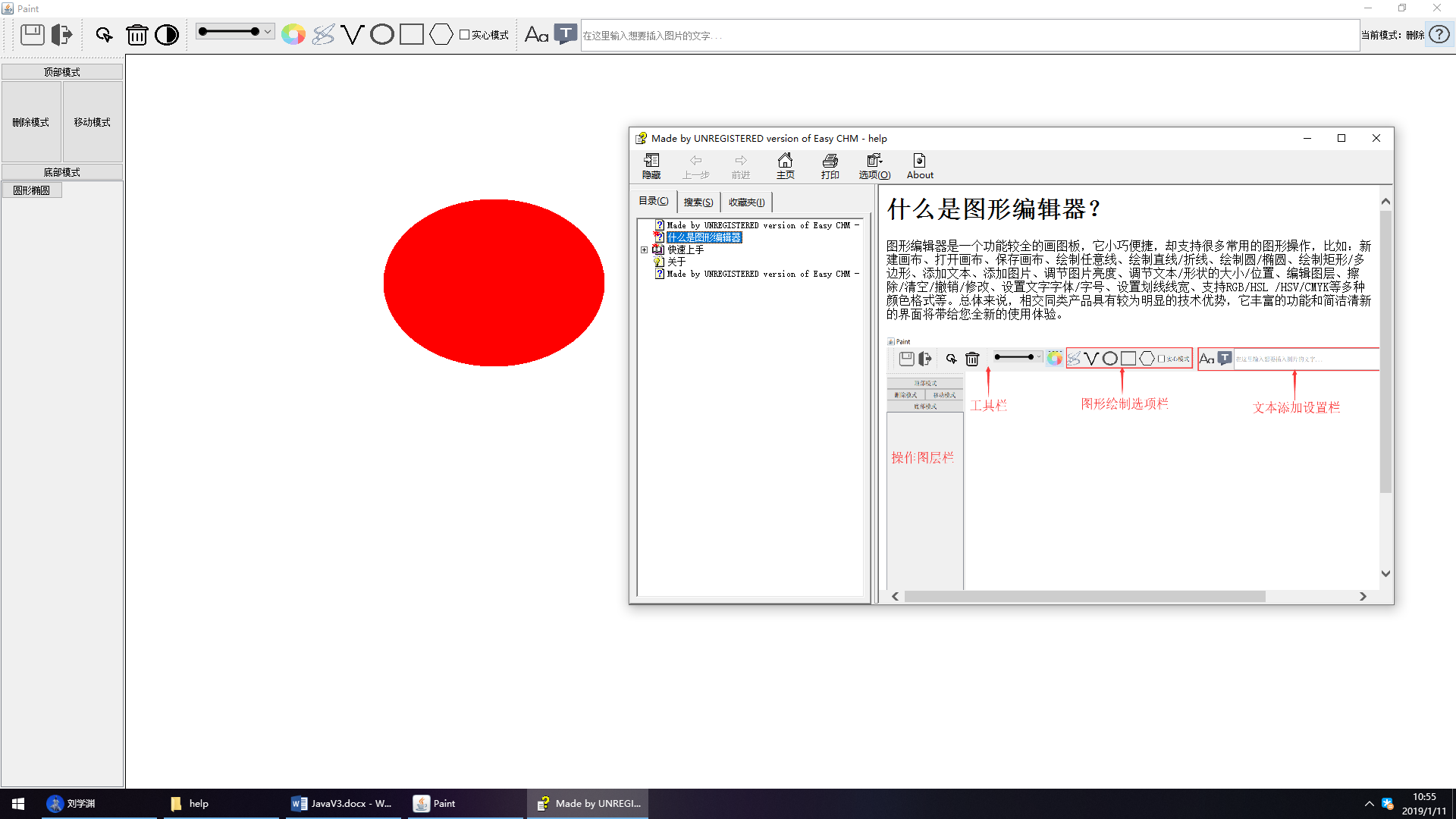
图层变更完成。

撤销操作是根据历史完成的，例如，去掉上图的矩形：

点击删除模式，点击矩形：

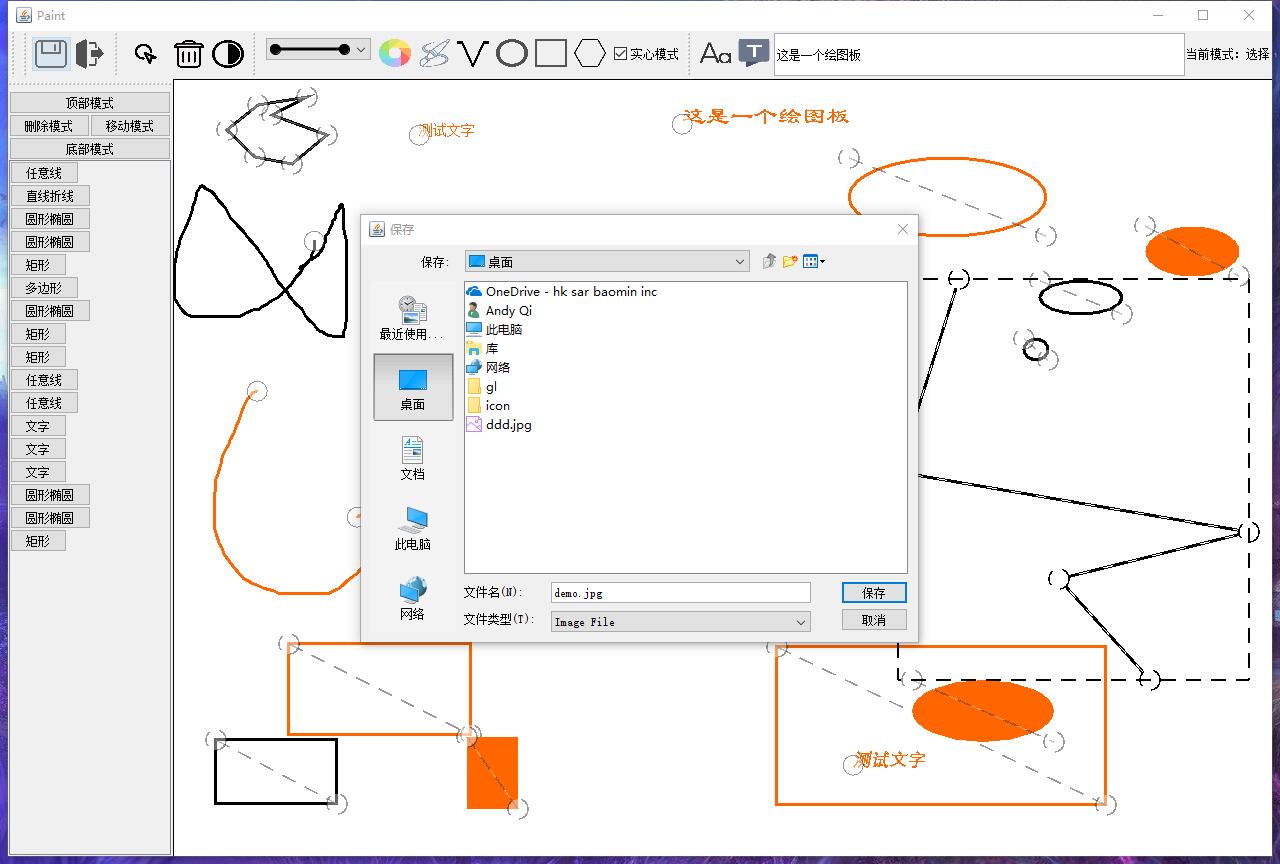


最后展示帮助：

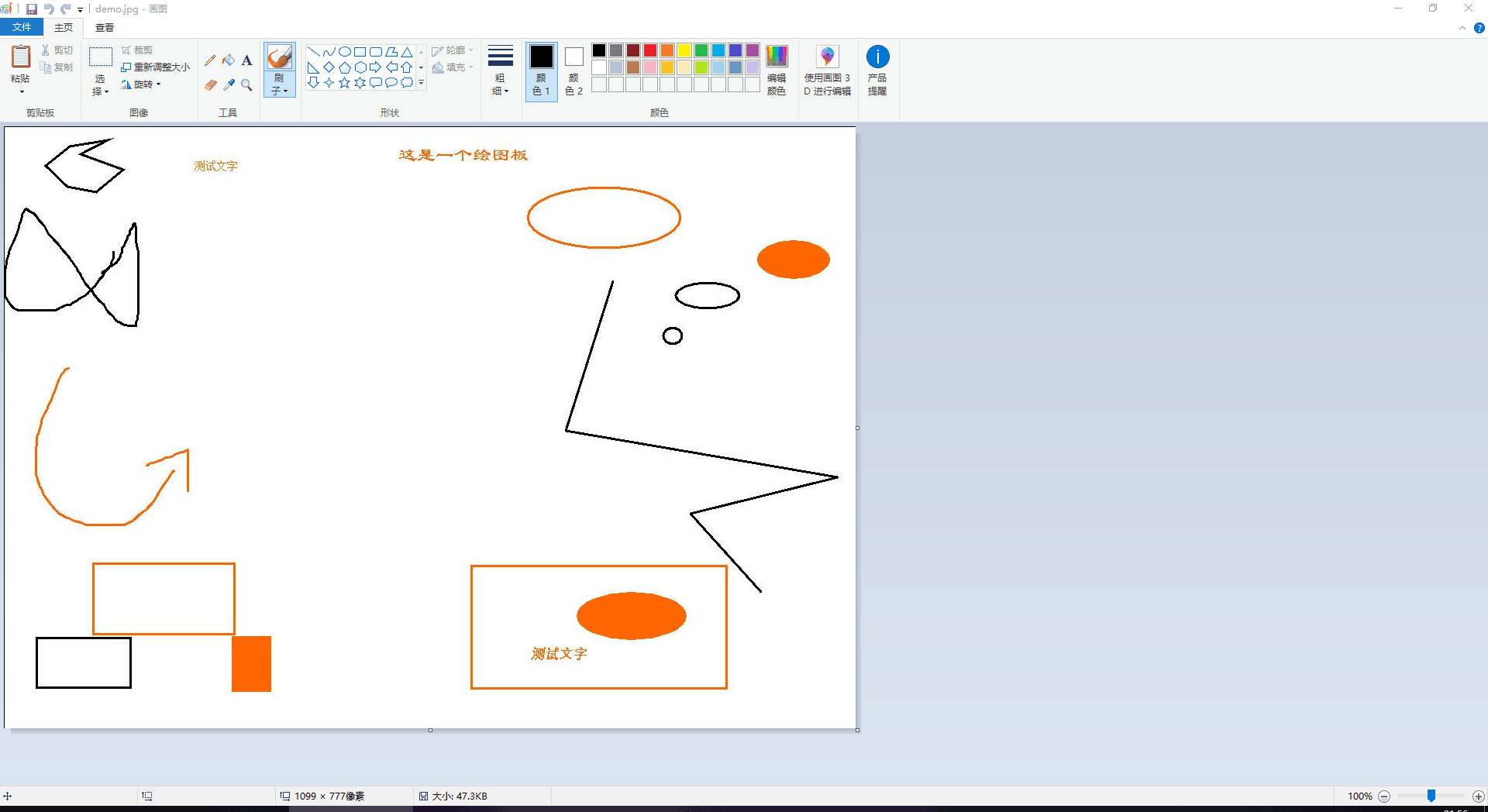


7、保存文件，并再次打开文件

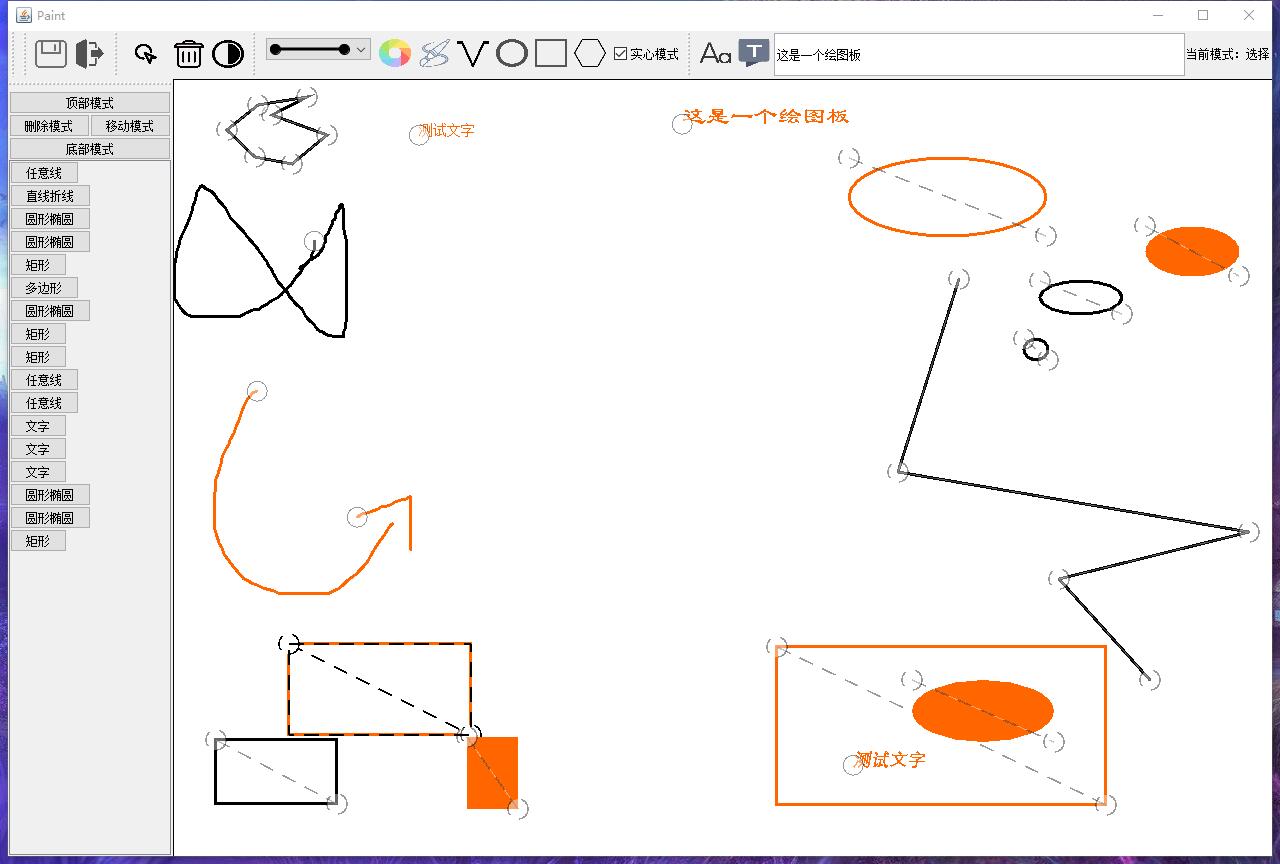
默认的文件后缀名为pnt/jpg/png等



使用 Windows画图 打开刚刚保存的文件：



再次打开编辑器，并打开文件，打开的文件依旧可以进行编辑



**七、任务分工与联系方式：**

|  |  |  |
| --- | --- | --- |
| 姓名 | 完成内容 | 联系方式（手机号） |
|  | 编写了DrawingItem类的派生类中的DrawingPolygon类和DrawingText类，完成DrawingBoard类，编写了BoardSettings类，编写了DrawingItem类的派生类中的DrawingLines类、DrawingPoints类，实现了亮度调节模块，编写了CHM帮助文档，撰写完成实验报告。 |  |
|  | 编写了ListIO类、HistoryButton类、SerializableStrokel类，编写了抽象类DrawingItem、完成了其派生类中的DrawingImage类、ResizePoint类、DrawingShape类，编写StrokeLibrary类，完成BoardMouseListener类，以及MainForm类函数，完成调试测试，撰写完成实验报告。 |  |

### 7.1 核心绘图代码

#### DrawingImage类（绘制图像）

package com.glede.paint;

import javax.imageio.ImageIO;

import javax.swing.\*;

import java.awt.\*;

import java.awt.image.BufferedImage;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.Vector;

/\*\*

\* Created by Glede on 1/10/2019

\*/

public class DrawingImage extends DrawingItem {

private transient Image \_image;

private transient JFrame \_imgObserver;

private Rectangle \_pos;

public DrawingImage(Image image,Rectangle pos,JFrame imgObserver){

this(image,pos,imgObserver,false);

}

public DrawingImage(Image image,Rectangle pos,JFrame imgObserver,boolean isPreview){

super(Type.图像,isPreview);

\_image=image;

\_pos=pos;

\_imgObserver=imgObserver;

initResizePoint();

}

@Override

public void reposition(Point pos) {

\_pos.x=pos.x;

\_pos.y=pos.y;

}

public void draw(Graphics g){

Graphics2D graphics2D=(Graphics2D)g;

graphics2D.drawImage(\_image,\_pos.x,\_pos.y,\_pos.width,\_pos.height,\_imgObserver);

}

@Override

public Rectangle getBounds() {

return \_pos;

}

@Override

public boolean contains(Point point) {

return \_pos.contains(point);

}

@Override

protected Vector<Point> getResizePoints() {

Vector<Point> points=new Vector<>();

points.add(new Point(\_pos.x,\_pos.y));

points.add(new Point(\_pos.x+\_pos.width,\_pos.y+\_pos.height));

return points;

}

@Override

public void resize(int resizePointRank, Point posTo) {

switch (resizePointRank){

case 0:

\_pos.x=posTo.x;

\_pos.y=posTo.y;

break;

case 1:

int x=Math.min(\_pos.x,posTo.x),y=Math.min(\_pos.y,posTo.y),

disX=Math.abs(posTo.x-\_pos.x),disY=Math.abs(posTo.y-\_pos.y);

\_pos=new Rectangle(x,y,disX,disY);

}

resizePoint.reposition(null);

}

@Override

public DrawingItem createPreview() {

