Java-MiniCAD

1 实验目的

- 学习Java的Awt和Swing知识
- 用Java的Awt和Swing做一个简单的绘图工具。要求以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。

2 实验环境

Java版本：1.8.0_351

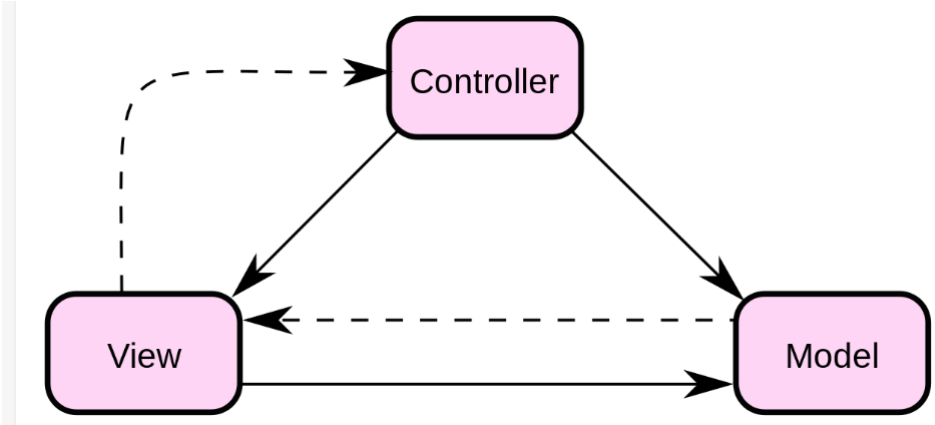
实验环境：Windows

3 实验原理

3.1 整体结构

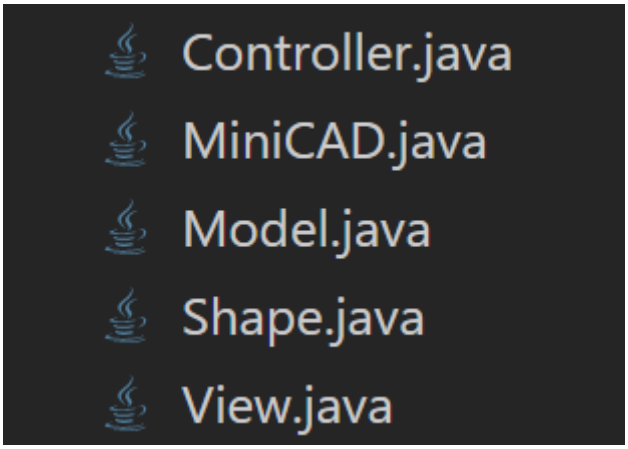
在本次实验中，学习了MVC架构，即Model-View-Controller（模型-视图-控制器） 模式。其逻辑模式为：

- **Model（模型）** - 模型代表一个存取数据的对象。在对象发送变化时，控制器会更新模型。
- **View（视图）** - 视图代表模型包含的数据的可视化。
- **Controller（控制器）** - 控制器作用于模型和视图上。它控制既控制程序中对象的加减变化，也在模型变化时即时更新视图。



3.2 框架说明

本项目工程一共包含五个java文件，分别是主程序-MiniCAD.java；控制器Controller.java；模型Model.java；视图View.java；以及服务于模型Model.java的Shape.java，包含在本次工程中使用的shape对象，譬如直线、矩形、椭圆、文字块。



3.3 MiniCAD

核心代码

```
public class MiniCAD extends JFrame{
    ......//实现JFrame的布局
}
public static void main(String[] args){
    MiniCAD frame=new MiniCAD();//创建新类
    View view=new View();
    ArrayList<Shape> shapes = new ArrayList<Shape>();
    Model model=new Model(shapes);//初始化两个model和view
    frame.add(view,BorderLayout.CENTER);//主界面放在中间
    frame.setFocusable(true);
    frame.setVisible(true);//视为可见
    frame.requestFocus();
    @SuppressWarnings("unused")
    Controller ctr=new Controller(model,view);//使用控制器进行控制
}
```

