软件设计实践报告

单　　位：信息工程学院

班　　级：软件工程(2)班

学　　号：2016551439

姓　　名：同正南

任课教师：谭貌

湘 潭 大 学

2019 年 06 月

软件设计实践

项目名称:《Java MiNi 画图工具》

湘潭大学信息工程学院　同正南

一、项目概述

1.1　概述

由于我们平时工作学习中，常常有作图编辑图片的需要，因此我们小组打算开发一个画图工具并兼备编辑图片的功能。这样在我们的日常生活工作学习中可极大地提高我们的工作效率。

由于本次项目开发时间短，任务重，组员技术面参差不齐，我们决定把技术面只限定在 java 上，去开发一个迷你画图工具，面向对象思维较强，分工容易，可扩展性较强,技术难度有层次感,这样每个组员都有能力参与到项目开发中去，最终大家的收获和实际的效果都会好一些。

我们的项目采用 Java 语言开发，界面使用了轻量级组件 Java Swing 编写。功能上，首先能想到的是实现自由画笔、一些基本的图形画笔（圆形，矩形等）、橡皮擦功能还有复杂多边形如五角星，并且提供可选择的颜色和画笔粗细程度，还有绘画步骤的回退、图片保存，可捕捉图形拖动、旋转、改变图形大小功能。但是在实际中几乎很少有用 java 做 PC 桌面端的应用开发,这个程序并不是标准的 C-S 结构或 B-S 结构，但是我们再做的时候又想让它接近有服务，有后台这种模式，所以我们想在后期，在画图工具大部分功能实现的基础上，再实现打开画图工具验证用户信息，实现添加用户、删除用户、修改用户信息，并且画图工具可以打开与用户绑定在一起的绘图文件。用户可将绘图文件属性信息存储到数据库中。

1.2　目的和用途

画图工具是一个具有广泛市场的实用软件。人们在娱乐、教学演示等方面时都会用到画图。画图工具的项目简洁而不失难度，对于 Java 语言、项目开发流程的学习都有极大的帮助，因此以画图工具为主题进行一次开发，是十分具有实际意义的。

画图用具的实际用途十分广泛。当其作为一个独立程序，在教学时，教师通常会通过绘图来使一些抽象的知识点具体化，达到提高教学水平的效果。某些高级绘图程序还可以用于完成专业的制图任务和专业处理；当其作为一个接口子程序，可以为上层程序提供基础服务，是一个极其重要的子程序。总而言之，画图工具与我们的日常生活息息相关。

1.3　需求说明

1.3.1　功能需求

表 1-1 功能需求

功能类别	功能名称	功能介绍
	画图	可选择从工具栏挑选直线、矩形、正方形、

基本功能		圆、椭圆、实心矩形、正方形、圆
	改变图形轮廓的颜色	可选择从工具栏挑选常用的几种颜色或者编辑菜单栏里从所有颜色列表里选择
	改变图形轮廓的粗细	可选择从工具栏挑选常用的几种粗细长度，或者编辑菜单栏里输入用户 想要的粗细
	帮助文档信息	可选择菜单栏里帮助菜单条款-关于画图板、关于作者信息
	画笔信息	在画图界面移动画笔时，在底部显示画笔位置(x,y)坐标信息。
	新建画图板	将画图板的图形清除，成为崭新的一张画板。
	退出画图板	在菜单栏里文件条款下点击退出，会终止程序的运行，可考虑检查画板是否保存的大牛股提示选项。
附加功能	复杂多边形	列如三角形、五边形、五角星、菱形、箭头
	喷漆点均匀分布	能在画板里喷出均匀的小点
	实现捕捉一个图形、矩形框选一些图形	捕捉一个图形或矩形框选一些图形能够在画板拖动并且删除
	实现所画图形的回退,复原，删除，重置	回退每次回退到上一个图形，复原同理。删除是删除最后一个图形。重置是重置会原来最多的图形数的状态。
	保存、另存为，打开	对所画的图形的另存为、保存到本地，可以考虑将图形转化为 png、jpg、gif 等图片格式,可以将保存到本地的画图文件打开
	曲线、橡皮擦	要实现能够画曲线和消除痕迹的橡皮擦
	旋转,移动	捕捉所画图形进行旋转、移动
	回退、恢复指定步数	在编辑菜单栏里有回退和恢复指定步数的条款
	插入文字	可以在画板插入输入的文本信息
	复制、粘贴选定我的图形	把所选定的图形可通过复制，重现粘贴在画板的其它位置上

1.3.2 非功能需求

表 1-2 非功能需求

需求类别	需求名称	需求介绍
运行需求	用户界面	有一个主界面，设想的是把每个分界面作为主界面的成员属性，按照某种布局方式组织起来
	操作定位	用户操作画笔的位置要精准
	界面提示	文件保存操作的提示语及相关的

		错误信息和显示画笔位置信息要简单明确
性能需求	配置要求	明确画图工具运行所需的最低配置及推荐配置
质量需求	正确性	用户在正常的操作下应能得到理想的画图结果
	响应时间	画图板对用户所做出的操作要尽可能的快速做出反应

1.4 环境要求

由于最后会使用 exe4j 将 JRE 运行环境与程序打包编译成.exe 对于系统的要求只是 200M 的硬盘空余空间和 1G 内存即可。

二、系统分析与设计

2.1 系统设计

2.1.1 系统结构图

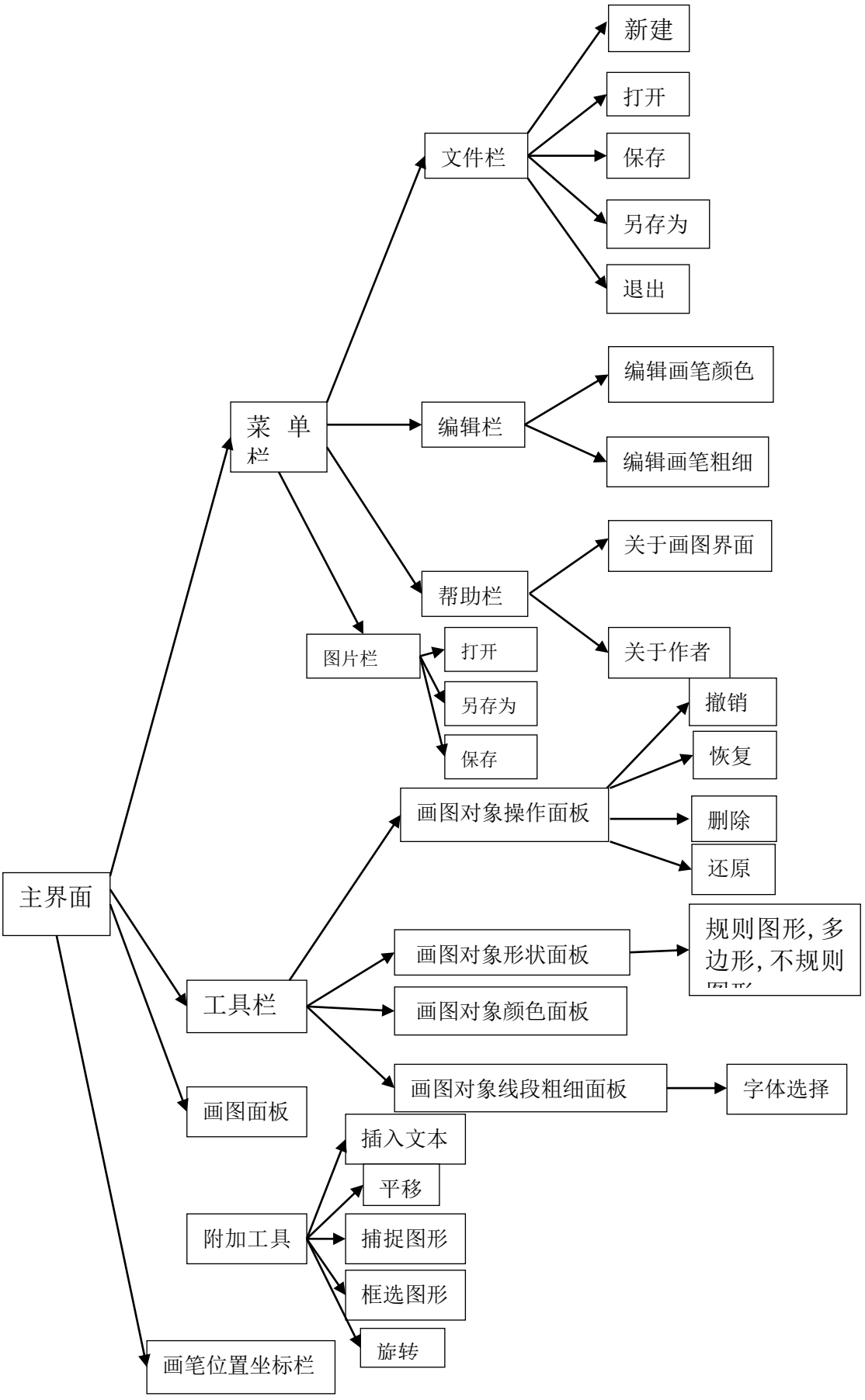

主界面
菜 单 栏
文件栏
新建
打开
保存
另存为
退出
编辑栏
编辑画笔颜色
编辑画笔粗细
帮助栏
关于画图界面
关于作者
图片栏
打开
另存为
保存
工具栏
画图对象操作面板
撤销
恢复
删除
还原
画图对象形状面板
规则图形, 多边形, 不规则
画图对象颜色面板
画图对象线段粗细面板
字体选择
画图面板
附加工具
插入文本
平移
捕捉图形
框选图形
旋转
画笔位置坐标栏

图 2-5 系统结构图

2.1.2 User Case 图

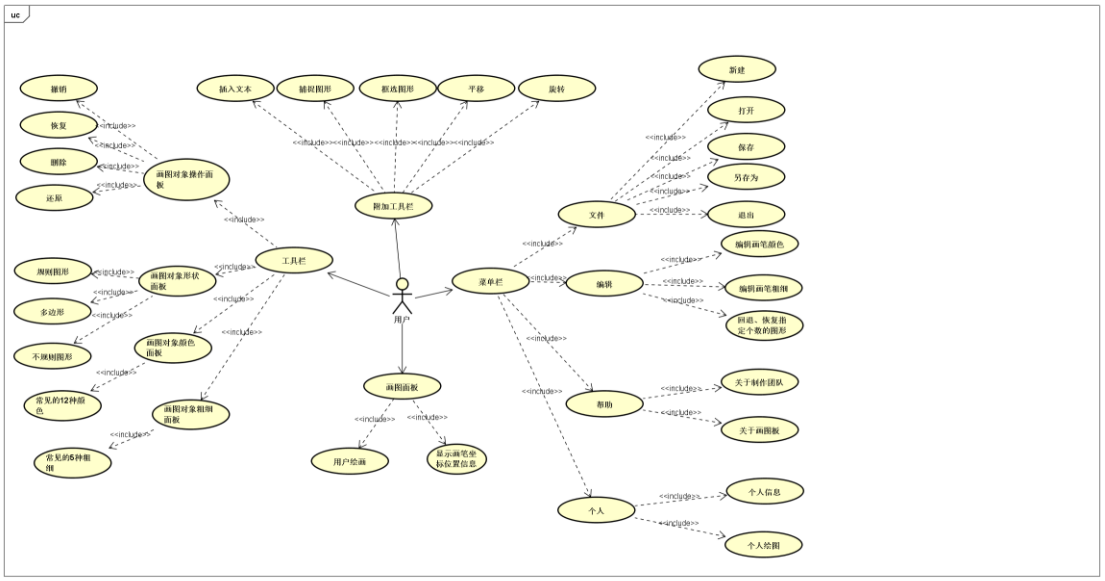


图 2-6 用例图

2.1.3 类图

1) Shape 类

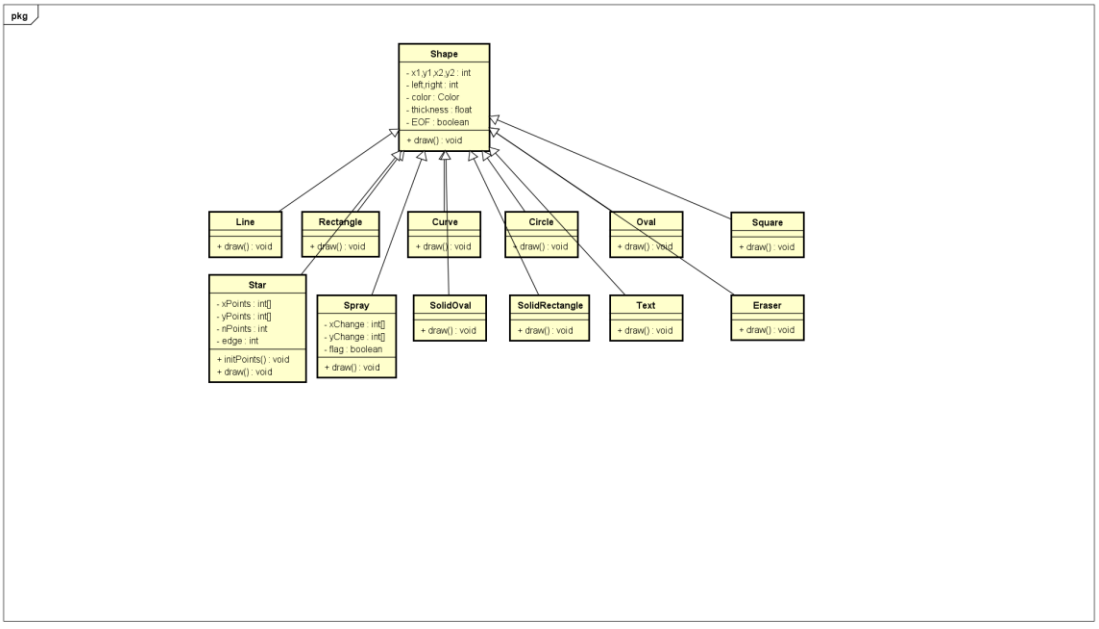


图 2-7 Shape 类图

2) 视图层、核心功能类

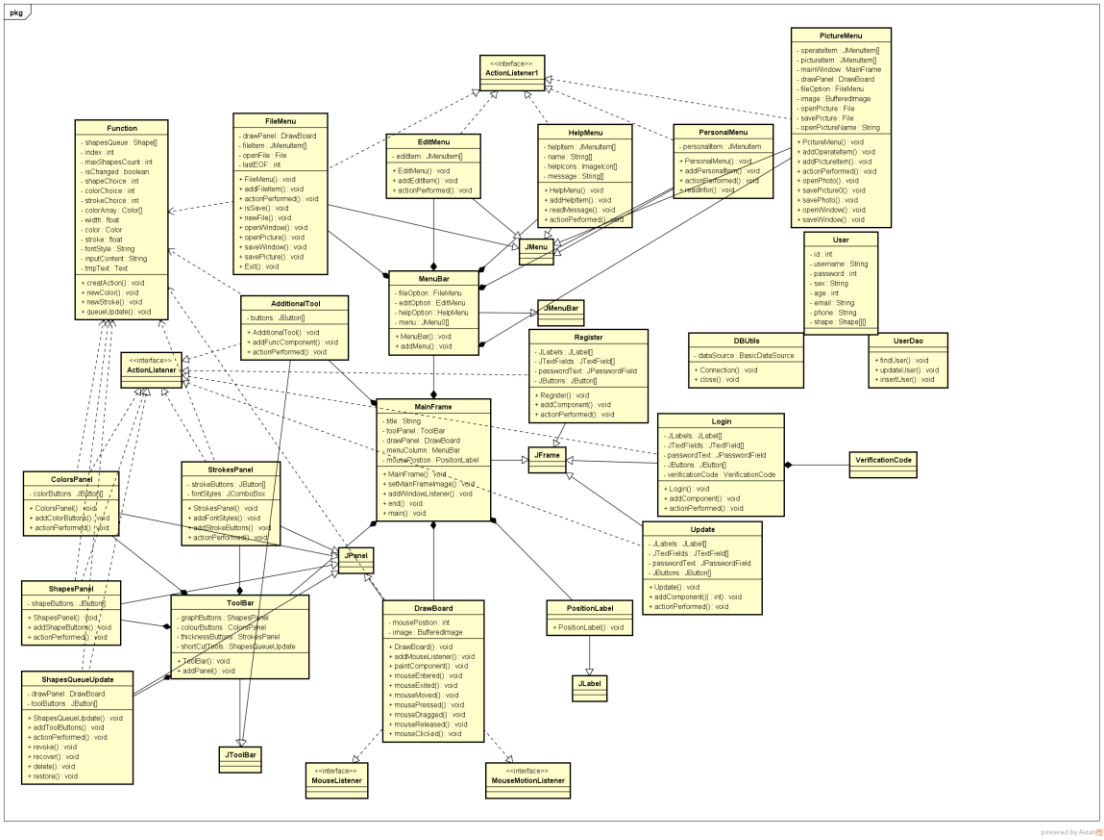


图 2-8 视图层、核心功能类图

2.2 模块/函数设计

1. 界面及总体组成结构

1) 界面整体设计

由上面视图层类图设计大概设计思路是

构思: MainFrame 类采用 BorderLayout 布局,分别设置了 MenuBar 菜单栏,工具栏 ToolBar 布置在主界面的北部,画板 DrawBoard 类布置在主界面的中部,画笔位置坐标栏 PositionLabel 类布置在主界面的南部。附加功能的工具栏放在主界面我的西部。 系统设计部分的原型界面就是根据这个界面布局去做的。

2) Panel(工具组成面板 ToolBar)

构思:包括 ColorsPanel(画图对象颜色面板)、ShapesPanel(画图对象形状面板)、StrokesPanel(画图对象线段粗细面板)、ShapesQueueUpdate(画图对象操作面板),这里还有一个 Function 类是作为一个功能函数类去使用的,里面是一些画图过程中会使用到的静态方法,其中 ColorsPanel、ShapesPanel、StrokesPanel 都继承 Jpanel。

ShapesQueueUpdate 静态导入 Function 类,会使用到它的一些静态变量,这里它是作为我的一个拓展功能区做的,可以实现对已画成的图撤销、恢复、对画图中间的一些图形实行修改或剪裁的功能。

3） 画图面板(DrawBoard)

构思:画板的实现想通过鼠标监听事件再借用 AWT 重型组件的重绘来完成的画图。

4） 菜单条(MenuBar)

构思:菜单条 MenuBar 这里打算设置 5 个 JMenu,分别为文件(FileMenu)、编辑(EditMenu)、帮助(HelpMenu)、图片(PictureMenu)。后面的个人菜单(PersonalMenu)打算最后整体做的差不多了再进行拓展。

5） 画笔位置坐标栏(PositionLabel)

构思: PositionLabel 类想做的简单点，就是 Jlabel 通过画板部分的鼠标监听器显示不同鼠标操作时的坐标状态。

2. 功能描述

1） 图形绘制的 draw 方法

思路:因为 Java 里的 Graphics 类中有一些基础图形的绘制方法已经有就可以直接调用，但是个别图形相比较已有的图形绘制方法，如曲线、橡皮擦、喷漆、多边形可能是没有的，对于曲线和橡皮擦的处理这里是利用了数学里微分的思想，把一条曲线无线去微分，可以看做是由很多条首位相接的短小直线拼接而成。对于喷漆实际上是在一个指定半径的圆内随机地画点，但是 java 里没有画点的办法，这里采用画直线的方法，只不过直线的起点和终点一样的时候，所画出的就是点。对于多边形，实际上还是计算各个点的坐标，这里可能会用到几何的一些知识，去构造直角三角形，利用正弦和余弦去计算一些点的坐标。一些特殊的正方形和圆是在矩形和椭圆的基础上对起点和终点取了最小值和最大值来绘制的。