return new DrawingShape(selectedColor,

new Rectangle(\_pos.x,\_pos.y,\_pos.width,\_pos.height),false,new BasicStroke(6f),true);

}

private void writeObject(ObjectOutputStream out) throws IOException {

out.defaultWriteObject();

ImageIO.write((BufferedImage)\_image, "png", out); // png

}

private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {

in.defaultReadObject();

\_image=ImageIO.read(in);

\_imgObserver=Entry.getMainFormInstance();

}

}

#### DrawingLines类（绘制直线和折线）

package com.glede.paint;

//import external.BetterBasicStroke;

import java.awt.\*;

import java.awt.geom.Line2D;

import java.util.Vector;

/\*\*

\* Created by Glede on 1/10/2019

\*/

public class DrawingLines extends DrawingItem {

private Color \_color;

private Vector<Point> \_points;

private Stroke \_stroke;

public DrawingLines(Color color,Vector<Point> points,Stroke stroke){

this(color,points,stroke,false);

}

public DrawingLines(Color color,Vector<Point> points,Stroke stroke,boolean isPreview){

super(Type.形状,isPreview);

\_color=color;

if (isPreview)

\_points=points;

else

\_points=(Vector<Point>) points.clone();

\_stroke= stroke;

initResizePoint();

}

@Override

public void resize(int resizePointRank, Point posTo) {

\_points.elementAt(resizePointRank).x=posTo.x;

\_points.elementAt(resizePointRank).y=posTo.y;

resizePoint.reposition(null);

}

public void reposition(Point pos) {

Point delta=new Point(pos.x-\_points.elementAt(0).x,pos.y-\_points.elementAt(0).y);

for (Point x:\_points){

x.x+=delta.x;

x.y+=delta.y;

}

}

@Override

protected Vector<Point> getResizePoints() {

return (Vector<Point>) \_points.clone();

}

@Override

public Rectangle getBounds() {

int minX = Integer.MAX\_VALUE, minY = Integer.MAX\_VALUE,

maxX = 0, maxY = 0;

for (Point p : \_points) {

maxX = Math.max(maxX, p.x);

maxY = Math.max(maxY, p.y);

minX = Math.min(minX, p.x);

minY = Math.min(minY, p.y);

}

return new Rectangle(minX, minY, maxX - minX, maxY - minY);

}

@Override

public boolean contains(Point point) {

for (int i = 1; i < \_points.size(); i++) {

Line2D.Double lineNow = new Line2D.Double(

\_points.elementAt(i - 1), \_points.elementAt(i));

if (lineNow.contains(point)) return true;

}

return false;

}

public void draw(Graphics g){

Graphics2D graphics2D=(Graphics2D)g;

graphics2D.setColor(\_color);

graphics2D.setStroke(\_stroke);

for(int i=1;i<\_points.size();i++){

Point x=\_points.elementAt(i),y=\_points.elementAt(i-1);

graphics2D.drawLine(x.x,x.y,y.x,y.y);

