简单的画图

孙博言　　2011756　　计算机学院

2021 年 11 月 28 日

1　功能展示

图 1: 界面展示

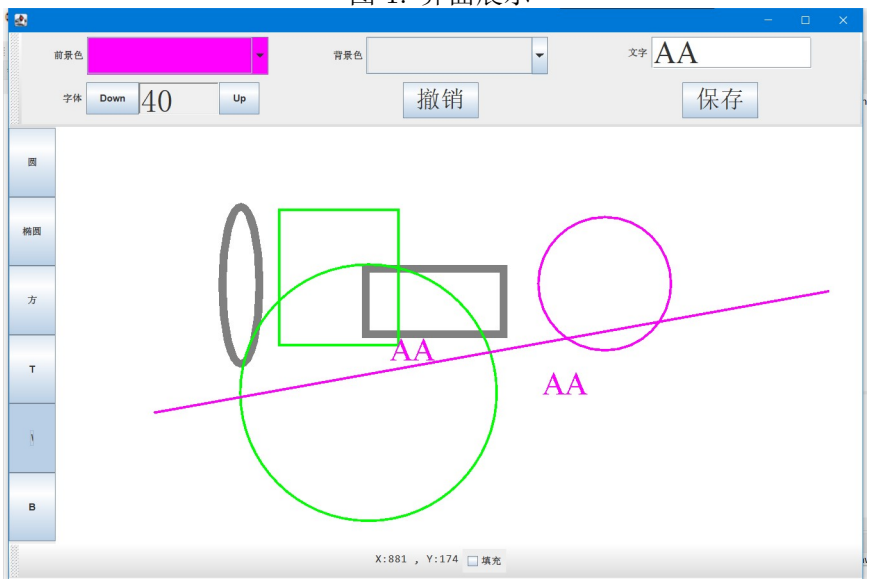


图 2: 形状总览

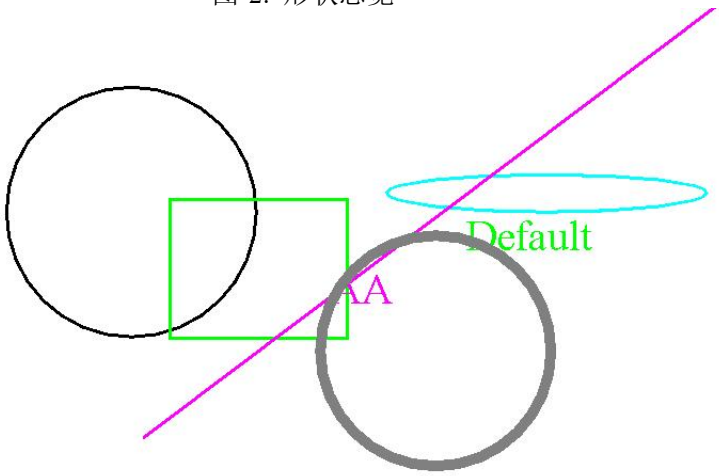


图 3: 线条加粗

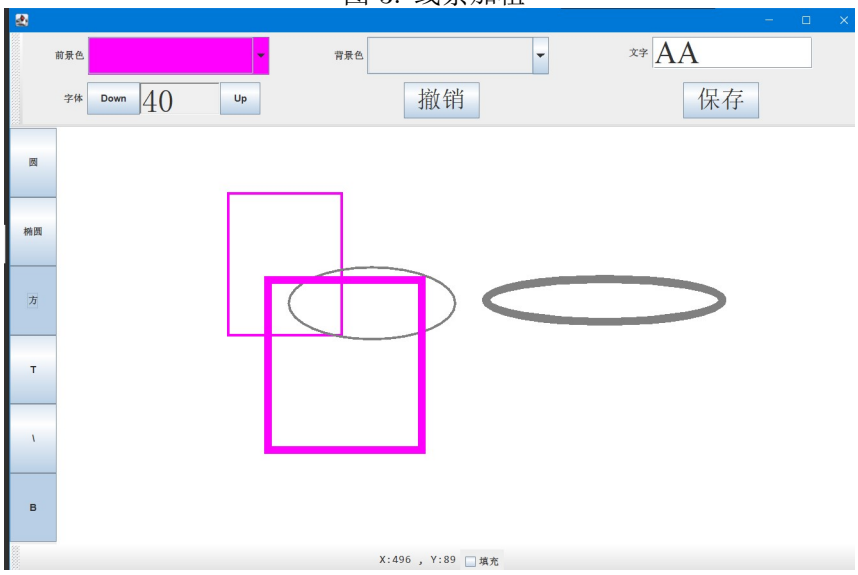


图 4: 局部填充