2） 图形重绘

思路:这里是打算把用户所选择的每一个图形都放置到一个队列里连续顺序入队，而每次用户绘制图形时都会调用 Awt 重型组件中的 repaint()函数把队列中的图形对象都重绘一遍。

3） 鼠标监听

功能: 监听鼠标动作并做出相应回应，鼠标动作包括鼠标左键按下、鼠标拖动和鼠标左键释放、鼠标在同一坐标单击。

调用关系:当鼠标指针移动到画板部分时坐标栏显示鼠标位置坐标,按下鼠标左键会在画板留下所画图形起点,拖动时图形大小会动态随鼠标指针移动发生变化。

4） 改变图形、颜色、粗细

思路:通过监听器来监听按钮，来改变 Shape 子类本身，对象中的 Color、thickness 值。

5） 图形回退、恢复、删除、复原

功能:鼠标单击工具栏上面的回退按钮画板上最后一个所绘制的图形会隐

藏，同时单击恢复按钮这个图形有会出现，如果单击删除则这个图形不可复原。

思路:这里是通过改变队列里元素的当前末尾指针所指向的位置，并且重绘来实现图形回退、恢复、删除、复原。

6） **文件新建、保存、打开**

思路:用户点击新建按钮会将画板清理会空白并且不可恢复，用户点击保存会将画板上的图形以对象输出流的形式保存到本地，同时打开按钮会将之前保存在本地的输出流文件已对象输入流的方式读取，再构造出新的对象进入队列再重绘，就会在画板上显示新的图形。

7） **图片打开、保存**

思路:这里是借用了 java 里的 ImageIO 类对 BufferedImage 来进行读写。保存采用 Rectangle 类对指定的矩形区域用 Robot 的截屏方法去截取保存成指定格式的图片。

8） **画板图形的捕捉框选**

思路:需要给抽象模型 Shape 再设立一些边沿轮廓点用户用户判断图形的边框，可能需要判断图形的实际坐标与用户鼠标坐标的关系，当前鼠标坐标是否在某个图形边沿或图形内。框选是根据所做出的框选矩形，将坐标在此矩形范围内的图形全部放入一个 Shape 的集合里，统一进行操作。

9） **画板图形的平移、旋转、放缩**

思路:这个功能需要建立在图形已被选定，本质都是根据当前鼠标的坐标变化情况来改变图形的属性坐标，这里对象可以是一个图形也可以是一个图形的集合，最后在通过重绘，就能达到所要效果。

10) **用户与作图文件的绑定**

思路:因为用户最后所做的 Shape 队列中的对象的属性信息每条是按顺序存放进去的。所以这里 User、Shape 实体可以设置一些公共关键字如用户 id 和图 id，在读取用户个人作图信息的时候就按照用户的 id 和每张图的 id，在查询的时候做自然连接，就可加载出用户的个人图片信息。

三、系统实现

3.1 关键模块/函数的实现

3.1.1 直线绘制模块

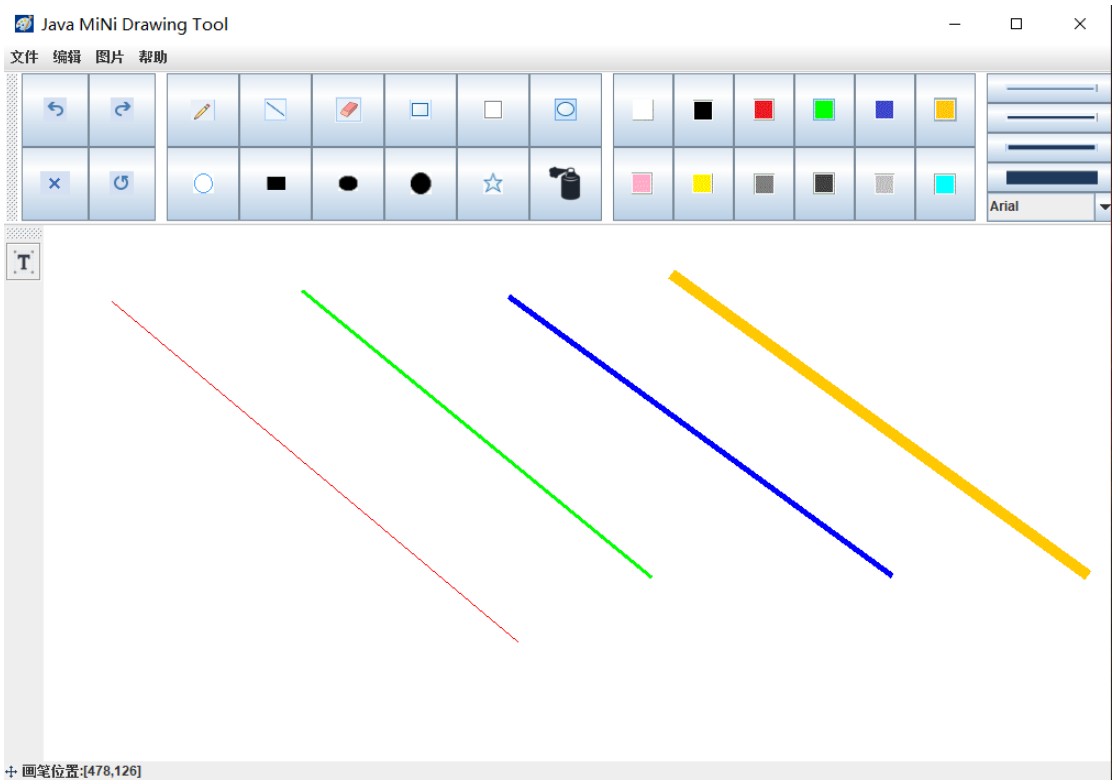


图 3-1 直线绘制案例

（1）程序框图

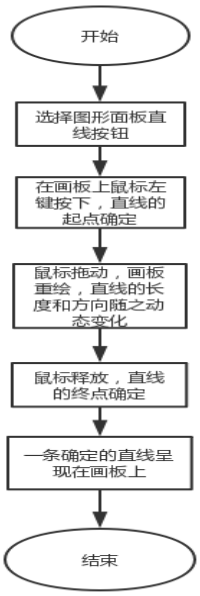


图 3-2 直线绘制模块程序流程图

（2）函数原型：

```
public void repaint()
protected void paintComponent(Graphics g)
public void drawLine(int x1, int y1, int x2, int y2)
```

（3）函数功能：

在此画图板的坐标系中，使用当前颜色在点（x1，y1）和（x2，y2）之间画一条线。

（4）参数：

x1 - 第一个点的 x 坐标。

y1 - 第一个点的 y 坐标。

x2 - 第二个点的 x 坐标。

y2 - 第二个点的 y 坐标。

shapesQueue[index] - 当前图形队列中 index 所指向的图形对象

3.1.2 矩形绘制模块

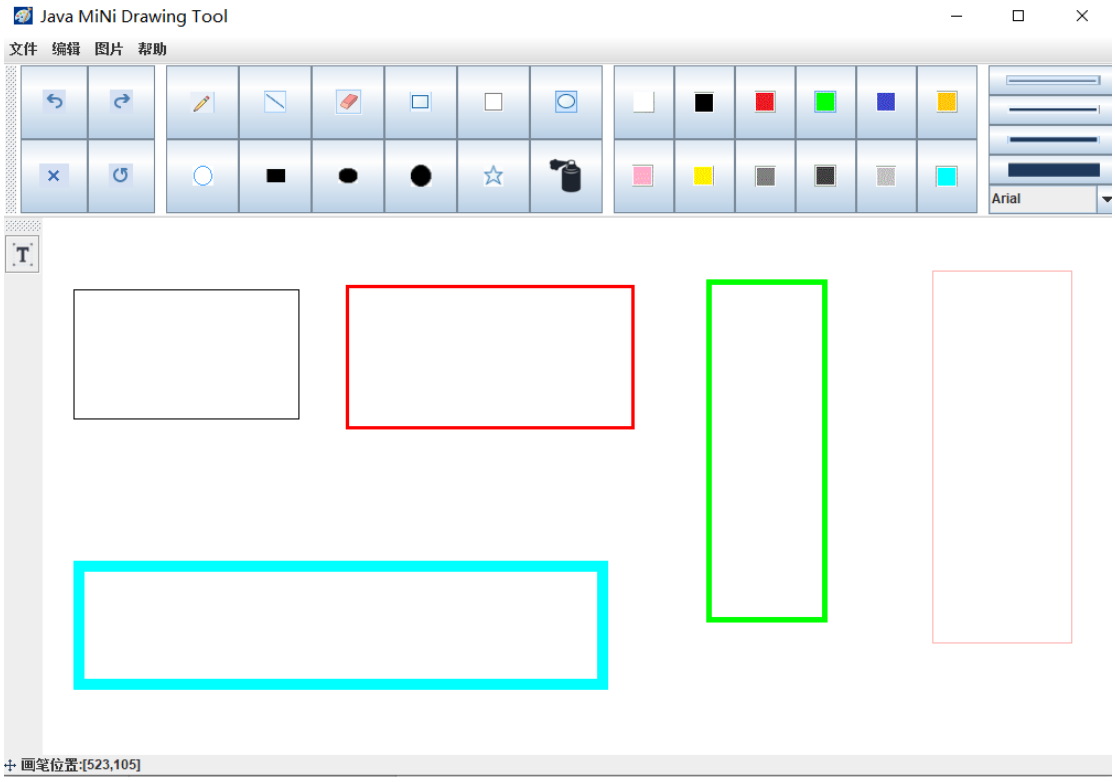


图 3-3 矩形绘制案例

（1） 程序框图

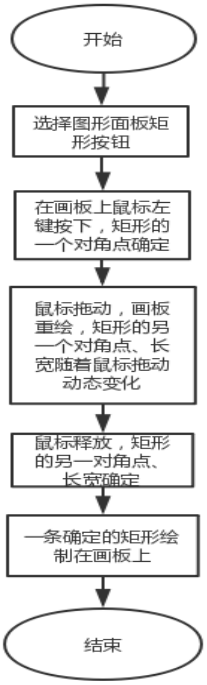


图 3-4 矩形绘制模块程序流程图

（2）函数原型：

```
public void repaint()
protected void paintComponent(Graphics g)
public void drawRect(int x, int y, int width, int height)
```

（3）函数功能：

此画图板坐标系中，绘制指定矩形的边框。矩形的左边缘和右边缘分别位于 x 和 x + width。上边缘和下边缘分别位于 y 和 y + height。使用图形上下文的当前颜色绘制该矩形。

（4）参数：

x - 要绘制矩形的 x 坐标。
y - 要绘制矩形的 y 坐标。
width - 要绘制矩形的宽度。
height - 要绘制矩形的高度。
shapesQueue[index] - 当前图形队列中 index 所指向的图形对象

3.1.3 椭圆绘制模块

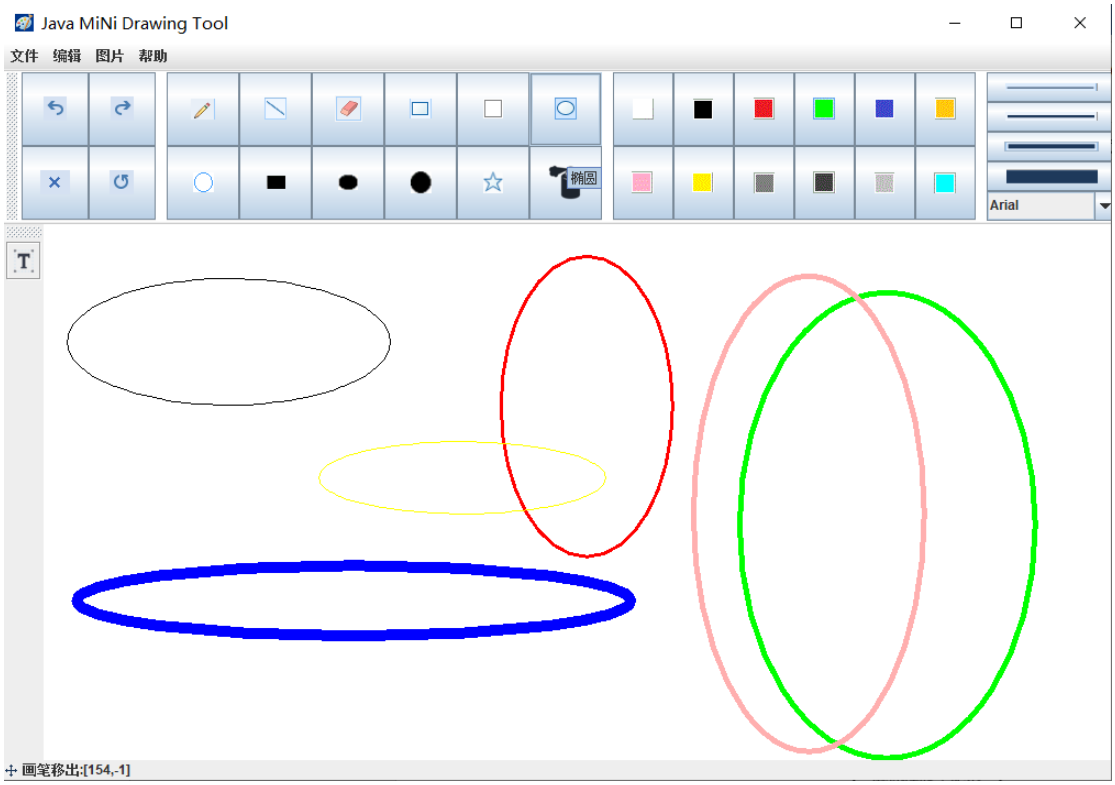


图 3-5 椭圆绘制案例

（1）程序框图

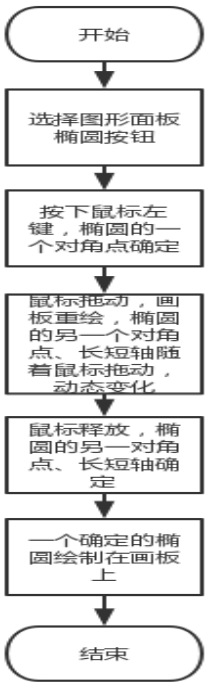


图 3-6 椭圆绘制模块程序流程图

（2） 函数原型：

```
public void repaint()
```

protected void paintComponent(Graphics g)
public void drawOval(int x, int y, int width, int height)

（3）函数功能：

此画图板坐标系中，绘制椭圆的边框。得到一个圆或椭圆，它刚好能放入由 x、y、width 和 height 参数指定的矩形中。 椭圆覆盖区域的宽度为 width + 1 像素，高度为 height + 1 像素。

（4）参数：

x – 要绘制椭圆的左上角的 x 坐标。
y – 要绘制椭圆的左上角的 y 坐标。
width – 要绘制椭圆的长轴。
height – 要绘制椭圆的短轴。
shapesQueue[index] – 当前图形队列中 index 所指向的图形对象

3.1.4 曲线绘制模块

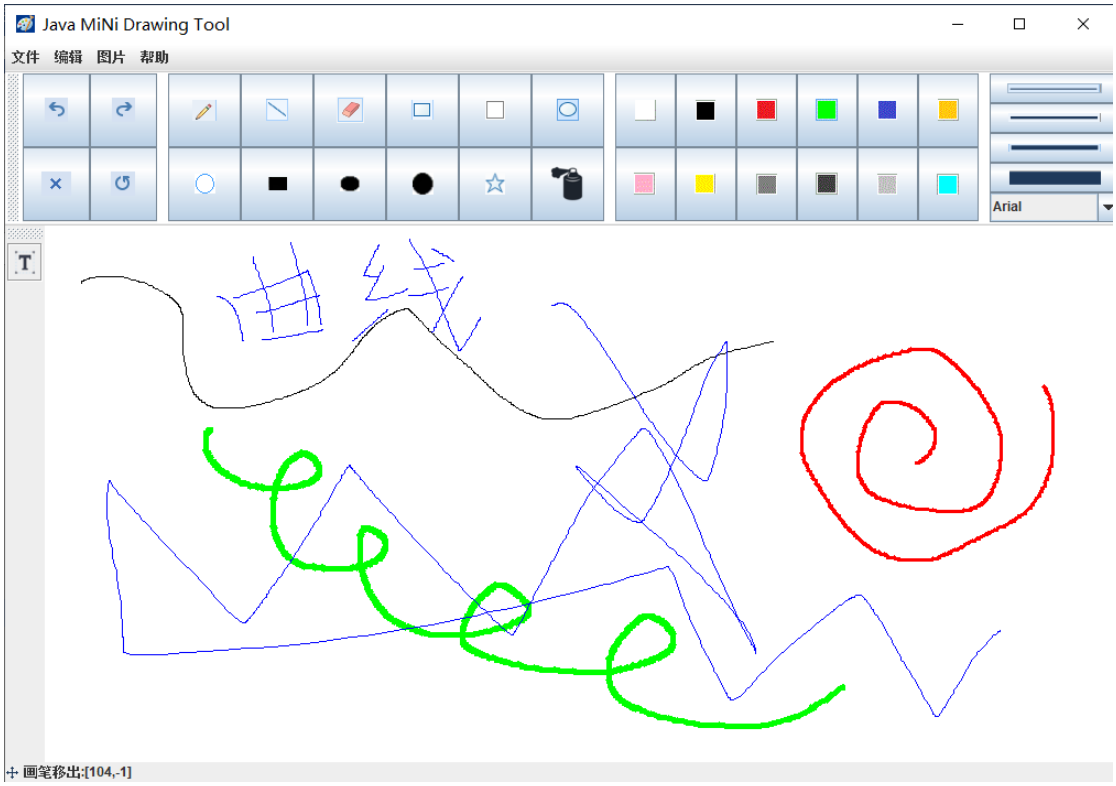


图 3-7 曲线绘制案例

（1） 程序框图

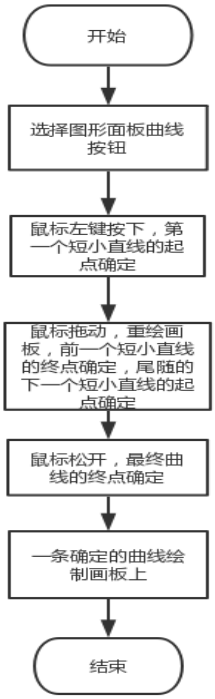


图 3-8 曲线绘制模块程序流程图

（2） 函数原型：

```
public void mouseDragged(MouseEvent e)
public void repaint()
protected void paintComponent(Graphics g)
public void drawLine(int x1, int y1, int x2, int y2)
```

（3）函数功能：

此画图板坐标系中，使用当前颜色在点 (x1, y1) 和 (x2, y2) 之间画一条短小线。在画板拖动时是连续的短小直线会拼凑成一条连续的曲线。

（4）参数：

x1 - 第一个点的 x 坐标。
y1 - 第一个点的 y 坐标。
x2 - 第二个点的 x 坐标。
y2 - 第二个点的 y 坐标。
shapesQueue[index] - 当前图形队列中 index 所指向的图形对象