}

// for(Point x:\_points){

// graphics2D.drawLine(x.x,x.y,x.x,x.y);

// }

}

@Override

public DrawingItem createPreview() {

return new DrawingLines(selectedColor,\_points,\_stroke,true);

}

}

#### DrawingPoints类（绘制任意线）

package com.glede.paint;

import java.awt.\*;

import java.util.Vector;

/\*\*

\* Created by Glede on 1/10/2019

\*/

public class DrawingPoints extends DrawingItem {

private Color \_color;

private Vector<Point> \_points;

private Stroke \_stroke;

public DrawingPoints(Color color,Vector<Point> points,Stroke stroke){

this(color,points,stroke,false);

}

public DrawingPoints(Color color,Vector<Point> points,Stroke stroke,boolean isPreview){

super(Type.任意线,isPreview);

\_color=color;

\_points=(Vector<Point>) points.clone();

\_stroke=stroke;

initResizePoint();

}

@Override

public void resize(int resizePointRank, Point posTo) {

reposition(posTo);

resizePoint.reposition(null);

}

@Override

public boolean contains(Point point) {

for (Point p : \_points) {

if (p.equals(point)) return true;

}

return false;

}

@Override

public Rectangle getBounds() {

int minX = Integer.MAX\_VALUE, minY = Integer.MAX\_VALUE,

maxX = 0, maxY = 0;

for (Point p : \_points) {

maxX = Math.max(maxX, p.x);

maxY = Math.max(maxY, p.y);

minX = Math.min(minX, p.x);

minY = Math.min(minY, p.y);

}

return new Rectangle(minX, minY, maxX - minX, maxY - minY);

}

@Override

protected Vector<Point> getResizePoints() {

Vector<Point> points=new Vector<>();

if (\_points.size()!=0)points.add(\_points.elementAt(0));

return points;

}

public void reposition(Point pos) {

Point delta=new Point(pos.x-\_points.elementAt(0).x,pos.y-\_points.elementAt(0).y);

for (Point x:\_points){

x.x+=delta.x;

x.y+=delta.y;

}

}

public void draw(Graphics g){

Graphics2D graphics2D=(Graphics2D)g;

graphics2D.setColor(\_color);

graphics2D.setStroke(\_stroke);

for(int i=1;i<\_points.size();i++){

Point x=\_points.elementAt(i),y=\_points.elementAt(i-1);

graphics2D.drawLine(x.x,x.y,y.x,y.y);

}

// for(Point x:\_points){

// graphics2D.drawLine(x.x,x.y,x.x,x.y);

// }

}

@Override

public DrawingItem createPreview() {

return new DrawingPoints(selectedColor,\_points,\_stroke,true);

}

}

#### DrawingPolygon类（绘制圆形和椭圆）

package com.glede.paint;

import java.awt.\*;

import java.util.Vector;

/\*\*

\* Created by Glede on 1/10/2019

\*/

public class DrawingPolygon extends DrawingItem {

private Color \_color;

private Polygon \_shape;

private Stroke \_stroke;

private boolean \_fill;

boolean \_isPolygon;

public DrawingPolygon(Color color,Polygon shape,boolean fill,Stroke stroke){

this(color,shape,fill,stroke,false);

}

public DrawingPolygon(Color color,Polygon shape,boolean fill,Stroke stroke,boolean isPreview){

super(Type.多边形,isPreview);

\_color=color;

\_shape=shape;

\_fill=fill;

\_stroke=fill?null:stroke;

initResizePoint();

}

public void reposition(Point pos) {

Point delta=new Point(pos.x-\_shape.xpoints[0],pos.y-\_shape.ypoints[0]);

for(int i=0;i<\_shape.npoints;i++){

\_shape.xpoints[i]+=delta.x;

\_shape.ypoints[i]+=delta.y;

}

}

public void draw(Graphics g){

Graphics2D graphics2D=(Graphics2D)g;

graphics2D.setColor(\_color);

if(\_fill) {

graphics2D.fillPolygon(\_shape);

}else {

graphics2D.setStroke(\_stroke);

graphics2D.drawPolygon(\_shape);

}

}

@Override

public void resize(int resizePointRank, Point posTo) {

\_shape.xpoints[resizePointRank]=posTo.x;

\_shape.ypoints[resizePointRank]=posTo.y;

resizePoint.reposition(null);

}

@Override

public boolean contains(Point point) {

return \_shape.contains(point);

}

@Override

public Rectangle getBounds() {

\_shape.invalidate();

return \_shape.getBounds();

}

@Override

protected Vector<Point> getResizePoints() {

Vector<Point> points=new Vector<>();

for(int i=0;i<\_shape.npoints;i++){

points.add(new Point(\_shape.xpoints[i],\_shape.ypoints[i]));

}

return points;

}

@Override

public DrawingItem createPreview() {

return new DrawingPolygon(selectedColor,\_shape,\_fill,\_stroke,true);

}

}

#### DrawingShape类（绘制形状基类）

package com.glede.paint;

import java.awt.\*;

import java.awt.geom.Ellipse2D;

import java.util.Vector;

/\*\*

\* Created by Glede on 1/10/2019

\*/

public class DrawingShape extends DrawingItem {

private Color \_color;

private Shape \_shape;

private Stroke \_stroke;

private boolean \_fill;

public DrawingShape(Color color,Shape shape,boolean fill,Stroke stroke,boolean isPreview){

super(Type.形状,isPreview);

init(color,shape,fill,stroke);

}

public DrawingShape(Color color,Shape shape,boolean fill,Stroke stroke){

this(color,shape,fill,stroke,false);

}

private void init(Color color,Shape shape,boolean fill,Stroke stroke){

\_color=color;

\_shape=shape;

\_fill=fill;

\_stroke=fill?null:stroke;

initResizePoint();

if(\_shape instanceof Rectangle){

setType(Type.矩形);

}

if(\_shape instanceof Ellipse2D.Double){

setType(Type.圆形椭圆);

}

if(\_shape instanceof Ellipse2D.Float){

setType(Type.圆形椭圆);

}

}

public void reposition(Point pos){

if(\_shape instanceof Rectangle){

Rectangle rectangle=(Rectangle)\_shape;

rectangle.x= pos.x;

rectangle.y=pos.y;

\_shape=rectangle;

}

if(\_shape instanceof Ellipse2D.Double){

Ellipse2D.Double oval=(Ellipse2D.Double) \_shape;

oval.x= pos.x;

oval.y=pos.y;

\_shape=oval;

}

if(\_shape instanceof Ellipse2D.Float){

Ellipse2D.Float oval=(Ellipse2D.Float) \_shape;

oval.x= pos.x;

oval.y=pos.y;

\_shape=oval;

}

}

@Override

public Rectangle getBounds() {

return \_shape.getBounds();

}

@Override

public boolean contains(Point point) {

return \_shape.contains(point);

}

@Override

protected Vector<Point> getResizePoints() {

Vector<Point> points=new Vector<>();

if(\_shape instanceof Rectangle){

Rectangle rectangle=(Rectangle)\_shape;

points.add(new Point(rectangle.x,rectangle.y));

points.add(new Point(rectangle.x+rectangle.width,rectangle.y+rectangle.height));

}

if(\_shape instanceof Ellipse2D.Double){

Ellipse2D.Double oval=(Ellipse2D.Double) \_shape;

points.add(new Point((int)oval.x,(int)oval.y));

points.add(new Point((int)(oval.x+oval.width),(int) (oval.y+oval.height)));

}

if(\_shape instanceof Ellipse2D.Float){

Ellipse2D.Float oval=(Ellipse2D.Float) \_shape;

points.add(new Point((int)oval.x,(int)oval.y));

points.add(new Point((int)(oval.x+oval.width),(int) (oval.y+oval.height)));

}

return points;

}

@Override

public void resize(int resizePointRank, Point posTo) {

switch (resizePointRank){

case 0:

if(\_shape instanceof Rectangle){

Rectangle rectangle=(Rectangle)\_shape;

rectangle.x=posTo.x;

rectangle.y=posTo.y;

}

if(\_shape instanceof Ellipse2D.Double){

Ellipse2D.Double oval=(Ellipse2D.Double) \_shape;

oval.x=posTo.x;

oval.y=posTo.y;

}

if(\_shape instanceof Ellipse2D.Float){

Ellipse2D.Float oval=(Ellipse2D.Float) \_shape;

oval.x=posTo.x;

oval.y=posTo.y;

}

break;

case 1:

if(\_shape instanceof Rectangle){

Rectangle rectangle=(Rectangle)\_shape;

int x=Math.min(rectangle.x,posTo.x),y=Math.min(rectangle.y,posTo.y),

disX=Math.abs(posTo.x-rectangle.x),disY=Math.abs(posTo.y-rectangle.y);

\_shape=new Rectangle(x,y,disX,disY);

}

if(\_shape instanceof Ellipse2D.Double){

Ellipse2D.Double oval=(Ellipse2D.Double) \_shape;

Double x=Math.min(oval.x,posTo.x),y=Math.min(oval.y,posTo.y),

disX=Math.abs(posTo.x-oval.x),disY=Math.abs(posTo.y-oval.y);

\_shape=new Ellipse2D.Double(x,y,disX,disY);

}

if(\_shape instanceof Ellipse2D.Float){

Ellipse2D.Float oval=(Ellipse2D.Float) \_shape;

Float x=Math.min(oval.x,posTo.x),y=Math.min(oval.y,posTo.y),

disX=Math.abs(posTo.x-oval.x),disY=Math.abs(posTo.y-oval.y);

\_shape=new Ellipse2D.Float(x,y,disX,disY);

}

break;

}

resizePoint.reposition(null);

}

public void draw(Graphics g){

// getResizePoint().setShow(true);

Graphics2D graphics2D=(Graphics2D)g;

graphics2D.setColor(\_color);

if(\_fill) {

graphics2D.fill(\_shape);

}else {

graphics2D.setStroke(\_stroke);

graphics2D.draw(\_shape);

}

}

@Override

public DrawingItem createPreview() {

return new DrawingShape(selectedColor,\_shape,\_fill,\_stroke,true);

}

}

#### DrawingText类（绘制文字）

package com.glede.paint;

import java.awt.\*;

import java.util.Vector;

/\*\*

\* Created by Glede on 1/10/2019

\*/

public class DrawingText extends DrawingItem {

private String text;

private Point pos;

private Font font;

private Color color;

public DrawingText(String text, Point pos,Font font,Color color){

this(text,pos,font,color,false);

