Pro 2:MiniCAD

[redacted]

目录

1 功能介绍

本 project 主要用 java 的 swing 设计了 MiniCAD 的界面，支持简单的直线、矩形、圆和文字绘制，主要操作功能如下：

- 图形绘制：支持直线，矩形，椭圆和文字的绘制

  分别点击图像图标，拂过图标图标变为绿色，选中图标变为蓝色。选中图标后进入画布区域，可以看见鼠标的

- 画笔修改：支持画笔粗细和颜色修改

  在菜单栏的调色块中点击修改画笔颜色，移动条的背景颜色为当前的画笔颜色。

  点击移动条上的按钮左右移动表示画笔的粗细变化。

- 图形移动：支持画布上的图像的相对位置移动和图形的缩放

  点击小手图标进入移动/缩放模式，鼠标进入可选中图形区域后显示移动/缩放符号，点击选中后移动/缩放。

- 图像保存：支持加载保存为 cad 格式的文件，保存当前画布到文件中

  点击上菜单栏，下拉显示 new，load，save 按钮，分别点击操作

为了设计的美观，我增加了背景和鼠标 icon 变化的功能。所有的图标为自己手画，其中鼠标 icon 变化包括选中 menu 板块的按钮颜色变化，选中按钮后进入画布的鼠标图标变化，选中图形的鼠标变化。

2 设计说明

整体分为三个设计模块，分别为上菜单栏设计 (文件操作)，右侧菜单栏设计 (模式按钮和画笔调整)，以及画布显示 (显示图形和图形区域响应) 三大模块。下面将根据实现功能详细介绍具体的事件步骤。

2.1 图像绘制

所有的图像为 shape 的子类，分别包括函数：

```
public abstract class Shape implements Serializable{
    private static final long serialVersionUID = 1L;
    int x1, x2, y1, y2;
    int _x1, _x2, _y1, _y2;
    Color color;
    double stroke;
    boolean flag;
    private HashSet <Point> points = new HashSet<Point>(); /

    public abstract void draw(Graphics2D g);
    public abstract void moveShape(int x, int y);
    public abstract void resizeShape(int x, int y);
    public void getClick(boolean flag) {···

    public void updateShape() {···
    public void addPoint(Point p)
    {
        points.add(p);
    }
    public boolean isContains(int x, int y) {···
}
```

Figure 2.1: shape 的主要函数

在 shape 函数的基础上通过，draw 函数在更新的时候不断画出 shape list 中的形状。支持的形状包括,line, text, oral, rectangle. 其中 draw 的函数大概形式如下：

```
public void draw(Graphics2D g) {
    g.setColor(color);
    g.setStroke(new BasicStroke((float)stroke));
    g.drawLine(x1, y1, x2, y2);
    initPoints();
}
```

Figure 2.2: draw 的函数

以上选取 line 的 draw，其他的大概类似。

在画图形的情况，通过点击不同的按钮选择绘制的模式，这部分放在 drawshape.java 的 code 中，主要通过 mouseclick 函数设置 press 和 release 的坐标点给到 shape 格式绘制，主要 code 结构如下：

```
public void mouseReleased(MouseEvent e) {
    // TODO Auto-generated method stub
    x2 = e.getX();
    y2 = e.getY();
    System.out.println("Release X:" + e.getX() + "Y:" + e.getY());
    switch (flag) {
    case 1:
        Shape line = new Line(x1, y1, x2, y2, color, stroke);
        list.add(line);
        menubar.setUpdated(true);
        System.out.println("add line");
        break;
```

Figure 2.3: 绘制的 mouseclick 响应函数内容

2.2 图像更改

主要包括检测图像的检测和选中之后的 shape 修改。

shape 的修改非常简单，只要选中的 list 的 shape 对象的相应属性进行修改就可以了，比如：

```
if(isclicked == 1) {
    switch (flag) {
    case 5:
        clicked.moveShape(e.getX()-x1, e.getY()-y1);
        menubar.setUpdated(true);
        break;
    case 6:
        clicked.resizeShape(e.getX()-x1, e.getY()-y1);
        menubar.setUpdated(true);
        break;
    default:
        break;
    }
}
```

Figure 2.4: 选中的 list 的属性更改

判断选中的函数比较麻烦，是通过记录绘制函数的 point 的 list，判断鼠标选中点在 list 中间，不同的函数绘制的方法不同，主要就是根据几何形

状的特性计算结果，其中的 contain 可以参见 rectangle 的情况, 其他基本相同：

```
private void initPoint(){
    int s = this.stroke>5?(int)(this.stroke/2):2;
    s = s<20?s:20;
    for (int k = -s;k<=s;k++)
    {
        for(int l = -s;l<=s;l++)
        {
            for (int i =Math.min(this.x1,this.x1 + this.x2);
                    i<=Math.max(this.x1,this.x1 + this.x2);i++)
            {
                addPoint(new Point(i+k,this.y1+l));
                addPoint(new Point(i+k,this.y1 + this.y2+l));
            }
            for (int i =Math.min(this.y1,this.y1 + this.y2);
                    i<=Math.max(this.y1,this.y1 + this.y2);i++)
            {
                addPoint(new Point(this.x1+k,i+l));
                addPoint(new Point(this.x1 + this.x2+k,i+l));
            }
        }
    }
}
```

Figure 2.5: 添加 points

2.3 图像保存

包括创建新的图像，保存当前图像到文件和加载之前保存的 cad 文件。主要的信息包括在 menubar.java 的信息中，通过增加 menubar 的响应函数给 savefile(),loadfile(),newfile() 的函数提供内容。

```
public MenuBar(ArrayList<Shape> list) {
    // TODO Auto-generated constructor stub
    this.list = list;
    this.setLayout(null);
    this.setBounds(100, 75, 650, 25);

    JMenu fileMenu = new JMenu("File");
    fileMenu.setFont(new Font("bold",5,15));
    fileMenu.setBounds(5, 5, 50, 15);

    JMenuItem newItem = new JMenuItem("New");
    newItem.setFont(new Font("bold",5,15));
    newItem.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent arg0) {
            // TODO Auto-generated method stub
            newfile();
        }
    });
    fileMenu.add(newItem);
```

