

```
1 import java.applet.*;
2 import java.awt.*;
3 import java.awt.image.FilteredImageSource;
4 import java.net.MalformedURLException;
5 import java.net.URL;
6
7 public class hover extends Applet
8 {
9     Color        BackColor;
10    Color        ForeColor;
11    Color        HoverColor;
12    String       Text;
13    Font         UsedFont;
14    FontMetrics  UsedMetrics;
15    Image        SourceImage;
16    Image        HoverImage;
17    Image        PressedImage;
18    Image        AnimImage;
19    String       URLText;
20    String       TargetFrame;
21    int          TextHAlign;
22    int          TextVAlign;
23    int          TextWidth;
24    URL          DocURL;
25    int          CurrentImage = 0;
26
27
28    //*****//
29    // functions: //
30    //*****//
31
32    //*****//
33    // gets the specified parameter //
34    // and decodes it to a Color //
35    //*****//
36    Color DecodeColor(String param)
37    {
38        String pString = getParameter(param);
39        Color Result = new Color(Integer.parseInt(pString.substring(0,2), 16),
40                                Integer.parseInt(pString.substring(2,4), 16),
41                                Integer.parseInt(pString.substring(4,6), 16));
42        return Result;
43    }
44
45    //*****//
46    // checks, if string contains substring //
47    //*****//
48    boolean SubStringExists(String MainString, String SubString)
49    {
```

```
50         int index = MainString.indexOf(SubString);
51         boolean Result;
52         if ((index >= 0) & (index < MainString.length())) Result = true;
53         else
54             Result = false;
55         return Result;
56     }
57
58     //*****//
59     // get the specified parameter //
60     // and decodes it to a Font //
61     //*****//
62     Font DecodeFont(String param)
63     {
64         int    FontStyle;
65         int    FontSize;
66         String FontName;
67         Font   Result;
68
69         String pString = getParameter(param);
70         if (pString == null) Result = new Font("Arial", Font.PLAIN, 12);
71         else
72             {
73                 int ende = pString.indexOf(",");
74                 FontSize = Integer.parseInt(pString.substring(0, ende));
75                 int ende1 = pString.indexOf(",", ende+1);
76                 FontName = pString.substring(ende+2, ende1);
77                 pString = pString.toUpperCase();
78                 FontStyle = Font.PLAIN;
79                 if (SubStringExists(pString, "ITALIC"))
80                     FontStyle += Font.ITALIC;
81                 if (SubStringExists(pString, "BOLD"))
82                     FontStyle += Font.BOLD;
83                 Result = new Font(FontName, FontStyle, FontSize);
84             }
85         return Result;
86     }
87
88     //*****//
89     // gets specified parameter //
90     // and converts it to integer //
91     //*****//
92     int DecodeInteger(String param)
93     {
94         String pString = getParameter(param);
95         return Integer.parseInt(pString);
96     }
97
98     //*****//
```

```
99     // checks, if param exists //
100    //*****//
101    boolean ParamExists(String param)
102    {
103        String pString = getParameter(param);
104        boolean Result;
105        if (pString == null) Result = false;
106        else
107            Result = true;
108        return Result;
109    }
110
111    //*****//
112    // lighthens or darkens a color //
113    // by the given parameter //
114    //*****//
115    Color ModifyColor(Color color, int amount)
116    {
117        int r = color.getRed();
118        int g = color.getGreen();
119        int b = color.getBlue();
120        r += amount;
121        if (r > 255) r = 255;
122        else
123            if (r < 0) r = 0;
124        g += amount;
125        if (g > 255) g = 255;
126        else
127            if (g < 0) g = 0;
128        b += amount;
129        if (b > 255) b = 255;
130        else
131            if (b < 0) b = 0;
132        Color Result = new Color(r, g, b);
133        return Result;
134    }
135