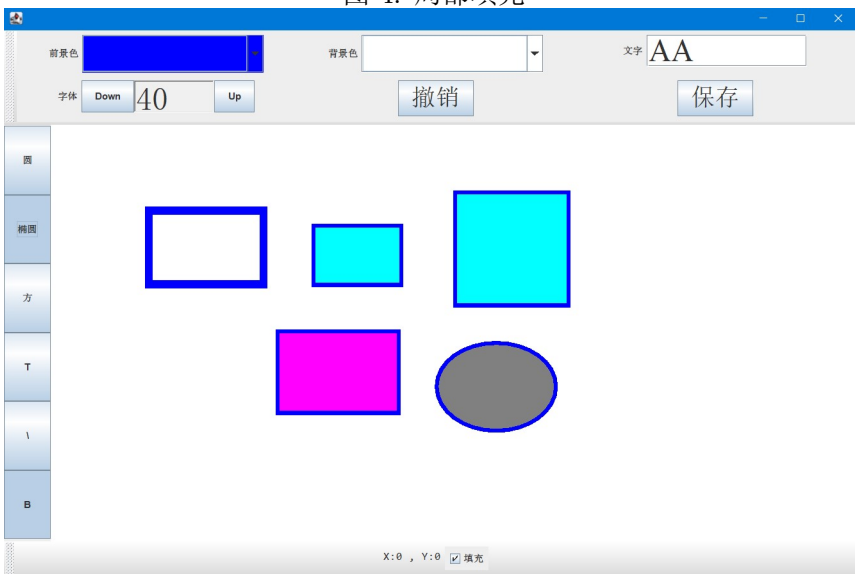


图 5: 字体大小

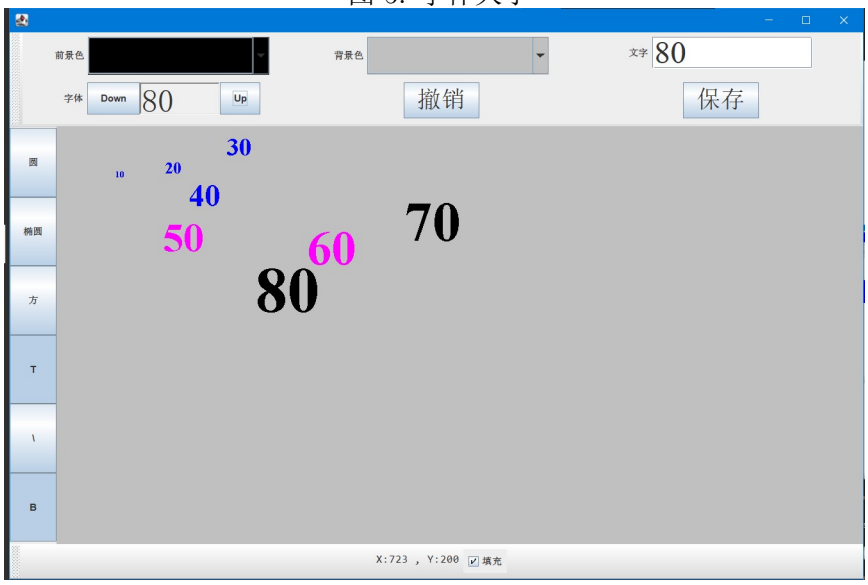


图 6: 自动缓存图片 (形状数目每超过 10)

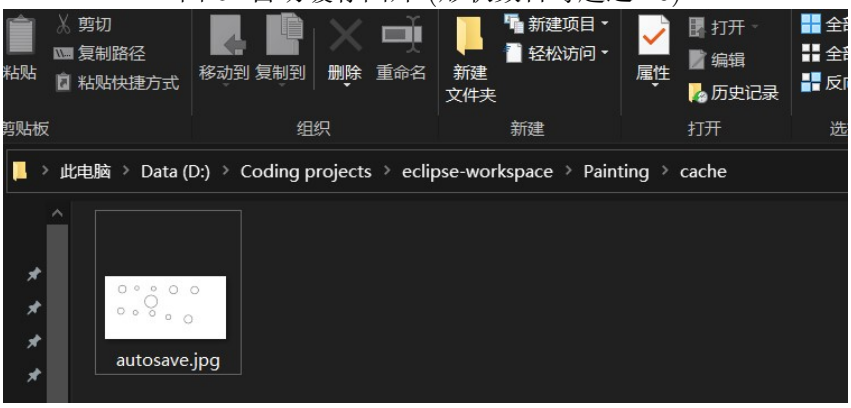


图 7: 自动缓存图片 (形状数目每超过 10)

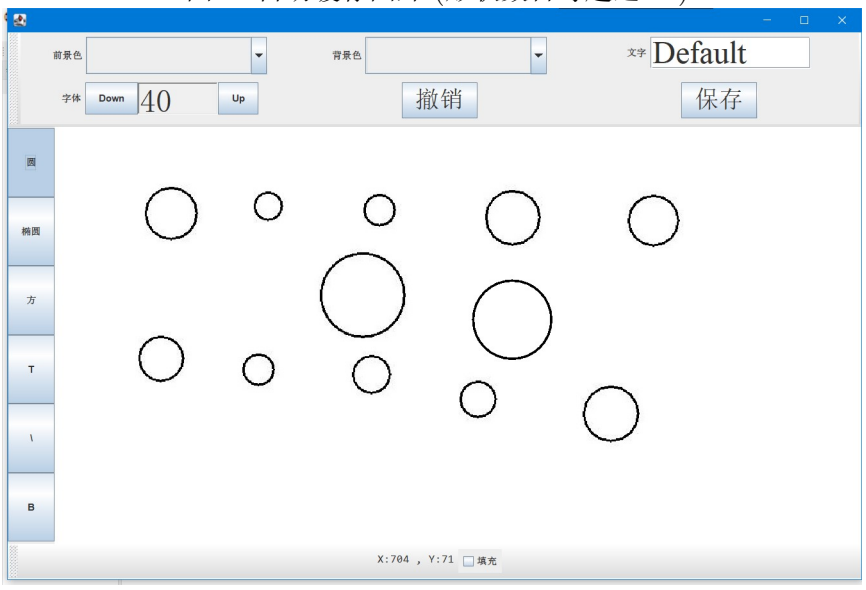


图 8: 在 10 步以内的动作可以撤回 (可自行运行代码尝试)