Figure 2.6: menubar 的菜单绘制

主要 file 操作主要就是将画布的 shapelist 序列化保存在文件中，和从文件中被序列化的 shapelist 序列化 reconstruction 到画布中，主要细节可以看到下列 code：

```
fileName.delete();
FileOutputStream fout=new FileOutputStream(fileName);
output=new ObjectOutputStream(fout);

output.writeInt(list.size());

for(int i=0;i < list.size();i++){
    Shape shape = (Shape)list.get(i);
    output.writeObject(shape);
    output.flush();
}

output.close();
fout.close();
updated = false;
saved = true;
```

Figure 2.7: 保存序列化结果

2.4 鼠标响应

鼠标相应就是添加 flag 信息和从按钮 comment 信息中提取 flag 后载入：

先载入 icon 图标到内容中：

```
// 设置鼠标icons
Cursor[] cursors = new Cursor[6];
Toolkit tk = Toolkit.getDefaultToolkit();
for(int i=1;i<=3;i++) {
    Image icon_cursor = new ImageIcon("image/cursor"+i+".png").getImage();//.getScaledIn
    cursors[i-1] = tk.createCustomCursor(icon_cursor, new Point(20,50), "cursor"+i);
}
```

Figure 2.8: 加载 icons

在一定的 flag 提醒下写入 icon：

```
// 设置鼠标移动时的图标
System.out.println("isclicke" + isclicked);
if(flag == 5) {
    for(int i=list.size()-1;i>=0;i--) {
        Shape shape = (Shape)list.get(i);
        if(shape.isContains(e.getX(), e.getY())) {
            canvas.setCursor(cursors[1]);
            isclicked = 1;
            clicked = shape;
            break;
        }
        else {
            isclicked = 0;
            canvas.setCursor(Cursor.getDefaultCursor());
        }
    }
```