136    public boolean mouseEnter(Event event, int i, int j)
137    {
138        CurrentImage = 2;
139        repaint();
140        return true;
141    }
142
143    public boolean mouseExit(Event event, int i, int j)
144    {
145        CurrentImage = 1;
146        repaint();
147        return true;

```

```
148     }
149
150     public void paint(Graphics g)
151     {
152         switch (CurrentImage)
153         {
154             default:
155                 break;
156             case 1:
157                 g.drawImage(SourceImage, 0, 0, this);
158                 break;
159             case 2:
160                 g.drawImage(HoverImage, 0, 0, this);
161                 break;
162             case 3:
163                 g.drawImage(PressedImage, 0, 0, this);
164                 break;
165         }
166     }
167
168     public boolean mouseUp(Event event, int i, int j)
169     {
170         if ((TargetFrame != null) || (TargetFrame != ""))
171         {
172             getAppletContext().showDocument(DocURL, TargetFrame);
173         }
174         else
175         {
176             getAppletContext().showDocument(DocURL);
177         }
178         return true;
179     }
180
181     public boolean mouseDown(Event event, int i, int j)
182     {
183         CurrentImage = 3;
184         repaint();
185         return true;
186     }
187
188     public void destroy()
189     {
190     }
191
192     public void update(Graphics g)
193     {
194         paint(g);
195     }
196
```

```
197     public void start()
198     {
199     }
200
201     public void stop()
202     {
203     }
204
205     public void init()
206     {
207         // get various colors:
208         if (ParamExists("background"))
209             BackColor = DecodeColor("background");
210         else
211             BackColor = Color.white;
212         if (ParamExists("foreground"))
213             ForeColor = DecodeColor("foreground");
214         else
215             ForeColor = Color.black;
216         if (ParamExists("hovercolor"))
217             HoverColor = DecodeColor("hovercolor");
218         else
219             HoverColor = Color.black;
220         // get the button text:
221         if (ParamExists("text"))
222             Text = getParameter("text");
223         else
224             Text = "no text given";
225         // get the url:
226         if (ParamExists("url"))
227             URLText = getParameter("url");
228         else
229             URLText = "";
230         // get the target frame:
231         if (ParamExists("target"))
232             TargetFrame = getParameter("target");
233         else
234             TargetFrame = "";
235         // get font and its style:
236         UsedFont = DecodeFont("font");
237         // get the font align:
238         if (ParamExists("align"))
239         {
240             /*****
241             /** 1 .. center **/
242             /** 2 .. left   **/
243             /** 3 .. right  **/
244             *****/
245             String align = getParameter("align");
```

```
246         align = align.toUpperCase();
247         if (SubStringExists(align, "CENTER"))
248             TextHAlign = 1;
249             else
250                 if (SubStringExists(align, "LEFT"))
251                     TextHAlign = 2;
252                     else
253                         if (SubStringExists(align, "RIGHT"))
254                             TextHAlign = 3;
255                             else
256                                 TextHAlign = 3;
257
258             /*****
259             /** 1 .. center **/
260             /** 2 .. top    **/
261             /** 3 .. bottom **/
262             *****/
262         if (SubStringExists(align, "CENTER"))
263             TextVAlign = 1;
264             else
265                 if (SubStringExists(align, "TOP"))
266                     TextVAlign = 2;
267                     else
268                         if (SubStringExists(align, "BOTTOM"))
269                             TextVAlign = 3;
270                             else
271                                 TextVAlign = 3;
272     }
273     else