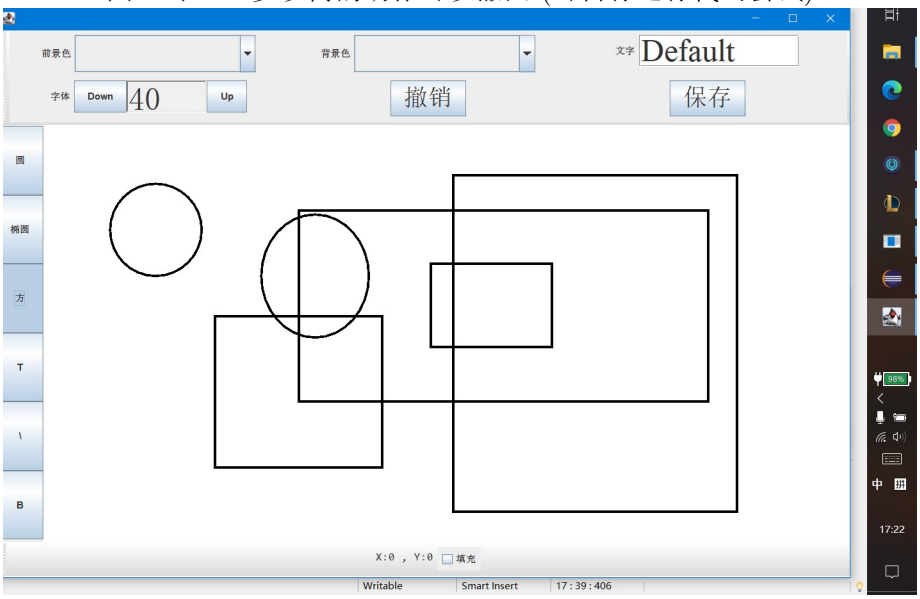


图 9: 撤回后

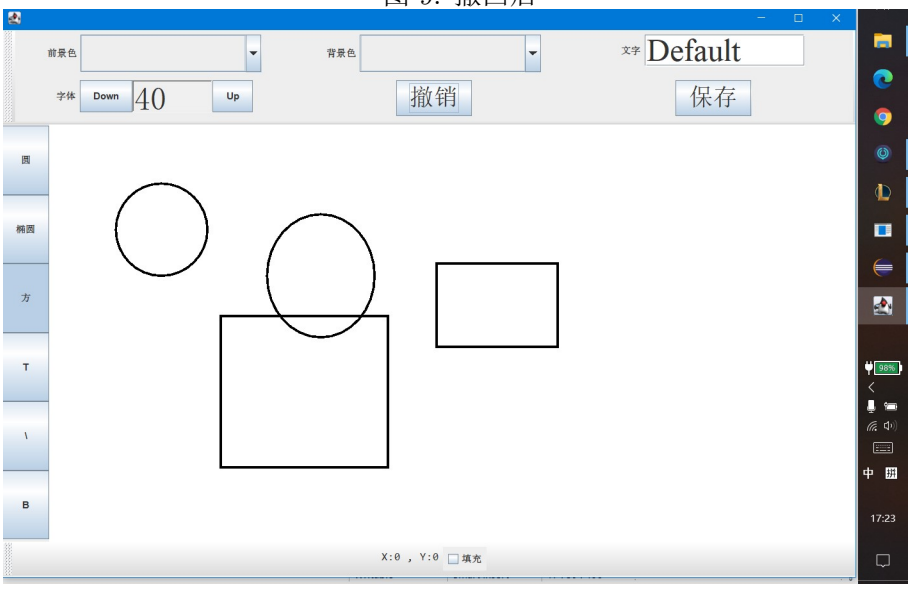


图 10: 另存为 jpg

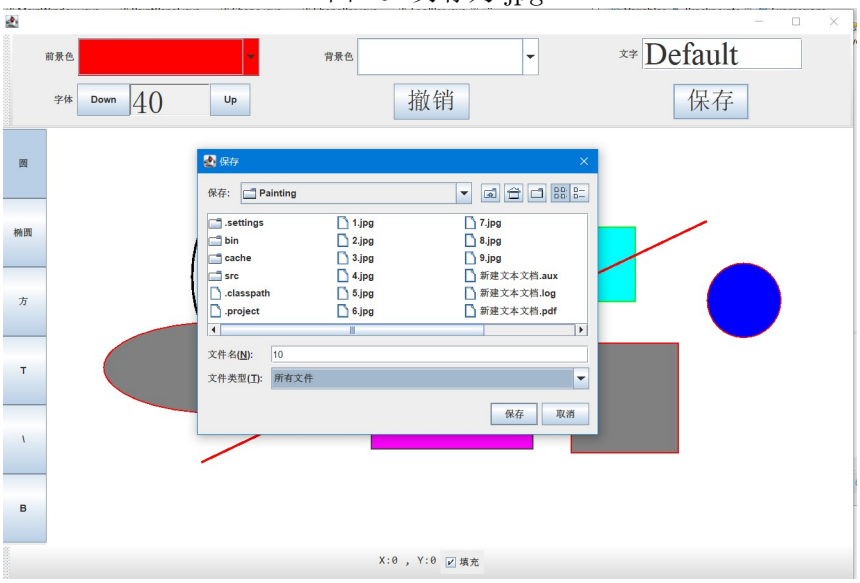
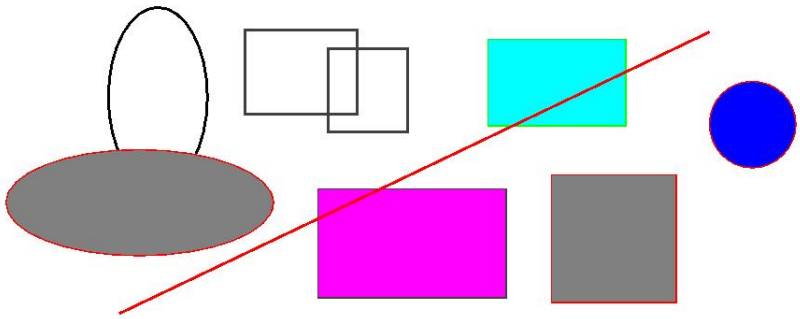
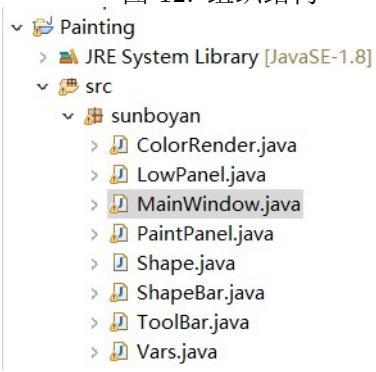