3.1.5 喷漆绘制模块

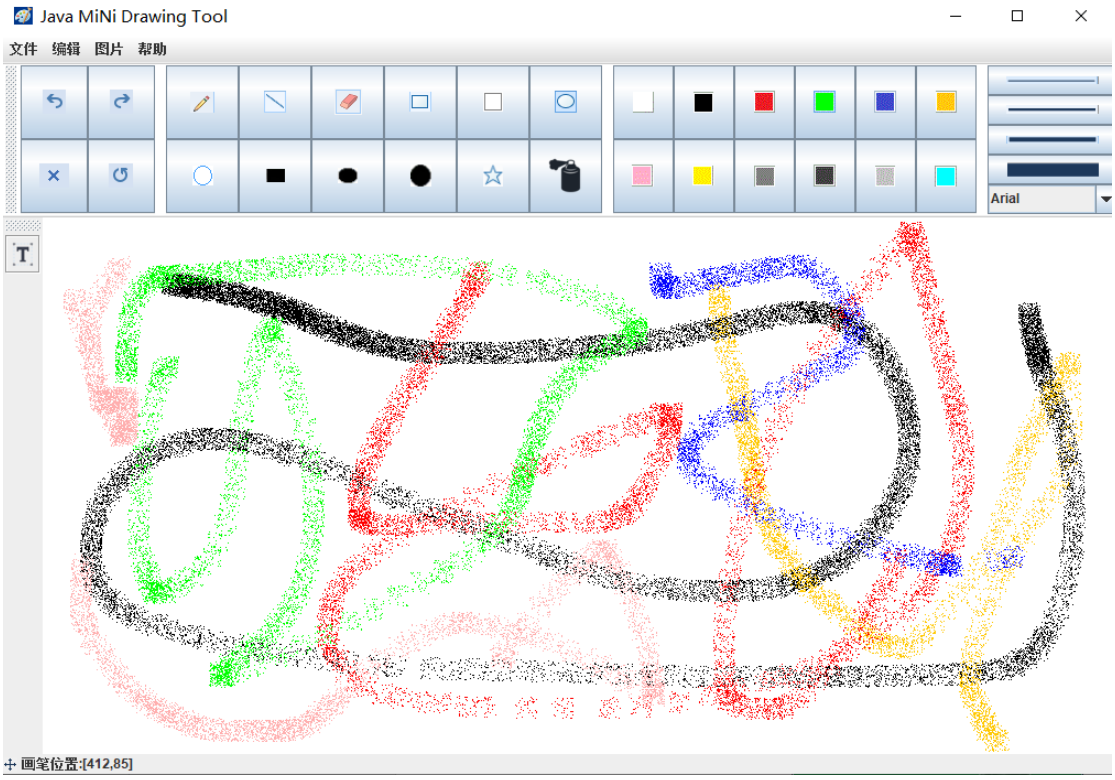


图 3-9 喷漆绘制案例

（1）程序框图

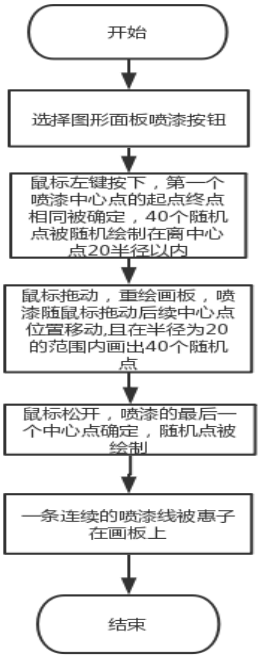


图 3-10 喷漆绘制模块程序流程图

（2） 函数原型：

```
public void mouseDragged(MouseEvent e)
public void repaint()
protected void paintComponent(Graphics g)
public void drawLine(int x1, int y1, int x2, int y2)
```

（3） **函数功能：**

此画图板坐标系中，这里 x1=x2,y1=y2。，使用当前颜色在点 (x1, y1) 和 (x2, y2) 之间画出一个点。在拖动时连续的半径内的随机点被绘制呈现出喷漆的形态。

（4）**参数：**

x1 - 第一个点的 x 坐标。
y1 - 第一个点的 y 坐标。
x2 - 第二个点的 x 坐标。
y2 - 第二个点的 y 坐标。
shapesQueue[index] - 当前图形队列中 index 所指向的图形对象

3.1.6 五角星绘制模块

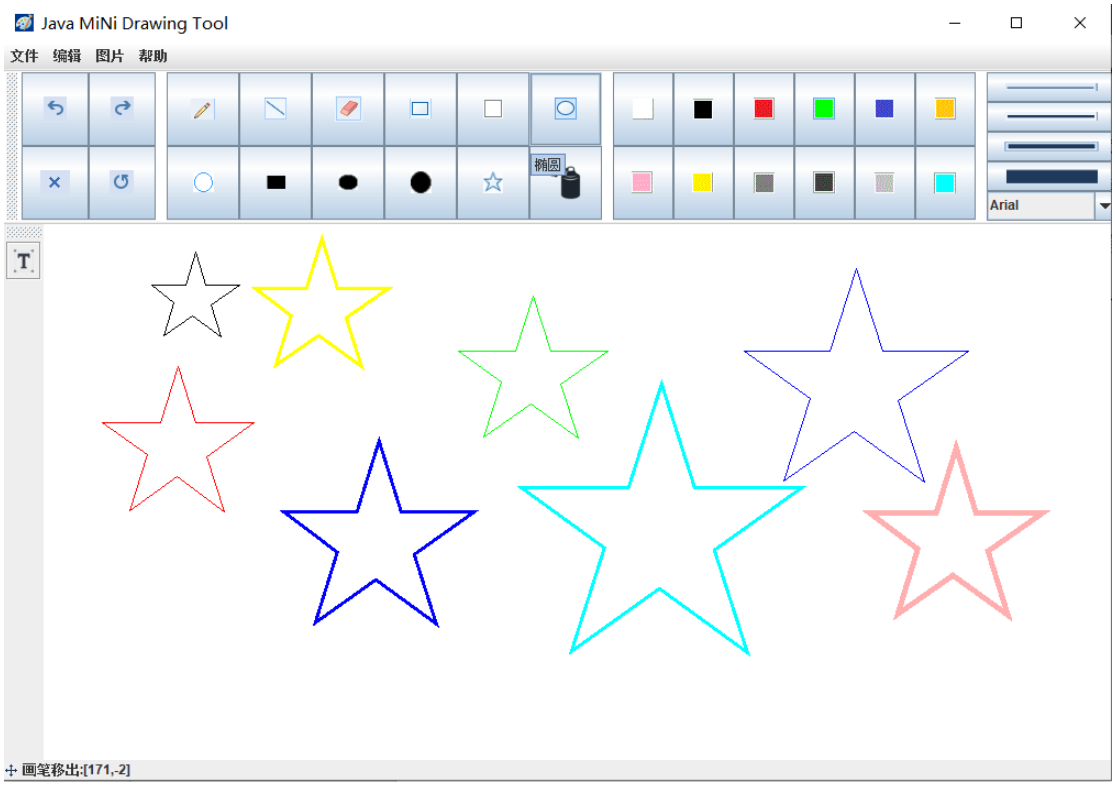


图 3-11 五角星绘制案例

（1）程序框图

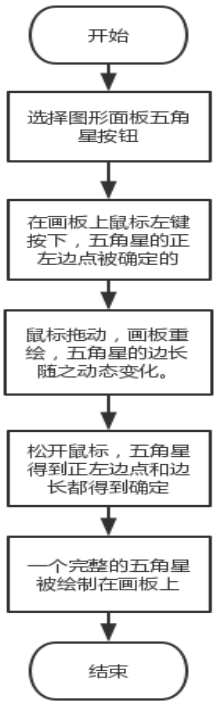


图 3-12 五角星绘制模块程序流程图

（2）函数原型：

public void repaint()

protected void paintComponent(Graphics g)

private void initPoints()

public void drawPolygon(int xPoints[], int yPoints[],int nPoints)

（3）函数功能：

此画图板坐标系中，绘制一个由 x 和 y 坐标数组定义的闭合多边形。每对 (x, y) 坐标定义一个点。

此方法绘制由 nPoint 个线段定义的多边形，其中前 nPoint - 1 个线段是 1 ≤ i ≤ nPoints 时从 (xPoints[i - 1], yPoints[i - 1]) 到 (xPoints[i], yPoints[i]) 的线段。如果最后一个点和第一个点不同，则图形将在这两点间绘制一条线段来自动闭合。

（4） 参数：

xPoints - 多边形 x 坐标的数组。

yPoints - 多边形 y 坐标的数组。

nPoints - 多边形点的总数。

shapesQueue[index] - 当前图形队列中 index 所指向的图形对象

3.1.7 文本绘制模块

图 3-13 文本绘制案例

（1）程序框图

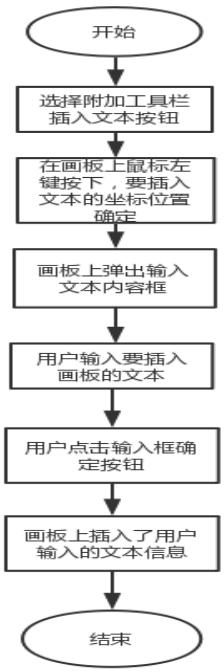


图 3-14 文本绘制模块程序流程图

（2） 函数原型：

```
public void repaint()
protected void paintComponent(Graphics g)
```

public void drawString(String str, int x, int y)

（3） **函数功能：**

此画图板坐标系中使用此图形上下文的当前字体和颜色绘制由指定 string 给定的文本。最左侧字符的基线位于此图形上下文坐标系的 (x, y) 位置处。

（4）**参数：**

str - 要绘制的 string。

x - x 坐标。

y - y 坐标。

shapesQueue[index] - 当前图形队列中 index 所指向的图形对象

3.1.8 正方形绘制模块

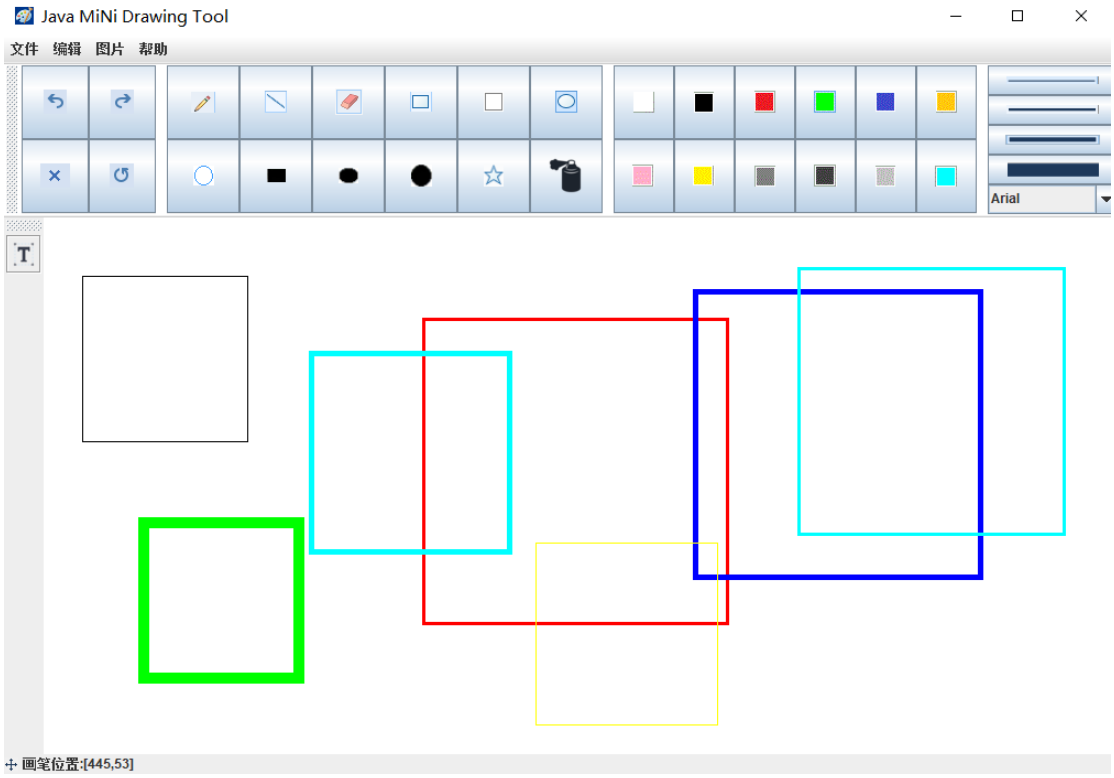


图 3-15 正方形绘制案例

（1）程序框图

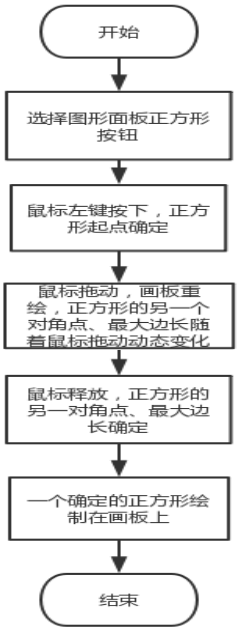


图 3-16 正方形绘制模块程序流程图

（3） 函数原型：

```
public void repaint()
protected void paintComponent(Graphics g)
public void drawRect(int x, int y, int edgeLen, int edgeLen)
```

（3）函数功能：

此画图板坐标系中绘制指定正方形的边框。正方形的左边缘和右边缘分别位于 x 和 x + edgeLen。上边缘和下边缘分别位于 y 和 y + edgeLen。使用图形上下文的当前颜色绘制该正方形。

（4）参数：

x - 要绘制矩形的 x 坐标。

y - 要绘制矩形的 y 坐标。

edgeLen - 要绘制正方形的最大边长。

shapesQueue[index] - 当前图形队列中 index 所指向的图形对象

3.1.9 圆形绘制模块

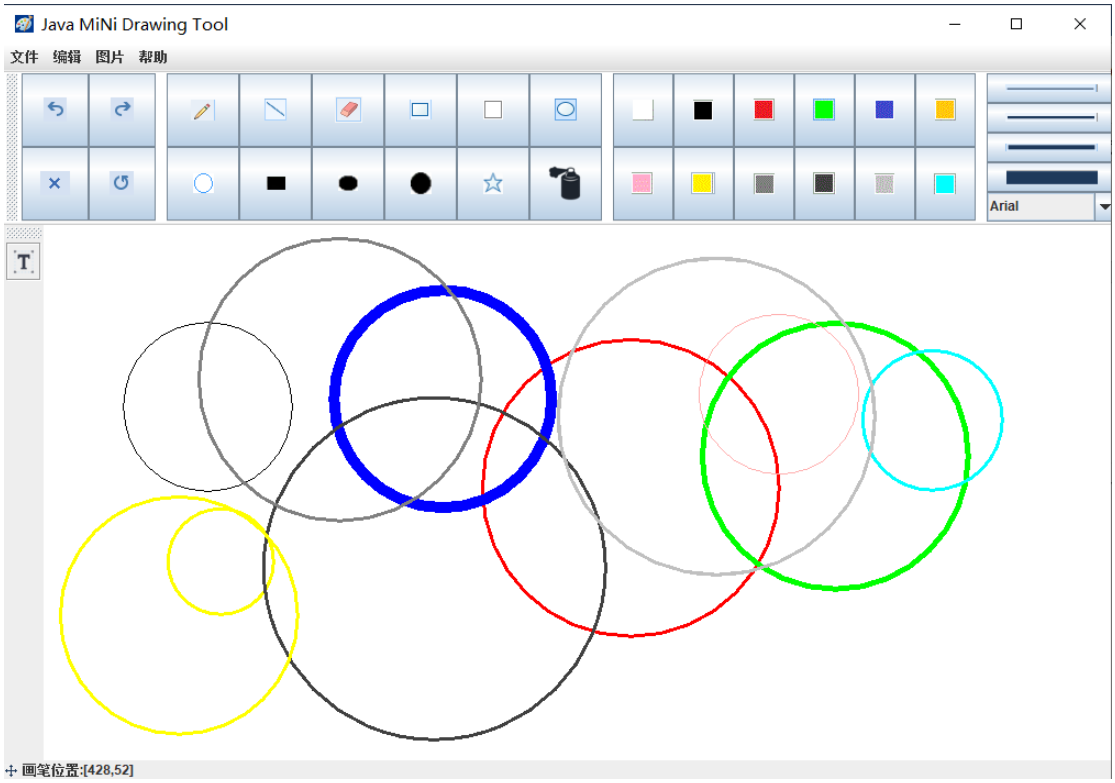


图 3-17 圆形绘制案例

（1）程序框图

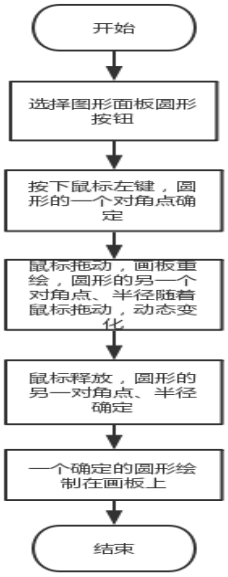


图 3-18 BOSS 圆形绘制模块程序流程图

（2）函数原型：

```
public void repaint()
protected void paintComponent(Graphics g)
public void drawOval(int x, int y, int diameter, int diameter)
```

（3）函数功能：

此画图板坐标系中，绘制圆形的边框。得到一个圆，它刚好能放入由 x、y、diameter、diameter 参数指定的正方形中。 圆形覆盖区域的直径为 diameter+1 像素。

（4）参数：

x - 要绘制椭圆的左上角的 x 坐标。

y - 要绘制椭圆的左上角的 y 坐标。

diameter - 要绘制圆形的直径。

shapesQueue[index] - 当前图形队列中 index 所指向的图形对象

3.1.10 橡皮擦绘制模块

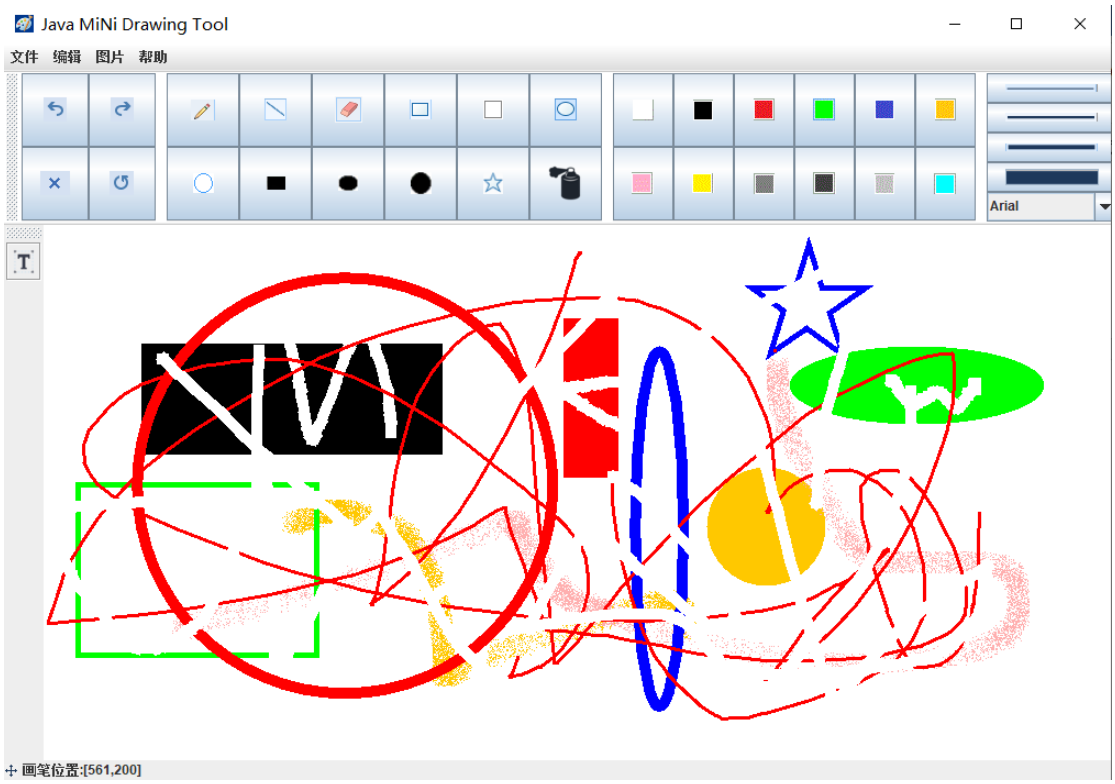


图 3-19 BOSS 橡皮擦绘制案例

（1）程序框图

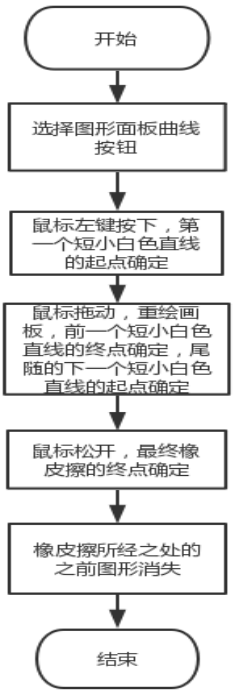


图 3-20 橡皮擦绘制模块程序流程图

（2）函数原型：

```
public void mouseDragged(MouseEvent e)
public void repaint()
protected void paintComponent(Graphics g)
public void drawLine(int x1, int y1, int x2, int y2)
```