Figure 2.9: set 鼠标的 icon

3 测试结果

连续的视频录制见 video 文件夹

3.1 图形绘制

下面为绘制 line 的结果，可以看见按钮的选中，图形的绘制和鼠标的 icon 的内容，其他内容可以通过 jar 文件的运行看见 oval 和 rec 的结果。

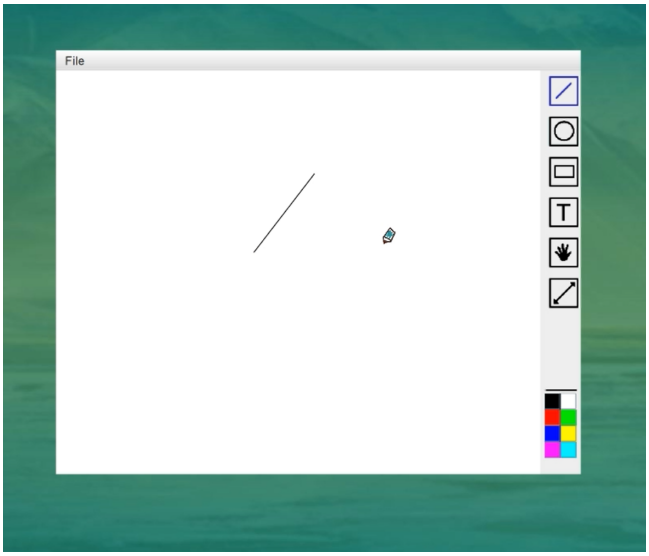

Figure 3.1: 绘制图像

以下为 text 的结果，可以看见鼠标还是箭头，输入文字的文本框弹出提示。

Figure 3.2: 绘制文本，添加文字

然后在鼠标点击的地方加入文字。

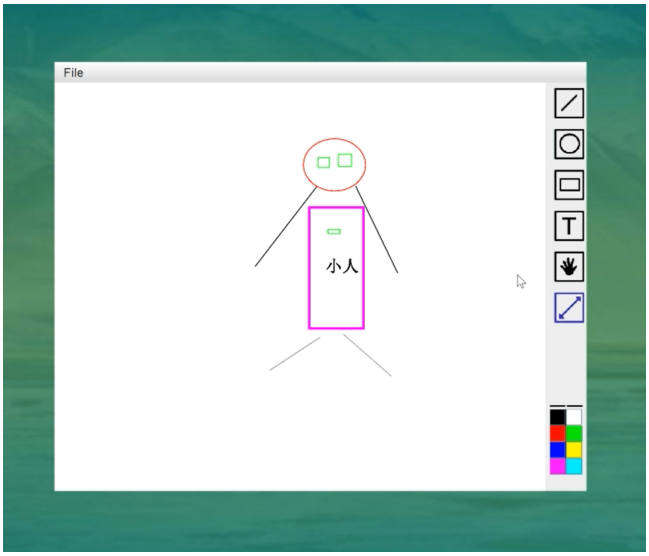

Figure 3.3: 绘制文本

3.2 图形选中

图像选中后会有一个移动的 icon 出现，点击后保持这个 icon 的显示移动图像：

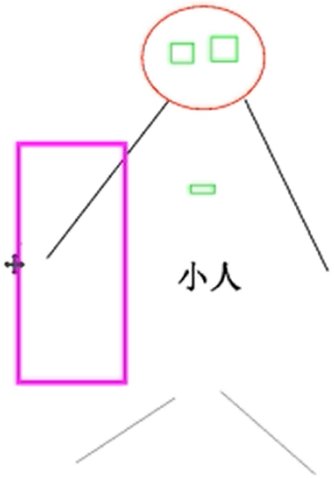

Figure 3.4: 保持 icon

可以看见包括形状的移动和大小的变化：

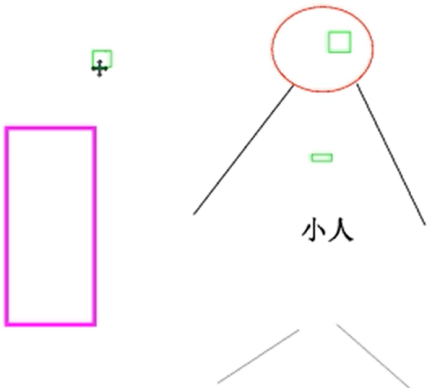

Figure 3.5: 形状大小

3.3 文件保存

可以看见下拉菜单包括：

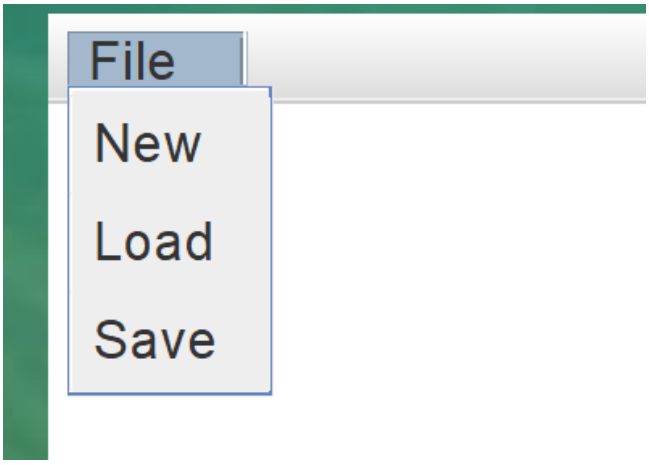

Figure 3.6: 下拉菜单

我们选择 load 文件夹内的信息：

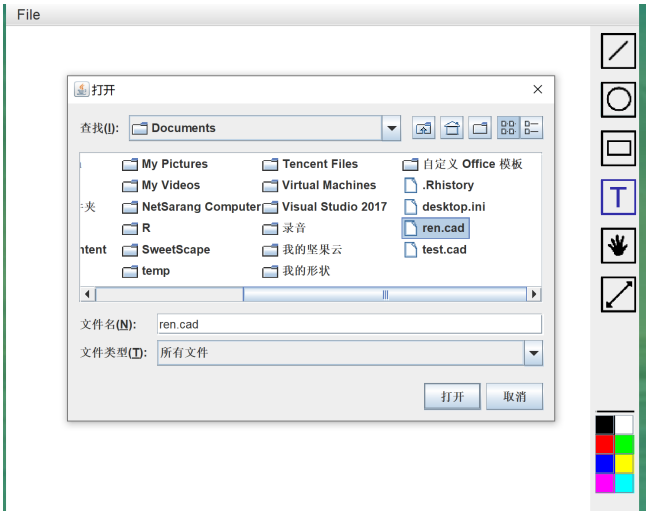

Figure 3.7: 加载 ren.java

可以看见 load 以后的结果：

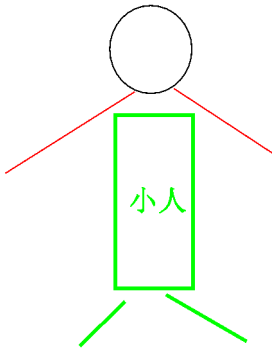

Figure 3.8: load 到画布里面

load 后依旧支持移动等修改、写入操作，可以看见如果当前画面不保存会弹出提醒消息：

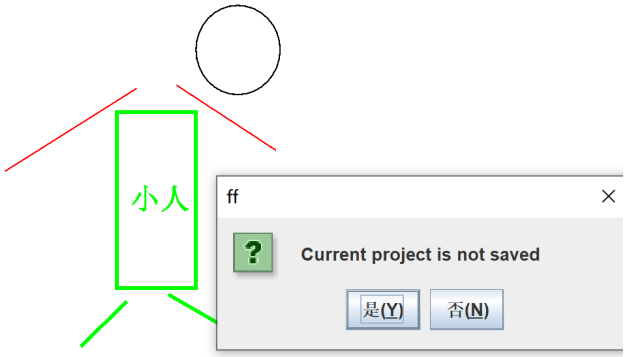

Figure 3.9: 新建结果

4 讨论与心得

这次实验主要是通过代码更加熟练地掌握了 swing 的 gui 设计的内涵，对于各种响应函数、shape 函数、鼠标 icon 设计和序列化格式设计的学习。其中主要遇到的问题是形状信息的不断刷新，这个过程开始的时候对于数据结构的设置挣扎了一下，想要把各个模块完全分开，但是最后还是把不断刷新绘制的部分放到了主画布显示模块。

在写不同画布，包括背景的时候，因为 java 只有三个层，叠加的时候要不断修改透明度和叠加顺序，最后找到的结果发现背景还是要放在第二场。

在写鼠标响应的时候，不知道为什么设置图标的标号一直失败，最后只能采取一个很笨的方法，把所有标号的图像加载，要用的时候从加载的图像 list 中选取要用的哪个，这一点在按钮的 comment 帮助下判断 flag 会比较好用。

在按钮设计上，因为想要实现经过，点下和点两下取消选中和同时只能在一组菜单中选中一个的方式，所以用来 buttongroup 的格式。还有调色盘的移动按钮，没有找到 java 的移动按钮，最终只能通过一个 canvas 的鼠标响应函数绘制按钮的位置来实现按钮的移动。