图 11: 存储结果



2 代码

图 12: 组织结构



代码位于：

Listing 1: Vars.Java

```
1  package sunboyan;
2
3  import ...
4  public class Vars {
5
6      private static ToolBar toolBar = null;
7      private static JComboBox<Color> shapeColorBox = null;
8      private static JComboBox<Color> backColorBox = null;
9      private static JTextField textField = null;
10     private static JButton upButton = null;
11     private static JButton downButton = null;
12     private static JPanel fontPanel = null;
13     private static Color[] colors = {Color.BLACK,Color.blue,Color.CYAN,Color.DARK_GRAY, Color.GRAY,
           Color.GREEN,Color.LIGHT_GRAY,Color.MAGENTA,Color.ORANGE,Color.RED,Color.WHITE,Color.YELLOW
           };
14     private static ColorRender render = null;
15     private static TextField fontLabel = null;
16
17     private static ShapeBar shapePanel = null;
18     private static JToggleButton circleButton = null;
19     private static JToggleButton squareButton = null;
20     private static JToggleButton textButton = null;
21     private static JToggleButton ellipseButton = null;
22     private static JToggleButton lineButton = null;
23     private static JToggleButton boldButton = null;
24
25     private static PaintPanel paintPanel = null;
26     private static Color color = Color.BLACK;
27     private static Color backColor = Color.white;
28     private static Shape shape = null;
29     public static Shape [] items = null;
30     public static int nums = 0;
31     private static String shapeString = "line";
32     private static Font font =null;
33
```

```
34          private static LowPanel lowerPanel =null;
35          private static JLabel stateLabel = null;
36          private static JCheckBox checkBox  = null;
37          public static boolean isFilled = false;
38          public static boolean isBold = false;
39
40          public static JButton saveButton = null;
41          public static JButton undoButton = null;
42
43          public static ToolBar getToolPanel() {
44              if(toolBar == null) {
45                  toolBar = new ToolBar();
46              }
47              return toolBar;
48          }
49
50          public static JComboBox<Color> getShapeColorBox() {
51              if(shapeColorBox == null) {
52                  shapeColorBox = new JComboBox<>(getColors());
53                  shapeColorBox.addItemListener(new ItemListener() {
54                      @Override
55                      public void itemStateChanged(ItemEvent e) {
56                          // TODO Auto-generated method stub
57                          Vars.color = (Color)shapeColorBox.getSelectedItem();
58                          shapeColorBox.setBackground(color);
59                          shapeColorBox.setFocusable(false);
60                      }
61                  });
62                  shapeColorBox.setRenderer(getColorRender());
63              }
64              return shapeColorBox;
65          }
66
67          public static JComboBox<Color> getBackColorBox() {
68              if(backColorBox == null) {
69                  backColorBox = new JComboBox<>(getColors());
70                  backColorBox.addItemListener(new ItemListener() {
71
72                      @Override
73                      public void itemStateChanged(ItemEvent e) {
74                          // TODO Auto-generated method stub
75                          Vars.backColor = (Color)backColorBox.getSelectedItem();
76                          backColorBox.setBackground(backColor);
77                          backColorBox.setFocusable(false);
78                      }
79                  });
80                  backColorBox.setRenderer(getColorRender());
81              }
82              return backColorBox;
83          }
84
85          public static ColorRender getColorRender() {
86              if(render == null) {
87                  render = new ColorRender();
88                  render.setPreferredSize(new Dimension(200,40));
89              }
90              return render;
91          }
92
93          public static JTextField getTextField() {
94              if(textField == null) {
95                  textField= new JTextField();
96                  textField.setPreferredSize(new Dimension(200,40));
```

```
97                 textField.setText("Default");
98                 textField.setFont(new Font("Times New Roman",Font.PLAIN,40));
99             }
100            return textField;
101        }
102
103        public static JButton getUpButton() {
104            if(upButton == null) {
105                upButton= new JButton();
106                upButton.setText("Up");
107                upButton.addMouseListener(new MouseAdapter() {
108                    @Override
109                    public void mouseReleased(MouseEvent e) {
110                        addSize();
111                        Integer integer = getFont().getSize();
112                        getFontLabel().setText(integer.toString());
113                    }
114                });
115            }
116            return upButton;
117        }
118
119        public static JButton getDownButton() {
120            if(downButton == null) {
121                downButton= new JButton();
122                downButton.setText("Down");
123                downButton.addMouseListener(new MouseAdapter() {
124                    @Override
125                    public void mouseReleased(MouseEvent e) {
126                        subSize();
127                        Integer integer = getFont().getSize();
128                        getFontLabel().setText(integer.toString());
129                    }
130                });
131            }
132            return downButton;
133        }
134
135        public static JPanel getFontPanel() {
136            if(fontPanel == null) {
137                fontPanel = new JPanel();
138                fontPanel.setLayout(new BorderLayout());
139                fontPanel.add(getFontLabel(),BorderLayout.CENTER);
140                fontPanel.add(getUpButton(),BorderLayout.EAST);
141                fontPanel.add(getDownButton(),BorderLayout.WEST);
142            }
143            return fontPanel;
144        }
145
146        public static Color [] getColors() {
147            return colors;
148        }
149
150        public static TextField getFontLabel() {
151            if(fontLabel == null) {
152                fontLabel = new TextField();
153                fontLabel.setPreferredSize(new Dimension(100,40));
154                fontLabel.setEditable(false);
155                fontLabel.setFont(new Font("Consolas", Font.PLAIN, 40));
156                fontLabel.setText("40");
157            }
158            return fontLabel;
159        }
```

```
160
161         public static ShapeBar getShapePanel() {
162             if(shapePanel == null) {
163                 shapePanel = new ShapeBar();
164             }
165             return shapePanel;
166         }
167 
168         public static JToggleButton getCircleButton() {
169             if(circleButton == null) {
170                 circleButton= new JToggleButton("圆",false);
171                 circleButton.setSize(60, 60);
172                 circleButton.addMouseListener(new MouseAdapter() {
173                     @Override
174                     public void mousePressed(MouseEvent e) {
175                         // TODO Auto-generated method stub
176                             getSquareButton().setSelected(false);
177                             getEllipseButton().setSelected(false);
178                             getTextButton().setSelected(false);
179                             getLineButton().setSelected(false);
180                             Vars.setString("circle");
181                         }
182 
183                 });
184             }
185             return circleButton;
186         }
187 
188         public static JToggleButton getSquareButton() {
189             if(squareButton == null) {
190                 squareButton= new JToggleButton("方",false);
191                 squareButton.setSize(60,60);
192                 squareButton.setText("方");
193                 squareButton.addMouseListener(new MouseAdapter() {
194                     @Override
195                     public void mousePressed(MouseEvent e) {
196                         // TODO Auto-generated method stub
197                             getCircleButton().setSelected(false);