（3）函数功能：

在此画图板的坐标系中，使用当前颜色在点 (x1, y1) 和 (x2, y2) 之间画一条短小白线。在画板拖动时是连续的短小白线会拼凑成一条连续的白色曲线。白色曲线覆盖掉了所经过的图形，成为用户所看到的橡皮擦擦去了之前画上去的图形。

（4）参数：

x1 - 第一个点的 x 坐标。
y1 - 第一个点的 y 坐标。
x2 - 第二个点的 x 坐标。
y2 - 第二个点的 y 坐标。
shapesQueue[index] - 当前图形队列中 index 所指向的图形对象

3.1.11 画笔图形选择模块

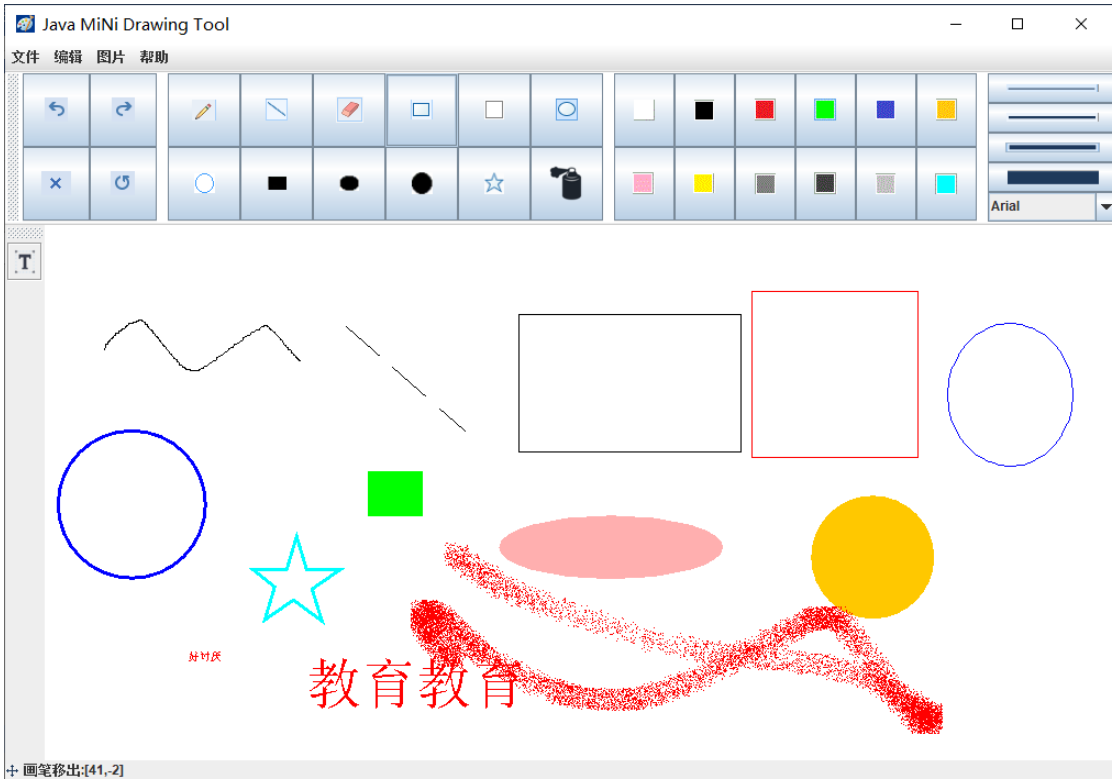


图 3-21 画笔图形选择案例

（1）程序框图

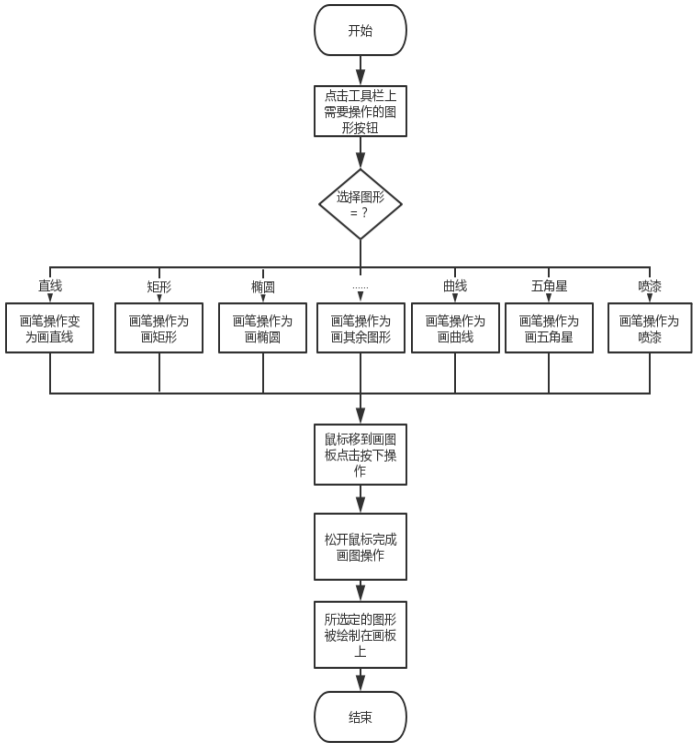


图 3-22 画笔图形选择模块程序流程图

（2）函数原型：

public static void creatAction()
public void actionPerformed(ActionEvent e)

（3）函数功能：

点击工具栏上的图形面板按钮切换即将在画板上所绘制的图形类型。

（4）参数：

shapeChoice - 用户所选择图形编号
shapesQueue[index] - 当前图形队列中 index 所指向的图形对象
maxShapesCount - 用户在画板上所绘制的最大图形对象数目
ActionEvent e - 点击动作事件

3.1.12 画笔颜色选择模块

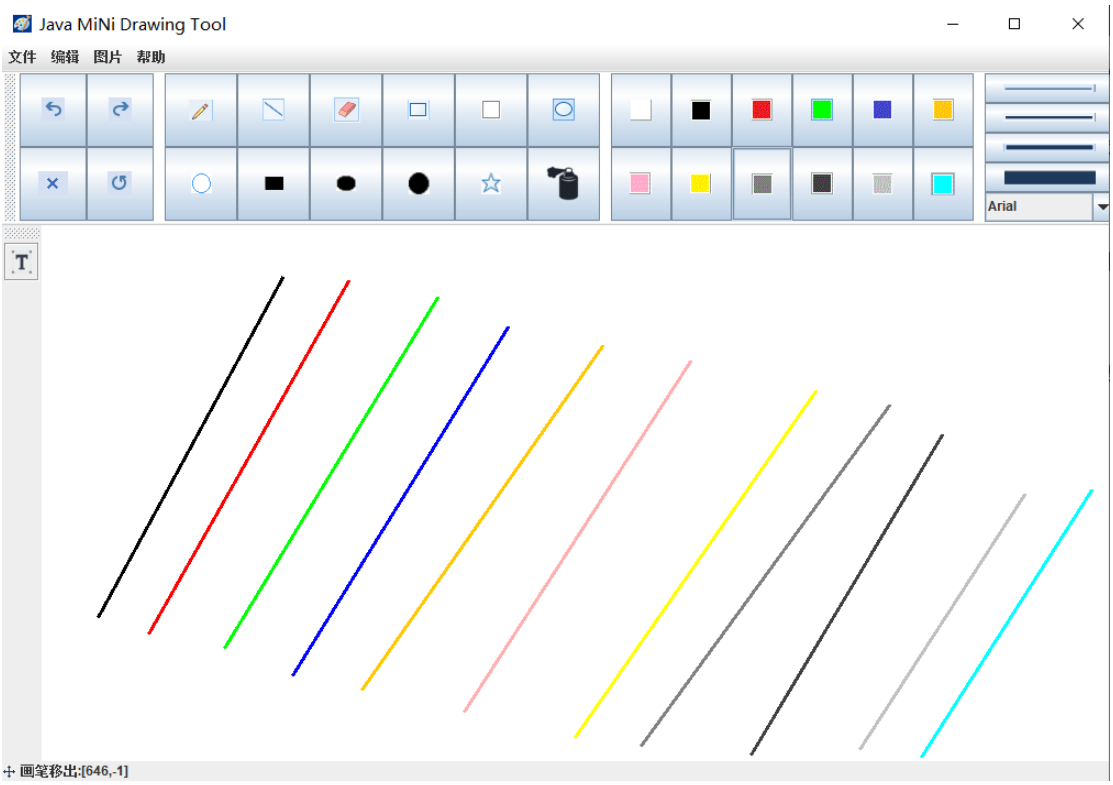


图 3-23 画笔颜色选择模块程序流程图

（1）程序框图

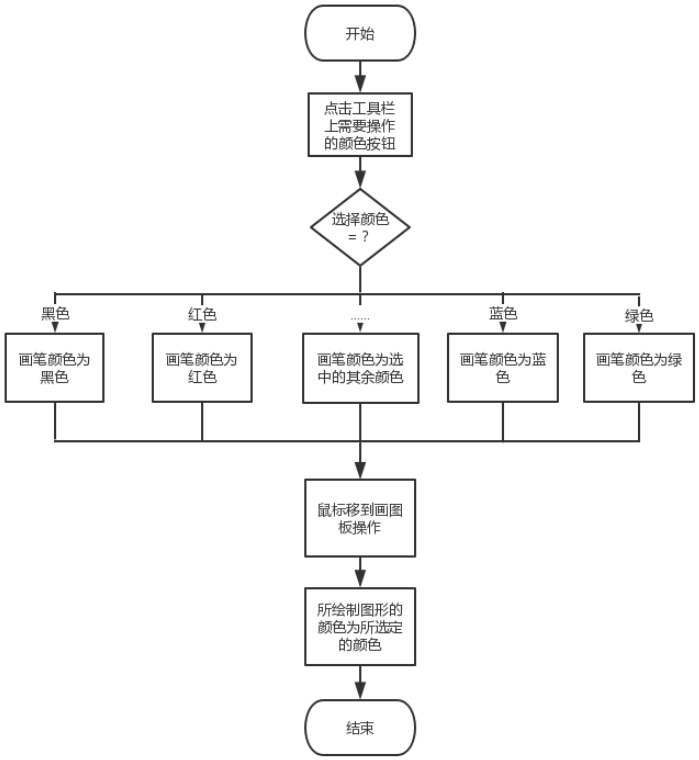


图 3-24 画笔颜色选择模块程序流程图

（2）函数原型：

```
protected static void newColor()
public void actionPerformed(ActionEvent e)
```

（3）函数功能：

点击工具栏上的颜色面板按钮切换即将在画板上所绘制的图形的颜色。

（4）参数：

colorChoice – 用户所选择的颜色编号

ActionEvent e – 点击动作事件

3.2.13 画笔粗细选择模块

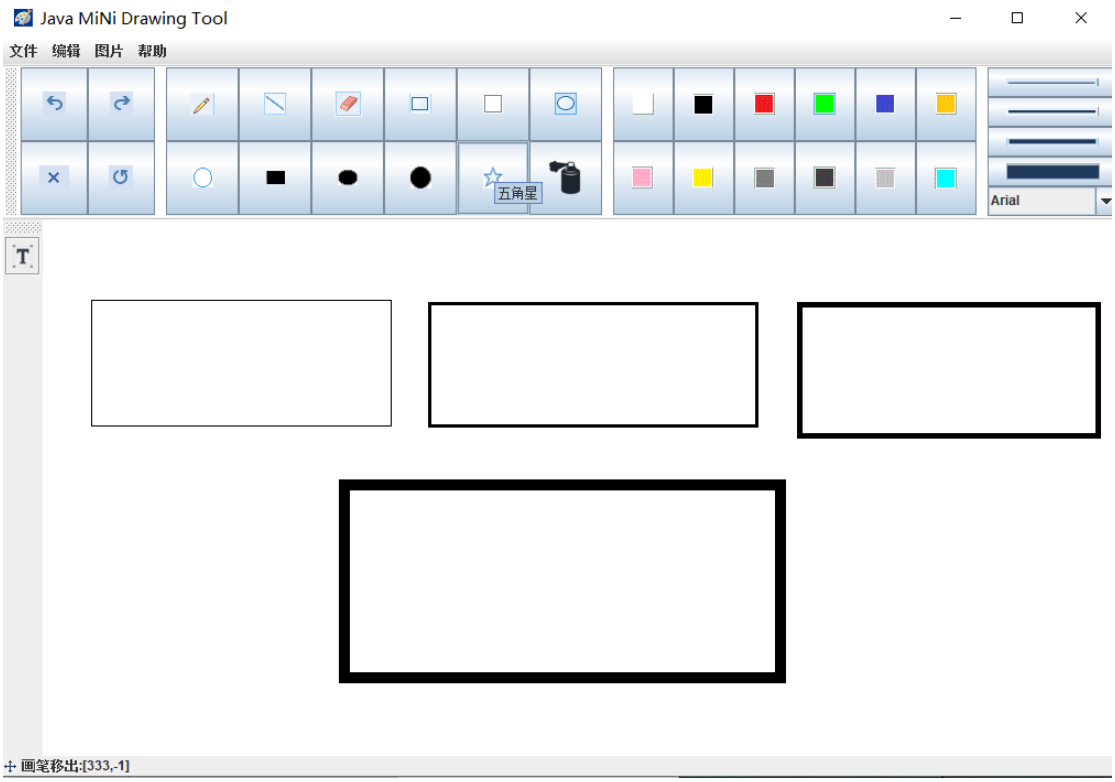


图 3-25 画笔粗细选择案例

（1）程序框图

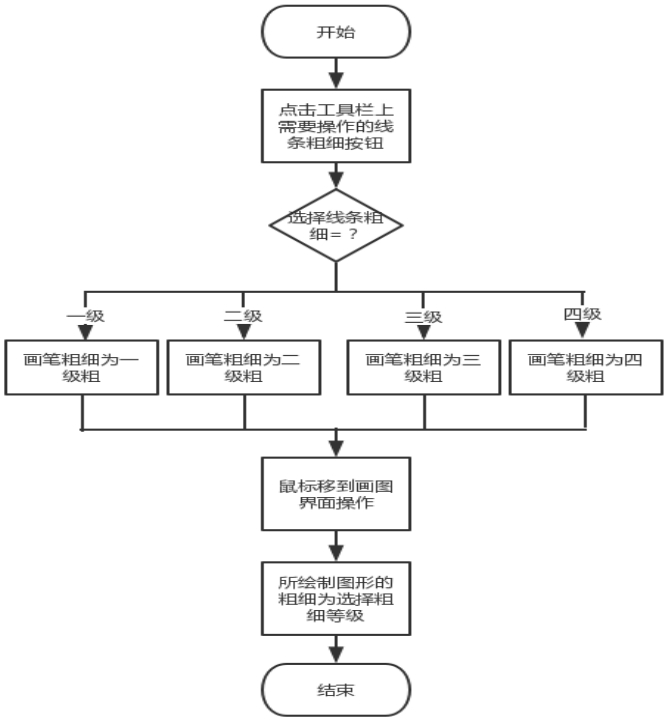


图 3-26 画笔粗细选择模块程序流程图

（2）函数原型：

```
protected static void newStroke()
```

public void actionPerformed(ActionEvent e)

（3）函数功能：

点击工具栏上的粗细面板按钮切换即将在画板上所绘制的图形的线条粗细程度。

（4）参数：

strokeChoice - 用户所选择画笔线条粗细编号

ActionEvent e - 点击动作事件

3.1.14 撤销图形模块

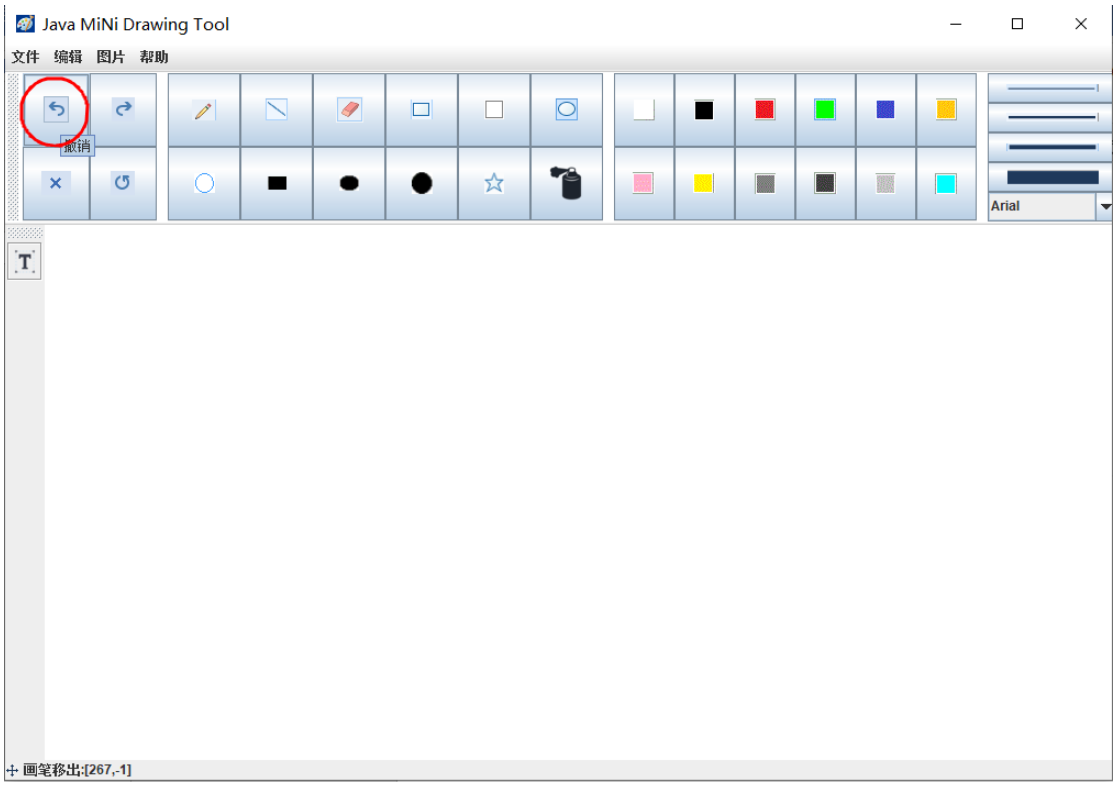


图 3-27 撤销图形案例

（1）程序框图

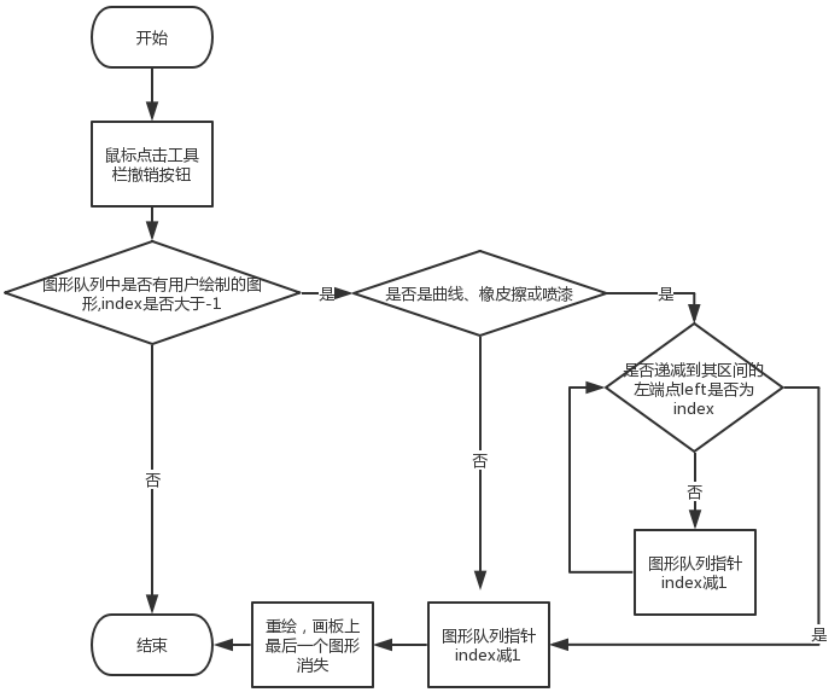


图 3-28 撤销图形模块程序流程图

（2）函数原型：

```
private void revoke()
public void repaint()
protected void paintComponent(Graphics g)
```