}

public DrawingText(String text, Point pos,Font font,Color color,boolean isPreview){

super(Type.文字,isPreview);

this.text=text;

this.font=font;

this.pos=pos;

this.color=color;

initResizePoint();

}

public void draw(Graphics g){

Graphics2D graphics2D=(Graphics2D )g;

graphics2D.setFont(font);

graphics2D.setColor(color);

graphics2D.drawString(text,pos.x,pos.y);

}

@Override

public boolean contains(Point point) {

return false;

}

@Override

public Rectangle getBounds() {

int sizeY = font.getSize();

int sizeX = sizeY \* text.length();

return new Rectangle(pos.x, pos.y - sizeY, sizeX, sizeY);

}

@Override

protected Vector<Point> getResizePoints() {

Vector<Point> points=new Vector<>();

points.add(pos);

return points;

}

@Override

public void resize(int resizePointRank, Point posTo) {

if(resizePointRank==0){

reposition(posTo);

}

resizePoint.reposition(null);

}

public void reposition(Point pos) {

this.pos.x=pos.x;

this.pos.y=pos.y;

}

@Override

public DrawingItem createPreview() {

return new DrawingText(text,pos,font,selectedColor,true);

}

}

### 7.2 GUI和I/O代码

#### ListIO类（负责打开保存等I/O操作）

package com.glede.paint;

import javax.swing.\*;

import java.io.\*;

import java.util.LinkedList;

public class ListIO {

public static LinkedList<DrawingItem> readList(JFrame mainForm, File file) throws IOException {

try {

// String fileName = file.getName();

// String tail = fileName.substring(fileName.lastIndexOf(".")+1);

// if (tail != "pnt") throw new IOException("File type error");

FileInputStream fis = new FileInputStream(file);

ObjectInputStream ois = new ObjectInputStream(fis);

LinkedList<DrawingItem> list = (LinkedList<DrawingItem>) ois.readObject();

ois.close();

fis.close();

return list;

} catch (IOException e) {

JOptionPane.showMessageDialog(mainForm, "Load file Error:" + e.getMessage());

e.printStackTrace();

} catch (ClassNotFoundException e) {

JOptionPane.showMessageDialog(mainForm, "Class not found");

e.printStackTrace();

}

return new LinkedList<>();

}

public static void saveList(JFrame mainForm, LinkedList<DrawingItem> target, File file) {

try {

String fileName = file.getName();

// String tail = fileName.substring(fileName.lastIndexOf(".")+1);

// if (tail != "pnt") throw new IOException("File type error");

if (!file.exists())

file.createNewFile();

} catch(IOException e) {

JOptionPane.showMessageDialog(mainForm,"Save file failed:"+e.getMessage());

}

try {

FileOutputStream fos = new FileOutputStream(file);

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(target);

oos.flush();

oos.close();

fos.close();

} catch(FileNotFoundException e) {

JOptionPane.showMessageDialog(mainForm,"File not found");

e.printStackTrace();

} catch(IOException e) {

JOptionPane.showMessageDialog(mainForm,"Saving error!:"+e.getMessage());

e.printStackTrace();

}

}

}

#### MainForm类（GUI实现）

package com.glede.paint;

//import com.sun.codemodel.internal.JOp;

import external.JFontChooser;

import external.StrokeChooserPanel;

import external.StrokeSample;

import javax.imageio.ImageIO;

import javax.swing.\*;

import javax.swing.event.DocumentEvent;

import javax.swing.event.DocumentListener;

import javax.swing.filechooser.FileNameExtensionFilter;

import org.omg.CORBA.PRIVATE\_MEMBER;

import com.sun.org.apache.xml.internal.security.utils.HelperNodeList;

import java.awt.\*;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.awt.event.FocusEvent;

import java.awt.event.FocusListener;

import java.awt.Desktop;

/\*\*

\* Created by Glede on 1/10/2019

\*/

class JTextFieldHintListener implements FocusListener {

    private String hintText;

    private JTextField textField;

    public JTextFieldHintListener(JTextField jTextField,String hintText) {

        this.textField = jTextField;

        this.hintText = hintText;

        jTextField.setText(hintText); //默认直接显示

        jTextField.setForeground(Color.GRAY);

    }

    @Override

    public void focusGained(FocusEvent e) {

        //获取焦点时，清空提示内容

        String temp = textField.getText();

        if(temp.equals(hintText)) {

            textField.setText("");

            textField.setForeground(Color.BLACK);

        }

    }

    @Override

    public void focusLost(FocusEvent e) {

        //失去焦点时，没有输入内容，显示提示内容

        String temp = textField.getText();

        if(temp.equals("")) {

            textField.setForeground(Color.GRAY);

            textField.setText(hintText);

        }

    }

}

public class MainForm extends JFrame {

    private int BUTTONWIDTH = 32, BUTTONHEIGHT = 28;

private DrawingBoard board;

private JToolBar select;

private JPanel mainPanel, history, historyMain;

private BoardSettings settings;

private JButton buttonOpenImg, buttonDelete,

buttonPoints, buttonPolygon, buttonOval,

buttonRect, buttonText, buttonLines,buttonContect,buttonLightAdjust,buttonHelp;

private JButton buttonColor, buttonFont, buttonClear,

buttonMove, buttonTop, buttonBottom, buttonSave, buttonSelect;

private JCheckBox checkBoxFill;

private JTextField textImput;

private StrokeChooserPanel strokeChooserPanel;

private JLabel labelModeNow;

public JPanel getHistory() {

return history;

}

public DrawingBoard getBoard() {

return board;

}

public StrokeChooserPanel getStrokeChooserPanel() {

return strokeChooserPanel;

}

public JButton changeIconSize(JButton button,String url,int width,int height,String tip){

button.setBounds(0,0,width,height);

ImageIcon buttonImg=new ImageIcon(url);

//改变图片的大小

Image temp=buttonImg.getImage().getScaledInstance(button.getWidth(), button.getHeight(), buttonImg.getImage().SCALE\_DEFAULT);

button=new JButton(new ImageIcon(temp));

button.setToolTipText(tip); //提示

return button;

}

private void initChoose() {

StrokeSample[] samples = StrokeLibrary.strokes;

strokeChooserPanel = new StrokeChooserPanel(samples[2], samples);

strokeChooserPanel.addSelectorListener(e -> {

settings.setStroke(strokeChooserPanel.getSelectedStroke());

});

}

private boolean asImageFile(File file) {

String name = file.getName(), extension;

int extBegin = name.lastIndexOf('.');

// boolean asImageFile;

if (extBegin == -1) return false;

else if (extBegin == name.length() - 1) return false;

else {

extBegin++;

extension = name.substring(extBegin);

String[] exts = {"jpg", "png", "gif", "jpeg"};

for (String ext : exts) {

if (ext.equals(extension)) return true;

}

}

return false;

}

private void initButtons() {

buttonSelect = new JButton("调整");

buttonSelect = changeIconSize(buttonSelect,"icon//resize.png",40,36,"调整位置");

buttonSelect.addActionListener(e -> {

settings.setType(BoardSettings.Type.选择);

settings.nextResizePoint(null);

});

buttonLines = new JButton("折线");

buttonLines = changeIconSize(buttonSelect,"icon//line2.png",BUTTONWIDTH,BUTTONHEIGHT,"使用鼠标左键逐个点按以绘制直线或者折线");

buttonLines.addActionListener(e -> {

settings.setType(BoardSettings.Type.直线折线);

board.repaint();

});

buttonPoints = new JButton("钢笔");

buttonPoints = changeIconSize(buttonPoints,"icon//drawfree.png",BUTTONWIDTH,BUTTONHEIGHT,"按住鼠标左键自由绘制");

buttonPoints.addActionListener(e ->

settings.setType(BoardSettings.Type.任意线));

buttonOval = new JButton("椭圆");

buttonOval = changeIconSize(buttonOval,"icon//circle.png",BUTTONWIDTH,BUTTONHEIGHT,"使用鼠标左键绘制圆形或者椭圆");

buttonOval.addActionListener(e ->

settings.setType(BoardSettings.Type.圆形椭圆));

buttonPolygon = new JButton("多边形");

buttonPolygon = changeIconSize(buttonPolygon, "icon//polygon.png",BUTTONWIDTH, BUTTONHEIGHT, "使用鼠标左键逐个点按以绘制多边形");

buttonPolygon.addActionListener(e ->

settings.setType(BoardSettings.Type.多边形));

buttonOpenImg = new JButton("打开");

buttonOpenImg = changeIconSize(buttonOpenImg, "icon//open.png", 32,36, "打开文件");

buttonOpenImg.addActionListener(e -> {

JFileChooser fileChooser = new JFileChooser();

fileChooser.setDialogTitle("Choose Image");

fileChooser.setFileFilter(new FileNameExtensionFilter("Image Files", "gif", "png", "jpg", "bmp", "jpeg"));

fileChooser.setFileFilter(new FileNameExtensionFilter("Painting Files", "pnt"));

while (fileChooser.showOpenDialog(this) != JFileChooser.APPROVE\_OPTION) {

int confirmDialog = JOptionPane.showConfirmDialog(this, "Selection failed, continue?");

if (confirmDialog != 0) {

return;

}

}

File file = fileChooser.getSelectedFile();

try {

if (asImageFile(file)) {

settings.setType(BoardSettings.Type.图像);

settings.setImgNow(ImageIO.read(file));

} else {

board.readList(ListIO.readList(this, file));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(this, "Read file failed");

}

settings.clearPoints();

});

buttonRect = new JButton("Rect");

buttonRect = changeIconSize(buttonRect, "icon//rect.png", BUTTONWIDTH, BUTTONHEIGHT, "使用鼠标左键绘制矩形");

buttonRect.addActionListener(e -> {

settings.setType(BoardSettings.Type.矩形);

});

buttonText = new JButton("Text");

buttonText = changeIconSize(buttonText, "icon//text.png", BUTTONWIDTH, BUTTONHEIGHT, "使用鼠标左键点击位置插入文本内容");

buttonText.addActionListener(e -> {

settings.setType(BoardSettings.Type.文字);

});

buttonContect = new JButton("contect");

buttonContect = changeIconSize(buttonContect, "icon//textcon.png", BUTTONWIDTH, BUTTONHEIGHT, "更改文本内容");

buttonContect.addActionListener(e ->{

    JOptionPane.showInputDialog(buttonContect,"Message");