总体还是比较好用的，基本实现所有功能并提供其他的功能。

A DrawFrame.java

```
//package src;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.util.ArrayList;

import javax.swing.*;
import javax.swing.border.BevelBorder;
//放置直线、矩形、圆和文字，能选中图形，修改参数，如颜色等，能拖动图形和调整
    大小
class DrawFrame extends JFrame{
  private ArrayList<Shape> list = new ArrayList<Shape>();
  JFrame frame = new JFrame("MiniCAD");
  JPanel panel = new JPanel();
  MenuBar menubar = new MenuBar(list);
  private String select_msg = "";
  private Graphics2D g;
  private ButtonGroup buttons = new ButtonGroup();
  private DrawShape drawShape;

  public DrawFrame() {
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);

        panel.setLayout(null);
        addbg(); // 添加背景
        addcanvas(); // 添加画板内容
        addmenu(); // 菜单
        addpaint(); // 添加右侧画面，包括图标变色，鼠标icons加载，调色盘
        frame.setBounds(150, 50, 900, 800);     //设置窗口大小和位置
  }
  private void addmenu() {
    panel.add(menubar);
    panel.setOpaque(false);
    frame.add(panel);
  }
  private void addbg() {
    //背景图片
    ImageIcon icon_bg=new ImageIcon("image/bg.jpg");
    JLabel label_bg=new JLabel(icon_bg);
    label_bg.setBounds(0, 0, 900,800);
```

```
  frame.getLayeredPane().add(label_bg,new Integer(Integer.MIN_VALUE));
  JPanel j=(JPanel)frame.getContentPane();
  j.setOpaque(false);
}
private void addcanvas(){
  JPanel canvas = new JPanel() {
    public void paint(Graphics g1) {
      super.paint(g1);
      Graphics2D g = (Graphics2D)g1;
      for(int i=0;i<list.size();i++) {
        Shape shape = (Shape)list.get(i);
        shape.draw(g);
      }
      this.repaint();
    }
  };
  canvas.setLayout(null);
  canvas.setBounds(100, 100, 600, 500);
  canvas.setBackground(Color.white);

  JTextField tesTextField = new JTextField();
  tesTextField.setVisible(true);
  tesTextField.setLocation(10, 10);
  tesTextField.setBackground(Color.green);
  canvas.add(tesTextField);

  // 设置鼠标icons
  Cursor[] cursors = new Cursor[6];
  Toolkit tk = Toolkit.getDefaultToolkit();
  for(int i=1;i<=3;i++) {
    Image icon_cursor = new ImageIcon("image/cursor"+i+".png").getImage();
   //.getScaledInstance(10, 10,Image.SCALE_DEFAULT);
    cursors[i-1] = tk.createCustomCursor(icon_cursor, new Point(20,50), "
   cursor"+i);
  }


  frame.setVisible(true);
  g=(Graphics2D)canvas.getGraphics();
  drawShape =new DrawShape (g, buttons, canvas, cursors,list, menubar);

  canvas.addMouseListener(drawShape);
  canvas.addMouseMotionListener(drawShape);

      panel.add(canvas);
      panel.setOpaque(false);
```

```
        frame.add(panel);
}

private void addpaint() {
  JPanel menu = new JPanel();
  menu.setLayout(null);
  menu.setBounds(700, 100, 50, 500);


  for(int i=1;i<=6;i++) {
    ImageIcon icon1 = new ImageIcon("image/icon0"+ i +"1.png");
    icon1.setImage(icon1.getImage().getScaledInstance(50, 50,Image.
  SCALE_DEFAULT));
    ImageIcon icon2 = new ImageIcon("image/icon0"+ i +"2.png");
    icon2.setImage(icon2.getImage().getScaledInstance(50, 50,Image.
  SCALE_DEFAULT));
    ImageIcon icon3 = new ImageIcon("image/icon0"+ i +"3.png");
    icon3.setImage(icon3.getImage().getScaledInstance(50, 50,Image.
  SCALE_DEFAULT));

    JRadioButton button = new JRadioButton(icon1);
    button.setBorderPainted(false);
    button.setContentAreaFilled(false);
    button.setPressedIcon(icon3);
    button.setSelectedIcon(icon3);

    button.setRolloverIcon(icon2);
    button.setActionCommand("icon" + i);
    button.addActionListener(new ActionListener() {
      @Override
            public void actionPerformed(ActionEvent e) {
              String msg = e.getActionCommand();
              if (msg.equals(select_msg)) {
                buttons.clearSelection();
                select_msg = "";
              }else {
                select_msg = buttons.getSelection().getActionCommand();
        }
            }
        });

    button.setBounds(0, 50 * (i-1) , 50, 50);
    buttons.add(button);
    menu.add(button);

  }
```

```
// 设置画笔大小
JPanel brush = new JPanel();
int brush_x1, brush_x2;
brush.setLayout(null);
brush.setBounds(5, 395, 40, 2);

JButton size = new JButton();
size.setBounds(0, 0, 2, 5);
size.setEnabled(false);
brush.addMouseMotionListener(new MouseMotionListener() {
  @Override
  public void mouseMoved(MouseEvent e) {
    // TODO Auto-generated method stub

  }

  @Override
  public void mouseDragged(MouseEvent e) {
    // TODO Auto-generated method stub
    int y = size.getY();
    int x = e.getX();
    if(x >= 0 && x <= 70) {
      size.setLocation(x, y);
      drawShape.setStroke(0.1 * x);
    }
  }
});
brush.add(size);
menu.add(brush);

// 调色板设置
JPanel pallet = new JPanel();
pallet.setLayout(null);
pallet.setBounds(5, 400, 40, 80);

Color []colors = {new Color(0,0,0),new Color(255,255,255),new Color
(255,0,0)
,new Color(0,255,0),new Color(0,0,255),new Color(255,255,0)
,new Color(255,0,255),new Color(0,255,255)};

for(int i=0;i<8;i++) {
  JButton bt = new JButton();
  bt.setBackground(colors[i]);
  bt.setBounds(20*(i%2),20*(int)(i/2), 20, 20);
  bt.addActionListener(new ActionListener() {
    @Override
```

```
        public void actionPerformed(ActionEvent e) {
          // TODO Auto-generated method stub
          JButton btButton = (JButton)e.getSource();
          Color c = btButton.getBackground();
          brush.setBackground(c);
          drawShape.setColor(c);
        }
      });
      pallet.add(bt);
    }
    drawShape.setColor(colors[0]); // 初始画笔黑色
    brush.setBackground(colors[0]);
    menu.add(pallet);



    panel.add(menu);
    panel.setOpaque(false);
    frame.add(panel);
  }


}
```

