



JavaOne™

java.sun.com/javaone

Nimbus: The New Face of Swing

Jasper Potts, Richard Bair
Swing Engineers
Sun Microsystems

TS-6096



See the new face of Swing and learn how to use and extend the Nimbus Look And Feel in your own applications

A large, light blue graphic consisting of a stylized arrow pointing to the right, followed by the word "GOAL" in a bold, sans-serif font. The arrow and text are semi-transparent and overlaid on a darker blue background.

Nimbus Overview

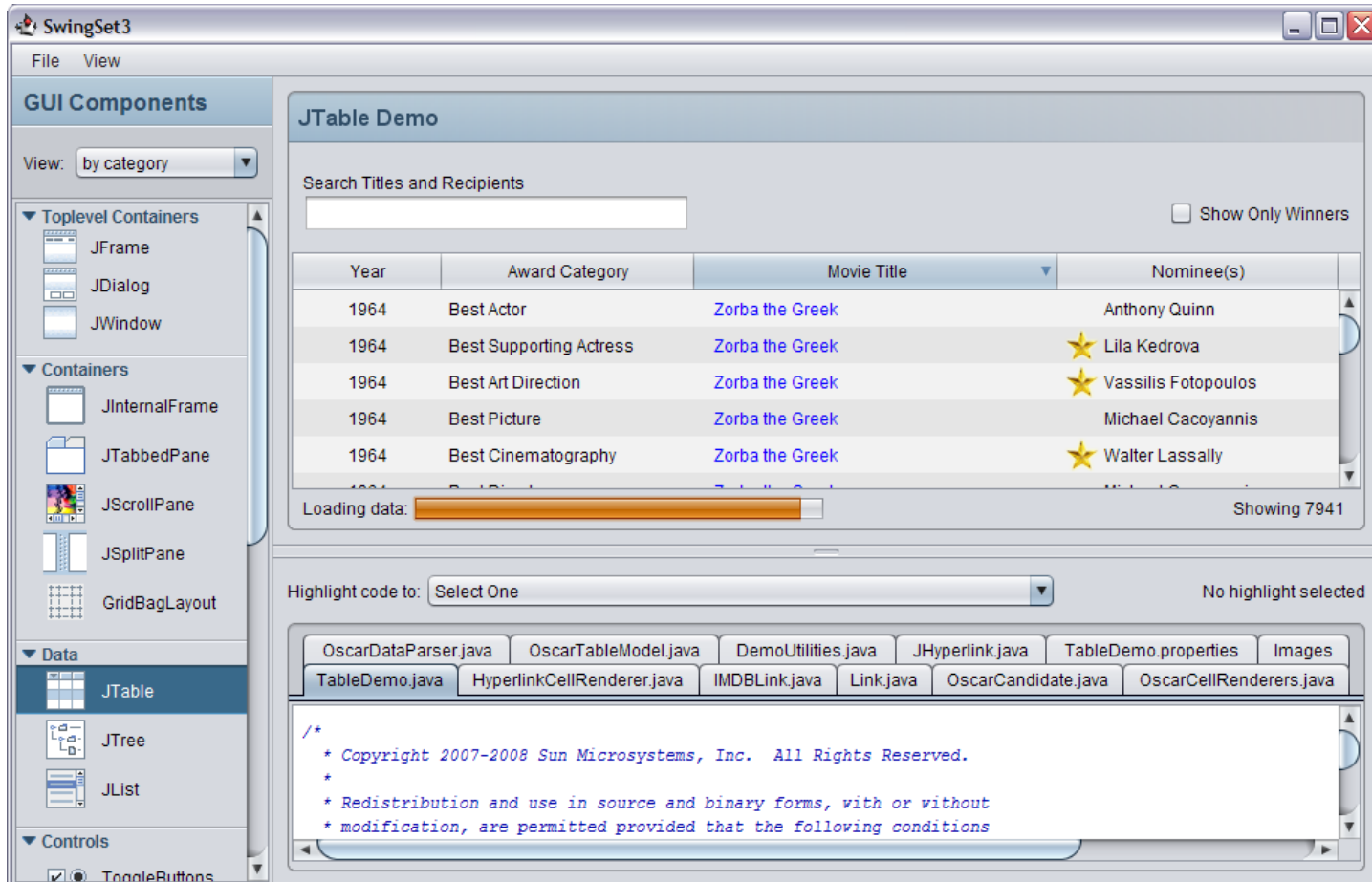
- Synth-based
- Vector-based
- Painters-based
- Customizable
- Java™ release 6u10