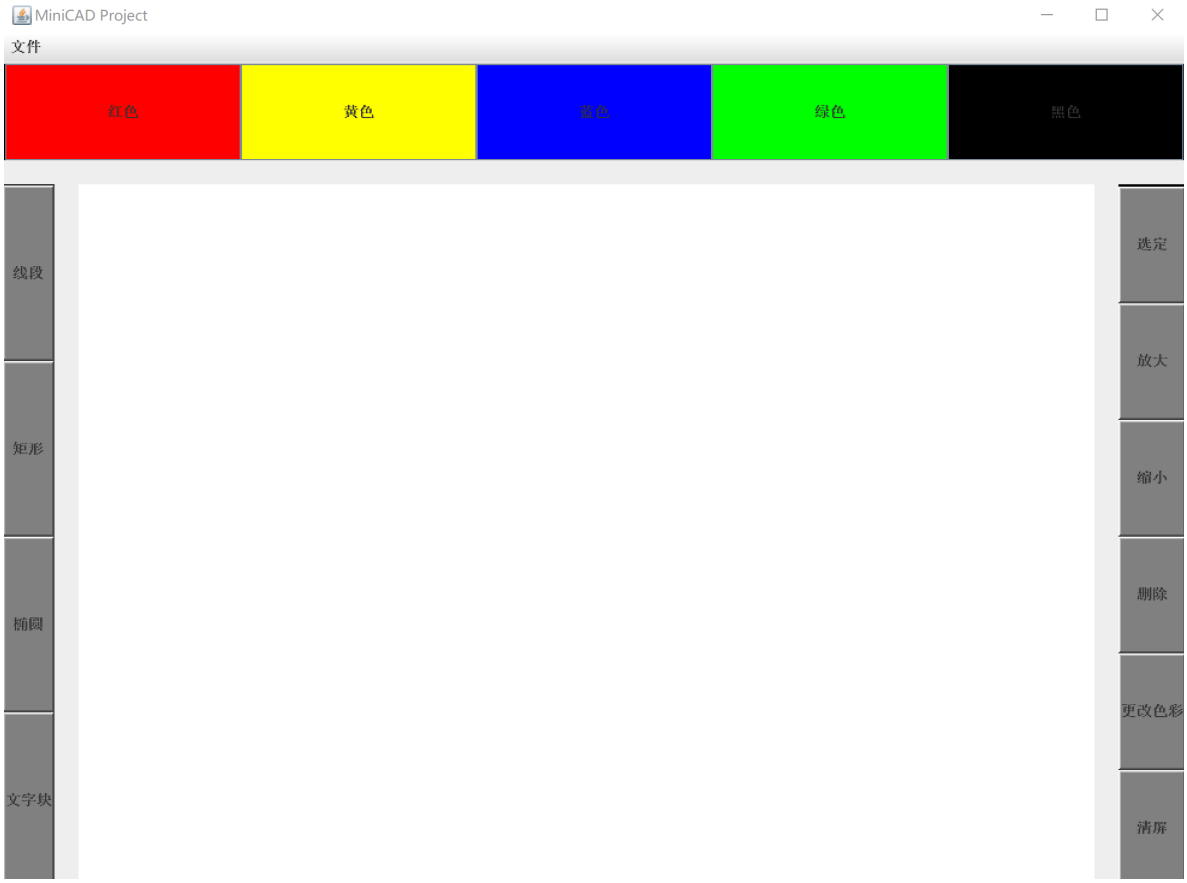
代码说明

在主程序中，首先创建一个MiniCAD类，初始化布局。

创建一个Shape的ArrayList以供Model使用，记录整个画布上存在的Shape对象。

创建一个View，放在MiniCAD的正中间，用来显示Model类中记录的所有Shape对象。

在省略的MiniCAD定义中，定义了WEST、EAST、NORTH部件和JMenuBar部件。



3.4 Shape

核心代码

```
public abstract class Shape implements Serializable{
    private static final long serialVersionUID = 1L;
    public ArrayList<Point> points=new ArrayList<Point>();//每种图形都是两个节点
    public Color color;//图形的颜色
    public float thick;//图形的粗细
    public Shape(Point p1,Point p2,Color CColor)//进行初始化定义
    {
        points.add(new Point(p1.x, p1.y));
        points.add(new Point(p2.x, p2.y));
        color = CColor;
        thick = 4;
    }//无法保证点的左右，因为后续拉拽的时候会导致第二个点的位置不确定
    public void CHANGE(int position){//用于键盘控制方向
        int del=4;
        int x1=points.get(0).x;
        int x2=points.get(1).x;
        int y1=points.get(0).y;
        int y2=points.get(1).y;
        switch (position){
            case 0://向上
                y1-=del;
                y2-=del;
                break;
            case 1://向下
                y1+=del;
                y2+=del;
```

```
                break;
            case 2://向左
                x1-=del;
                x2-=del;
                break;
            case 3://向右
                x1+=del;
                x2+=del;
                break;
        }
        points.get(0).y=y1;
        points.get(1).y=y2;
        points.get(0).x=x1;
        points.get(1).x=x2;
    }