198                             getEllipseButton().setSelected(false);
199                             getTextButton().setSelected(false);
200                             getLineButton().setSelected(false);
201                             Vars.setString("square");
202                         }
203 
204                 });
205             }
206             return squareButton;
207         }
208 
209         public static JToggleButton getEllipseButton() {
210             if(ellipseButton == null) {
211                 ellipseButton= new JToggleButton("椭圆",false);
212                 ellipseButton.setSize(60,60);
213                 ellipseButton.addMouseListener(new MouseAdapter() {
214                     @Override
215                     public void mousePressed(MouseEvent e) {
216                         // TODO Auto-generated method stub
217                             getCircleButton().setSelected(false);
218                             getSquareButton().setSelected(false);
219                             getTextButton().setSelected(false);
220                             getLineButton().setSelected(false);
221                             Vars.setString("ellipse");
222                         }
```

```
223
224                 });
225             }
226             return ellipseButton;
227         }
228
229         public static JToggleButton getTextButton() {
230             if(textButton == null) {
231                 textButton= new JToggleButton("T",false);
232                 textButton.setSize(60,60);
233                 textButton.addMouseListener(new MouseAdapter() {
234                     @Override
235                     public void mousePressed(MouseEvent e) {
236                         // TODO Auto-generated method stub
237                             getCircleButton().setSelected(false);
238                             getSquareButton().setSelected(false);
239                             getEllipseButton().setSelected(false);
240                             getLineButton().setSelected(false);
241                             Vars.setString("text");
242                     }
243
244                 });
245             }
246             return textButton;
247         }
248
249         public static JToggleButton getLineButton() {
250             if(lineButton == null) {
251                 lineButton = new JToggleButton("\\",false);
252                 lineButton.setSize(60,60);
253                 lineButton.addMouseListener(new MouseAdapter() {
254                     @Override
255                     public void mousePressed(MouseEvent e) {
256                         // TODO Auto-generated method stub
257                             getCircleButton().setSelected(false);
258                             getSquareButton().setSelected(false);
259                             getEllipseButton().setSelected(false);
260                             getTextButton().setSelected(false);
261                             Vars.setString("line");
262                     }
263
264                 });
265             }
266             return lineButton;
267         }
268
269         public static JToggleButton getBoldButton() {
270             if(boldButton==null) {
271                 boldButton = new JToggleButton("B");
272                 boldButton.addItemListener(new ItemListener() {
273
274                     @Override
275                     public void itemStateChanged(ItemEvent e) {
276                         // TODO Auto-generated method stub
277                         isBold = boldButton.isSelected();
278                         System.out.println(isBold);
279                     }
280                 });
281             }
282             return boldButton;
283         }
284
285         public static JButton getUndoButton() {
```

```
286             if(undoButton == null) {
287                 undoButton = new JButton("撤销");
288                 undoButton.setFont(new Font("宋体",Font.PLAIN,30));
289                 undoButton.addMouseListener(new MouseAdapter() {
290                     @Override
291                     public void mouseReleased(MouseEvent e) {
292                         subItems();
293                         paintPanel.repaint();
294                     }
295                 });
296             }
297             return undoButton;
298 
299         }
300 
301         public static JButton getSaveButton() {
302             if(saveButton == null) {
303                 saveButton = new JButton("保存");
304                 saveButton.setFont(new Font("宋体",Font.PLAIN,30));
305                 saveButton.addMouseListener(new MouseAdapter() {
306                     @Override
307                     public void mouseReleased(MouseEvent e) {
308                         getPaintPanel().save();
309                     }
310                 });
311             }
312             return saveButton;
313         }
314 
315         public static Color getColor() {
316             return color;
317         }
318 
319         public static void setColor(Color c) {
320             color = c;
321         }
322 
323         public static Color getBackColor() {
324             return backColor;
325         }
326 
327         public static void setBackColor(Color c) {
328             backColor = c;
329         }
330 
331         public static Shape getShape() {
332             if(shape == null) {
333                 shape = new Shape();
334             }
335             return shape;
336         }
337 
338         public static Shape[] getItems() {
339             if(items ==null) {
340                 items = new Shape[100];
341             }
342             return items;
343         }
344 
345         public static int getNums() {
346             return nums;
347         }
348 
```

```
349         public static void addItems() {
350             items[nums] = new Shape(shape);
351             nums++;
352         }
353 
354         public static void subItems() {
355             if(nums>0) {
356                 items[--nums] = null;
357             }
358         }
359 
360         public static void setString(String string) {
361             shapeString = string;
362         }
363 
364         public static String getString() {
365             return shapeString;
366         }
367 
368         public static PaintPanel getPaintPanel() {
369             if(paintPanel == null) {
370                 paintPanel = new PaintPanel();
371             }
372             return paintPanel;
373         }
374 
375         public static Font getFont() {
376             if(font==null) {
377                 font = new Font("Times New Roman", Font.PLAIN, 40);
378             }
379             return font;
380         }
381 
382         public static void addSize() {
383             if(font==null) {
384                 font = new Font("Times New Roman", Font.PLAIN, 42);
385             }else if(font.getSize()<80){
386                 font = new Font("Times New Roman",Font.PLAIN,font.getSize()+2);
387             }
388         }
389 
390         public static void subSize() {
391             if(font==null) {
392                 font = new Font("Times New Roman", Font.PLAIN, 38);
393             }else if(font.getSize()>2) {
394                 font = new Font("Times New Roman",Font.PLAIN,font.getSize()-2);
395             }
396         }
397 
398         public static  LowPanel getLowPanel() {
399             if(lowerPanel ==null) {
400                 lowerPanel = new LowPanel();
401             }
402             return lowerPanel;
403         }
404 
405         public static JLabel getStateLabel() {
406             if(stateLabel ==null) {
407                 stateLabel = new JLabel();
408                 stateLabel.setFont(new Font("Consolas", Font.PLAIN, 14));
409                 stateLabel.setText("X:0 , Y:0");
410             }
411             return stateLabel;
```

```
412         }
413
414         public static JCheckBox getCheckBox() {
415             if (checkBox == null) {
416                 checkBox = new JCheckBox("填充");
417                 checkBox.addItemListener(new ItemListener() {
418
419                     @Override
420                     public void itemStateChanged(ItemEvent e) {
421                         // TODO Auto-generated method stub
422                         isFilled = checkBox.isSelected();
423                     }
424                 } );
425             }
426             return checkBox;
427         }
428     }
```

