MiniCAD-report

3200104705 陆遥

功能介绍

1.概述

- 题目要求

  用Java的awt和swing做一个简单的绘图工具，以CAD的方式操作，能放置直线、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，可以保存和恢复。功能请参考视频演示。

- 名词解释：

  画布：主界面中用于绘制图形的区域。

  图形：所有可以绘制的图形，包括直线，矩形，椭圆和文本。

2.具体实现

- 界面：

  左侧的工具栏实现图形绘制

  - 直线：绘制一条直线
  - 矩形：绘制一个矩形
  - 椭圆：绘制一个椭圆
  - 文本：输入待绘制文本后，绘制文本。

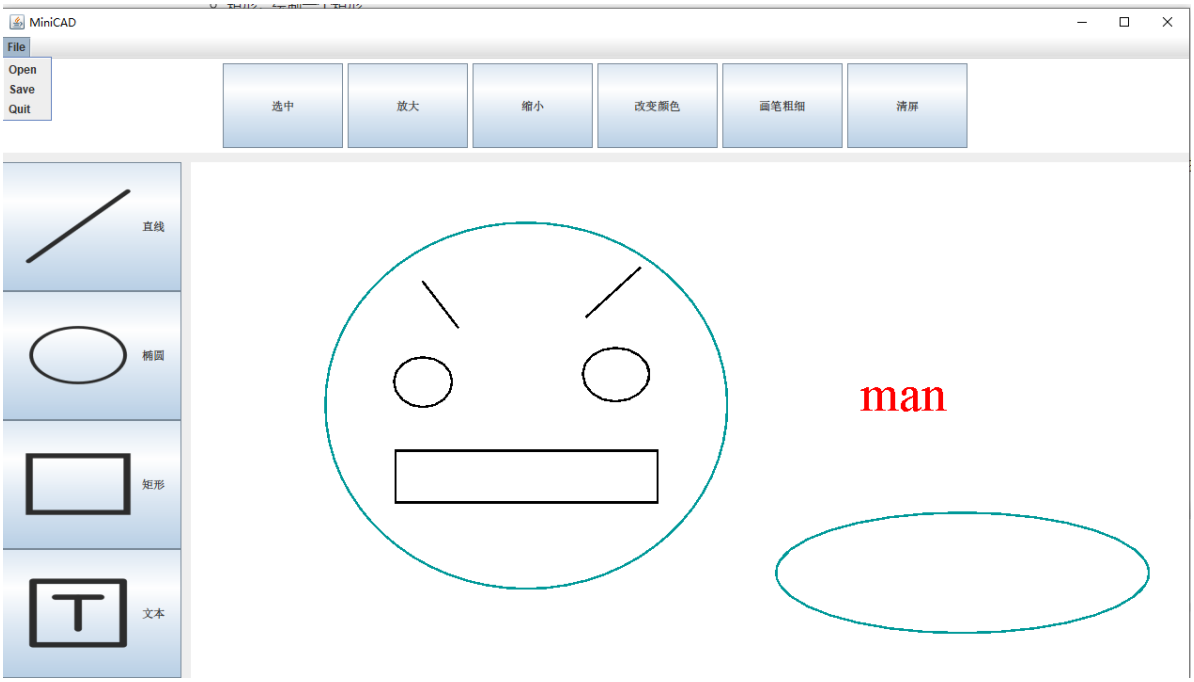
- 上方工具栏主要用于对图形的修改。实现以下功能：
  - 选中，选中图形（选中后可以直接拖动图形），如果未选中，则默认最后一个绘制的图形为当前选中的图形，如果选中图形时图形有重叠，默认选择最先绘制的图形
  - 放大：放大已经选中的图形的大小，默认每次放大五分之一
  - 缩小：缩小已经选中的图形的大小，默认每次缩小五分之一
  - 改变颜色：自动改变已选中的图形的颜色，且之后的图形颜色会切换至最新选择的颜色
  - 画笔粗细：改变画笔的粗细，画笔粗细默认为2.5
  - 清屏：清除当前画布上的所有内容。