    public void setColor(Color ccolor){//用于更改图形颜色
        color=ccolor;
    }
    public abstract void draw(Graphics g);//画图
    public abstract boolean isSelected(Point p);//确认是否鼠标选中
    public abstract void ChangeSize(int delta);//更改图形大小
}
class Line extends Shape{......}
class Rectangle extends Shape{......}
class Oval extends Shape{......}
class Text extends Shape{......}
```

代码说明

对于Shape类，有3个内部量。分别是颜色color，粗细thick，以及记录图形位置的两个点Points。

有六个函数，三个函数已实现：

- Shape函数，进行初始化，记录信息；
- CHANGE函数，接收控制器发来的键盘信息，通过上下左右的分辨，更改图形位置
- setColor函数，接收控制器发来的颜色信息，并设置当前Shape的颜色

此外，还有三个函数需要由子类实现：

- draw函数，用来在view上画出此图形，根据直线、矩形、椭圆、文本的不同，有不同的实现
- isSelected函数，接收当前控制器发来的鼠标点击的坐标值，根据与此shape的位置进行对比，判断是否选中
- ChangeSize函数，通过接受控制器发来的放大或缩小命令，对不同的shape坐标进行改变，实现放大缩小。

细节补充

对于Line，放大缩小需要先算出斜率，然后计算。

对于Rectangle、Oval、Text，Points记录的是两个对角的坐标，由于绘画需要的是左上方点的横纵坐标和宽度高度，并且由于首次绘画的拖拽和各种改变，因此每次绘画时需要通过对比找到这四个值。

3.5 Model

核心代码

```
public class Model {
    private ArrayList<Shape> shapes; //存放所有图形
    Model(ArrayList<Shape> s) {
        shapes = s;
    }

    public void setAll(ArrayList<Shape> s) {//用于打开文件
        shapes = s;
    }

    public ArrayList<Shape> getAll() {
        return shapes;
    }

    public Shape Last() {
        return shapes.get(shapes.size()-1);
    }

    public void add(Shape shape) {
        shapes.add(shape);
    }

    public void removeAll() {
        shapes.clear();
    }
}
```

代码说明

Model类包含一个ArrayList来记录所有的shape，包含了初始化函数，赋值函数，返回值函数，添加shape的函数，移除所有shape的函数，以及辅助控制器中拖拽的Last函数。