Listing 2: PaintPanel.java

```
1   package sunboyan;
2   import ...
3   public class PaintPanel extends JPanel implements MouseListener,MouseMotionListener{
4       private static final long serialVersionUID = 1L;
5
6       @Override
7       public void mouseClicked(MouseEvent e) {
8           // TODO Auto-generated method stub
9
10      }
11
12      @Override
13      public void mousePressed(MouseEvent e) {
14          // TODO Auto-generated method stub
15          int x1 = e.getX();
16          int y1 = e.getY();
17          Vars.getShape().setX1(x1);
18          Vars.getShape().setY1(y1);
19      }
20
21      @Override
22      public void mouseReleased(MouseEvent e) {
23          // TODO Auto-generated method stub
24          int x2 = e.getX();
25          int y2 = e.getY();
26          Vars.getShape().setX2(x2);
27          Vars.getShape().setY2(y2);
28          Vars.getShape().setColor();
29          Vars.getShape().setString();
30          Vars.getShape().setFont(Vars.getFont());
31          Vars.getShape().setBackColor(Vars.getBackColor());
32          Vars.getShape().setTextString(Vars.getTextField().getText());
33          Vars.getShape().filled = Vars.isFilled;
34          Vars.getShape().bold = Vars.isBold;
35          Vars.addItems();
36          paintComponent(getGraphics());
37      }
38
39      @Override
40      public void mouseExited(MouseEvent e) {
41          // TODO Auto-generated method stub
42          Vars.getStateLabel().setText("X:0 , Y:0");
```

```
43          }
44
45
46
47          @Override
48          public void mouseDragged(MouseEvent e) {
49              // TODO Auto-generated method stub
50
51          }
52
53          @Override
54          public void mouseMoved(MouseEvent e) {
55              // TODO Auto-generated method stub
56              Vars.getStateLabel().setText("X:"+e.getX()+" , Y:"+e.getY());
57          }
58
59          public PaintPanel() {
60              setBackground(Color.white);
61              setCursor(Cursor.getPredefinedCursor(Cursor.CROSSHAIR_CURSOR));
62              addMouseListener(this);
63              addMouseMotionListener(this);
64          }
65
66          @Override
67          public void paintComponent(Graphics g) {
68              super.paintComponent(g);
69              try {
70                  g.drawImage(ImageIO.read(new File("D:\\Coding projects\\eclipse-workspace\\Painting\\
                        cache\\autosave.jpg")), 0, 0, Color.WHITE, this);
71              } catch (IOException e) {
72                  // TODO Auto-generated catch block
73                  ;
74              }
75              setBackground(Vars.getBackColor());
76              Shape[] shapes = Vars.getItems();
77              if(shapes == null) {
78                  return;
79              }
80              for (Shape shape : shapes) {
81                  draw((Graphics2D)g, shape);
82              }
83              if(Vars.nums>10) {
84                  clear();
85              }
86          }
87          public void clear() {
88              Vars.nums = 0;
89              autoSave();
90              Vars.items = new Shape[100];
91          }
92
93          public void draw(Graphics2D g, Shape shape) {
94              if(shape ==null) {
95                  return;
96              }
97              switch (shape.getString()) {
98              case "circle":
99                  g.setColor(shape.getColor());
100                 if(shape.bold) {
101                     g.setStroke(new BasicStroke(10));
102                 }else {
103                     g.setStroke(new BasicStroke(3));
104                 }
```

```
105                 g.drawOval(Math.min(shape.getX1(), shape.getX2()), Math.min(shape.getY1(), shape.getY2()
                          ), Math.abs(shape.getX1()-shape.getX2()), Math.abs(shape.getX1()-shape.getX2()));
106                 if (shape.filled) {
107                     g.setColor(shape.getBackColor());
108                     g.fillOval(Math.min(shape.getX1(), shape.getX2()), Math.min(shape.getY1(), shape.
                              getY2()), Math.abs(shape.getX1()-shape.getX2()), Math.abs(shape.getX1()-shape.
                              getX2()));
109
110                 }else {
111
112                 }
113                 break;
114             case "square":
115                 g.setColor(shape.getColor());
116                 if(shape.bold) {
117                     g.setStroke(new BasicStroke(10));
118                 }else {
119                     g.setStroke(new BasicStroke(3));
120                 }
121                 g.drawRect(Math.min(shape.getX1(), shape.getX2()), Math.min(shape.getY1(), shape.getY2()
                          ), Math.abs(shape.getX1()-shape.getX2()), Math.abs(shape.getY1()-shape.getY2()));
122                 if (shape.filled) {
123                     g.setColor(shape.getBackColor());
124                     g.fillRect(Math.min(shape.getX1(), shape.getX2()), Math.min(shape.getY1(), shape.
                              getY2()), Math.abs(shape.getX1()-shape.getX2()), Math.abs(shape.getY1()-shape.
                              getY2()));
125
126                 }else {
127
128                 }
129                 break;
130             case "ellipse":
131                 g.setColor(shape.getColor());
132                 if(shape.bold) {
133                     g.setStroke(new BasicStroke(10));
134                 }else {
135                     g.setStroke(new BasicStroke(3));
136                 }
137                 g.drawOval(Math.min(shape.getX1(), shape.getX2()), Math.min(shape.getY1(), shape.getY2()
                          ), Math.abs(shape.getX1()-shape.getX2()), Math.abs(shape.getY1()-shape.getY2()));
138                 if (shape.filled) {
139                     g.setColor(shape.getBackColor());
140                     g.fillOval(Math.min(shape.getX1(), shape.getX2()), Math.min(shape.getY1(), shape.
                              getY2()), Math.abs(shape.getX1()-shape.getX2()), Math.abs(shape.getY1()-shape.
                              getY2()));
141
142                 }else {
143                 }
144                 break;
145             case "text":
146                 g.setColor(shape.getColor());
147                 if(shape.bold) {
148                     Font font = shape.getFont().deriveFont(Font.BOLD);
149                     g.setFont(font);
150                 }else {
151                     g.setFont(shape.getFont());
152                 }
153                 g.drawString(shape.getTextString(),Math.min(shape.getX1(), shape.getX2()), Math.min(
                          shape.getY1(), shape.getY2()));
154                 break;
155             case "line":
156                 g.setColor(shape.getColor());
157                 if(shape.bold) {
```

```
158                     g.setStroke(new BasicStroke(10));
159                 }else {
160                     g.setStroke(new BasicStroke(3));
161                 }
162                 g.drawLine(shape.getX1(), shape.getY1(), shape.getX2(), shape.getY2());
163                 break;
164             default:
165                 break;
166             }
167         }
168 
169         @Override
170         public void mouseEntered(MouseEvent e) {
171             // TODO Auto-generated method stub
172 
173         }
174 
175         public void save() {
176             BufferedImage image  = new BufferedImage(getWidth(), getHeight(), BufferedImage.TYPE_INT_RGB
                    );
177             Graphics2D g = (Graphics2D)image.getGraphics();
178             paint(g);
179             JFileChooser chooser=new JFileChooser();//文件保存对话框
180             chooser.setCurrentDirectory(new File("."));
181             if(chooser.showSaveDialog(null)==JFileChooser.APPROVE_OPTION){
182                 File oFile=chooser.getSelectedFile();
183                 try{
184                      File file = new File(oFile.getName()+".jpg");
185                     ImageIO.write(image, "jpg", file);//保存图像文件
186                 }catch(IOException ex){
187                     ex.printStackTrace();
188                 }
189             }
190         }
191         public void autoSave() {
192             BufferedImage image  = new BufferedImage(getWidth(), getHeight(), BufferedImage.TYPE_INT_RGB
                    );
193             Graphics2D g = (Graphics2D)image.getGraphics();
194             paint(g);
195             try {
196                 File cacheFile = new File("D:\\Coding projects\\eclipse-workspace\\Painting\\cache\\
                        autosave.jpg");
197                 ImageIO.write(image, "jpg", cacheFile);
198             } catch (Exception e) {
199                 // TODO: handle exception
200             }
201         }
202     }
```