});

buttonColor = new JButton("Color");

buttonColor = changeIconSize(buttonColor, "icon//color.png", BUTTONWIDTH,BUTTONHEIGHT, "更换颜色");

buttonColor.addActionListener(e -> {

Color color = JColorChooser.showDialog(null, "Choose Color", Color.BLACK);

settings.setColor(color);

});

buttonFont = new JButton("Font");

buttonFont = changeIconSize(buttonFont, "icon//font.png", BUTTONWIDTH, BUTTONHEIGHT, "更换字体");

buttonFont.addActionListener(e -> {

JFontChooser fontChooser = new JFontChooser();

fontChooser.showDialog(this);

settings.setFont(fontChooser.getSelectedFont());

});

buttonClear = new JButton("Clear");

buttonClear = changeIconSize(buttonClear, "icon//clear.png", BUTTONWIDTH, BUTTONHEIGHT, "清空所有更改");

buttonClear.addActionListener(e -> {

if (JOptionPane.showConfirmDialog(settings.getMainFrame(),

"Clear the paint board and discard all changes?") == 0) {

board.clearBoard();

}

});

textImput = new JTextField(15);

textImput.addFocusListener(new JTextFieldHintListener(textImput, "在这里输入想要插入图片的文字..."));

textImput.getDocument().addDocumentListener(new DocumentListener() {

@Override

public void insertUpdate(DocumentEvent e) {

setText();

}

@Override

public void removeUpdate(DocumentEvent e) {

setText();

}

@Override

public void changedUpdate(DocumentEvent e) {

setText();

}

private void setText() {

settings.setText(textImput.getText());

}

});

buttonLightAdjust = new JButton();

buttonLightAdjust = changeIconSize(buttonLightAdjust, "icon//light.png", BUTTONWIDTH, BUTTONHEIGHT, "调节亮度");

buttonLightAdjust.addActionListener(e ->{

    BufferedImage image = board.getImage();

    File file = new File("cache.jpg");

    try {

    ImageIO.write(image, "jpg", file);

    }catch (IOException ioe) {

                // TODO: handle exception

        JOptionPane.showMessageDialog(this, "Save Failed :" + ioe.getMessage());

            }

    LightImage li = new LightImage();

    if(li.getClosed() == 1)

    {

        try {

settings.setType(BoardSettings.Type.图像);

settings.setImgNow(ImageIO.read(file));

    }catch (IOException ioe) {

                // TODO: handle exception

        JOptionPane.showMessageDialog(this, "Save Failed :" + ioe.getMessage());

            }

    }

}

);

buttonSave = new JButton("保存");

buttonSave = changeIconSize(buttonSave, "icon//save.png", BUTTONWIDTH, BUTTONHEIGHT, "保存文件");

buttonSave.addActionListener(e -> {

BufferedImage image = board.getImage();

JFileChooser fileChooser = new JFileChooser();

fileChooser.addChoosableFileFilter

(new FileNameExtensionFilter("Image File", "jpg", "png", "gif", "jpeg"));

fileChooser.setFileFilter

(new FileNameExtensionFilter("Paint Save File", "pnt"));

if (fileChooser.showSaveDialog(this) == JFileChooser.APPROVE\_OPTION) {

File file = fileChooser.getSelectedFile();

// save to file

try {

String name = file.getName();

String path = file.getAbsolutePath();

String filter = fileChooser.getFileFilter().getDescription();

if (filter.contains("Image")) {

if (name.lastIndexOf('.') == -1) {

file = new File(path + ".jpg");

}

ImageIO.write(image, "jpg", file);

} else {

if (!name.substring(name.lastIndexOf('.') + 1).equals("pnt")) {

file = new File(path + ".pnt");

}

ListIO.saveList(this, board.getItemsList(), file);

}

} catch (IOException ioe) {

JOptionPane.showMessageDialog(this, "Save Failed :" + ioe.getMessage());

}

}

});

buttonHelp = new JButton();

buttonHelp = changeIconSize(buttonHelp, "icon//help.png", BUTTONWIDTH, BUTTONHEIGHT, "帮助");

buttonHelp.addActionListener(e ->{

    try {

    Desktop.getDesktop().open(new File("help.chm"));

    }catch (IOException ioe) {

                // TODO: handle exception

            }

});

checkBoxFill = new JCheckBox("实心模式");

checkBoxFill.addActionListener(e -> {

settings.setFill(checkBoxFill.isSelected());

});

JToolBar barSaveOpen=new JToolBar();

JToolBar barChoose=new JToolBar();

JToolBar barShapes=new JToolBar();

JToolBar barText=new JToolBar();

barSaveOpen.add(buttonSave);

barSaveOpen.add(buttonOpenImg);

barChoose.add(buttonSelect);

barChoose.add(buttonClear);

barChoose.add(buttonLightAdjust);

barShapes.add(strokeChooserPanel);

barShapes.add(buttonColor);

barShapes.add(buttonPoints);

barShapes.add(buttonLines);

barShapes.add(buttonOval);

barShapes.add(buttonRect);

barShapes.add(buttonPolygon);

barShapes.add(checkBoxFill);

barText.add(buttonFont);

barText.add(buttonText);

barText.add(textImput);

// barText.add(buttonContect);

select.add(barSaveOpen);

select.add(barChoose);

select.add(barShapes);

select.add(barText);

select.add(labelModeNow);

select.add(buttonHelp);

}

private void initHistory() {

buttonTop = new JButton("顶部模式");

buttonTop.addActionListener(e -> {

if (settings.getType()== BoardSettings.Type.选择){

settings.getPointNow().getItem().getRelatedButton().goTop();

settings.getPointNow().getItem().getRelatedButton().reposition();

}

else {

settings.setType(BoardSettings.Type.顶部);

}

});

buttonBottom = new JButton("底部模式");

buttonBottom.addActionListener(e -> {

if (settings.getType() == BoardSettings.Type.选择) {

settings.getPointNow().getItem().getRelatedButton().goBottom();

settings.getPointNow().getItem().getRelatedButton().reposition();

} else {

settings.setType(BoardSettings.Type.底部);

}

});

buttonDelete = new JButton("删除模式");

buttonDelete.addActionListener(e -> {

if (settings.getType()== BoardSettings.Type.选择){

settings.getPointNow().getItem().getRelatedButton().delete();

settings.getPointNow().getItem().getRelatedButton().reposition();

}

else {

settings.setType(BoardSettings.Type.删除);

}

});

buttonMove = new JButton("移动模式");

buttonMove.addActionListener(e -> {

if (settings.getType() == BoardSettings.Type.选择) {

settings.getPointNow().getItem().getRelatedButton().move();

settings.getPointNow().getItem().getRelatedButton().reposition();

} else {

settings.setType(BoardSettings.Type.移动);

}

});

historyMain = new JPanel();

historyMain.setLayout(new BoxLayout(historyMain, BoxLayout.PAGE\_AXIS));

JPanel historyOptions = new JPanel();

historyOptions.setLayout(new BorderLayout());

historyOptions.add(buttonTop, BorderLayout.NORTH);

historyOptions.add(buttonBottom, BorderLayout.SOUTH);

historyOptions.add(buttonMove, BorderLayout.EAST);

historyOptions.add(buttonDelete, BorderLayout.WEST);

historyMain.add(historyOptions);

history = new JPanel();

JScrollPane historyScroll = new JScrollPane(history);

historyScroll.setBounds(0, 0, 0, 50);

history.setAutoscrolls(true);

historyScroll.setPreferredSize(new Dimension(50, 800));

history.setSize(new Dimension(10, 50));

history.setLayout(new BoxLayout(history, BoxLayout.PAGE\_AXIS));

historyMain.add(historyScroll);

}

public MainForm() {

try {

this.setUndecorated(false);

// UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

SwingUtilities.updateComponentTreeUI(this);

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "LOAD UI FAILED");

}

Dimension d = Toolkit.getDefaultToolkit().getScreenSize();

this.setSize(d.width \*2/3, d.height \*4/5);

this.setVisible(true);

this.setLocationRelativeTo(null);

this.setVisible(true);

this.setTitle("Paint");

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

switch (JOptionPane.showConfirmDialog(settings.getMainFrame(),

"Discard all changes?")) {

case 0:

e.getWindow().dispose();

System.exit(0);

break;

case 1:

buttonSave.doClick();

break;

}

}

});

this.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

mainPanel = new JPanel();

mainPanel.setSize(d);

mainPanel.setLayout(new BorderLayout(1, 1));

select = new JToolBar();

initHistory();

select.setSize(new Dimension(700, 30));

initChoose();

labelModeNow=new JLabel("当前模式 : "+BoardSettings.INITIAL\_TYPE.toString());

settings = new BoardSettings(this);

board = new DrawingBoard(settings,

new Dimension(d.width - historyMain.getWidth(),

d.height - historyMain.getHeight()));

initButtons();

mainPanel.add(board);

mainPanel.add(select, BorderLayout.NORTH);

// mainPanel.add(history,BorderLayout.WEST);

JToolBar toolBarHistory=new JToolBar(JToolBar.VERTICAL);

toolBarHistory.add(historyMain);

mainPanel.add(toolBarHistory, BorderLayout.WEST);

this.add(mainPanel);

select.revalidate();

}

public JLabel getLabelModeNow() {

return labelModeNow;

}

}

### 7.3 配置和图形操作代码

#### BoardSettings类（画板配置）

package com.glede.paint;

import javax.swing.\*;

import java.awt.\*;

import java.util.Vector;

/\*\*

\* Created by Adam on 2017/9/22.