B DrawShape.java

```
//package src;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.rmi.activation.ActivationGroupDesc.CommandEnvironment;
import java.util.ArrayList;

import javax.swing.*;

//public class

public class DrawShape implements MouseListener,MouseMotionListener{
  private Graphics2D g;
  private ButtonGroup buttons;
  private JPanel canvas;
  private Color color;
  private Cursor[] cursors;
  private int x1,x2,y1,y2;
```

```
private double stroke = 1;
private int flag = -1;
private int isclicked = 0;
private ArrayList<Shape> list;
private MenuBar menubar;

private Shape clicked;


public DrawShape(Graphics g, ButtonGroup buttons, JPanel canvas, Cursor[]
   cursors, ArrayList<Shape> list, MenuBar menubar) {
  this.g = (Graphics2D)g;
  this.buttons = buttons;
  this.canvas = canvas;
  this.cursors = cursors;
  this.list = list;
  this.menubar = menubar;
}
public void setColor(Color color) {
  this.color = color;
}
public Color getColor() {
  return this.color;
}
public void setStroke(double stroke) {
  this.stroke = stroke;
}
public double getStroke() {
  return this.stroke;
}

@Override
public void mouseClicked(MouseEvent e) {
  // TODO Auto-generated method stub

}

@Override
public void mouseEntered(MouseEvent e) {
  // 设置button作用
  if(buttons.getSelection() != null) {
    ButtonModel eButtonModel = buttons.getSelection();
    String command = eButtonModel.getActionCommand();
    flag = command.charAt(4)-'0';
  }else {
    flag = -1;
```

```
    }

    // 设置进入画布的鼠标图案
    //System.out.print(flag);
    if(flag >= 1 && flag <= 3) {
      canvas.setCursor(cursors[0]);
    }
    else {
      canvas.setCursor(Cursor.getDefaultCursor());
    }

  }

  @Override
  public void mouseExited(MouseEvent e) {
    // TODO Auto-generated method stub

  }

  @Override
  public void mousePressed(MouseEvent e) {
    // TODO Auto-generated method stub
    x1 = e.getX();
    y1 = e.getY();

//    System.out.println("Press X:" + e.getX() + "Y:" + e.getY());
  }

  @Override
  public void mouseReleased(MouseEvent e) {
    // TODO Auto-generated method stub
    x2 = e.getX();
    y2 = e.getY();
//    System.out.println("Release X:" + e.getX() + "Y:" + e.getY());
    switch (flag) {
    case 1:
      Shape line = new Line(x1, y1, x2, y2, color, stroke);
          list.add(line);
          menubar.setUpdated(true);
//          System.out.println("add line");
      break;
    case 2:
      Shape oval = new Oval(Math.min(x2, x1), Math.min(y2, y1), Math.abs(x2-
    x1),Math.abs(y1-y2), color, stroke);
          list.add(oval);
          menubar.setUpdated(true);
```

```
//          System.out.println("add oval");
      break;
    case 3:
      Shape rec = new Rectangle(Math.min(x2, x1),Math.min(y2, y1), Math.abs(
    x2-x1),Math.abs(y1-y2), color, stroke);
      list.add(rec);
      menubar.setUpdated(true);
//          System.out.println("add rec");
      break;
    case 4:
      String str = JOptionPane.showInputDialog("Please input the text: ");
      if(str != null) {
        Text text = new Text(x2, y2, color, stroke, str);
        list.add(text);
        menubar.setUpdated(true);
      }
    case 5:
      if(isclicked == 1) {
        clicked.updateShape();
        menubar.setUpdated(true);
      }
      isclicked = 0;
    case 6:
      if(isclicked == 1) {
        clicked.updateShape();
        menubar.setUpdated(true);
      }
      isclicked = 0;
    default:
      break;
    }
  }

  @Override
  public void mouseDragged(MouseEvent e) {
    // TODO Auto-generated method stub

    if(isclicked == 1) {
      switch (flag) {
      case 5:
        clicked.moveShape(e.getX()-x1, e.getY()-y1);
        menubar.setUpdated(true);
        break;
      case 6:
        clicked.resizeShape(e.getX()-x1, e.getY()-y1);
        menubar.setUpdated(true);
```

```
        break;
      default:
        break;
      }
    }

  }

  @Override
  public void mouseMoved(MouseEvent e) {
    // TODO Auto-generated method stub
    // 设置鼠标移动时的图标
//    System.out.println("isclicke" + isclicked);
    if(flag == 5) {
      for(int i=list.size()-1;i>=0;i--) {
        Shape shape = (Shape)list.get(i);
        if(shape.isContains(e.getX(), e.getY())) {
          canvas.setCursor(cursors[1]);
          isclicked = 1;
          clicked = shape;
          break;
        }
        else {
          isclicked = 0;
          canvas.setCursor(Cursor.getDefaultCursor());
        }
      }
    }else if(flag == 6) {
      for(int i=list.size()-1;i>=0;i--) {
        Shape shape = (Shape)list.get(i);
        if(shape.isContains(e.getX(), e.getY())) {
          canvas.setCursor(cursors[2]);
          isclicked = 1;
          clicked = shape;
          break;
        }else {
          isclicked = 0;
          canvas.setCursor(Cursor.getDefaultCursor());
        }
      }
    }

  }

}
```

