

# MiniCAD 实验报告

MiniCAD for zju JAVA course

## 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。

要求上传：

1. 源码；
2. 实验报告；
3. 可执行的jar文件。

实验报告中须注明使用的Java版本、在Linux平台上编译源码及运行的命令。

## 程序运行效果示意图



## 运行方式

用console打开./jar文件夹，执行命令

```
1 | java -jar MiniCAD.jar
```

## 程序功能介绍

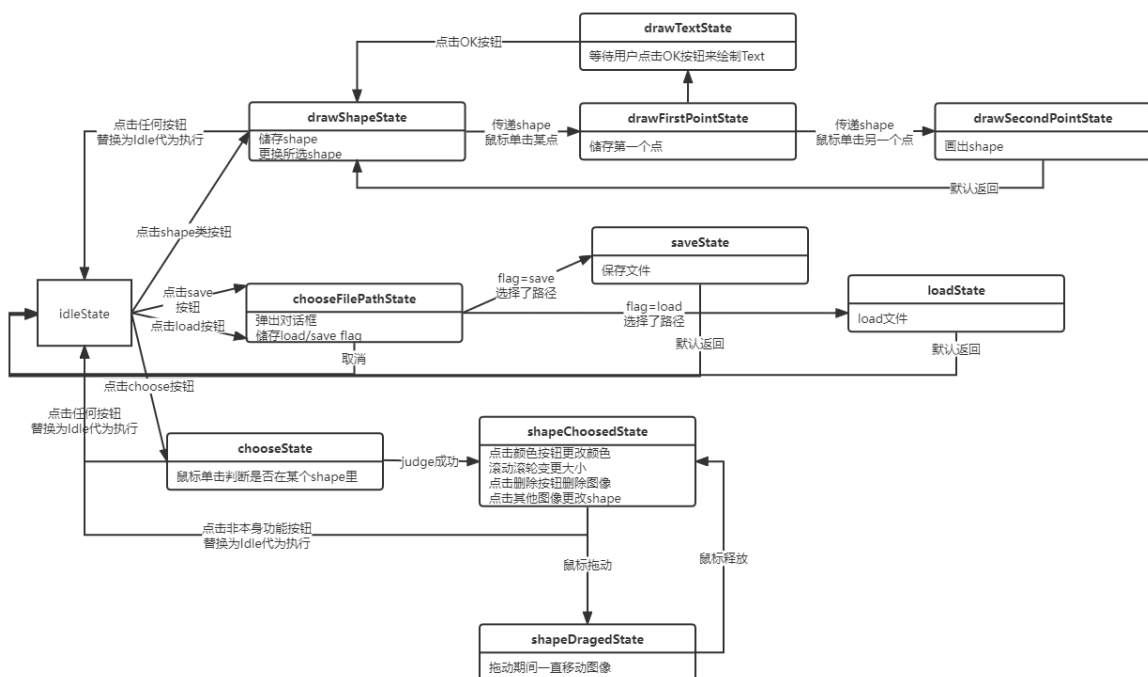
请随时留意程序状态提示框的指引。

- 选择图形

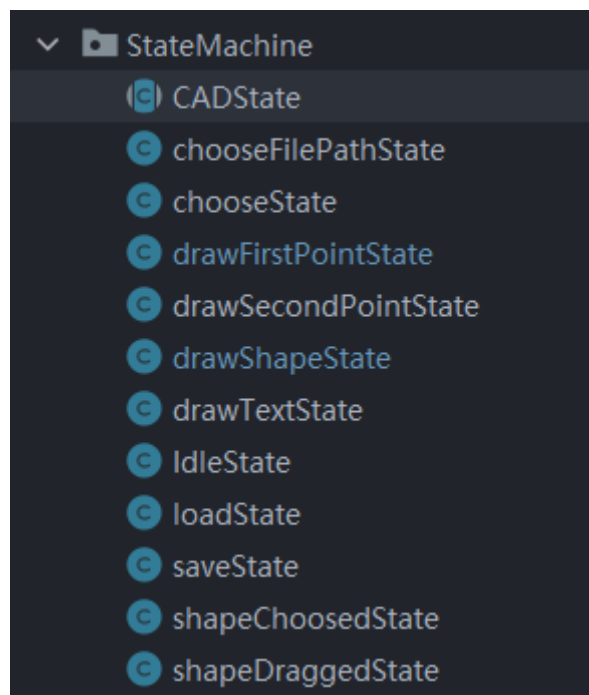
- 点击图形即可选中图形
- 在图形被选中的情况下
  - **点击颜色按钮可以调整图形颜色\*\***
  - **可以拖动图形调整位置**
  - **可以使用鼠标滚轮调整图形大小**
  - 可以点击删除按钮删除图形
  - 点击其他图形来选择其他图形
- 矩形
  - 点击两个点来绘制矩形
- 圆形
  - 点击一个点选择圆心
  - 点击第二个点来决定半径
- 直线
  - 点击第一个点来选择直线的第一个点
  - 点击第二个点来选择直线的第二个点
- 文本
  - 点击一个点来选择文本框生成位置
  - 输入文本生成文字
- 删除
  - 选中图形的条件下，点击可以删除图形
- 载入
  - 载入\*.cad文件
- 保存
  - 保存\*.cad文件