\*/

/\*\*

\* A class that saves current global settings

\*/

public class BoardSettings {

public enum Type{

POINTS,IMAGE,POLYGON,OVAL,RECT,TEXT,DELETE,MOVE,TOP,BOTTOM,LINES,SELECT

,直线折线,图像,多边形,圆形椭圆,矩形,文字,删除,移动,顶部,底部,任意线,选择,未选择

}

Stroke stroke;

Color color;

boolean fill;

Image imgNow;

String text;

Vector<Point> points;

Type type;

MainForm mainFrame;

JPanel history;

DrawingItem itemReplacing;

ResizePoint pointNow;

int selectPoint;

JLabel labelMode;

public int getSelectPoint(){return selectPoint;}

public void setSelectPoint(int i){

selectPoint=i;

}

public MainForm getMainFrame() {

return mainFrame;

}

public ResizePoint getPointNow() {

return pointNow;

}

public boolean typeIsFunction() {

switch (type) {

case 选择:

case 移动:

case 顶部:

case 底部:

case 删除:

return true;

default:

return false;

}

}

public void nextResizePoint(ResizePoint pointNext){

if(pointNow!=null){

pointNow.setShow(false);

}

if(pointNext!=null){

pointNext.setShow(true);

}

pointNow=pointNext;

}

public void replace(Point pos){

if (getItemReplacing()!=null){

getItemReplacing().reposition(pos);

}

}

public DrawingItem getItemReplacing() {

return itemReplacing;

}

public void setItemReplacing(DrawingItem item){

itemReplacing=item;

}

public Font getFont() {

return font;

}

public void setFont(Font font) {

this.font = font;

}

Font font;

public BoardSettings(MainForm mainFrame) {

type = INITIAL\_TYPE;

this.mainFrame = mainFrame;

this.color=Color.BLACK;

this.stroke=mainFrame.getStrokeChooserPanel().getSelectedStroke();

this.imgNow=null;

this.fill=false;

this.history=mainFrame.getHistory();

this.font=new JLabel().getFont();

this.text="";

this.itemReplacing=null;

this.labelMode=mainFrame.getLabelModeNow();

pointNow=null;

}

public void clearPoints(){

setPoints(null);

mainFrame.getBoard().repaint();

}

public JPanel getHistory() {

return history;

}

public void setType(Type type) {

this.type = type;

labelMode.setText("当前模式："+type.toString());

clearPoints();

}

public Type getType() {

return type;

}

public Stroke getStroke() {

return stroke;

}

public void setStroke(Stroke stroke) {

this.stroke = stroke;

}

public Color getColor() {

return color;

}

public void setColor(Color color) {

this.color = color;

}

public boolean isFill() {

return fill;

}

public void setFill(boolean fill) {

this.fill = fill;

}

public Image getImgNow() {

return imgNow;

}

public void setImgNow(Image imgNow) {

this.imgNow = imgNow;

}

public String getText() {

return text;

}

public void setText(String text) {

this.text = text;

}

public Vector<Point> getPoints() {

if(points==null){

points = new Vector<>();

}

return points;

}

public void setPoints(Vector<Point> points) {

this.points = points;

}

public static final Type INITIAL\_TYPE=Type.未选择;

}

#### DrawingBoard类（画板操作）

package com.glede.paint;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.event.MouseMotionListener;

import java.awt.geom.Ellipse2D;

import java.awt.image.BufferedImage;

import java.util.LinkedList;

import java.util.ListIterator;

import java.util.Vector;

/\*\*

\* Created by Glede on 1/10/2019

\*/

/\*\*

\* The listener of DrawingBoard

\*

\* @author glede, xueyuan

\* @see DrawingBoard

\* @see java.awt.event.MouseListener

\* @see java.awt.event.MouseMotionListener

\*/

class BoardMouseListener implements MouseListener, MouseMotionListener {

BoardSettings settings;

DrawingBoard drawingBoard;

// LinkedList<JButton> buttons;

Point begin, now, end;

private void setPreview() {

drawingBoard.setPreview(null);

}

private void setPreview(DrawingItem item) {

drawingBoard.setPreview(item);

}

/\*\*

\* Set preview

\*

\* @param beg The point that drawing begins

\* @param mouse The point that mouse at now

\*/

private void setPreview(Point beg, Point mouse) {

int xx = Math.min(beg.x, mouse.x),

yy = Math.min(beg.y, mouse.y),

disX = Math.abs(beg.x - mouse.x),

disY = Math.abs(beg.y - mouse.y);

switch (settings.getType()) {

case 矩形:

drawingBoard.setPreview(new DrawingShape(settings.getColor(),

new Rectangle(xx, yy, disX, disY), settings.isFill(), settings.getStroke()));

break;

case 图像:

drawingBoard.setPreview(new DrawingImage(settings.getImgNow(),

new Rectangle(xx, yy, disX, disY), settings.getMainFrame()));

break;

case 圆形椭圆:

drawingBoard.setPreview(new DrawingShape(settings.getColor(),

new Ellipse2D.Double(xx, yy, disX, disY), settings.isFill(), settings.getStroke()));

break;

case 任意线:

drawingBoard.setPreview(new DrawingPoints(settings.getColor(),

settings.getPoints(), settings.getStroke()));

break;

case 文字:

drawingBoard.setPreview(new DrawingText(settings.getText(), now,

settings.getFont(), settings.getColor()));

break;

}

}

/\*\*

\* Reposition current preview to pos

\* @param pos

\*/

private void repositionPreview(Point pos) {

if (drawingBoard.getPreview() != null)

drawingBoard.getPreview().reposition(pos);

}

public BoardMouseListener(BoardSettings settings,

DrawingBoard drawingBoard) {

this.settings = settings;

this.drawingBoard = drawingBoard;

}

@Override

public void mouseDragged(MouseEvent e) {

now = new Point(e.getPoint());

if (settings.getType() == BoardSettings.Type.任意线) {

settings.getPoints().addElement(now);

setPreview(begin, now);

} else if (settings.getType() == BoardSettings.Type.选择) {

ResizePoint resizePoint = settings.getPointNow();

if (resizePoint != null && resizePoint.valid())

resizePoint.getItem().resize(settings.getSelectPoint(), now);

} else if (settings.getType() != BoardSettings.Type.移动)

setPreview(begin, now);

else {

repositionPreview(now);

}

drawingBoard.repaint();

}

@Override

public void mouseMoved(MouseEvent e) {

Point mouse = new Point(e.getPoint());

switch (settings.getType()) {

case 选择:

selectResizePoint(mouse);

break;

case 多边形:

case 直线折线:

Vector<Point> previewPoints =

(Vector<Point>) settings.getPoints().clone();

previewPoints.add(mouse);

drawingBoard.setPreview(new DrawingPoints(settings.color,

previewPoints, settings.getStroke(), true));

drawingBoard.repaint();

break;

}

}

@Override

public void mouseClicked(MouseEvent e) {

now = new Point(e.getPoint());

if (begin != null) {

if (settings.getType() == BoardSettings.Type.多边形) {

settings.getPoints().addElement(now);

setPreview(begin, now);

} else if (settings.getType() == BoardSettings.Type.直线折线) {

settings.getPoints().addElement(now);

setPreview(begin, now);

} else if (settings.getType() != BoardSettings.Type.移动)

setPreview(begin, now);

else {

repositionPreview(now);

}

}

drawingBoard.repaint();

}

private void selectResizePoint(Point point) {

for (ListIterator<DrawingItem> iter =

drawingBoard.getItemsList().

listIterator(drawingBoard.getItemsList().size()); iter.hasPrevious(); ) {

DrawingItem item = iter.previous();

int result;

if ((result = item.getResizePoint().selected(point)) >= 0) {

settings.nextResizePoint(item.getResizePoint());

settings.setSelectPoint(result);

drawingBoard.repaint();

return;

}

}

for (ListIterator<DrawingItem> iter =

drawingBoard.getItemsList().

listIterator(drawingBoard.getItemsList().size()); iter.hasPrevious(); ) {

DrawingItem item = iter.previous();

if (item.contains(point)) {

settings.nextResizePoint(item.getResizePoint());

settings.setSelectPoint(0);

drawingBoard.repaint();

return;

}

}

}

@Override

public void mousePressed(MouseEvent e) {

begin = new Point(e.getPoint());

if (settings.getType() == BoardSettings.Type.选择) {

selectResizePoint(begin);

} else if (settings.getType() != BoardSettings.Type.多边形 &&

settings.getType() != BoardSettings.Type.直线折线) {

settings.getPoints().addElement(begin);

}

}

@Override

public void mouseReleased(MouseEvent e) {

if (begin != null) {

end = new Point(e.getPoint());

if (settings.getType() != BoardSettings.Type.多边形 &&

settings.getType() != BoardSettings.Type.直线折线) {

settings.getPoints().addElement(end);

}

Vector<Point> points = settings.getPoints();

if (points.isEmpty()) return;

double xx = Math.min(points.firstElement().x, points.lastElement().x),

yy = Math.min(points.firstElement().y, points.lastElement().y),

disX = Math.abs(points.firstElement().x - points.lastElement().x),

disY = Math.abs(points.firstElement().y - points.lastElement().y);

switch (settings.getType()) {

case 圆形椭圆:

addListItem(new DrawingShape(settings.color,

new Ellipse2D.Double(xx, yy, disX, disY),

settings.isFill(), settings.getStroke()));

setPreview();

break;

case 矩形:

addListItem(new DrawingShape(settings.color,

new Rectangle((int) xx, (int) yy, (int) disX, (int) disY),

settings.isFill(), settings.getStroke()));

setPreview();

break;

case 图像:

addListItem(new DrawingImage(settings.getImgNow(),

new Rectangle((int) xx, (int) yy, (int) disX, (int) disY), settings.mainFrame));

setPreview();

break;

case 任意线:

addListItem(new DrawingPoints(settings.color, points, settings.getStroke()));

setPreview();

break;

case 文字:

addListItem(new DrawingText(settings.getText(), end, settings.getFont(), settings.getColor()));

setPreview();

break;

case 移动:

settings.replace(end);

default:

break;

}

drawingBoard.repaint();

}

}

@Override

public void mouseEntered(MouseEvent e) {

if (settings.getType() == BoardSettings.Type.多边形) {

begin = null;

}

}

@Override

public void mouseExited(MouseEvent e) {

if (drawingBoard.getPreview() != null

&& drawingBoard.getPreview().isSelectedPreview()

&& settings.getType() != BoardSettings.Type.移动) {

setPreview();

drawingBoard.repaint();

}

Vector<Point> points = settings.getPoints();

if (points == null || points.size() <= 1) return;

if (settings.getType() == BoardSettings.Type.多边形) {

int[] xpoints = new int[points.size()], ypoints = new int[points.size()];

int i = 0;

for (Point point : points) {

xpoints[i] = (point.x);

ypoints[i] = (point.y);

i++;

}

addListItem(new DrawingPolygon(settings.color,

new Polygon(xpoints, ypoints, points.size()), settings.isFill(), settings.getStroke()));

setPreview();

drawingBoard.repaint();

} else if (settings.getType() == BoardSettings.Type.直线折线) {

addListItem(new DrawingLines(settings.getColor(), points, settings.getStroke()));

setPreview();

drawingBoard.repaint();

}

}

/\*\*

\* Add the item to the list of all items

\* @param item

\*/

private void addListItem(DrawingItem item) {

if (settings.getPoints() != null) {

drawingBoard.getItemsList().add(item);

HistoryButton button = new HistoryButton(drawingBoard, settings.getType().toString(), item);

item.setRelatedButton(button);

settings.getHistory().add(button);

settings.getHistory().revalidate();

settings.setPoints(null);

}

}

}

/\*\*

\* The mighty drawing board!!!