C Shape.java

```
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionListener;
import java.awt.font.FontRenderContext;
import java.awt.geom.Rectangle2D;
import java.io.Serializable;
import java.util.HashSet;

public abstract class Shape implements Serializable{
  private static final long serialVersionUID = 1L;
  int x1, x2, y1, y2;
  int _x1, _x2, _y1, _y2;
  Color color;
  double stroke;
  boolean flag;
  private HashSet <Point> points = new HashSet<Point>(); // 用一组points

  public abstract void draw(Graphics2D g);
  public abstract void moveShape(int x, int y);
  public abstract void resizeShape(int x, int y);
  public void getClick(boolean flag) {
    this.flag = flag;
  }

  public void updateShape() {
    this._x1 = this.x1;
    this._x2 = this.x2;
    this._y1 = this.y1;
    this._y2 = this.y2;
    points.clear();
  }
  public void addPoint(Point p)
  {
    points.add(p);
  }
  public boolean isContains(int x, int y) {
    return points.contains(new Point(x,y));
  }
}
```

```
class Line extends Shape{
  public Line(int x1, Integer y1, int x2, int y2, Color color, double stroke
    ) {
    // TODO Auto-generated constructor stub
    this.x1 = x1;this._x1 = x1;
    this.x2 = x2;this._x2 = x2;
    this.y1 = y1;this._y1 = y1;
    this.y2 = y2;this._y2 = y2;
    this.color = color;
    this.stroke = stroke;
  }

  public void draw(Graphics2D g) {
    g.setColor(color);
    g.setStroke(new BasicStroke((float)stroke));
    g.drawLine(x1, y1, x2, y2);
    initPoints();
  }

  private void initPoints(){
    float k = (this.y2-this.y1)/(float)(this.x2-this.x1);
    float k2 = (this.x2-this.x1)/(float)(this.y2-this.y1);
    int s = this.stroke>5?(int)(this.stroke/2):2;
    s = s<20?s:20;
    for (int j =-s;j<=s;j++)
    {
      if (k<=1&&k>=-1)
      {
        for (int i = Math.min(this.x1, this.x2)+j;
            i<=Math.max(this.x1, this.x2)+j;i++)
        {
          int x = i;
          int y = (int)(k*(x-(this.x1+j))+this.y1);
          addPoint(new Point(x,y));
        }
        for (int i = Math.min(this.x1, this.x2);
            i<=Math.max(this.x1, this.x2);i++)
        {
          int x = i;
          int y = (int)(k*(x-this.x1)+this.y1+j);
          addPoint(new Point(x,y));
        }
      }
      else if(k2<=1&&k2>=-1)
      {
        for (int i = Math.min(this.y1, this.y2)+j;
```

```
            i<=Math.max(this.y1, this.y2)+j;i++)
        {
          int y = i;
          int x = (int)(k2*(y-(this.y1+j))+this.x1);
          addPoint(new Point(x,y));
        }
        for (int i = Math.min(this.y1, this.y2);
            i<=Math.max(this.y1, this.y2);i++)
        {
          int y = i;
          int x = (int)(k2*(y-this.y1)+this.x1+j);
          addPoint(new Point(x,y));
        }
      }
      else
      {
        for(int i = -s;i<=s;i++)
          addPoint(new Point(this.x1+i,this.y1+j));
      }
    }
  }
  public void moveShape(int x, int y) {
    this.x1 = this._x1 + x;
    this.x2 = this._x2 + x;
    this.y1 = this._y1 + y;
    this.y2 = this._y2 + y;
  }
  public void resizeShape(int x, int y) {
    this.x1 = this._x1 - x;
    this.x2 = this._x2 + x;
    this.y1 = this._y1 - y;
    this.y2 = this._y2 + y;
  }
}

class Oval extends Shape{
  public Oval(int x1, Integer y1, int x2, int y2, Color color, double stroke
    ) {
    // TODO Auto-generated constructor stub
    this.x1 = x1;this._x1 = this.x1;
    this.x2 = x2;this._x2 = this.x2;
    this.y1 = y1;this._y1 = this.y1;
    this.y2 = y2;this._y2 = this.y2;
    this.color = color;
    this.stroke = stroke;
  }
```

```
  public void draw(Graphics2D g) {
    g.setColor(color);
    g.setStroke(new BasicStroke((float)stroke));
    g.drawOval(x1, y1, x2, y2);
    initPoint();
  }

  private void initPoint(){
    for (int i =0;i<360;i++)
    {
      int s = this.stroke>5?(int)(this.stroke/2):2;
      s = s<20?s:20;
      for(int j =-s;j<=s;j++)
      {
        for(int k =-s;k<=s;k++)
        {
          double a = (double)Math.abs(this.x2)/2;
          double b = (double)Math.abs(this.y2)/2;
          int x = (int)((double)Math.min(this.x1,this.x1 + this.x2)+a+j
              +a*Math.cos((double)i/360*Math.PI*2));
          int y = (int)((double)Math.min(this.y1,this.y1 + this.y2)+b+k
              +b*Math.sin((double)i/360*Math.PI*2));
          addPoint(new Point(x,y));
        }
      }
    }
  }
  public void moveShape(int x, int y) {
    this.x1 = this._x1 + x;
    this.y1 = this._y1 + y;
  }
  public void resizeShape(int x, int y) {
    this.x1 = this._x1 - x;
    this.y1 = this._y1 - y;
    this.x2 = this._x2 + 2*x;
    this.y2 = this._y2 + 2*y;
  }

}

class Rectangle extends Shape{
  private static final long serialVersionUID = 1L;

  public Rectangle(int x1, int y1, int x2, int y2, Color color, double
     stroke) {
```

```
  // TODO Auto-generated constructor stub
  this.x1 = x1;this._x1 = this.x1;
  this.x2 = x2;this._x2 = this.x2;
  this.y1 = y1;this._y1 = this.y1;
  this.y2 = y2;this._y2 = this.y2;
  this.color = color;
  this.stroke = stroke;
}

public void draw(Graphics2D g) {
  g.setColor(color);
  g.setStroke(new BasicStroke((float)stroke));
  g.drawRect(x1, y1, x2, y2);
  initPoint();
}

private void initPoint(){
  int s = this.stroke>5?(int)(this.stroke/2):2;
  s = s<20?s:20;
  for (int k = -s;k<=s;k++)
  {
    for(int l = -s;l<=s;l++)
    {
      for (int i =Math.min(this.x1,this.x1 + this.x2);
          i<=Math.max(this.x1,this.x1 + this.x2);i++)
      {
        addPoint(new Point(i+k,this.y1+l));
        addPoint(new Point(i+k,this.y1 + this.y2+l));
      }
      for (int i =Math.min(this.y1,this.y1 + this.y2);
          i<=Math.max(this.y1,this.y1 + this.y2);i++)
      {
        addPoint(new Point(this.x1+k,i+l));
        addPoint(new Point(this.x1 + this.x2+k,i+l));
      }
    }
  }
}
public void moveShape(int x, int y) {
  this.x1 = this._x1 + x;
  this.y1 = this._y1 + y;
}
public void resizeShape(int x, int y) {
  this.x1 = this._x1 - x;
  this.y1 = this._y1 - y;
  this.x2 = this._x2 + 2*x;
```

```
    this.y2 = this._y2 + 2*y;
  }
}
class Text extends Shape{
  private static final long serialVersionUID = 1L;
  private double _stroke;
  private String str;
  public Text(int x1, int y1, Color color, double stroke, String str) {
    this.x1 = x1;this._x1 = x1;
    this.y1 = y1;this._y2 = y2;
    this.color = color;
    this.stroke = stroke;this._stroke=stroke;
    this.str = str;
  }
  public void draw(Graphics2D g) {
    g.setColor(color);
    g.setStroke(new BasicStroke((float)stroke));
    g.setFont(new Font("宋体",Font.BOLD,(int)stroke * 10));
    g.drawString(str, x1, y1);
    initPoint(g);
  }
  private void initPoint(Graphics2D g){
    double fontheight = g.getFont().getStringBounds(str, g.
    getFontRenderContext()).getHeight();
    double fontwidth = g.getFont().getStringBounds(str, g.
    getFontRenderContext()).getWidth();
    for (int i = (int)(this.x1);
        i<=this.x1+fontwidth;i++)
      for (int j = (int) (this.y1-fontheight);
          j<=this.y1;j++)
        addPoint(new Point(i,j));
  }
  public void moveShape(int x, int y) {
    this.x1 = this._x1 + x;
    this.y1 = this._y1 + y;
  }
  public void resizeShape(int x, int y) {
    this.stroke = this._stroke + 0.1*Math.abs(x);
  }
}
```