## 关键实现介绍

### 状态机



项目使用了状态机来管理整个程序的运行。



我们以状态机的父类 `CADState` 为例：

```
1 public abstract class CADState {
2     protected String stateName;
3     protected String notify;
4     private MiniCAD miniCAD;
5
6     CADState(MiniCAD miniCAD) {
7         this.miniCAD = miniCAD;
8     }
9
10    public void setStateName(String stateName) {
11        System.out.println("\nnow state: " + stateName);
12        this.stateName = stateName;
13    }
14
15    public void setNotify(String notify) {
16        this.notify = notify;
17        miniCAD.setNotifyLable(getStateName()+" : "+notify);
18    }
19
20    public abstract void handle();
21    public abstract void clickButton(ActionEvent event);
22    public abstract void mouseDragged(MouseEvent event);
23    public abstract void mouseReleased(MouseEvent event);
24    public abstract void mouseMoved(MouseEvent event);
25    public abstract void mouseEntered(MouseEvent event);
26    public abstract void mouseExited(MouseEvent event);
27    public abstract void mouseClicked(MouseEvent event);
28    public abstract void mousePressed(MouseEvent event);
29    public abstract void mouseWheelMoved(MouseWheelEvent event);
30    public String getNotify() {
31        return notify;
32    }
33
34    public String getStateName() {
```

```

35         return stateName;
36     }
37
38     protected void setNextState(CADState nextState){
39         System.out.println("(state change:" +
miniCAD.nowState.getStateName() + " -> " + nextState.getStateName()+")");
40         //不能在此处设置nowState，因为构造函数会先于设置nowState出发，导致状态机的实际
状态与nowState发生冲突
41         //正确的设置方法是在状态的构造函数内，handle()之前设置nowState
42     }
43
44     public MiniCAD getMiniCAD(){
45         return miniCAD;
46     }
47
48     /*
49     * @description: 用于统一处理按钮的函数。写在这里是为了避免在不同状态机一直写同一大
段代码
50     * @param event: 按钮事件
51     * @return void
52     */
53     protected void logButtonName(ActionEvent event){
54         System.out.println("button clicked: "+event.getActionCommand());
55     }
56     protected void logSwitchDefault(String switchCaseName){
57         System.out.println("An switch default is occurred: " +
switchCaseName);
58     }
59
60 }

```

一个 CADState 里面包括了这个名字，这个 State 对应的状态提示框的信息，基本的行动函数 `handle()`、完整的事件监听处理函数以及一些其他行为函数。函数 `handle()` 将会在状态构造函数的最后调用。

可以看到我们在 CADState 中设置了各种事件的回调函数，这使得我们在 MiniCAD 类中的监听器可以直接调用现在的状态的事件处理函数。

MiniCAD 中对于事件的回调处理如下：

当某一监听事件发生时，直接调用 `nowState` 对应的监听处理方法来处理监听事件的发生，从而达成状态机切换。

```

1 public CADState nowState; //目前的状态
2
3 public MiniCAD(){
4     //顺序不能随意调换，需要先初始化state和actionListener,最后才能加鼠标监听
5     nowState = new IdleState(this);
6     actionListenerInit();
7     /*
8     * 设置各种组件，略
9     */
10    setMouseListener();
11    mainJFrame.setVisible(true);
12 }

```

```
13
14 {...略}
15
16 private void setMouseListener(){
17     drawBoard.addMouseListener(new MouseListener() {
18         @Override
19         public void mouseClicked(MouseEvent e) {
20             nowState.mouseClicked(e);
21         }
22
23         @Override
24         public void mousePressed(MouseEvent e) {
25             nowState.mousePressed(e);
26         }
27
28         @Override
29         public void mouseReleased(MouseEvent e) {
30             nowState.mouseReleased(e);
31         }
32
33         @Override
34         public void mouseEntered(MouseEvent e) {
35             nowState.mouseEntered(e);
36         }
37
38         @Override
39         public void mouseExited(MouseEvent e) {
40             nowState.mouseExited(e);
41         }
42     });
43     drawBoard.addMouseMotionListener(new MouseMotionListener() {
44         @Override
45         public void mouseDragged(MouseEvent e) {
46             nowState.mouseDragged(e);
47         }
48
49         @Override
50         public void mouseMoved(MouseEvent e) {
51             nowState.mouseMoved(e);
52         }
53     });
54     drawBoard.addMouseWheelListener(new MouseWheelListener() {
55         @Override
56         public void mouseWheelMoved(MouseWheelEvent e) {
57             nowState.mouseWheelMoved(e);
58         }
59     });
60 }
61 private void actionListenerInit(){
62     actionListener = new ActionListener() {
63         @Override
64         public void actionPerformed(ActionEvent e) {
65             nowState.clickButton(e);
66         }
67     };
68 }
```