274     {
275         TextHAlign = 1;
276         TextVAlign = 1;
277     }
278     // set background color and font
279     setBackground(BackColor);
280     setFont(UsedFont);
281     //get the font metrics
282     UsedMetrics = getFontMetrics(UsedFont);
283     TextWidth = UsedMetrics.stringWidth(Text);
284     // get the document URL:
285     if ((URLText != "") || (URLText != null))
286         try
287         {
288             DocURL = new URL(getDocumentBase(), URLText);
289         }
290     catch(MalformedURLException e) {}
291     // create 3 images and clear them:
292     //     - main image, when mouse not in applet
293     //     - hover image, when mouse is in applet
294     //     - pressed image, when mouse button is pressed
```

```
295     int w = getSize().width;
296     int h = getSize().height;
297     int d_c = 20; //holds delta color
298     if (ParamExists("framecol"))
299         d_c = DecodeInteger("framecol");
300     {
301         SourceImage = createImage(w, h);
302         Graphics g = SourceImage.getGraphics();
303         g.setColor(BackColor);
304         g.fillRect(0, 0, w, h);
305         Color bevel = ModifyColor(BackColor, -d_c);
306         {
307             g.setColor(bevel);
308             g.drawLine(2, 2, getSize().width-3, 2);
309             g.drawLine(2, 2, 2, getSize().height-3);
310         }
311         bevel = ModifyColor(bevel, -d_c);
312         {
313             g.setColor(bevel);
314             g.drawLine(1, 1, getSize().width-2, 1);
315             g.drawLine(1, 1, 1, getSize().height-2);
316         }
317         bevel = ModifyColor(bevel, -d_c);
318         {
319             g.setColor(bevel);
320             g.drawLine(0, 0, getSize().width, 0);
321             g.drawLine(0, 0, 0, getSize().height);
322         }
323         bevel = ModifyColor(BackColor, d_c);
324         {
325             g.setColor(bevel);
326             g.drawLine(3, getSize().height-3, getSize().width-4, getSize().height-3);
327             g.drawLine(getSize().width-3, getSize().height-3, getSize().width-3, 3);
328         }
329         bevel = ModifyColor(bevel, d_c);
330         {
331             g.setColor(bevel);
332             g.drawLine(2, getSize().height-2, getSize().width-3, getSize().height-2);
333             g.drawLine(getSize().width-2, getSize().height-2, getSize().width-2, 2);
334         }
335         bevel = ModifyColor(bevel, d_c);
336         {
337             g.setColor(bevel);
338             g.drawLine(1, getSize().height-1, getSize().width-2, getSize().height-1);
339             g.drawLine(getSize().width-1, getSize().height-1, getSize().width-1, 1);
340         }
341         g.setColor(ForeColor);
342         int l = w/2 - UsedMetrics.stringWidth(Text)/2;
343         int t = h/2 + UsedFont.getSize()/2;
```

```
344         g.drawString(Text, l, t-1);
345     }
346     {
347         HoverImage = createImage(w, h);
348         Graphics g = HoverImage.getGraphics();
349         g.setColor(HoverColor);
350         g.fillRect(0, 0, w, h);
351         Color bevel = ModifyColor(HoverColor, -d_c);
352         {
353             g.setColor(bevel);
354             g.drawLine(2, 2, getSize().width-3, 2);
355             g.drawLine(2, 2, 2, getSize().height-3);
356         }
357         bevel = ModifyColor(bevel, -d_c);
358         {
359             g.setColor(bevel);
360             g.drawLine(1, 1, getSize().width-2, 1);
361             g.drawLine(1, 1, 1, getSize().height-2);
362         }
363         bevel = ModifyColor(bevel, -d_c);
364         {
365             g.setColor(bevel);
366             g.drawLine(0, 0, getSize().width, 0);
367             g.drawLine(0, 0, 0, getSize().height);
368         }
369         bevel = ModifyColor(HoverColor, d_c);
370         {
371             g.setColor(bevel);
372             g.drawLine(3, getSize().height-3, getSize().width-4, getSize().height-3);
373             g.drawLine(getSize().width-3, getSize().height-3, getSize().width-3, 3);
374         }
375         bevel = ModifyColor(bevel, d_c);
376         {
377             g.setColor(bevel);