- 左上角的menubar支持对当前画布上的内容的保存和恢复。
  - Open: 打开一个本软件绘制的cad文件。
  - Save：保存当前画布上的所有内容为一个cad文件，保存时后缀名没有限制。文件保存时将序列化所有对象，因此保存的形式为字符形式。
  - Quit：退出本软件

- jar封装：本程序以封装为jar包，使用时需先解压缩，将jar包中的icon文件夹和jar包放在同一目录下：然后使用命令运行程序：

```
java -jar ./minicad.jar
```

一个案例如下：



代码框架

1.class分析

- Myframe

  定义了minicad的顶层容器以及所有component的布局，放置menubar，工具栏以及画布。核心代码如下：

```
public class Myframe extends JFrame {

    private Mycanvas mycanvas;
    private Dimension framesize;
    private Myactionlistener actionlisten;
    private Itemmanage itemmanage;

    public Myframe(Itemmanage itemmanage, Myactionlistener actionlisten) {
        this.actionlisten = actionlisten;
        this.itemmanage = itemmanage;
        InitFrame();
    }

    private void InitFrame(){} //用于初始化JFrame
    private void makemenubar(){} //用于初始化menubar
    public void maketoolbar(){} //用于初始化按钮工具栏
}
```

- Mycanvas

  定义了minicad绘制图形的组件：画布，当调用JFram的repaint方法时，会调用画布中的paint函数以重画已有的所有图形。核心代码如下：

```
public class Mycanvas extends JPanel {
    private Itemmanage c_itemmanage;
    private Myactionlistener c_actionlistener;

    public Mycanvas(Itemmanage itemmanage,Myactionlistener actionlistener) {
        this.c_actionlistener = actionlistener;
        this.c_itemmanage = itemmanage;

        this.setBackground(new Color(255, 255, 255));
        this.addMouseListener(c_actionlistener);
        this.addMouseMotionListener(c_actionlistener);
    }

    public void paint(Graphics g){
        super.paint(g);
        for(Myitem m: c_itemmanage.Itemlist){
            m.draw(g);
        }
    }
}
```

- Myitem

  该类用于维护minicad的图形类，实现了创建图形，四种图形的绘制方式，以及判断图形是否可以被选中，移动图形位置，改变图形大小等功能。核心代码如下：

```
public class Myitem implements Serializable{


    private int i_type = 0;  //用于记录该图形的类型

    // private boolean i_chosed = false;

    private Color i_color = new Color(0, 0, 0); // 颜色
    private Font i_font; // 字体
    private float i_width; // 设置字体粗细
    // private Graphics2D i_2D; //画笔
    private String i_content; //主要用于存储文本的内容，线段，椭圆和矩形自动设置为空字符串

    //图形的坐标1
    private int x1;
    private int x2;
    //图形的坐标2
    private int y1;
    private int y2;

    // 用于记录图形初始化后的长和宽，用于放大缩小
    private int initwidth = 0;
```

```
    private int initheight = 0;

    // private Rectangle2D t_bound; // 设置边界

    public Color getColor() {
        return i_color;
    }

    public void setColor(Color color) {
        this.i_color = color;
    }

    // 初始化Myitem
    public Myitem(Color color, int type, int x1, int y1, int x2, int y2,
float width, String content) {
        // i_2D = g2d;
        i_width = width;
        i_color = color;
        // i_stroke = stroke;
        // i_font = new Font("Serif", Font.PLAIN, 20);
        i_content = content;
        i_type = type;
        this.x1 = x1;
        this.y1 = y1;
        this.x2 = x2;
        this.x2 = y2;

    }

    // 绘制图形方法，在frame或者JPanel的repaint时自动调用
    public void draw(Graphics g) {
        Graphics2D i_2D = (Graphics2D) g;
        i_2D.setColor(i_color);
        i_2D.setStroke(new BasicStroke(i_width));

        switch (i_type) {
            case Itemmanage.LINE_TYPE: {
                i_2D.drawLine(x1, y1, x2, y2);
                break;
            }
            case Itemmanage.CIRCLE_TYPE: {
                int xmin = Math.min(x1, x2);
                int xmax = Math.max(x1, x2);
                int ymin = Math.min(y1, y2);
                int ymax = Math.max(y1, y2);
                Ellipse2D ellispse = new Ellipse2D.Double(xmin, ymin, xmax -
xmin, ymax - ymin);
                i_2D.draw(ellispse);
                break;
            }
            case Itemmanage.REC_TYPE: {
                int xmin = Math.min(x1, x2);
                int xmax = Math.max(x1, x2);
                int ymin = Math.min(y1, y2);
                int ymax = Math.max(y1, y2);

                i_2D.drawRect(xmin, ymin, xmax - xmin, ymax - ymin);
                break;
```

```
            }

            case Itemmanage.TEXT_TYPE: {
                int xmin = Math.min(x1, x2);
                int ymax = Math.max(y1, y2);
                int ymin = Math.min(y1, y2);
                i_font = new Font("Serif", Font.PLAIN, ymax - ymin);
                i_2D.setFont(i_font);
                i_2D.drawString(i_content, xmin, ymax);
            }
        }
    }
```