D MenuBar.java

```
import java.awt.*;
import java.awt.event.ActionEvent;
```

```java
import java.awt.event.ActionListener;
import java.io.*;
import java.util.*;
import javax.swing.*;

public class MenuBar extends JMenuBar{

  private static final long serialVersionUID = 1L;
  private ArrayList<Shape> list;
  private boolean updated = false; // 表示文件被更新
  private boolean saved = false; // 标识save文件加载
  private ObjectInputStream  input;
  private ObjectOutputStream output;
  private File fileName;
  JMenuBar menu = new JMenuBar();

  public MenuBar(ArrayList<Shape> list) {
    // TODO Auto-generated constructor stub
    this.list = list;
    this.setLayout(null);
    this.setBounds(100, 75, 650, 25);

    JMenu fileMenu = new JMenu("File");
    fileMenu.setFont(new Font("bold",5,15));
    fileMenu.setBounds(5, 5, 50, 15);

    JMenuItem newItem = new JMenuItem("New");
    newItem.setFont(new Font("bold",5,15));
    newItem.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent arg0) {
        // TODO Auto-generated method stub

      }
    });
    fileMenu.add(newItem);

    JMenuItem loadItem = new JMenuItem("Load");
    loadItem.setFont(new Font("bold",5,15));
    loadItem.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        loadfile();
      }
    });
```

```
    fileMenu.add(loadItem);

    JMenuItem saveItem = new JMenuItem("Save");
    saveItem.setFont(new Font("bold",5,15));
    saveItem.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        savefile();
      }
    });
    fileMenu.add(saveItem);

    this.add(fileMenu);

  }
  public void newfile() {
    int yes = 0;
    if(list.size() !=0 && updated) {
      yes = JOptionPane.showConfirmDialog(null, "Current project is not
    saved","ff",JOptionPane.YES_NO_OPTION);
    }
    if(yes == 0) {
      list.clear();
      saved = false;
      updated = false;
    }
  }

  public void loadfile() {
    int yes = 0;
    if(list.size() !=0 && updated) {
      yes = JOptionPane.showConfirmDialog(null, "Current project is not
    saved","",JOptionPane.YES_NO_OPTION);
    }
    if(yes == 0) {
      JFileChooser fileChooser=new JFileChooser();
        fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
        if(fileChooser.showOpenDialog(this)==JFileChooser.CANCEL_OPTION)
              return;
        fileName=fileChooser.getSelectedFile();
        if (fileName==null||fileName.getName().equals(""))
           JOptionPane.showMessageDialog(fileChooser,"Invalid File Name","
    Invalid File Name", JOptionPane.ERROR_MESSAGE);
        else {
          try {
```

```
            FileInputStream fin=new FileInputStream(fileName);
            input=new ObjectInputStream(fin);
            list.clear();
            Shape inShape;
            int count=input.readInt();
            for(int i=0 ;i<count ;i++)
            {
              Shape shape = (Shape)input.readObject();
              list.add(shape);
            }
            input.close();
            saved = true;
            updated = false;
            }catch(EOFException endofFileException){
              JOptionPane.showMessageDialog(this,"no more record in file","
   class not found",JOptionPane.ERROR_MESSAGE );
            }catch(ClassNotFoundException classNotFoundException){
              JOptionPane.showMessageDialog(this,"Unable to Create Object","
   end of file",JOptionPane.ERROR_MESSAGE );
            }catch (IOException ioException){
              JOptionPane.showMessageDialog(this,"error during read from
   file","read Error",JOptionPane.ERROR_MESSAGE );
            }
          }
      }
}

public void savefile() {
  if(saved)
  {
    try {
      fileName.delete();
      FileOutputStream fout=new FileOutputStream(fileName);
      output=new ObjectOutputStream(fout);

      output.writeInt(list.size());
      for(int i=0;i < list.size();i++){
            Shape shape = (Shape)list.get(i);
            output.writeObject(shape);
            output.flush();
        }

      output.close();
      fout.close();
      updated = false;
    }catch(IOException e){
```

```
        e.printStackTrace();
      }
    }
    else {
      JFileChooser fileChooser=new JFileChooser();
        fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
        if(fileChooser.showSaveDialog(this)==JFileChooser.CANCEL_OPTION)
                return ;
        fileName=fileChooser.getSelectedFile();

        if (fileName==null||fileName.getName().equals(""))
        JOptionPane.showMessageDialog(fileChooser,"Invalid File Name","
    Invalid File Name", JOptionPane.ERROR_MESSAGE);
        else{
          try {
            fileName.delete();
            FileOutputStream fout=new FileOutputStream(fileName);
            output=new ObjectOutputStream(fout);

            output.writeInt(list.size());

            for(int i=0;i < list.size();i++){
              Shape shape = (Shape)list.get(i);
              output.writeObject(shape);
              output.flush();
            }

            output.close();
            fout.close();
            updated = false;
            saved = true;
           }catch(IOException e){
             e.printStackTrace();
           }
        }
    }
  }

  public void setUpdated(boolean set) {
    this.updated = set;
  }
}
```

```
//package src;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.util.ArrayList;