3.6 View

核心代码

```
public class View extends JPanel {
    private static final long serialVersionUID = 1L;
    ArrayList<Shape> shapes = new ArrayList<Shape>();
    View() {//初始化定义,对view添加鼠标的事件监听
        setBackground(Color.white);
        addMouseListener(Controller.pl);
        addMouseMotionListener(Controller.pl);
        addKeyListener(Controller.pl);
    }
    public void paintAll(ArrayList<Shape> s) {
        shapes = s;
        repaint();//无法重写的内置函数，进行重新绘制
        //也就是将图面全白，然后调用paint函数
    }
    public void paint(Graphics g) {
        super.paint(g);
        if(!shapes.isEmpty()) {
            for(Shape shape: shapes)
                shape.draw(g);
```

```
        }
    }
}
```

代码说明

包含初始化函数，设置中心画布的颜色，给中心画布添加鼠标和键盘的控制器。以及设置设置自带的paint和paintAll（**repaint会自动清空幕布，然后调用paint函数**）

3.7 Controller

核心代码

```
public class Controller {
    public static Model model;
    public static View view;
    public static paintListener pl= new paintListener();
    public static String state = "idle";
    Controller(Model m, View v) {
        model = m;
        view = v;
    }
    public static void updateView() {
        view.paintAll(model.getAll());//每次出现更改就要调用一次次全屏幕的清空和重画
    }
}

class paintListener implements ActionListener, MouseListener,
MouseMotionListener,KeyListener {
    String state="idle";
    Color colornow=Color.black;//默认颜色为
    String text;
    Shape Shapenow;
    int colorchange=0;
    String[] ColorName = {"红色","黄色","蓝色","绿色","黑色"};
    Point tmp[]=new Point[2];
    Point p1,p2;//每个图形都是由两个点所标注而生的
    public void actionPerformed(ActionEvent e){......}
    public void mousePressed(MouseEvent e){......}
    public void mouseDragged(MouseEvent e){......}
    public void keyPressed(KeyEvent e){......}
    private void saveFile() {......}
    private void openFile() {......}
}
```