（3）函数功能：

将当前画板上用户最后一个绘制的图形从画板上撤销，消失不见。

（4）参数：

index - shapesQueue[]队列中当前所指向的对象在队列中的序号

shapesQueue - 用户在画板上所绘制的图形按照先后顺序继承的 shape 队列集合

3.1.15 恢复图形模块

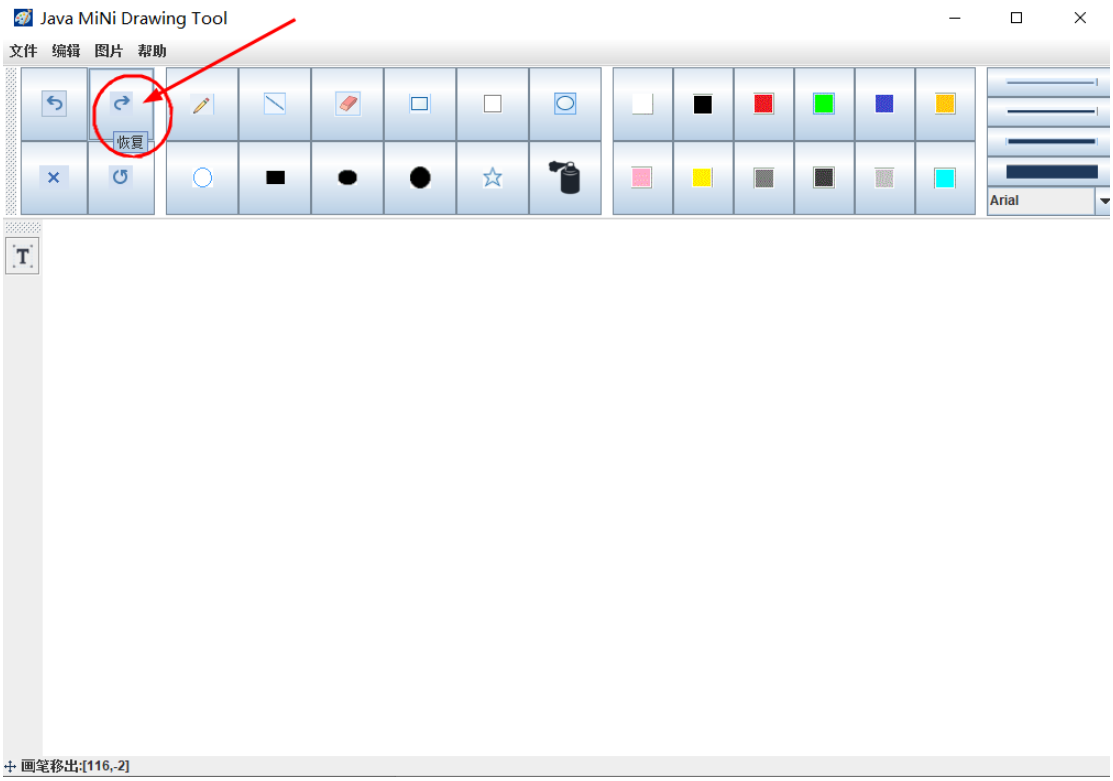


图 3-28 恢复图形案例

（1）程序框图

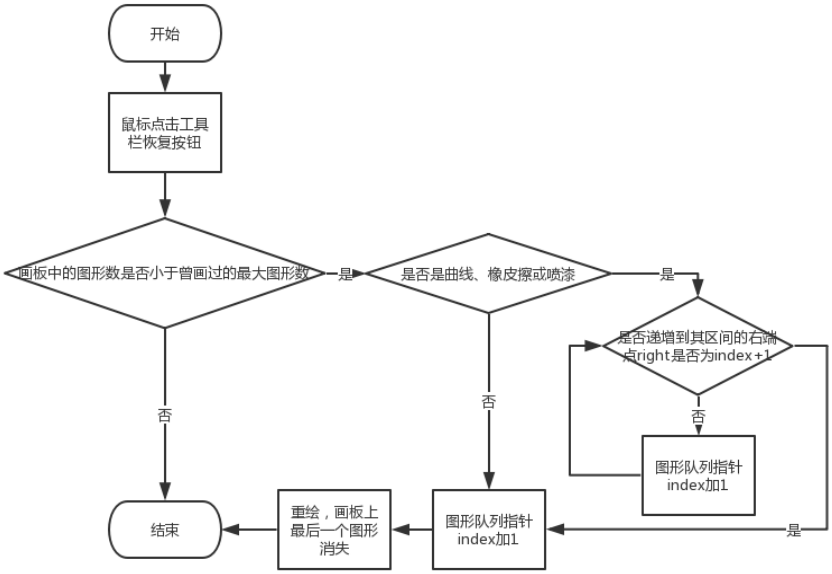


图 3-29 恢复图形模块程序流程图

（2）函数原型：

```
private void recover()
public static void queueUpdate(int pastIndex,int newIndex)
public void repaint()
protected void paintComponent(Graphics g)
```

（3）函数功能：

将当前画板上用户之前撤销的最后一个图形从画板上恢复，重新呈现在画板上。

（4）参数：

index - shapesQueue[]队列中当前所指向的对象在队列中的序号

shapesQueue - 用户在画板上所绘制的图形按照先后顺序继承的 shape 队列集合

left -作为一个记录图形区间边界的临时变量

right - 用来记录中间因为插入图形而截断的图形的右区间序号

3.1.16 删除图形模块

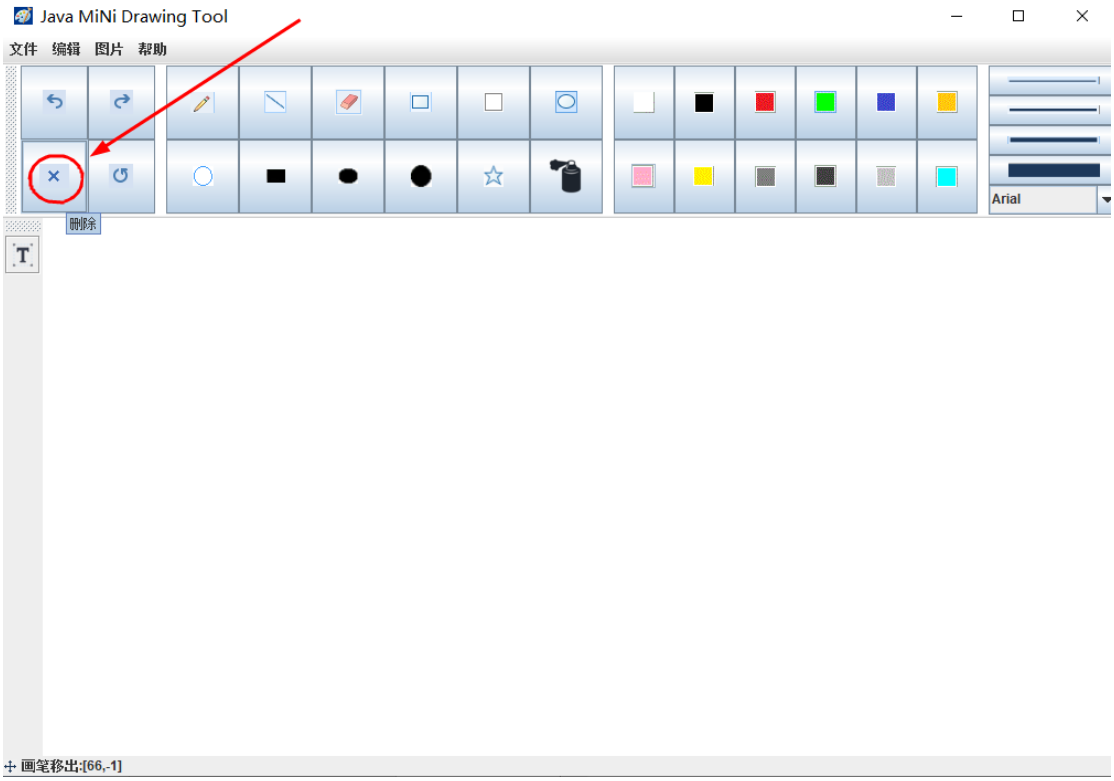


图 3-30 删除图形案例

（1）程序框图

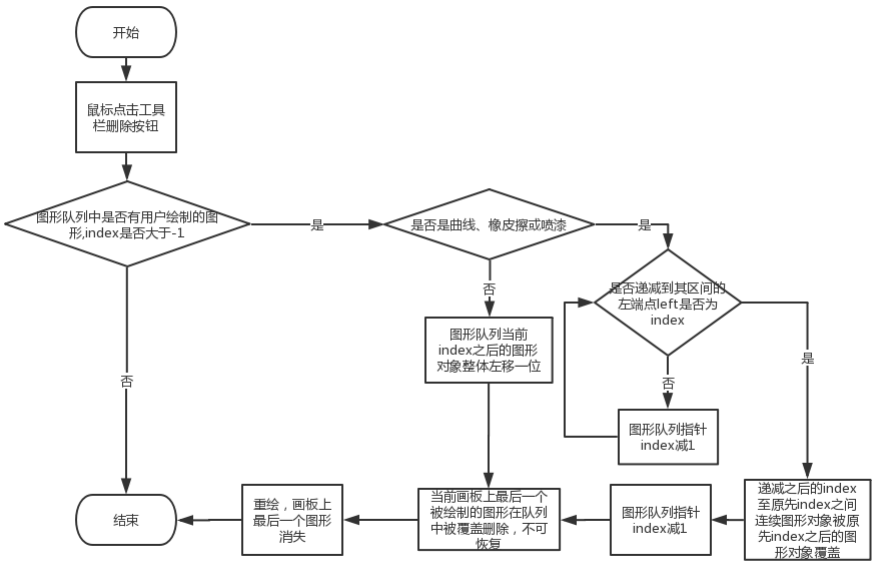


图 3-31 删除图形模块程序流程图

（2）函数原型：

```
private void delete()
public static void queueUpdate(int pastIndex,int newIndex)
public void repaint()
protected void paintComponent(Graphics g)
```

（3）函数功能：

将当前画板上用户最后一个绘制的图形从画板上删除，消失不见，并且不可恢复。

（4）参数：

index - shapesQueue[]队列中当前所指向的对象在队列中的序号

shapesQueue - 用户在画板上所绘制的图形按照先后顺序继承的 shape 队列集合

left -用来记录所要删除的多对象图形的区间左端点

right - 用来记录所要删除的多对象图形的区间右端点

3.1.17 还原图形模块

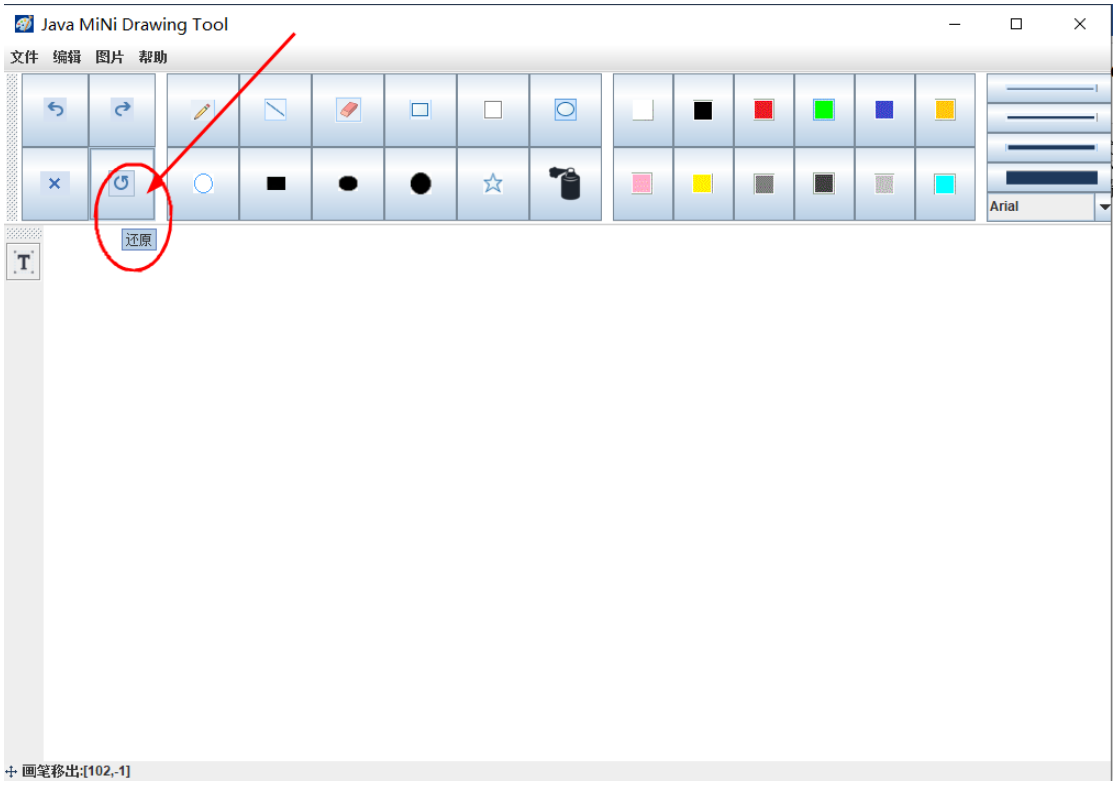


图 3-32 还原图形案例

（1）程序框图

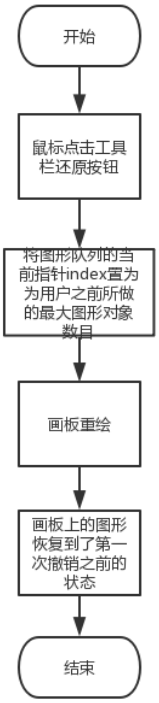


图 3-33 还原图形模块程序流程图

（2）函数原型：

```
private void restore()
```

```
public static void queueUpdate(int pastIndex,int newIndex)
public void repaint()
protected void paintComponent(Graphics g)
```

（3）函数功能：

将当前画板上用户最后一个绘制的图形从画板上删除，消失不见，并且不可恢复。

（4）参数：

index - shapesQueue[]队列中当前所指向的对象在队列中的序号

shapesQueue - 用户在画板上所绘制的图形按照先后顺序继承的 shape 队列集合

left -作为一个记录图形区间边界的临时变量

right - 用来记录中间因为插入图形而截断的图形的右区间序号

3.1.18 文件菜单栏新建模块

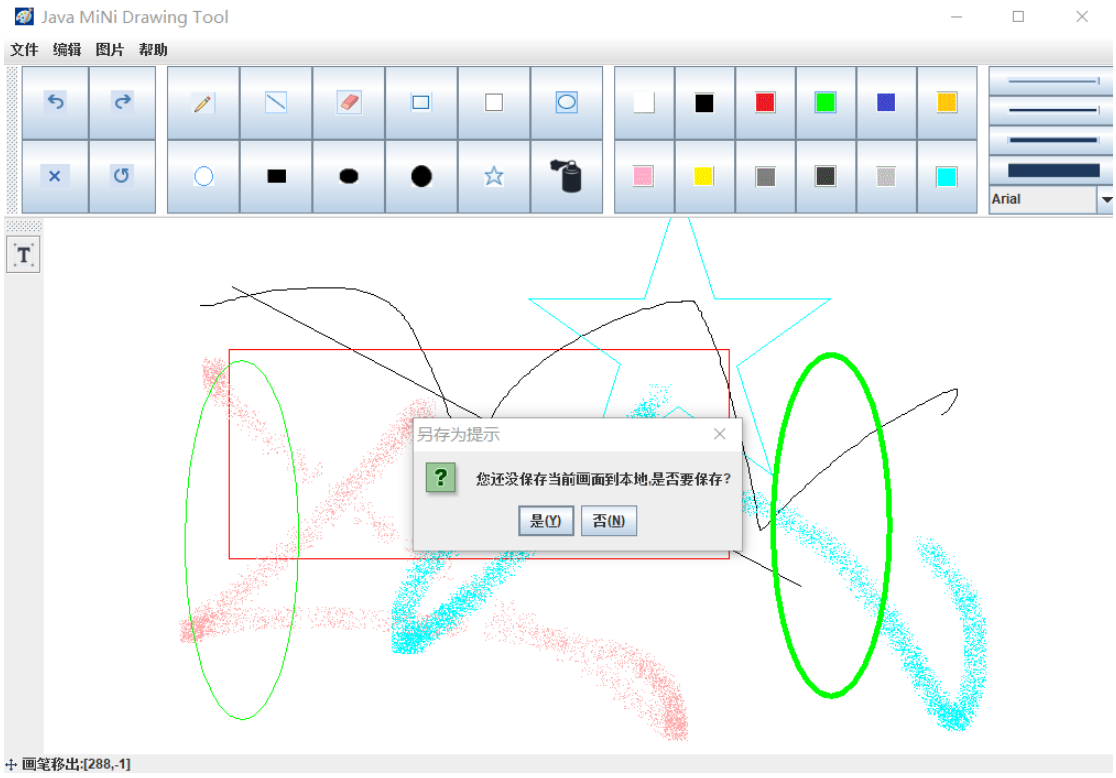


图 3-34 文件菜单栏新建案例

（1）程序框图

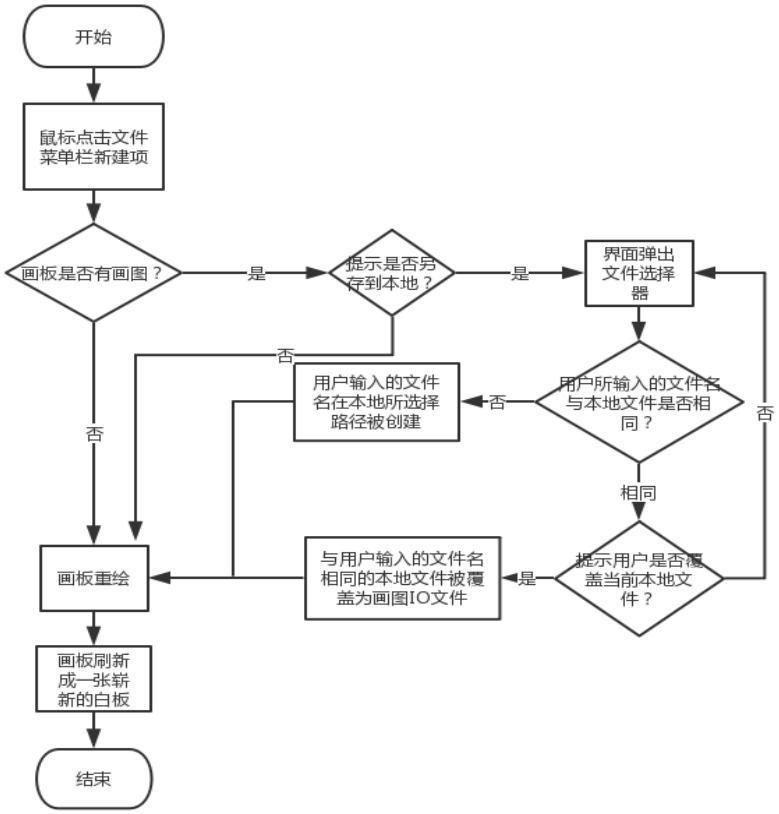


图 3-35 文件菜单栏新建模块程序流程图

（2）函数原型：

```
public void isSave(boolean flag)
public void newFile()
public void repaint()
```