Agenda

- How it looks
- Use it
- How it works
- Q & A

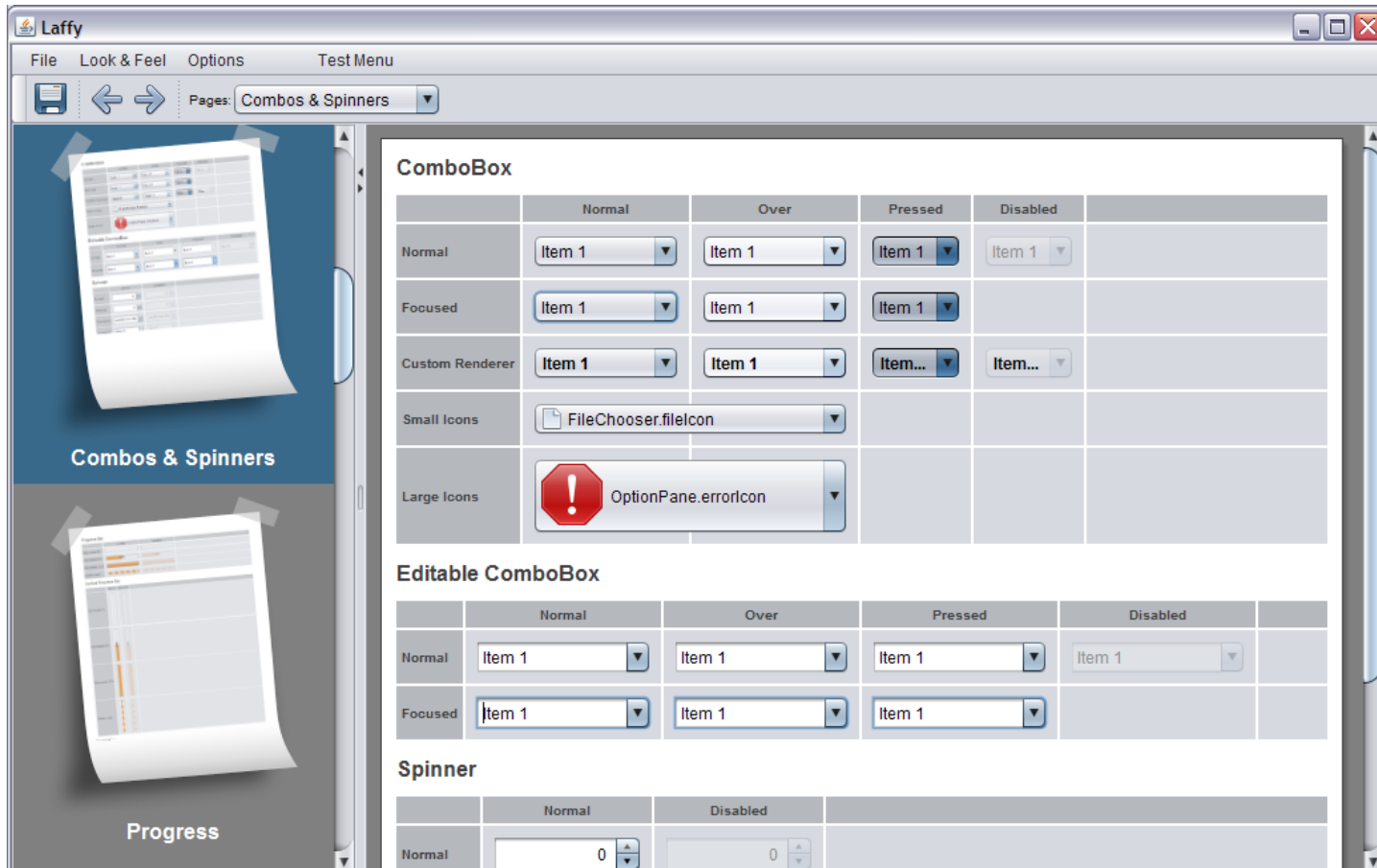
How it looks

SwingSet3



How it looks

Laffy



Laffy

Look and Feel Browser

DEMO

Run Laffy Now!