Listing 3: ToolBar.java

```
1   package sunboyan;
2 
3   import ...
4   public class ToolBar extends JToolBar{
5       private JPanel panel1 = null;
6       private JPanel panel2 = null;
7       private JPanel panel3 = null;
8       private JPanel panel4 = null;
9       private JPanel panel5 = null;
10      private JPanel panel6 = null;
11      public ToolBar() {
```

```
            panel1 = new JPanel();
            panel2 = new JPanel();
            panel3 = new JPanel();
            panel4 = new JPanel();
            panel5 = new JPanel();
            panel6 = new JPanel();
            panel2.add(new JLabel("背景色"));
            panel2.add(Vars.getBackColorBox());
            panel1.add(new JLabel("前景色"));
            panel1.add(Vars.getShapeColorBox());
            panel3.add(new JLabel("文字"));
            panel3.add(Vars.getTextField());
            panel4.add(new JLabel("字体"));
            panel4.add(Vars.getFontPanel());
            panel5.add(Vars.getUndoButton());
            panel6.add(Vars.getSaveButton());
            setLayout(new GridLayout(2,3));
            add(panel1,getLayout());
            add(panel2,getLayout());
            add(panel3,getLayout());
            add(panel4,getLayout());
            add(panel5,getLayout());
            add(panel6,getLayout());
        }

    }
```