\*

\* @author glede,xueyuan

\*/

public class DrawingBoard extends JPanel {

BoardSettings settings;

LinkedList<DrawingItem> itemsList;

DrawingItem preview;

BoardMouseListener listener;

private BufferedImage image;

DrawingBoard(BoardSettings settings, Dimension dimension) {

this.settings = settings;

setSize(dimension);

itemsList = new LinkedList<>();

listener = new BoardMouseListener(settings, this);

this.addMouseListener(listener);

this.addMouseMotionListener(listener);

}

/\*\*

\* Clears the drawing board

\*/

public void clearBoard() {

itemsList.clear();

settings.getHistory().removeAll();

this.setPreview(null);

this.repaint();

if (settings.typeIsFunction())

settings.setType(BoardSettings.Type.任意线);

settings.getHistory().revalidate();

settings.getHistory().repaint();

}

public LinkedList<DrawingItem> getItemsList() {

return itemsList;

}

/\*\*

\* Set current preview

\* @param item

\*/

void setPreview(DrawingItem item) {

this.preview = item;

}

public DrawingItem getPreview() {

return preview;

}

/\*\*

\* Used when saving as image

\* @return An image of the copy of the drawing board

\*/

public BufferedImage getImage() {

image = new BufferedImage(this.getWidth(), this.getHeight(), BufferedImage.TYPE\_3BYTE\_BGR);

Graphics g = image.createGraphics();

g.fillRect(0, 0, image.getWidth(), image.getHeight());

this.setPreview(null);

this.print(g);

return image;

}

/\*\*

\* Copy param list to item list

\* @param list

\*/

public void readList(LinkedList<DrawingItem> list) {

clearBoard();

itemsList.addAll(list);

for (DrawingItem item : itemsList) {

item.initResizePoint();

item.setRelatedButton(new HistoryButton

(this, item.getType().toString(), item));

settings.getHistory().add(item.getRelatedButton());

}

settings.getHistory().revalidate();

settings.getHistory().repaint();

repaint();

}

/\*\*

\* Draw all item and preview on g

\* @param g

\*/

@Override

public void paint(Graphics g) {

super.paintComponent(g);

g.setColor(Color.WHITE);

g.fillRect(0, 0, getWidth(), getHeight());

g.setColor(Color.BLACK);

g.drawRect(0, 0, getWidth(), getHeight());

if (itemsList != null)

itemsList.forEach(item -> {

item.draw(g);

if (settings.getType() == BoardSettings.Type.选择) {

item.drawResizePoint(g);

}

});

if (settings.getType() == BoardSettings.Type.选择 &&

settings.getPointNow() != null) {

settings.getPointNow().getItem().drawBounds(g);

}

if (preview != null) {

preview.draw(g);

}

// selectBoard.repaint(new Rectangle(0,0,700,30));

}

}

#### SerializableStroke类和StrokeLibrary类（基础结构）

package com.glede.paint;

import java.awt.\*;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

/\*\*

\* Created by Glede on 1/10/2019

\*/

public class SerializableStroke implements Serializable,Stroke {

public transient BasicStroke basicStroke;

public SerializableStroke(float width, int cap, int join, float miterlimit,

float dash[], float dash\_phase) {

basicStroke = new BasicStroke(width, cap, join, miterlimit, dash, dash\_phase);

}

// float width, int cap, int join, float miterlimit,

// float dash[], float dash\_phase)

private void writeObject(ObjectOutputStream out) throws IOException {

out.writeFloat(basicStroke.getLineWidth());

out.writeInt(basicStroke.getEndCap());

out.writeInt(basicStroke.getLineJoin());

out.writeFloat(basicStroke.getMiterLimit());

out.writeObject(basicStroke.getDashArray());

out.writeFloat(basicStroke.getDashPhase());

}

private void readObject(ObjectInputStream in) throws ClassNotFoundException, IOException {

float width = in.readFloat();

int cap = in.readInt();

int join = in.readInt();

float miterlimit = in.readFloat();

float dash[] = (float[]) in.readObject();

float dash\_phase = in.readFloat();

basicStroke = new BasicStroke(width, cap, join, miterlimit, dash, dash\_phase);

}

public SerializableStroke(float width, int cap, int join, float miterlimit) {

this(width, cap, join, miterlimit, null, 0.0f);

}

public SerializableStroke(float width, int cap, int join) {

this(width, cap, join, 10.0f, null, 0.0f);

}

public SerializableStroke(float width) {

this(width, CAP\_ROUND, JOIN\_MITER, 10.0f, null, 0.0f);

}

public SerializableStroke() {

this(1.0f, CAP\_SQUARE, JOIN\_MITER, 10.0f, null, 0.0f);

}

public static final int JOIN\_MITER = BasicStroke.JOIN\_MITER;

public static final int JOIN\_ROUND = BasicStroke.JOIN\_ROUND;

public static final int JOIN\_BEVEL = BasicStroke.JOIN\_BEVEL;

public static final int CAP\_BUTT = BasicStroke.CAP\_BUTT;

public static final int CAP\_ROUND = BasicStroke.CAP\_ROUND;

public static final int CAP\_SQUARE = BasicStroke.CAP\_SQUARE;

@Override

public Shape createStrokedShape(Shape p) {

return basicStroke.createStrokedShape(p);

}

}

package com.glede.paint;

import external.StrokeSample;

import java.io.Serializable;

public class StrokeLibrary {

public static final StrokeSample[] strokes = {

new StrokeSample(new SerializableStroke(1f)),

new StrokeSample(new SerializableStroke(2f)),

new StrokeSample(new SerializableStroke(3f)),

new StrokeSample(new SerializableStroke(4f)),

new StrokeSample(new SerializableStroke(5f)),

new StrokeSample(new SerializableStroke(6f)),

new StrokeSample(new SerializableStroke(7f)),

new StrokeSample(new SerializableStroke(8f)),

new StrokeSample(new SerializableStroke(9f)),

new StrokeSample(new SerializableStroke(10f)),

new StrokeSample(new SerializableStroke(11f)),

new StrokeSample(new SerializableStroke(12f)),

new StrokeSample(new SerializableStroke(1.5f, SerializableStroke.CAP\_BUTT,

SerializableStroke.JOIN\_ROUND, 2.5f, new float[]{15, 10,},

0f)),

new StrokeSample(new SerializableStroke(2.5f, SerializableStroke.CAP\_BUTT,

SerializableStroke.JOIN\_ROUND, 3.5f, new float[]{15, 10,},

0f)),

new StrokeSample(new SerializableStroke(3.5f, SerializableStroke.CAP\_BUTT,

SerializableStroke.JOIN\_ROUND, 4.5f, new float[]{15, 10,},

0f)),

new StrokeSample(new SerializableStroke(4.5f, SerializableStroke.CAP\_BUTT,

SerializableStroke.JOIN\_ROUND, 5.5f, new float[]{15, 10,},

0f)),

new StrokeSample(new SerializableStroke(5.5f, SerializableStroke.CAP\_BUTT,

SerializableStroke.JOIN\_ROUND, 6.5f, new float[]{15, 10,},

0f)),

new StrokeSample(new SerializableStroke(6.5f, SerializableStroke.CAP\_BUTT,

SerializableStroke.JOIN\_ROUND, 7.5f, new float[]{15, 10,},

0f)),

};

}

#### ResizePoint类

package com.glede.paint;

import java.awt.\*;

import java.util.Vector;

public class ResizePoint extends DrawingItem {

Vector<Point> points;

DrawingItem item;

boolean show;

private final double radius = 10;

@Override

public Rectangle getBounds() {

return null;

}

@Override

public boolean contains(Point point) {

return false;

}

private static final Stroke dash = new SerializableStroke(1.5f,

SerializableStroke.CAP\_BUTT,

SerializableStroke.JOIN\_ROUND,

1f, new float[]{15, 10,},

0f);

private static final Stroke ordinary=new SerializableStroke();

private static final Color colorWhite = new Color(1.0f, 1.0f, 1.0f, 0.4f);

private static final Color colorBlack = new Color(0.0f, 0.0f, 0.0f, 0.4f);

public ResizePoint(DrawingItem itemRef) {

super(Type.调整, true);

show = false;

points = itemRef.getResizePoints();

item = itemRef;

}

void setShow(boolean show) {

this.show = show;

}

void setItem(DrawingItem itemRef) {

points = itemRef.getResizePoints();

item = itemRef;

}

public ResizePoint() {

super(Type.调整, true);

}

public boolean valid() {

return points != null && item != null;

}

@Override

public void reposition(Point point) {

if (point == null) {

points = item.getResizePoints();

} else {

Point delta = new Point(point.x - points.elementAt(0).x,

point.y - points.elementAt(0).y);

for (Point x : points) {

x.x += delta.x;

x.y += delta.y;

}

}

}

@Override

public void resize(int resizePointRank, Point posTo) {

reposition(null);

}

public DrawingItem getItem() {

return item;

}

public int selected(Point point) {

for (int i = 0; i < points.size(); i++) {

double a = point.x - points.elementAt(i).x, b = point.y - points.elementAt(i).y;

a = a \* a;

b = b \* b;

if (a + b <= radius \* radius) {

return i;

}

}

return -1;

}

private void setDrawColor(Graphics2D graphics2D,boolean white){

if (white) {

if (show)

graphics2D.setColor(Color.WHITE);

else

graphics2D.setColor(colorWhite);

}

else {

if (show) {

graphics2D.setColor(Color.BLACK);

} else {

graphics2D.setColor(colorBlack);

}

}

}

@Override

public void draw(Graphics g) {

Graphics2D graphics2D = (Graphics2D) g;

graphics2D.setStroke(new BasicStroke(1));

for (int i = 1; i < points.size(); i++) {

Point x = points.elementAt(i - 1), y = points.elementAt(i);

setDrawColor(graphics2D,true);

graphics2D.setStroke(ordinary);

graphics2D.drawLine(x.x, x.y, y.x, y.y);

setDrawColor(graphics2D,false);

graphics2D.setStroke(dash);

graphics2D.drawLine(x.x, x.y, y.x, y.y);