（3）函数功能：

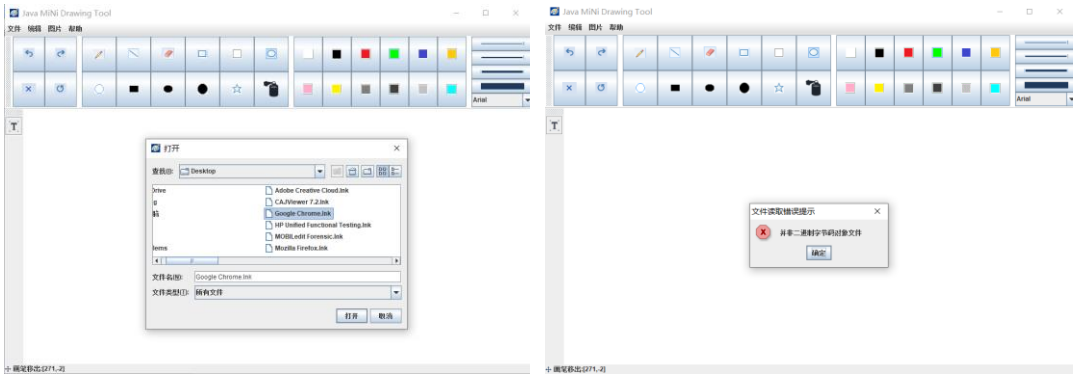
将画板刷新成一张崭新的白板的同时，提示是否保存画板图像信息到本地。

（4）参数：

image - 画板上之前打开的图片的 BufferedImage 对象句柄
flag - 保存信息提示完根据参数的值确实是否退出画图工具
openFile - 打开的图画对象 IO 文件的 File 对象句柄
index - shapesQueue[]队列中当前所指向的对象在队列中的序
maxShapesCount - 用户在画板上所绘制的最大图形对象数目
lastEOF - 用来记录用户打开本地图像 IO 文件加载到画板后 shapesQueue 中末尾图形对象的位置
isChanged - 用来判断画板内容是否修改更新过

3.1.19 文件菜单栏打开模块

打开一个快捷链接文件，发现不是我们所识别的二进制图像对象 IO 文件



Test 是之前保存好在本地的一个图像 IO 文件

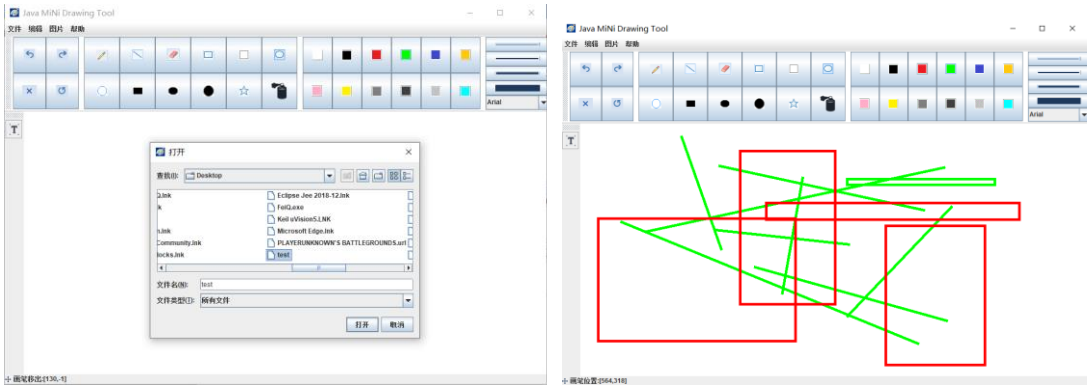


图 3-36 文件菜单栏打开案例

（1）程序框图

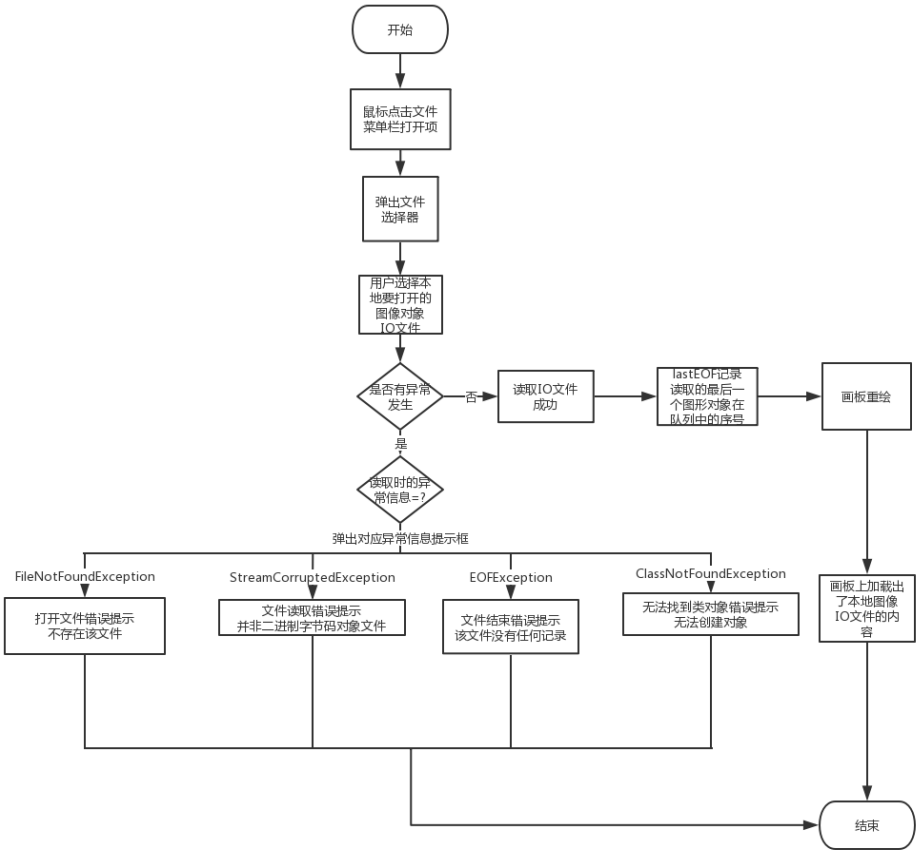


图 3-37 文件菜单栏打开模块程序流程图

（2）函数原型：

public void isSave(boolean flag)
private void openPicture()
private File openWindow()
public void repaint()

（3）函数功能：

可打开之前用户保存在本地的图像对象 IO 文件，将图画重新加载到画图板上。

（4）参数：

flag - 保存信息提示完根据参数的值确实是否退出画图工具
openFile - 打开的图画对象 IO 文件的 File 对象句柄
index - shapesQueue[]队列中当前所指向的对象在队列中的序
maxShapesCount - 用户在画板上所绘制的最大图形对象数目
lastEOF - 用来记录用户打开本地图像 IO 文件加载到画板后 shapesQueue 中末尾图形对象的位置
pictureChooser - 弹出的文件选择器

3.1.20 文件菜单栏保存、另存为模块

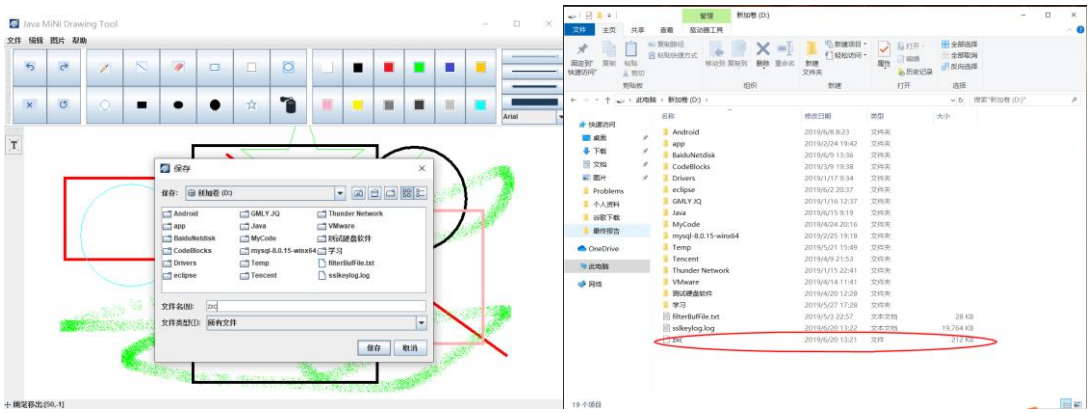


图 3-38 文件菜单栏保存、另存为案例

（1）程序框图

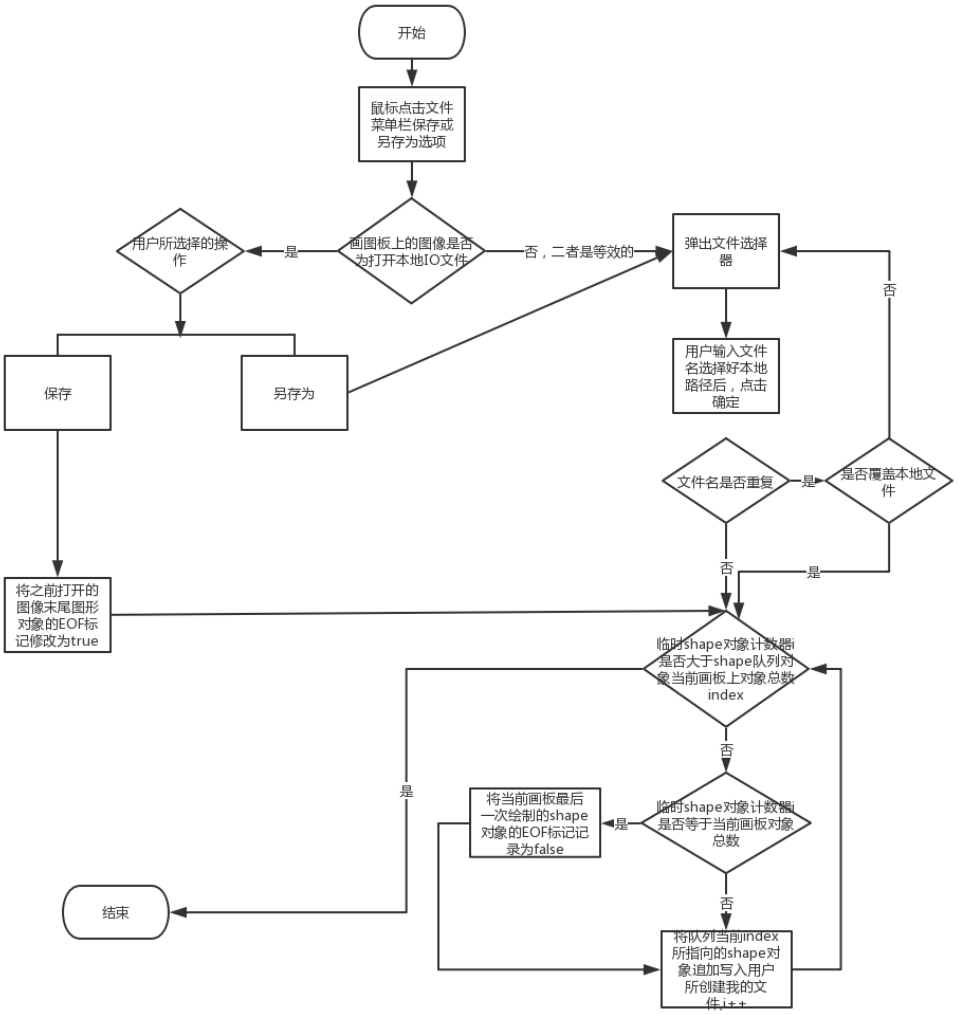


图 3-39 文件菜单栏保存、另存为模块程序流程图

（2）函数原型：

```
public void isSave(boolean flag)
private void savePicture(boolean flag)
```

```
private  File saveWindow()
public void repaint()
```

（3）函数功能：

将画板上的打开的图形 IO 文件的修改结果更新到本地，或将画板上的图画另存到本地为一个 IO 文件。

（4）参数：

flag - 保存信息提示完根据参数的值确实是否退出画图工具

openFile - 打开的图画对象 IO 文件的 File 对象句柄

index - shapesQueue[]队列中当前所指向的对象在队列中的序

maxShapesCount - 用户在画板上所绘制的最大图形对象数目

lastEOF - 用来记录用户打开本地图像 IO 文件加载到画板后 shapesQueue 中末尾图形对象的位置

pictureChooser - 弹出的文件选择器

isChanged - 用来判断画板内容是否修改更新过

3.1.21 文件菜单栏退出模块

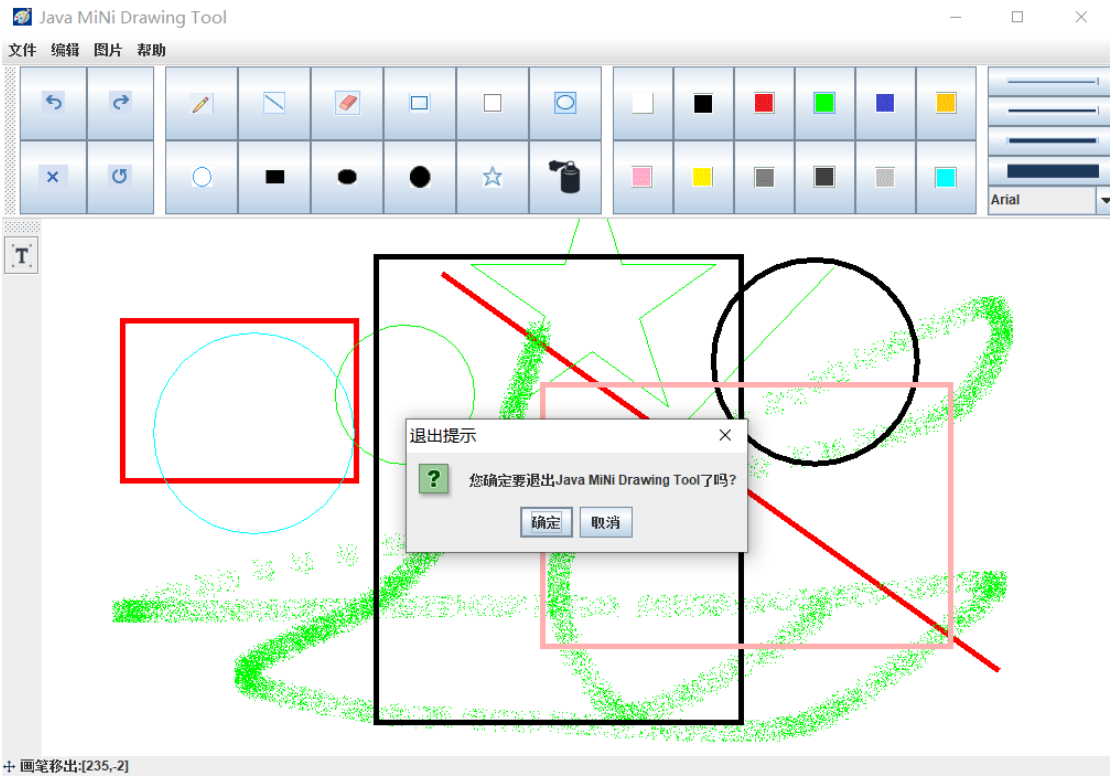


图 3-39 文件菜单栏退出案例

（1）程序框图

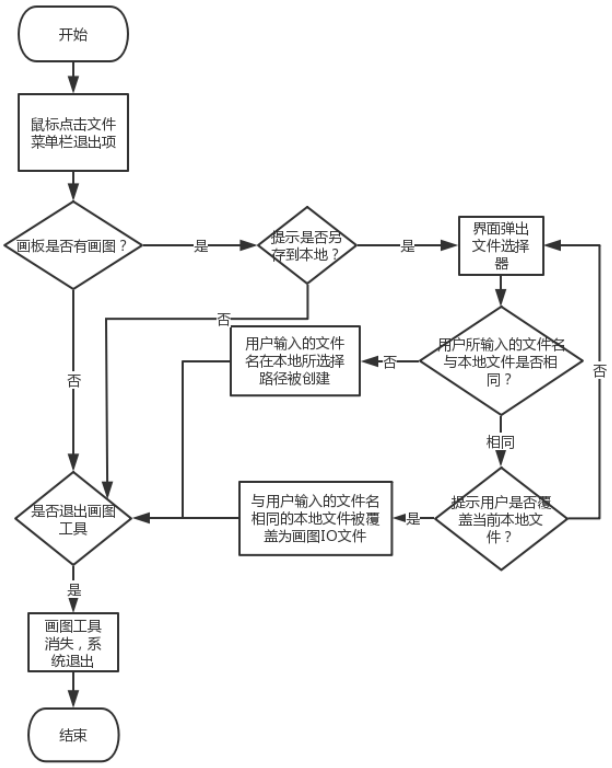


图 3-40 文件菜单栏退出模块程序流程图

（2）函数原型：

public void isSave(boolean flag)

private void Exit()