- Myactionlistener

  该类定义了所有按钮以及画布上的事件监听，主要重写了事件监听的三个方法：actionperformed方法用于监听按钮，根据按钮的command决定接下来的事件，mousePressed方法定义鼠标按下的事件，主要用于图形绘制，mousedragged定义鼠标拖动时的事件，主要用于图形绘制。重要变量currentitem决定当前正在操纵的图形对象。

```
public class Myactionlistener implements ActionListener, MouseListener,
MouseMotionListener {

    private Point2D presspoint = new Point2D.Double(); // 用于决定鼠标点击的点，
基本未使用
    private static int event_type = -1; // 决定当前事件类型

    // 以下用于记录画笔的位置，x1，y1一般记录画笔按下的位置，x2，y2记录画笔拖动后的位置。
    private int x1 = 0;
    private int y1 = 0;
    private int x2 = 0;
    private int y2 = 0;
    private float pen = (float) 2.5; // 记录当前画笔粗细
    private Color m_color = Color.BLACK; // 记录当前画笔颜色
    private String text; // 记录文本框输入的内容

    private Myitem currentitem = null; // 始终为正在处理（包括绘制，拖动，改变颜色
的图形对象）
    private Itemmanage a_itemmanage; // item管理类
    private Myframe mainframe = null; // 顶层容器，用于绘制图片

    public Myactionlistener(Itemmanage itemmanage) {
        a_itemmanage = itemmanage;
    }

    public void setframe(Myframe mainframe) {
        this.mainframe = mainframe;
    }

    // 该方法重写所有按钮按下后会发生的事件。
    @Override
    public void actionPerformed(ActionEvent e) {}

    @Override
    public void mouseDragged(MouseEvent e){}
```

```
    @Override
    public void mousePressed(MouseEvent e){}

}
```

- Itemmanage

  该类管理一个ArrayList,该list存储了当前画布上的所有图形对象，且该类决定了所有按钮的command代码，事件监听函数依据该代码决定哪个按钮提交事件。核心代码如下：

```
public class Itemmanage {
    // 以下为事件的actioncommand，用于按钮提交时间类型以及做出相应处理。
    public static final int LINE_TYPE = 1;
    public static final int CIRCLE_TYPE = 2;
    public static final int REC_TYPE = 3;
    public static final int TEXT_TYPE = 4;

    public static final int CLEAR_TYPE = 5;
    public static final int SELECT_TYPE = 6;
    public static final int BIGGER_TYPE = 7;
    public static final int SMALLER_TYPE = 8;

    public static final int COLOR_TYPE = 9;
    public static final int PEN_WIDTH = 10;

    public static final int SAVE = 11;
    public static final int OPEN = 12;
    ArrayList<Myitem> Itemlist = new ArrayList<>();

}
```