使用状态机的好处就是让整个代码结构变得更加清晰，极大地增加了代码的可读性和可维护性。

各个状态机的切换一目了然，Debug信息（打log）也十分的方便。

## Shape类

基础的图形类。

```

1  public abstract class Shape implements Serializable {
2      protected int x,y;
3      protected Color color;
4
5      public Shape(int x, int y, Color color) {
6          this.x = x;
7          this.y = y;
8          this.color = color;
9      }
10     public void setColor(Color color){
11         this.color = color;
12     }
13     public void moveTo(int toX,int toY){
14         x = toX; y = toY;
15     }
16     public abstract void addSize(float dsize);
17     public abstract void subSize(float dsize);
18     public abstract void draw(Graphics graphics);
19     public abstract boolean isIn(int nowX, int nowY);
20 }
```

有四个子类 `Rectangular`、`Circle`、`Line`、`Text`。

其中需要实现的抽象函数有：

- `addSize()` 增大图形
- `subSize()` 减小图形
- `draw()` 绘制图形
- `isIn()` 查看某个点是否在图形内

## 绘图与重绘

用于重绘整个`drawBoard`。一般在图形有改变时调用

具体思路为：清除整个程序并重绘。

```

1 public void paintAllShapes(Graphics2D graphics){
2     graphics2D = (Graphics2D) drawBoard.getGraphics();
3     graphics2D.setColor(DefaultSettings.DRAW_BOARD_BGCOLOR);
4     drawBoard.getGraphics().clearRect(0,0,
5         DefaultSettings.WINDOW_WIDTH,
6         DefaultSettings.WINDOW_HEIGHT);
7     for(Shape shape: shapes){
8         shape.draw(graphics2D);
9     }
10 }

```

## DefaultSettings

用于储存程序内使用的各种数据，从而避免程序中直接出现硬编码，以便于程序的维护。

```

1 package MiniCAD;
2
3 import java.awt.*;
4
5 public class DefaultSettings {
6
7     //region UI default settings
8     public static final int WINDOW_WIDTH = 1200; //窗口宽度
9     public static final int WINDOW_HEIGHT = 815; //窗口高度
10    public static final int TOOLBAR_WIDTH = 90; //工具栏宽度
11    public static final int BOTTOMBAR_WIDTH = WINDOW_WIDTH -
12    TOOLBAR_WIDTH; //底部栏宽度
13    public static final int BOTTOMBAR_HEIGHT = 30; //底部栏高度
14    public static final int TOOLBAR_HEIGHT = WINDOW_HEIGHT -
15    BOTTOMBAR_HEIGHT; //工具栏高度
16    public static final int TOOL_ICON_SIDE_LENGTH = 50; //工具图标边长
17    public static final int COLOR_SIDE_LENGTH = 40; //颜色选择器边长
18    public static final Color TOOLBAR_BGCOLOR = Color.lightGray; //ToolBar的背景颜色
19    public static final Color TOOLBUTTON_BGCOLOR = Color.white; //ToolButton的背景颜色
20    public static final Color BOTTOMBAR_BGCOLOR = Color.white; //BottomBar的背景颜色
21    public static final Color NOTIFY_COLOR = Color.black; //Notify文字的颜色
22    public static final Color DRAW_BOARD_BGCOLOR = Color.white; //drawBoard的背景颜色
23    public static final String TEXT_LABEL_CONTENT = "Input Text(No longer than 100)"; //输入text的弹出框的label的文字
24    public static final String TEXT_FRAME_NAME = "Text"; //输入text的弹出框的框名
25    public static final int TEXT_WIDTH = 400; //输入text的弹出框的框宽度
26    public static final int TEXT_HEIGHT = 150; //输入text的弹出框的框高度
27    public static final int TEXT_COLOMN = 32; //输入text的弹出框的文字列数
28    //endregion
29
30    //region shape default settings
31    public static final float SHAPE_DEFAULT_DELTA_SIZE = 5.0f; //shape的默认增大增量度
32    public static final int TEXT_DEFAULT_SIZE = 20;

```

```
31     public static final float SHAPE_MIN_SIZE = 10.0f; //shape的最小大小
32     public static final float LINE_NEAR_RANGE = 3.0f; //用于判断某个点是否在Line
    的旁边，基本用于Line的isIn
33     public static final int FONT_SIZE_BASE = 1; //font-size的基数
34     public static final Color DEFAULT_SHAPE_COLOR = Color.black; //默认shape颜
    色
35     //endregion
36 }
37
```