（3）函数功能：

用户从操作系统界面退出画图工具。

（4）参数：

flag - 保存信息提示完根据参数的值确实是否退出画图工具

openFile - 打开的图画对象 IO 文件的 File 对象句柄

index - shapesQueue[]队列中当前所指向的对象在队列中的序

lastEOF - 用来记录用户打开本地图像 IO 文件加载到画板后 shapesQueue 中末尾图形对象的位置

3.1.22 编辑菜单栏编辑画笔颜色模块

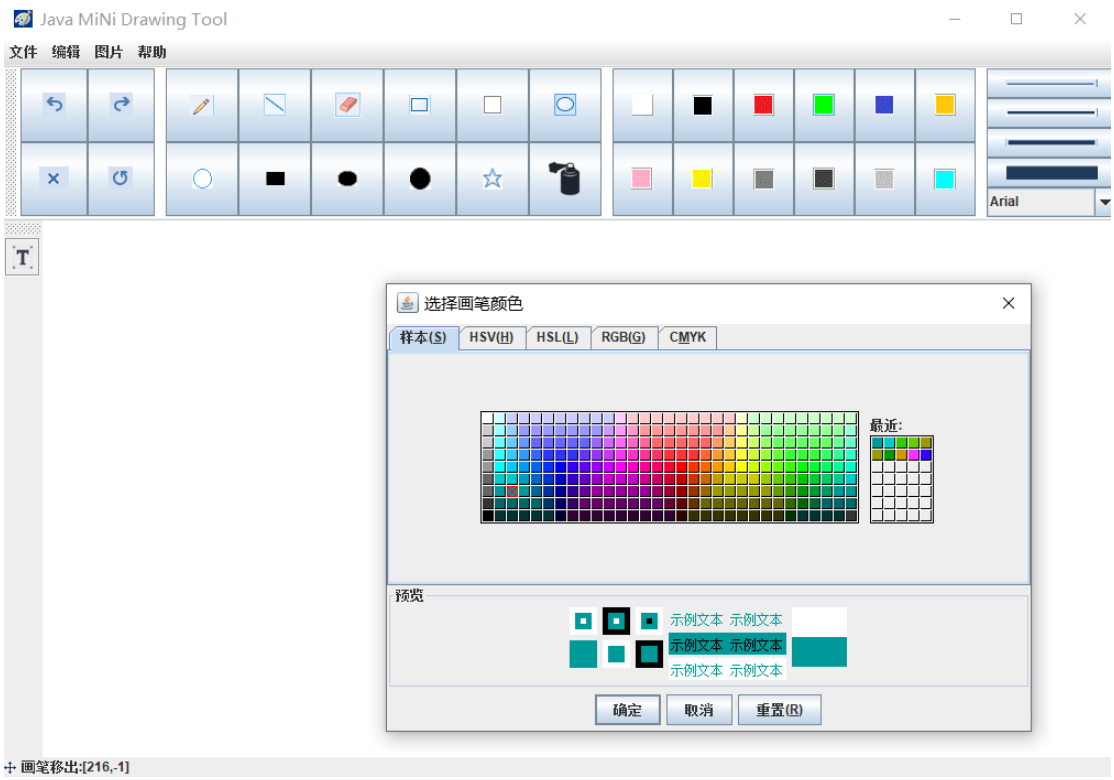


图 3-40 编辑菜单栏编辑画笔颜色案例

（1）程序框图

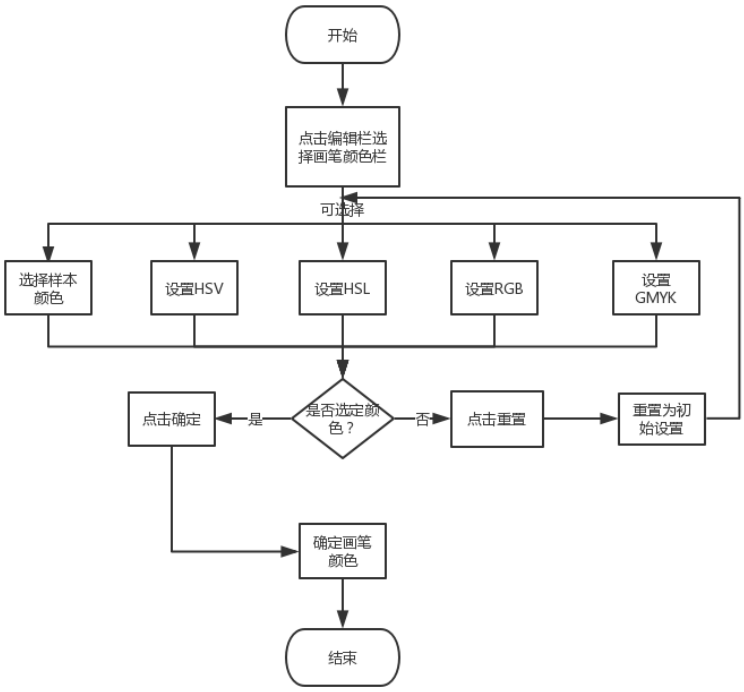


图 3-41 编辑菜单栏编辑画笔颜色模块程序流程图

（2）函数原型：

public void actionPerformed(ActionEvent e)
public static Color showDialog(Component component,String title, Color initialColor)

（3）函数功能：

用户可根据自己的喜好从颜色选择器 JColorChooser 选取自己特定的颜色样本。

（4）参数：

color - 用于记录用户当前做选择画笔颜色属性

3.1.23 编辑菜单栏编辑画笔粗细模块

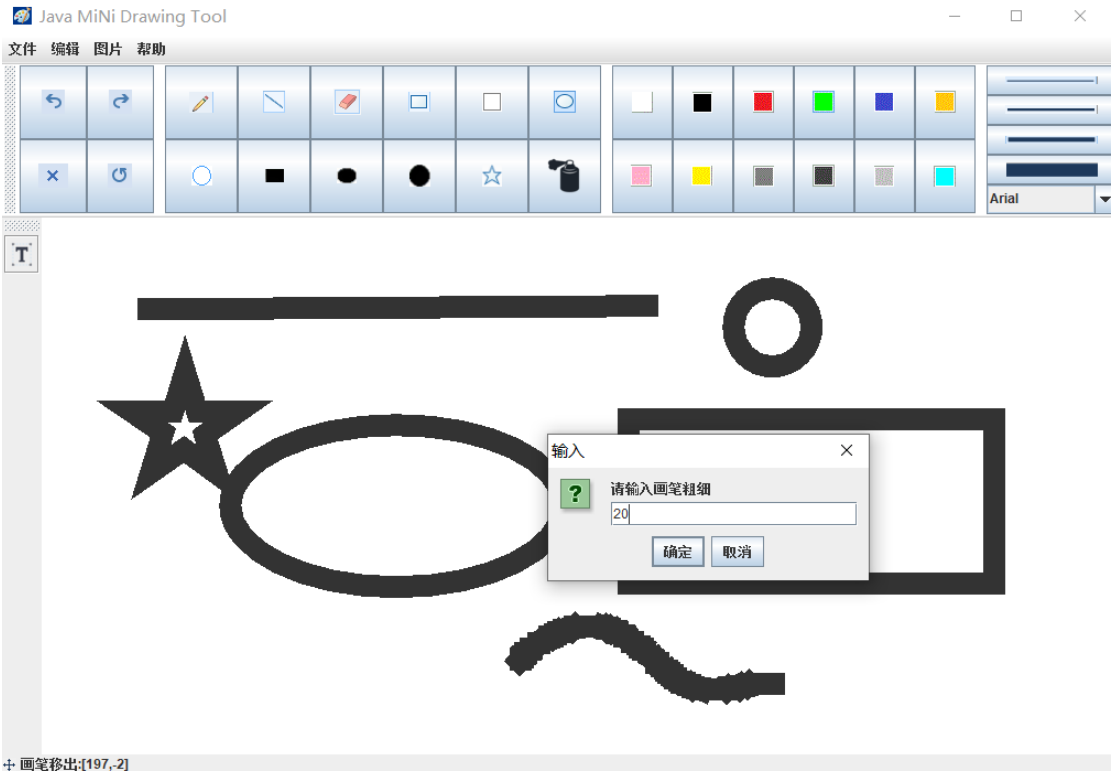


图 3-42 编辑菜单栏编辑画笔粗细案例

（1）程序框图

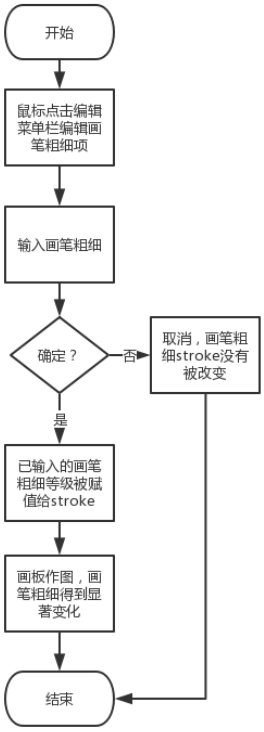


图 3-43 编辑菜单栏编辑画笔粗细模块程序流程图

（2）函数原型：

```
public void actionPerformed(ActionEvent e)
public static String showInputDialog(Object message)
public static float parseFloat(String s)
public static void showMessageDialog(Component parentComponent,
        Object message, String title, int messageType)
```

（3）函数功能：

用户可输入自己所要求的画笔粗细来绘制图形。

（4）参数：

stroke – 用于记录用户当前做选择正式画笔粗细属性

tmp – 用户输入画笔的粗细待转换成浮点数的用来记录画笔粗细的临时变量

input – 用户输入画笔的粗细待转换成浮点数的字符串

3.1.24 图片菜单栏打开模块

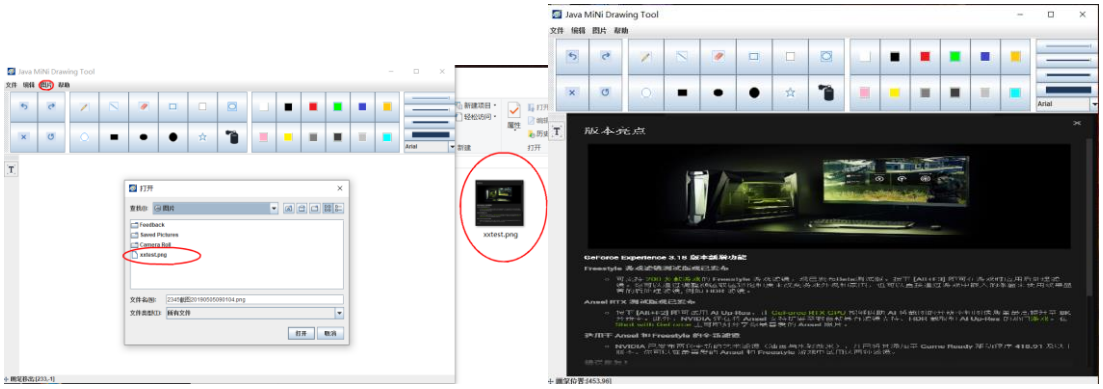


图 3-44 图片菜单栏打开图片案例

（1）程序框图

图 3-45 图片菜单栏打开图片模块程序流程图

（2）函数原型：

```
private void openPhoto()
private File openWindow()
public static BufferedImage read(File input)
public void newFile()
public void repaint()
```

（3）函数功能：

用画图工具可打开本地的一个标准的图片格式的文件，并且将图片加载至画图板上。

（4）参数：

openPicture – 用户打开的图片文件的对象句柄
input – 待读取本地文件对象句柄
image – 画板上的 BufferedImage 打开图片文件的对象句柄
fileOption – 文件菜单栏，用以调用其新建功能来重绘画板和重置 index 指向图形队列中的指针位置为-1

3.1.25 图片菜单栏保存、另存为模块

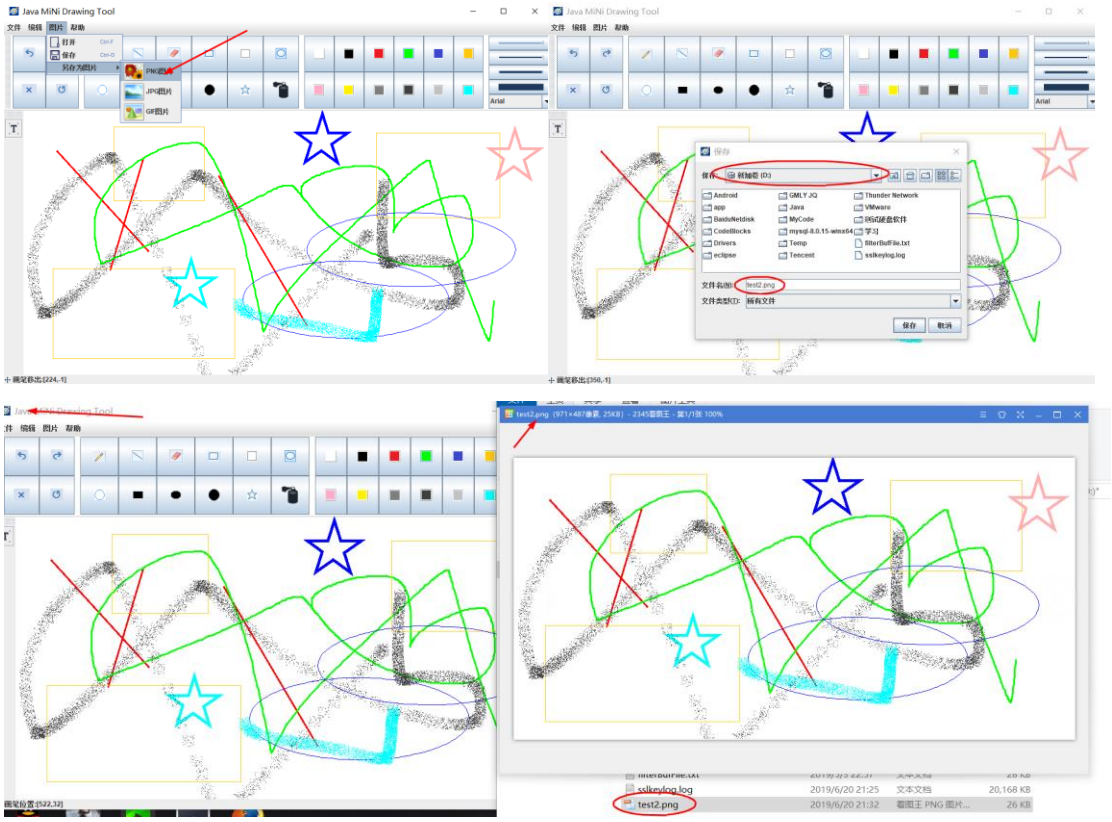


图 3-46 图片菜单栏保存、另存为图片案例

（1）程序框图

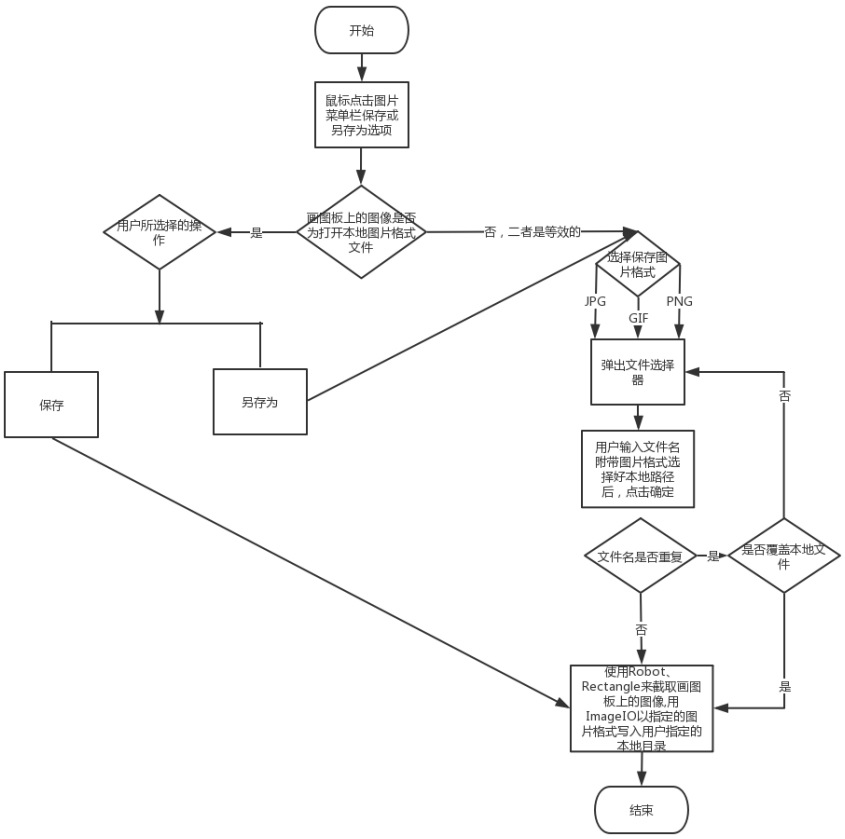


图 3-47 图片菜单栏保存、另存为模块程序流程图

（2）函数原型：

```
private void savePicture0(String format)
private  File saveWindow()
public static native void sleep(long millis)
private void savePhoto(File src,String format)
public synchronized BufferedImage createScreenCapture(Rectangle screenRect)
public static boolean write(RenderedImage im,String formatName, File output)
```

（3）函数功能：

用画图工具可将画板上所画的图像以指定的图片格式(png、jpg、gif)保存到本地，或将本地打开的图片提供编辑功能将修改更新到本地文件。

（4）参数：

openPicture – 用户打开的图片文件的对象句柄

format – 要保存的图片的格式

savePicture – 要保存到的本地文件路径形成的 File 对象句柄

millis – 线程休眠的时间以 ms 计，这里是为了防止用户在点击文件选择器确定按钮时，窗体还没完全消失，截取画板图像方法已截取图像将文件选择器也截取了进去，设置休眠时间为的是能让文件选择器的窗体完全消失之后再去截屏。

screenRect - 将在屏幕坐标中捕获的 Rect 画板
image - 以指定格式截取画板上图像所创建的 BufferedImage
output - 将在其中写入截屏画板图像数据的 File

3.1.26 帮助菜单栏模块

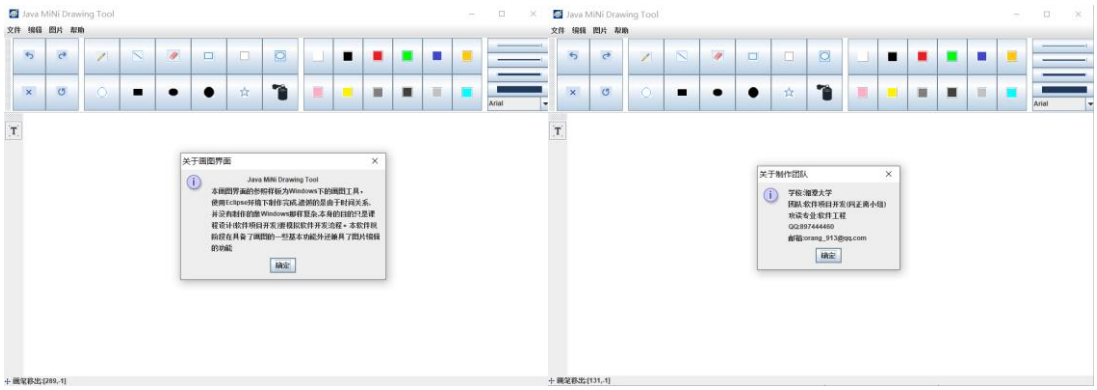


图 3-48 帮助菜单栏案例

（1）程序框图

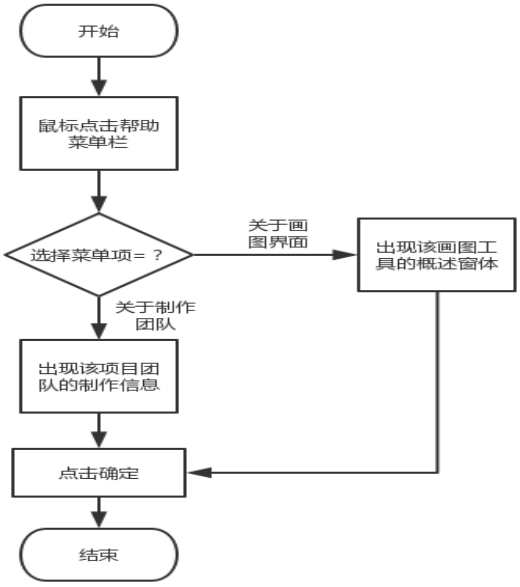


图 3-49 帮助菜单栏模块程序流程图

（2）函数原型：

```
public void actionPerformed(ActionEvent e)
private void readMessage()
public int read(byte b[])
public static void showMessageDialog(Component parentComponent,
        Object message, String title, int messageType, Icon icon)
```