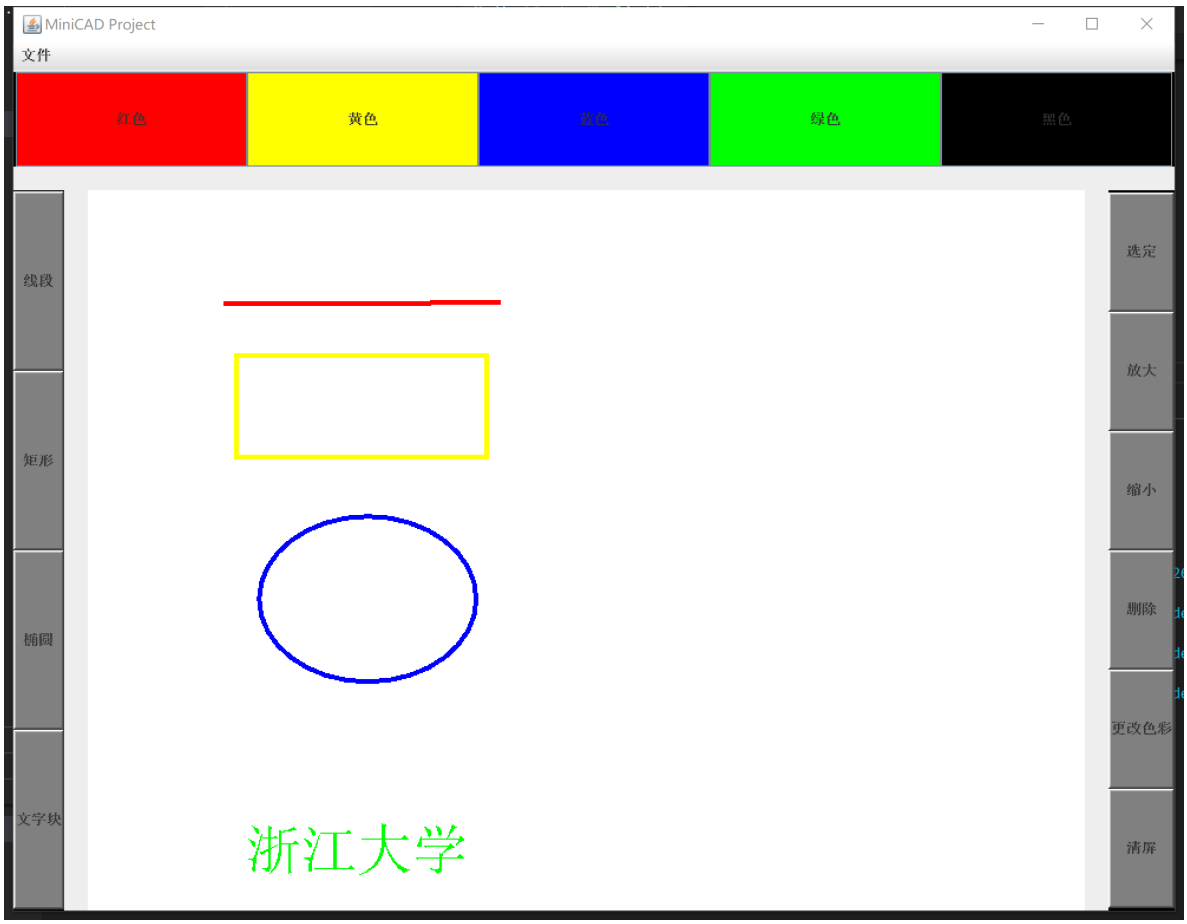
代码说明

- actionPerformed中，监控鼠标点击了什么按钮，调整控制器的state状态为对应状态
- mousePressed中，如果需要画新图形，检测首个点击点，并初始化一个shape到model中；如果是对已有的进行选择，则记录当前选择图形的坐标
- mouseDragged中，如果是新画图形，则动态根据拖拽点，更新此shape，并调用画出；如果是选中拖拽，亦如此。
- keyPressed中，检测键盘上下左右键，根据选中的图形，调用shape中的上下左右变化。

