

# AnimatedImage.java

```
1 import javax.swing.JApplet;
2 import javax.swing.JButton;
3 import javax.swing.JPanel;
4 import javax.swing.JComboBox;
5 import javax.swing.JLabel;
6 import javax.swing.ImageIcon;
7 import javax.swing.SwingUtilities;
8 import java.awt.*;
9 import java.awt.event.ActionEvent;
10 import java.awt.event.ActionListener;
11 //import java.awt.geom.Point2D;
12
13
14 public class AnimatedImage extends JApplet implements Runnable
15 {
16
17     private JPanel canvas, buttonPanel;
18     private JButton stopButton, startButton;
19     private JComboBox<String> modePicker;
20     private JLabel modelLabel;
21
22     private int width,
23             height,
24             mesh,
25             steps,
26             deltaT,
27             deltaW,
28             deltaH;
29     private boolean runThread = true,
30             doAnimate = false,
31             toggle = true,
32             animationThreadRunning = false;
33     private Image blueSphere;
34     private String mode;
35
36     @Override
37     public void init()
38     {
39         width = Integer.parseInt(getParameter("width"));
40         height = Integer.parseInt(getParameter("height")) - 40;
41         mesh = Integer.parseInt(getParameter("mesh"));
42         steps = Integer.parseInt(getParameter("steps"));
43         deltaT = Integer.parseInt(getParameter("deltaT"));
44         deltaW = width / mesh;
45         deltaH = height / mesh;
```

```

46 mode = getParameter("mode");
47 blueSphere = new ImageIcon(getClass().getResource("sphere.jpg")).getImage();
48 blueSphere = blueSphere.getScaledInstance(width, height, Image.SCALE_DEFAULT);
49
50 canvas = new JPanel()
51 {
52     @Override
53     public void paintComponent(Graphics g)
54     {
55         super.paintComponent(g);
56         setPreferredSize(new Dimension(width, height));
57         Graphics2D g2D = (Graphics2D)g;
58         /*
59         g2D.setPaint(new GradientPaint(
60             new Point2D.Double(width / 8, height / 8), Color.GREEN,
61             new Point2D.Double(width / 4, height / 4), Color.CYAN,
62             true));
63         g2D.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, .5f));
64         g2D.fillRect(0, 0, width, height);
65         */
66         g2D.drawImage(blueSphere, 0, 0, width, height, Color.BLACK, this);
67         g2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
68             RenderingHints.VALUE_ANTIALIAS_ON);
69         g2D.setStroke(new BasicStroke(1));
70         for (int i = 0; i < mesh; i++)
71         {
72
73             if (mode.equals("curtains"))
74             {
75                 g2D.setColor(Color.BLUE);
76                 g2D.drawLine(0, i * deltaH, (i + 1) * deltaW, height);
77                 g2D.setColor(Color.RED);
78                 g2D.drawLine(width, height - i * deltaH, width - (i + 1) * deltaW, 0);
79                 g2D.setColor(Color.MAGENTA);
80                 g2D.drawLine(0, height - (i * deltaH), (i + 1) * deltaW, 0);
81                 g2D.setColor(Color.MAGENTA);
82                 g2D.drawLine(width, i * deltaH, width - ((i + 1) * deltaW), height);
83             }
84             else
85             if (mode.equals("cradle") || toggle)
86             {
87                 g2D.setColor(Color.BLUE);
88                 g2D.drawLine(0, i * deltaH, (i + 1) * deltaW, height);
89                 g2D.setColor(Color.RED);
90                 g2D.drawLine(width, height - i * deltaH, width - (i + 1) * deltaW, 0);