import javax.swing.*;
import javax.swing.border.BevelBorder;
//放置直线、矩形、圆和文字，能选中图形，修改参数，如颜色等，能拖动图形和调整
    大小
class DrawFrame extends JFrame{
  private ArrayList<Shape> list = new ArrayList<Shape>();
  JFrame frame = new JFrame("MiniCAD");
  JPanel panel = new JPanel();
  MenuBar menubar = new MenuBar(list);
  private String select_msg = "";
  private Graphics2D g;
  private ButtonGroup buttons = new ButtonGroup();
  private DrawShape drawShape;

  public DrawFrame() {
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);

        panel.setLayout(null);
        addbg(); // 添加背景
        addcanvas(); // 添加画板内容
        addmenu(); // 菜单
        addpaint(); // 添加右侧画面，包括图标变色，鼠标icons加载，调色盘
        frame.setBounds(150, 50, 900, 800);     //设置窗口大小和位置
  }
  private void addmenu() {
    panel.add(menubar);
    panel.setOpaque(false);
    frame.add(panel);
  }
  private void addbg() {
    //背景图片
    ImageIcon icon_bg=new ImageIcon("image/bg.jpg");
    JLabel label_bg=new JLabel(icon_bg);
    label_bg.setBounds(0, 0, 900,800);
    frame.getLayeredPane().add(label_bg,new Integer(Integer.MIN_VALUE));
    JPanel j=(JPanel)frame.getContentPane();
    j.setOpaque(false);
  }
  private void addcanvas(){
    JPanel canvas = new JPanel() {
```

```
    public void paint(Graphics g1) {
      super.paint(g1);
      Graphics2D g = (Graphics2D)g1;
      for(int i=0;i<list.size();i++) {
        Shape shape = (Shape)list.get(i);
        shape.draw(g);
      }
      this.repaint();
    }
  };
  canvas.setLayout(null);
  canvas.setBounds(100, 100, 600, 500);
  canvas.setBackground(Color.white);

  JTextField tesTextField = new JTextField();
  tesTextField.setVisible(true);
  tesTextField.setLocation(10, 10);
  tesTextField.setBackground(Color.green);
  canvas.add(tesTextField);

  // 设置鼠标icons
  Cursor[] cursors = new Cursor[6];
  Toolkit tk = Toolkit.getDefaultToolkit();
  for(int i=1;i<=3;i++) {
    Image icon_cursor = new ImageIcon("image/cursor"+i+".png").getImage();
  //.getScaledInstance(10, 10,Image.SCALE_DEFAULT);
    cursors[i-1] = tk.createCustomCursor(icon_cursor, new Point(20,50), "
   cursor"+i);
  }


  frame.setVisible(true);
  g=(Graphics2D)canvas.getGraphics();
  drawShape =new DrawShape (g, buttons, canvas, cursors,list, menubar);

  canvas.addMouseListener(drawShape);
  canvas.addMouseMotionListener(drawShape);

      panel.add(canvas);
      panel.setOpaque(false);
      frame.add(panel);
}

private void addpaint() {
  JPanel menu = new JPanel();
  menu.setLayout(null);
```

```
menu.setBounds(700, 100, 50, 500);


for(int i=1;i<=6;i++) {
  ImageIcon icon1 = new ImageIcon("image/icon0"+ i +"1.png");
  icon1.setImage(icon1.getImage().getScaledInstance(50, 50,Image.
SCALE_DEFAULT));
  ImageIcon icon2 = new ImageIcon("image/icon0"+ i +"2.png");
  icon2.setImage(icon2.getImage().getScaledInstance(50, 50,Image.
SCALE_DEFAULT));
  ImageIcon icon3 = new ImageIcon("image/icon0"+ i +"3.png");
  icon3.setImage(icon3.getImage().getScaledInstance(50, 50,Image.
SCALE_DEFAULT));

  JRadioButton button = new JRadioButton(icon1);
  button.setBorderPainted(false);
  button.setContentAreaFilled(false);
  button.setPressedIcon(icon3);
  button.setSelectedIcon(icon3);

  button.setRolloverIcon(icon2);
  button.setActionCommand("icon" + i);
  button.addActionListener(new ActionListener() {
    @Override
          public void actionPerformed(ActionEvent e) {
            String msg = e.getActionCommand();
            if (msg.equals(select_msg)) {
              buttons.clearSelection();
              select_msg = "";
            }else {
              select_msg = buttons.getSelection().getActionCommand();
      }
          }
      });

  button.setBounds(0, 50 * (i-1) , 50, 50);
  buttons.add(button);
  menu.add(button);

}
// 设置画笔大小
JPanel brush = new JPanel();
int brush_x1, brush_x2;
brush.setLayout(null);
brush.setBounds(5, 395, 40, 2);
```

```
JButton size = new JButton();
size.setBounds(0, 0, 2, 5);
size.setEnabled(false);
brush.addMouseMotionListener(new MouseMotionListener() {
  @Override
  public void mouseMoved(MouseEvent e) {
    // TODO Auto-generated method stub

  }

  @Override
  public void mouseDragged(MouseEvent e) {
    // TODO Auto-generated method stub
    int y = size.getY();
    int x = e.getX();
    if(x >= 0 && x <= 70) {
      size.setLocation(x, y);
      drawShape.setStroke(0.1 * x);
    }
  }
});
brush.add(size);
menu.add(brush);

// 调色板设置
JPanel pallet = new JPanel();
pallet.setLayout(null);
pallet.setBounds(5, 400, 40, 80);

Color []colors = {new Color(0,0,0),new Color(255,255,255),new Color
(255,0,0)
,new Color(0,255,0),new Color(0,0,255),new Color(255,255,0)
,new Color(255,0,255),new Color(0,255,255)};

for(int i=0;i<8;i++) {
  JButton bt = new JButton();
  bt.setBackground(colors[i]);
  bt.setBounds(20*(i%2),20*(int)(i/2), 20, 20);
  bt.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
      // TODO Auto-generated method stub
      JButton btButton = (JButton)e.getSource();
      Color c = btButton.getBackground();
      brush.setBackground(c);
      drawShape.setColor(c);
```

```
        }
      });
      pallet.add(bt);
    }
    drawShape.setColor(colors[0]); // 初始画笔黑色
    brush.setBackground(colors[0]);
    menu.add(pallet);



    panel.add(menu);
    panel.setOpaque(false);
    frame.add(panel);
  }


}
```