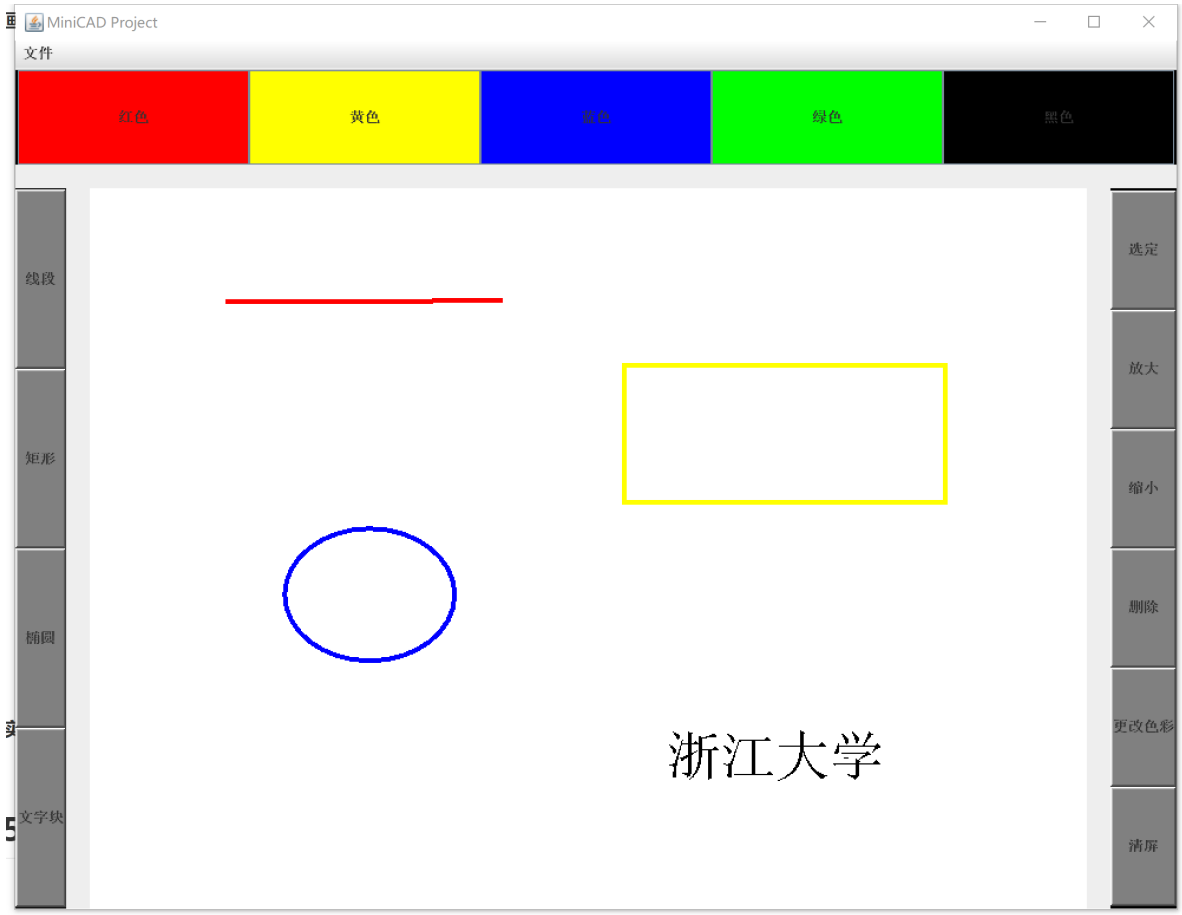
- saveFile打印，调用保存函数，实现保存
- openfile打开，调用打开函数，打开之前savefile保留的文件