（3）函数功能：

可展示关于画图工具的一些开发概述和关于制作团队的一些信息。

（4）参数：

ActionEvent e – 点击动作事件

input – 缓冲区输入流

message – 保存读取本地的.txt 文本信息后将其以字节的形式转化为的字符串

3.1.27 画笔坐标栏模块

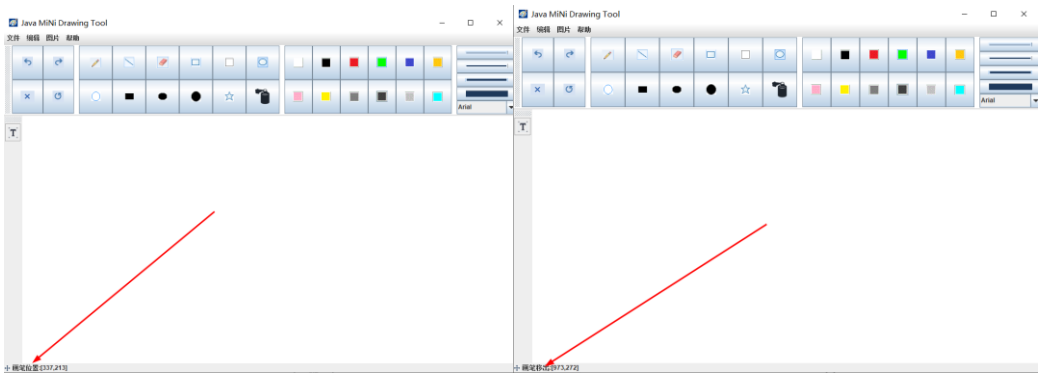


图 3-50 画笔坐标栏案例

（1）程序框图

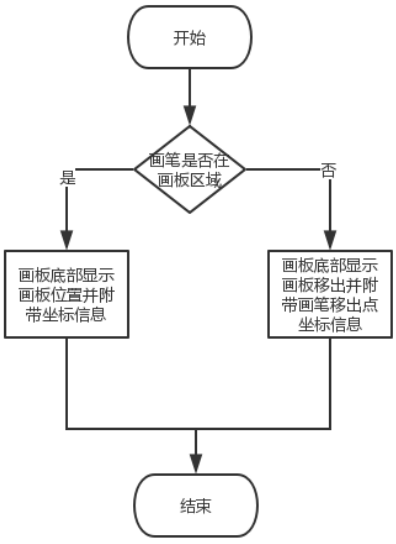


图 3-51 画笔坐标栏模块程序流程图

（2）函数原型：

```
public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
public void mouseMoved(MouseEvent e)
```

（3）函数功能：

在画板底部显示画笔位置的一些信息。

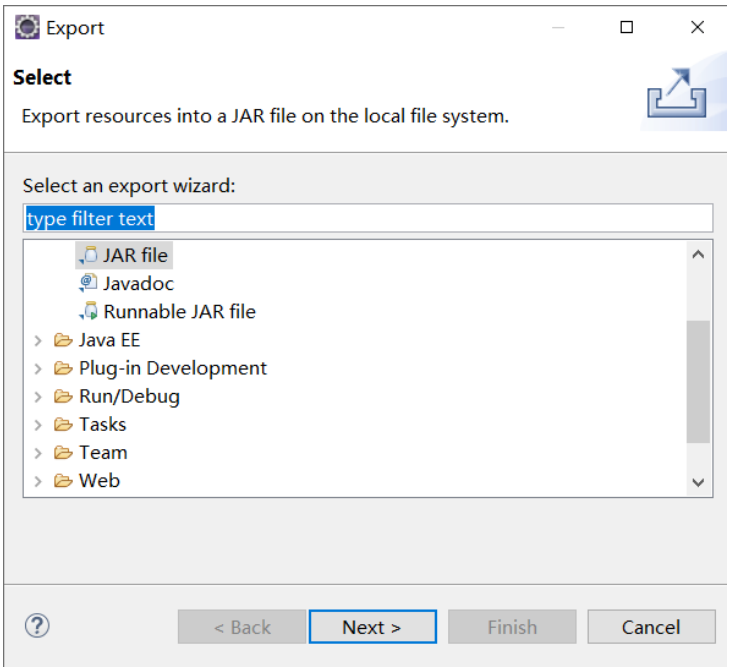
（4）参数：

MouseEvent e – 鼠标移动事件

3.2 系统安装与使用

3.2.1 系统集成打包

1） 将 Eclipse 工程项目打包成 jar 文件



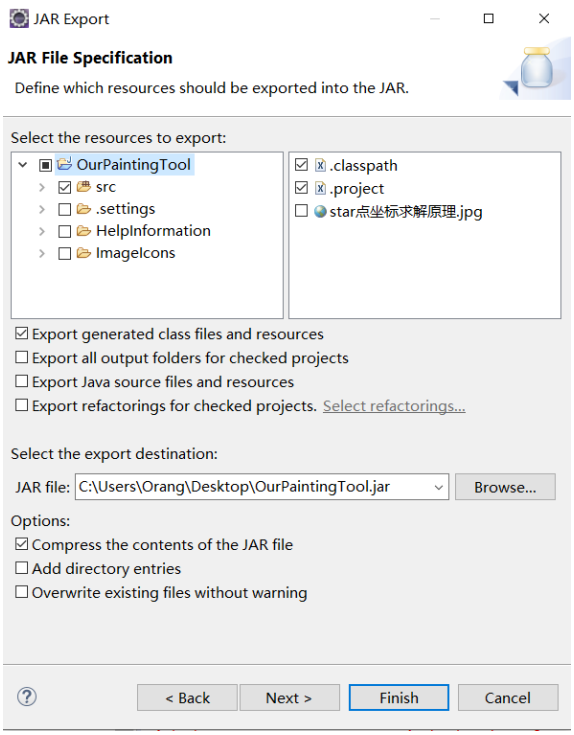


图 3-52 将项目导出打包成 jar 文件

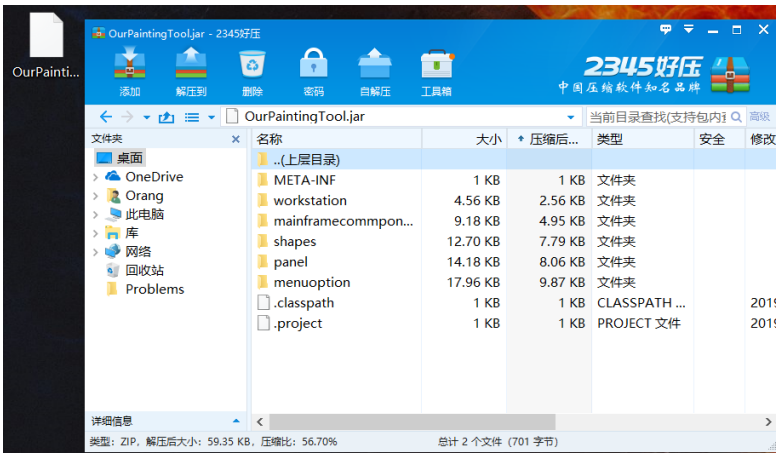


图 3-53 打包后的 jar 文件

打包时注意要将图片与项目分离，为了避免真正执行时加载图片的相对路径出错。这样只要用户系统上安装有 jre 1.8 及以上，就可以运行该软件。但是一般用户所使用的操作系统多数为 Windows，而且用户可能没有 JRE 运行环境，所以我们可以考虑把它编译集成打包成一个.exe 可执行文件。

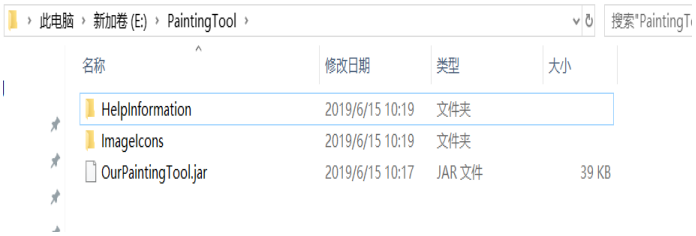
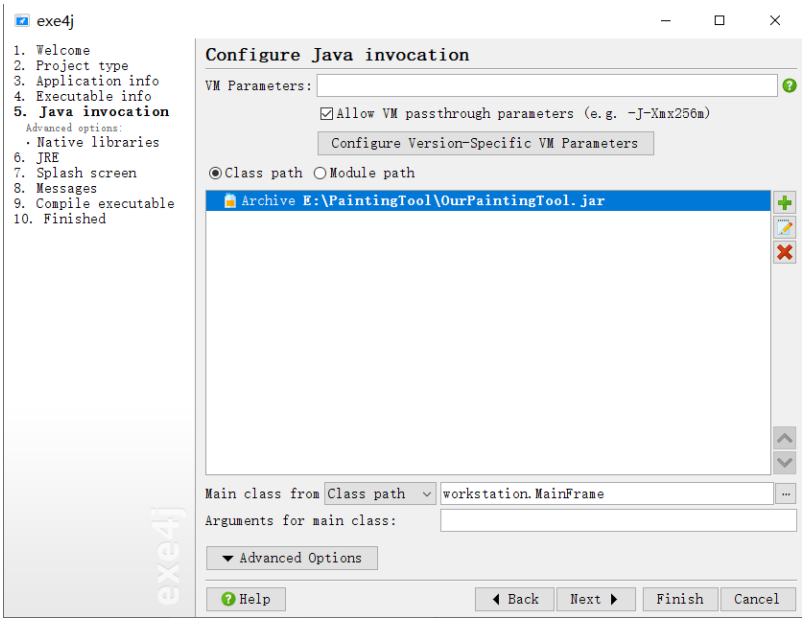
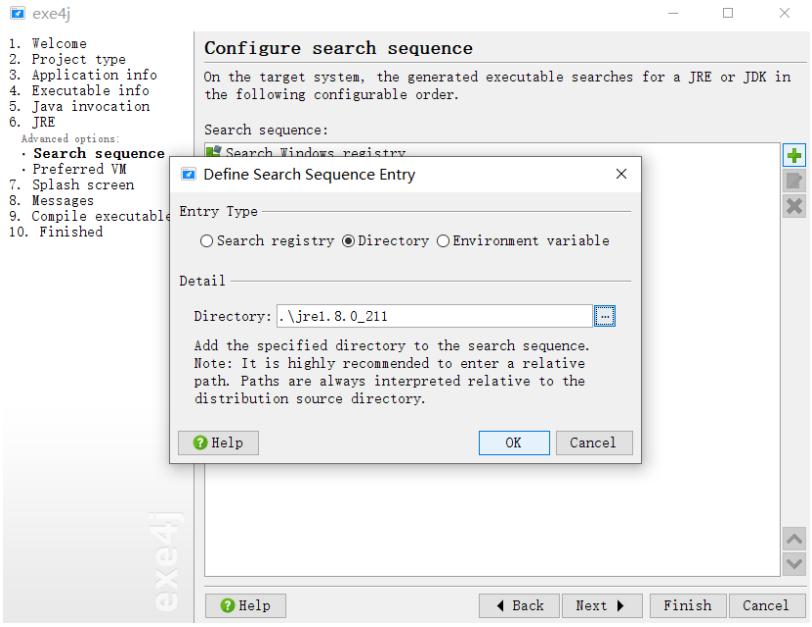


图 3-54 将需要用到的图片文件和 jar 包放置在一个文件夹下

2） 将 jar 文件用 exe4j 工具打包编译成. exe 可执行文件





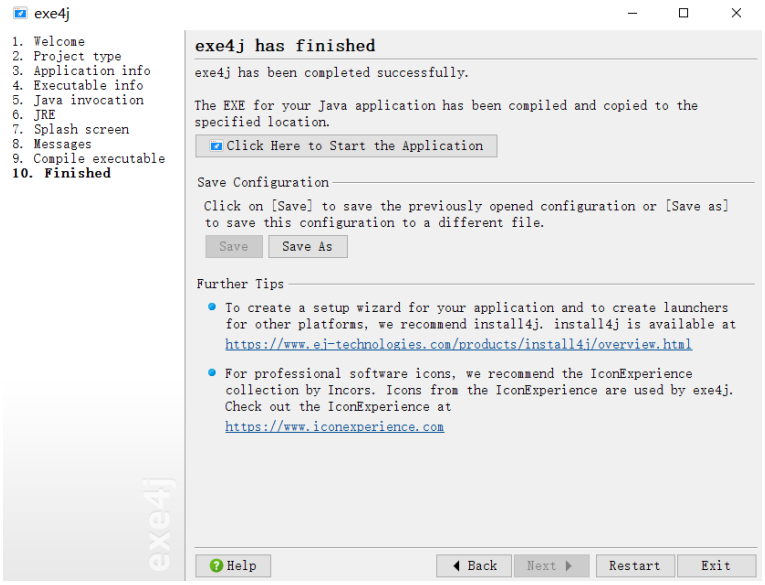


图 3-55 用 exe4j 将 jar 包编译成.exe 的步骤过程

3.2.2 系统安装方法

1） **安装解压缩工具**

从网上下载解压缩工具，并安装好。推荐使用好压。

2） **下载程序安装包**

直接从网上下载画图工具压缩包

3.2.3 使用界面

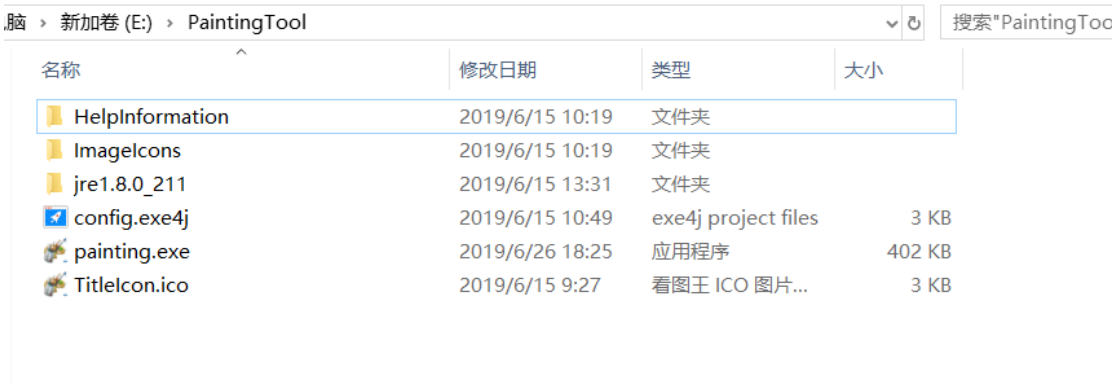


图 3-56 打包编译后的目录

双击 **painting.exe** 运行后，出现画图界面

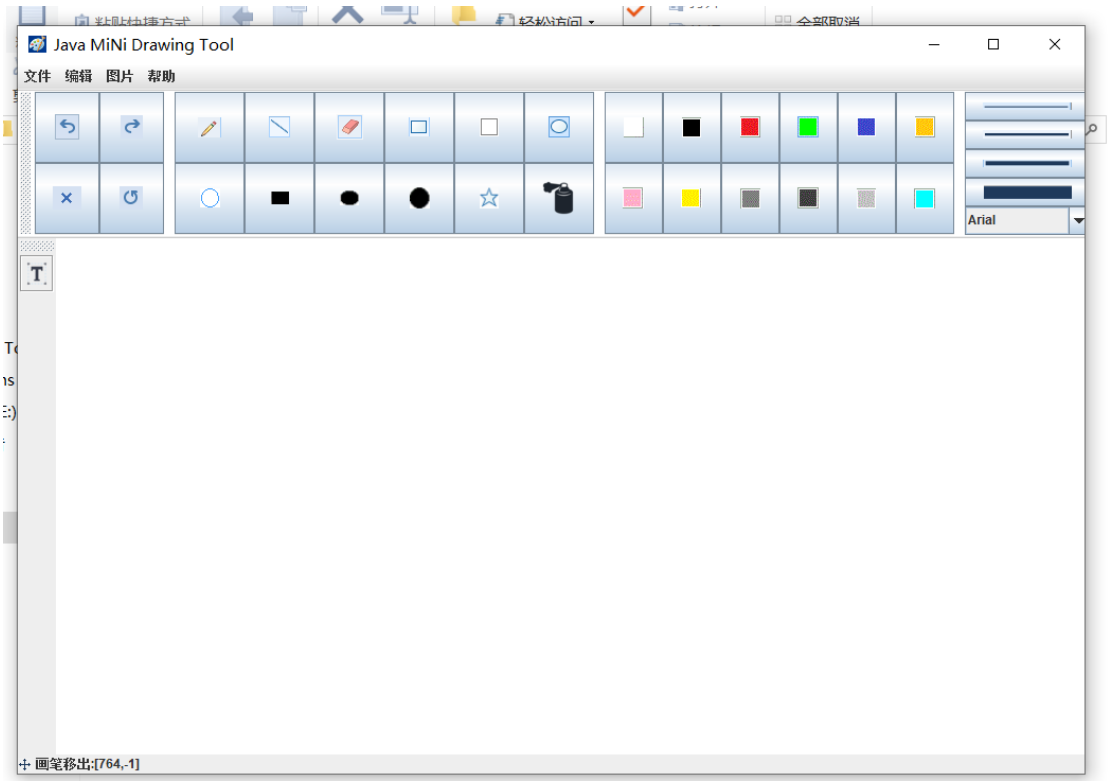


图 3-57 画图工具界面

3.2.4 使用规则

1） 解压

使用压缩工具解压画图工具压缩包。

2） 运行

双击解压的文件夹内的 painting.exe 开始画图。

四、参考文献

1. 《面向对象 java 程序设计 案例教程》,(中) 刘新 编制，机械工业出版社，2017 年 8 月
2. 《数据结构与算法分析》 Mark Allen Weiss 著，冯舜玺 译，机械工业出版社，2016 年 10 月。
3. 《软件需求工程与 UML 建模第 2 版》 毋国庆 梁正平 著，机械工业出版社，2017 年 8 月。
4. 《数据库系统概念 第 6 版》 Abraham Silberschatz Henry F.Korth 著，杨冬青、李红燕 译，机械工业出版社，2017 年 7 月。
5. 《软件体系结构原理、方法与实践 第 2 版》 张友生 著，清华大学出版社，2014 年 1 月。
6. 《软件工程概论 第 2 版》 郑人杰 马素霞 著，机械工业出版社，2017 年 11 月。
7. 《软件项目管理》 Bob Hughes Mike Cotterell 著，廖彬山、周卫华译，机械工业出版社，2010 年 7 月。
8. 《操作系统 第 3 版》 孟庆昌 牛欣源 著，电子工业出版社，2017 年 1 月。
9. 《软件测试教程 第 2 版》 宫云战 主编，机械工业出版社，2019 年 2 月。
10. 《软件测试的工具使用和实验指导》 李枚毅 主编，湘潭大学出版社，2019 年 2 月。