```

```

91         g2D.setColor(Color.MAGENTA);
92         g2D.drawLine(width - i * deltaW, 0, 0, (i + 1) * deltaH);
93         g2D.setColor(Color.MAGENTA);
94         g2D.drawLine(i * deltaW, height, width, height - i * deltaH);
95     }
96     else
97     {
98         g2D.setColor(Color.BLUE);
99         g2D.drawLine(width - i * deltaW, height, 0, height - (i + 1) * deltaH);
100        g2D.setColor(Color.RED);
101        g2D.drawLine(i * deltaW, 0, width, (i + 1) * deltaH);
102        g2D.setColor(Color.MAGENTA);
103        g2D.drawLine(0, height - (i * deltaH), (i + 1) * deltaW, 0);
104        g2D.setColor(Color.MAGENTA);
105        g2D.drawLine(width, i * deltaH, width - ((i + 1) * deltaW), height);
106    }
107 }
108 }
109 };
110 add(canvas, BorderLayout.CENTER);
111 buttonPanel = new JPanel();
112 buttonPanel.setPreferredSize(new Dimension(width, 40));
113 buttonPanel.setBackground(Color.WHITE);
114 modeLabel = new JLabel("Mode:");
115 buttonPanel.add(modeLabel);
116 modePicker = new JComboBox<String>(new String[] {"curtains", "cradle", "wheel"});
117 modePicker.setSelectedItem(mode);
118 modePicker.addActionListener(
119     new ActionListener()
120     {
121         public void actionPerformed(ActionEvent e)
122         {
123             destroy();
124             stopButton.setEnabled(true);
125             startButton.setEnabled(false);
126             new Thread()
127             {
128                 public void run()
129                 {
130                     while (animationThreadRunning)
131                     {
132                         try {
133                             Thread.sleep(deltaT);
134                         } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }
135                     }

```

```

136         runThread = true;
137         doAnimate = true;
138         deltaW = width / mesh;
139         deltaH = height / mesh;
140         toggle = true;
141         mode = (String)modePicker.getSelectedItem();
142         repaint();
143         new Thread(AnimatedImage.this).start();
144     }
145     }.start();
146 }
147 }
148 );
149 buttonPanel.add(modePicker);
150 startButton = new JButton("Start");
151 startButton.addActionListener(
152     new ActionListener()
153     {
154         public void actionPerformed(ActionEvent e)
155         {
156             start();
157         }
158     }
159 );
160 buttonPanel.add(startButton);
161 stopButton = new JButton("Stop");
162 stopButton.addActionListener(
163     new ActionListener()
164     {
165         public void actionPerformed(ActionEvent e)
166         {
167             stop();
168         }
169     }
170 );
171 buttonPanel.add(stopButton);
172 add(buttonPanel, BorderLayout.SOUTH);
173 new Thread(this).start();
174 setVisible(true);
175 }
176
177 @Override
178 public void run()
179 {
180     synchronized(this)

```

```

181 {
182     animationThreadRunning = true;
183
184     while(runThread)
185     {
186
187         while(runThread && !doAnimate)
188             try {
189                 wait();
190             } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }
191
192         while(runThread && doAnimate)
193         {
194
195             try {
196                 wait(3 * deltaT);
197             } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }
198
199             for (int i = steps - 1; runThread && i >= 0; i--)
200             {
201                 if (doAnimate)
202                 {
203                     deltaW = (width * i) / (mesh * steps);
204                     deltaH = (height * i) / (mesh * steps);
205                     repaint();
206                     try {
207                         wait(deltaT);
208                     } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }
209                 }
210                 else
211                 {
212                     try {
213                         wait();
214                     } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }
215                     i--;
216                 }
217             }
218             toggle = !toggle;
219
220             try {
221                 wait(3 * deltaT);
222             } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }
223
224             for (int i = 1; runThread && i <= steps; i++)
225             {

```

```

226         if (doAnimate)
227         {
228             deltaW = (width * i) / (mesh * steps);
229             deltaH = (height * i) / (mesh * steps);
230             repaint();
231             try {
232                 wait(deltaT);
233             } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }
234         }
235         else
236         {
237             try {
238                 wait();
239             } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }
240             i--;
241         }
242     }
243     toggle = !toggle;
244 }
245 }
246 animationThreadRunning = false;
247 }
248 }
249
250 @Override
251 public void start()
252 {
253     synchronized(this)
254     {
255         doAnimate = true;
256         notifyAll();
257     }
258     SwingUtilities.invokeLater(
259         new Runnable()
260         {
261             public void run()
262             {
263                 stopButton.setEnabled(true);
264                 startButton.setEnabled(false);
265             }
266         }
267     );
268 }
269
270 @Override

```

```
271 public void stop()
272 {
273     synchronized(this)
274     {
275         doAnimate = false;
276         notifyAll();
277     }
278     SwingUtilities.invokeLater(
279         new Runnable()
280         {
281             public void run()
282             {
283                 stopButton.setEnabled(false);
284                 startButton.setEnabled(true);
285             }
286         }
287     );
288 }
289
290 @Override
291 public void destroy()
292 {
293     synchronized(this)
294     {
295         runThread = false;
296         doAnimate = true;
297         notifyAll();
298     }
299 }
300
301 }
302
```