4 实验结果

画图形

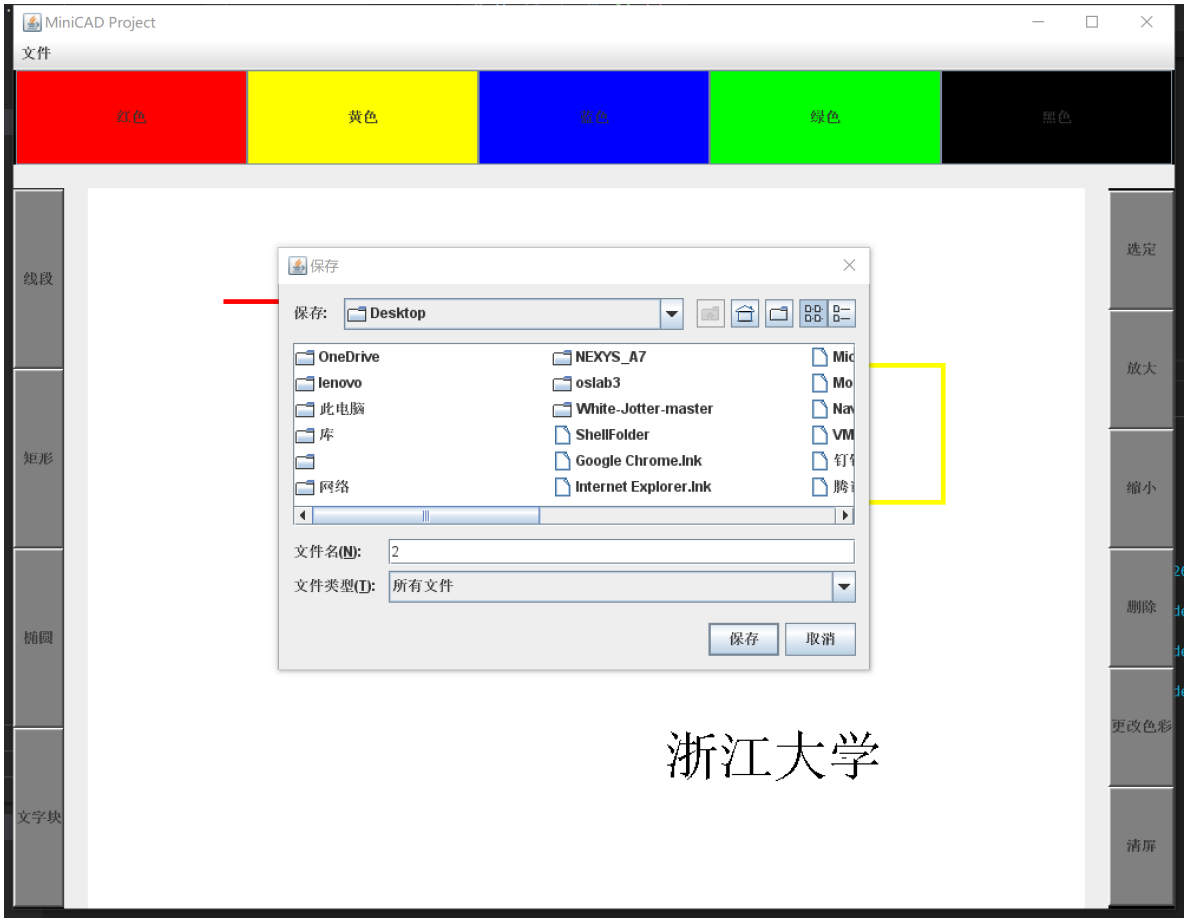


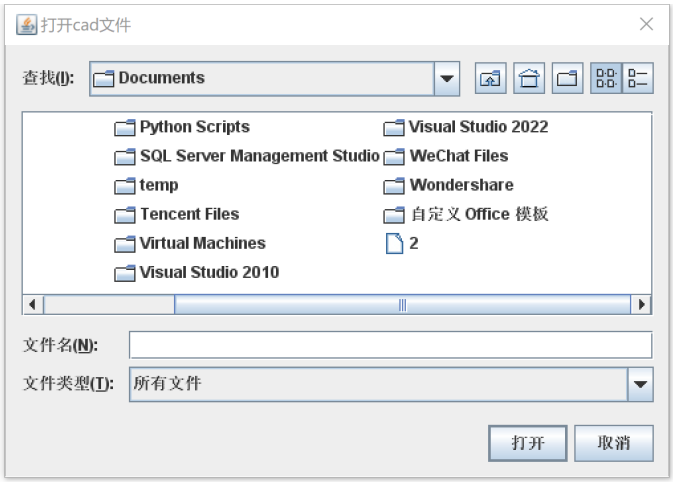
实现右侧功能

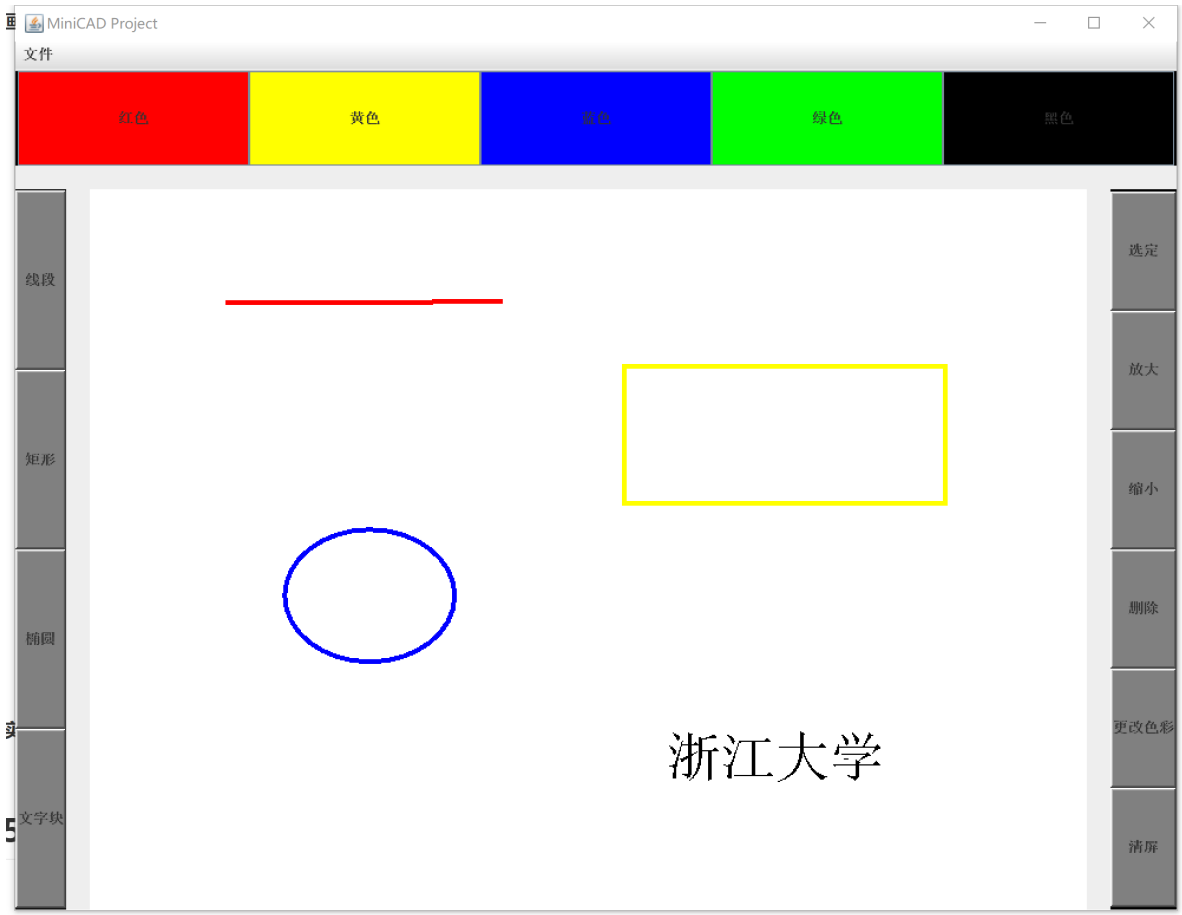


在此实现了选定拖动线段，放大长方形，缩小椭圆，更改文字颜色能功能

保存与打开







在此实现了文件的保存与打开，保存与打开的命名都是2.

5 实验心得

在本次实验中，实现的是之前大一C大程中已经尝试过的MiniCAD。相比于大一胡乱的去做，这次再学习了MVC模式后，整个代码模块会更有逻辑。而在我实现的过程中，碰到了几个问题比较突出，这里进行记录：

（1）还是关于图形首次画出时的拖拽问题，这个在大一就碰到过，这次实验中本来在初始化设置中设置了前一个点左后一个点右，后来发现由于首次画出的拖拽，会使第二个点到第一个点的左边，初始化的设置失败。因此在除线段之外的shape，都直接进行了一次数学比较，直接求出四个值然后去实现操作。其中后三个shape的操作非常像，思考也许可以在基类中加一个函数去求这几个值在后续只要直接调用即可

（2）关于键盘和鼠标的控制设置上，发现键盘控制一直没有反应，后来发现是要去找焦点

（3）关于如何更改坐标，java是对类进行传递的，对一个类进行传递（不是new一个然后进行相等赋值）后可以进行值的更改，这一点是和C++内的指针区别开来的，也是我觉得比较重要的地方。