}

for (Point point : points) {

setDrawColor(graphics2D,true);

graphics2D.fillOval(point.x - (int) radius, point.y - (int) radius,

(int) radius \* 2, (int) radius \* 2);

setDrawColor(graphics2D,false);

graphics2D.drawOval(point.x - (int) radius, point.y - (int) radius,

(int) radius \* 2, (int) radius \* 2);

}

}

@Override

protected Vector<Point> getResizePoints() {

return null;

}

@Override

public DrawingItem createPreview() {

return null;

}

}

#### LightImage类（亮度调节）

package com.glede.paint;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Component;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.util.Dictionary;

import java.util.Hashtable;

import javax.imageio.ImageIO;

import javax.swing.Box;

import javax.swing.BoxLayout;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JSlider;

import javax.swing.event.ChangeEvent;

import javax.swing.event.ChangeListener;

import java.awt.event.\*;

import javax.swing.\*;

public class LightImage extends JFrame

{

private JPanel panel = null;

private JSlider slider = null;

private BufferedImage img = null;

private int value = 0;

public int isClosed = 0;

public LightImage()

{

initComponent();

this.setVisible(true);

}

public int getClosed() {

    return isClosed;

}

private void initComponent()

{

this.setTitle("Brightness Adjustment");

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setSize(550, 450);

this.setLocationRelativeTo(null);

initImg(new File("cache.jpg"));

panel = new JPanel()

{

@Override

protected void paintComponent(Graphics g)

{

super.paintComponent(g);

Graphics2D g2 = (Graphics2D) g;

g2.drawImage(img, 0, 0, this.getWidth(), this.getHeight(), null);

}

};

this.add(panel, BorderLayout.CENTER);

        this.addWindowListener(new WindowAdapter(){

            public void windowClosing(WindowEvent e){

         File file = new File("cache.jpg");

         try {

         ImageIO.write(img, "jpg", file);

         }catch (IOException ioe) {

                    // TODO: handle exception

                }

                isClosed = 1;

            }

        }

        );

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE);

panel.setLayout(new BorderLayout());

slider = new JSlider();

slider.setValue(50);

slider.setPaintLabels(true);

slider.setPaintTicks(true);

slider.setMajorTickSpacing(100);

slider.setMinorTickSpacing(5);

Dictionary<Integer, Component> labels = new Hashtable<Integer, Component>();

for (int i = 0; i <= 100; i += 5)

{

labels.put(i, new JLabel("" + i));

}

slider.setLabelTable(labels);

value = slider.getValue();

slider.addChangeListener(new ChangeListener()

{

@Override

public void stateChanged(ChangeEvent e)

{

if (slider.getValueIsAdjusting())

{

int changeValue = slider.getValue() - value;

processImg(changeValue);

value = slider.getValue();

}

}

});

Box box = new Box(BoxLayout.X\_AXIS);

box.add(new JLabel("BrightnessAdjustment:"));

box.add(slider);

this.add(box, BorderLayout.NORTH);

}

private void initImg(File file)

{

try

{

img = ImageIO.read(file);

} catch (IOException e)

{

JOptionPane.showMessageDialog(null, "NO IMAGE!");

}

}

private void processImg(int changeValue)

{

for (int x = 0; x < img.getWidth(); x++)

{

for (int y = 0; y < img.getHeight(); y++)

{

int rgb = img.getRGB(x, y);

Color color = new Color(rgb);

int r = color.getRed() + changeValue;

int g = color.getGreen() + changeValue;

int b = color.getBlue() + changeValue;

if (r > 255)

{

r = 255;

} else if (r < 0)

{

r = 0;

}

if (g > 255)

{

g = 255;

} else if (g < 0)

{

g = 0;

}

if (b > 255)

{

b = 255;

} else if (b < 0)

{

b = 0;

}

color = new Color(r, g, b);

img.setRGB(x, y, color.getRGB());

}

}

panel.repaint();

}

/\*

public static void main(String[] args)

{

new LightImage();

}

\*/

}

**八、Java课程学习心得与改进意见**

本学期的Java课程使我受益良多。从学到的知识方面来说，我又学会了一门主流编程语言，无论是对于自己日后的科研需要还是应对将来可能面对的各种面试、笔试都十分有帮助。Java入门其实不是很难，尤其是在学习过C/C++的基础上，我对于数据类型、条件语句、循环语句等语言学习的基本方面都有了很熟练的掌握；对于面向对象的编程、类的声明定义与使用、成员变量和成员函数的关系、类的继承、重载、多态、虚基类等知识点也都有较为深刻的理解，在这样的知识背景下，学习Java便轻松了不少。对于与C/C++相一致的知识，可以当作复习、巩固来对待，而对于Java所特有的用法，如：类或方法的嵌套定义、接口interface，Java异常处理、包的导入等是我学习这门课程的重点，只有真正掌握了自己曾经不会的内容，才算真正有所收获。因此，在学习这门课之前，我便有了一个十分清晰的规划，因而学习也具有了目标性和针对性，最终达到了预期的目标。看着自己从一开始只能用写C/C++的思路写Java程序，到现在能够写出“具有Java特色”的代码，实现一个功能丰富的、有较高实用价值的可视化程序，一种巨大的成就感油然而生。

从本次课程设计的角度来说，设计并实现一个画图板，在一开始总觉得难度不小，但真正做起来，发现采用模块化的设计思想，将大的工程问题划分为多个小的模块，逐一攻破，便可以使问题得到极大的简化，设计的思路也变得清晰了不少。在课程结束之前，我就早早开始动手写画图板，但由于当时对Java的理解还不够深刻、还缺乏编程实践。我虽然写了一部分，但总感觉像是在写C++，不能自觉地用Java的思维想问题，导致写出的代码并没有体现出Java语言的特色和优势，灵活性欠佳，而且很多类和方法都堆砌在一个文档中，维护变得十分不方便，最后迫于杂乱无章而只能另起炉灶、推倒重来，虽然之前也投入了不少精力，但确是一次有益的尝试，不仅使我发现了学习过程中存在的问题，也为之后重写画图板做了技术的积累。在之后的程序实现中，我汲取之前的教训，避免踩之前踩过的坑、更加注重体现Java编程的思想，代码管理上也更加有条理，先划分功能模块后动手编写程序，分文件实现各功能模块，在进度和速度上相比于之前也有了很大的提高。

当然，大作业中也存在一些遗憾，比如：亮度调节模块使用的方法是根据进度条SliderValue减去一个调整参数value，作为改变RGB三通道值的偏移量，这种方法虽然实现起来较为简单，但是可能存在白噪声，当图片亮度值改变后，这种变化常常是不可逆的，而导致图片的质量被降低，这也是在算法上还有待改进的地方，其中一种可行的改进思路是先将RGB颜色空间转换到HSV颜色空间，再对图像的亮度或对比度进行操作，这种方法更科学，但是可能需要使用到Opencv API for Java，因此需要额外配置Opencv的链接库。此外，删除操作做的还不够流畅，这也是今后值得改进和注意的地方。

最后，还要十分感谢王波涛老师严谨认真的教学，王老师是少有的能给我留下很深印象的老师。王老师不仅是一位责任心很强的好老师，尤其在与同学相处方面，也是一位“有感情”、对一切充满感恩心、言传重于身教的好老师。当然，也要感谢本次课程设计中的队友，感谢队友的默契配合和及时有益的帮助。

首先，对于Java学习。基础语法和Java原理是地基，地基不牢靠，犹如沙地上建摩天大厦，是相当危险的。学习Java也是如此，必须要有扎实的基础，才能在J2EE、J2ME领域游刃有余。在学习Java的语法时，Java的语法是类似c语言的，所以学习的比较轻松。唯一需要注意的是有几个不容易搞清楚的关键字的用法，public，protected，private，static，什么时候用，为什么要用，怎么用，和同学一起讨论了好久才得以解决。在学习Java的面向对象的编程语言的特性。比如继承，构造器，抽象类，接口，方法的多态，重载，覆盖，Java的异常处理机制。

而对于这次课程设计，首先，大体分析一下一个画图板界面可以分成那些部分：整体的一个大窗体，处于最顶部的一个菜单栏，左侧的工具栏，中间空白的画图纸，下部的颜色选择器以及底部有一条帮助提示栏。最早我所做的画图板把这些生成界面的相关代码都给写在了同一个类里面，其中包括各种组件的添加和布局，那样写的话就会显得十分的乱，想要修改某一部分就要在里面找上好久。后来这个最新弄得画板在生成这个界面的时候采用的方法是每一部分的组件单独成一类，在每一个类中写一个初始化的方法，并且重载他的构造方法，这样只要在主类中调用他的构造方法再合理的安排一下布局就可以了。这样做每一部分组件单独成类显得十分清晰，但也有一定的不足，就是在这些类与后来定义的监听器类之间相互传递参数的时候就会很麻烦，经常传参数传的自己头昏脑胀的，但肯定没有万全的方法吧，看自己习惯怎么弄了。

期间遇到了许多的挑战，比如以画矩形为例，我们查看API中drawRect这个方法发现他也有四个参数：int x, int y, int width, int height。很显然这四个参数是一个点的一对坐标和长还有宽。这里如果还是像之前画直线那样把x1赋给x，y1赋给y。x2-x1赋给width，y2-y1赋给height。如果这样的话，运气好的话我们是可以话出来一个矩形的，但只能正确的画出一种矩形，就是鼠标拖动的方向是从左上向右下的。其他的矩形会偏离我们鼠标所滑动的区域。那这是为什么呢？其实只要我们好好看一下drawRect这个方法，好好看下他的四个参数代表的东西就会发现问题：坐标xy是矩形的左上点的坐标。所以我们鼠标按下松开获取的x1x2y1y2这四个坐标，我们就要从中找出左上角的那个坐标。不难分析，窗体左上角的坐标为（0，0）。所以左上角的坐标是较小的，所以只需调用一个取较小值的一个方法Math.min(x1,x2)，Math.min(y1, y2)这样分别赋给x,y就可以了。另外，长宽的设置，我们也不能直接就x2-x1，y2-y1，需要对其取一个绝对值，那就是Math. abs (x1-x2), Math.abs(y1-y2)。这样就可以正确的画出一个矩形了。

说学到一定程度要尝试着自己做东西，但觉得仅仅经过一学期的学习，还远远不够，希望在以后学习中继续努力，能够真真正正拥有一门编程语言。