Listing 4: shape.java

```
    package sunboyan;

    import java.awt.Color;
    import java.awt.Font;

    public class Shape {
        private int x1 = 0;
        private int x2 = 0;
        private int y1 = 0;
        private int y2 = 0;
        private String string = null;
        private String textString =null;
        private Color color = null;
        private Color backColor = null;
        private Font font = new Font("Times New Roman",Font.PLAIN,20);
        public boolean filled =false;
        public boolean bold = false;
        public int getX1() {
            return x1;
        }
        public void setX1(int x1) {
            this.x1 = x1;
        }
        public int getX2() {
            return x2;
        }
        public void setX2(int x2) {
            this.x2 = x2;
        }
        public int getY1() {
            return y1;
        }
        public void setY1(int y1) {
```

```
34              this.y1 = y1;
35          }
36          public int getY2() {
37              return y2;
38          }
39          public void setY2(int y2) {
40              this.y2 = y2;
41          }
42          public String getString() {
43              return string;
44          }
45          public void setString() {
46              this.string = Vars.getString();
47          }
48          public Color getColor() {
49              return color;
50          }
51          public void setColor() {
52              color = Vars.getColor();
53          }
54          public void setFont(Font font) {
55              this.font = font;
56          }
57          public Font getFont() {
58              return font;
59          }
60          public Color getBackColor() {
61              return backColor;
62          }
63          public void setBackColor(Color c) {
64              backColor =c;
65          }
66          public Shape() {
67
68          }
69          public Shape(Shape shape) {
70              string = shape.string;
71              color = shape.color;
72              backColor = shape.backColor;
73              font = shape.font;
74              filled =shape.filled;
75              bold = shape.bold;
76              textString = shape.textString;
77              x1 = shape.x1;
78              x2 = shape.x2;
79              y1 = shape.y1;
80              y2 = shape.y2;
81          }
82          public String getTextString() {
83              return textString;
84          }
85          public void setTextString(String string) {
86              textString = string;
87          }
88      }
```

Listing 5: shapeBar.java

```
1   package sunboyan;
2
3   import java.awt.FlowLayout;
4   import java.awt.GridBagLayout;
```

```
    import java.awt.GridLayout;
    import javax.swing.JPanel;
    import javax.swing.JToggleButton;

    public class ShapeBar extends JPanel {
        private static JToggleButton circleButton = null;
        private static JToggleButton squareButton = null;
        private static JToggleButton textButton = null;
        private static JToggleButton ellipseButton = null;
        private static JToggleButton lineButton = null;
        private static JToggleButton boldButton = null;
        public ShapeBar() {
            circleButton = Vars.getCircleButton();
            squareButton = Vars.getSquareButton();
            textButton = Vars.getTextButton();
            ellipseButton = Vars.getEllipseButton();
            lineButton = Vars.getLineButton();
            boldButton = Vars.getBoldButton();
            setLayout(new GridLayout(6,1));
            add(circleButton);
            add(ellipseButton);
            add(squareButton);
            add(textButton);
            add(lineButton);
            add(boldButton);
        }
}
```

Listing 6: LowPanel.java

```
    package sunboyan;

    import java.awt.FlowLayout;
    import javax.swing.JCheckBox;
    import javax.swing.JLabel;
    import javax.swing.JPanel;
    import javax.swing.JToolBar;

    public class LowPanel extends JToolBar{
        private static JLabel stateLabel = null;
        private static JCheckBox checkBox =null;
        public LowPanel() {
            stateLabel = Vars.getStateLabel();
            checkBox = Vars.getCheckBox();
            setLayout(new FlowLayout(FlowLayout.CENTER));
            add(stateLabel);
            add(checkBox);
        }
    }
```

Listing 7: ColorRender.java

```
    package sunboyan;

    import ...
    public class ColorRender extends JLabel implements ListCellRenderer<Color>{

        private static final long serialVersionUID = 1L;
        public ColorRender() {
            setOpaque(true);
            setHorizontalAlignment(CENTER);
```

```
            setVerticalAlignment(CENTER);
        }
        @Override
        public Component getListCellRendererComponent(JList list, Color value, int index, boolean
            isSelected,
              boolean cellHasFocus) {
            // TODO Auto-generated method stub
            if(index == -1) {

            }else {
                setBackground(value);
            }
            return this;
        }

    }
```

Listing 8: MainWindow.java

```
    package sunboyan;

    import ...
    public class MainWindow extends JFrame{
        public static void main(String[]args) {
            EventQueue.invokeLater(new Runnable() {
                public void run() {
                    try {
                        MainWindow frame = new MainWindow();
                        frame.setVisible(true);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            });
        }
        public MainWindow() {
            JPanel panel = new JPanel();
            setContentPane(panel);
            setDefaultCloseOperation(EXIT_ON_CLOSE);
            setBounds(0, 0, 1080, 720);
            getContentPane().setLayout(new BorderLayout());
            getContentPane().add(Vars.getToolPanel(), BorderLayout.NORTH);
            getContentPane().add(Vars.getShapePanel(),BorderLayout.WEST);
            getContentPane().add(Vars.getPaintPanel(),BorderLayout.CENTER);
            getContentPane().add(Vars.getLowPanel(),BorderLayout.SOUTH);
        }
        public void update(Graphics g) {
            // TODO Auto-generated method stub
            super.update(g);
            Vars.getPaintPanel().paintComponent(g);
        }
    }
```