378             g.drawLine(2, getSize().height-2, getSize().width-3, getSize().height-2);
379             g.drawLine(getSize().width-2, getSize().height-2, getSize().width-2, 2);
380         }
381         bevel = ModifyColor(bevel, d_c);
382         {
383             g.setColor(bevel);
384             g.drawLine(1, getSize().height-1, getSize().width-2, getSize().height-1);
385             g.drawLine(getSize().width-1, getSize().height-1, getSize().width-1, 1);
386         }
387         g.setColor(ForeColor);
388         int l = w/2 - UsedMetrics.stringWidth(Text)/2;
389         int t = h/2 + UsedFont.getSize()/2;
390         g.drawString(Text, l-4, t-3);
391     }
392 }
```

```
393         PressedImage = createImage(w, h);
394         Graphics g = PressedImage.getGraphics();
395         Color PressColor = ModifyColor(HoverColor, 40);
396         g.setColor(PressColor);
397         g.fillRect(0, 0, w, h);
398         Color bevel = ModifyColor(PressColor, -d_c);
399         {
400             g.setColor(bevel);
401             g.drawLine(2, 2, getSize().width-3, 2);
402             g.drawLine(2, 2, 2, getSize().height-3);
403         }
404         bevel = ModifyColor(bevel, -d_c);
405         {
406             g.setColor(bevel);
407             g.drawLine(1, 1, getSize().width-2, 1);
408             g.drawLine(1, 1, 1, getSize().height-2);
409         }
410         bevel = ModifyColor(bevel, -d_c);
411         {
412             g.setColor(bevel);
413             g.drawLine(0, 0, getSize().width, 0);
414             g.drawLine(0, 0, 0, getSize().height);
415         }
416         bevel = ModifyColor(PressColor, d_c);
417         {
418             g.setColor(bevel);
419             g.drawLine(3, getSize().height-3, getSize().width-4, getSize().height-3);
420             g.drawLine(getSize().width-3, getSize().height-3, getSize().width-3, 3);
421         }
422         bevel = ModifyColor(bevel, d_c);
423         {
424             g.setColor(bevel);
425             g.drawLine(2, getSize().height-2, getSize().width-3, getSize().height-2);
426             g.drawLine(getSize().width-2, getSize().height-2, getSize().width-2, 2);
427         }
428         bevel = ModifyColor(bevel, d_c);
429         {
430             g.setColor(bevel);
431             g.drawLine(1, getSize().height-1, getSize().width-2, getSize().height-1);
432             g.drawLine(getSize().width-1, getSize().height-1, getSize().width-1, 1);
433         }
434         int c_r = ForeColor.getRed();
435         int c_g = ForeColor.getGreen();
436         int c_b = ForeColor.getBlue();
437         Color FontColor = new Color(0xFF-c_r, 0xFF-c_g, 0xFF-c_b);
438         g.setColor(FontColor);
439         int l = w/2 - UsedMetrics.stringWidth(Text)/2;
440         int t = h/2 + UsedFont.getSize()/2;
441         g.drawString(Text, l-4, t-3);
```

```
442         }
443         CurrentImage = 1;
444     }
445 }
446
447 public String getAppletInfo()
448 {
449     return "Title: Hover-Applet \nAuthor: Klaus Burgstaller \nlast modified on 15.11.2001 \ncopyright (c) by Klaus
Burgstaller. \nwww.crosswinds.net/~burgstaller";
450 }
451
452
453 public String[][] getParameterInfo()
454 {
455     String[][] info = {
456         {"text", "string", "Contains text to be displayed. Has no default!! The component doesn't
work otherwise!"},
457         {"foreground", "hex color", "Foreground color in hex format: RRGGBB"},
458         {"background", "hex color", "Works the same way for the back ground color"},
459         {"hovercolor", "hex color", "Works the same way for the back ground color"},
460         {"target", "string", "Optional, may define target frame."},
461         {"url", "string", "No default!! Contains URL to be linked."},
462         {"font", "font size, font name, font style", "contains description about the font style: \
nfirst parameter is font size, second font name, \nadditional font stylings can be given: BOLD, ITALIC, PLAIN. \nYou may write
the font stylings upper od lower case. \nDefault = Arial, plain, size:12 ."}
463     };
464     return info;
465 }
466 }
467
```