- Download and install Java 6 Update 10
 - <http://download.java.net/jdk6>
- Run Laffy
 - <http://download.java.net/javadesktop/laffy/Laffy.jnlp>
- Laffy is open source on <http://laffy.dev.java.net>

Agenda

- How it looks
- Use it
- How it works
- Q & A

<http://download.java.net/jdk6>

<http://download.java.net/javadesktop/laffy/Laffy.jnlp>

Using Nimbus

```
for (LookAndFeelInfo laf :
    UIManager.getInstalledLookAndFeels() {
    if ("Nimbus".equals(laf.getName())) {
        UIManager.setLookAndFeel(laf.getClassName());
    }
}
```

<http://download.java.net/jdk6>

<http://download.java.net/javadesktop/laffy/Laffy.jnlp>

Why not this?

```
UIManager.setLookAndFeel(  
    "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");
```

<http://download.java.net/jdk6>

<http://download.java.net/javadesktop/laffy/Laffy.jnlp>

Why not this?

```
UIManager.setLookAndFeel(  
    "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");
```

- In Java version 7...
 - Nimbus will move to javax.swing.plaf.nimbus
 - java.* packages cannot be modified in update releases
- Does not fall back to the default if Nimbus is not available

New Goodies

- Scalable Components
- Branding
 - Color Themes
 - Skinning
 - Customizing
- Designer Friendly

<http://download.java.net/jdk6>

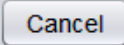
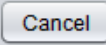


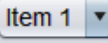
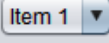
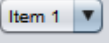
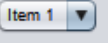










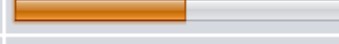







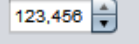





<http://download.java.net/javadesktop/laffy/Laffy.jnlp>

Scalable Components

- Support for small and large components
 - Small size variants of a component useful for palettes
- Supply a client property to the component
 - `JComponent.sizeVariant.small`
 - `JComponent.sizeVariant.large`
 - `JComponent.sizeVariant.medium`

Size Variants

Easy scaling

	Large	Regular	Small	Mini
Buttons				
Radio Buttons	<input type="radio"/> Cancel	<input type="radio"/> Cancel	<input type="radio"/> Cancel	<input type="radio"/> Cancel
Check Boxes	<input type="checkbox"/> Cancel	<input type="checkbox"/> Cancel	<input type="checkbox"/> Cancel	<input type="checkbox"/> Cancel
Combo Boxes				
Combo Boxes				
Text Fields	<input type="text" value="Example Text"/>	<input type="text" value="Example Text"/>	<input type="text" value="Example Text"/>	<input type="text" value="Example Text"/>
Scroll Bars				
Progress Bars				
Progress Bars				
Spinners				
Sliders				

Using Size Variants

```
JButton btn = new JButton("Click Me");
```

```
btn.putClientProperty("JComponent.sizeVariant", "large");
```

```
// or
```

```
btn.putClientProperty("JComponent.sizeVariant", "small");
```

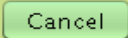
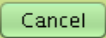


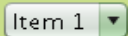
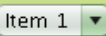













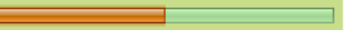

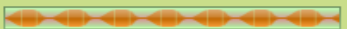
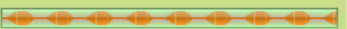





```
// or
```

```
btn.putClientProperty("JComponent.sizeVariant", "mini");
```

Color Themes

Easily change colors

Size Variants
























	Large	Regular	Small	Mini
Buttons				
Radio Buttons	<input type="radio"/> Cancel	<input type="radio"/> Cancel	<input type="radio"/> Cancel	<input type="radio"/> Cancel
Check Boxes	<input type="checkbox"/> Cancel	<input type="checkbox"/> Cancel	<input type="checkbox"/> Cancel	<input type="checkbox"/> Cancel
Combo Boxes				
Combo Boxes				
Text Fields	<input type="text" value="Example Text"/>	<input type="text" value="Example Text"/>	<input type="text" value="Example Text"/>	<input type="text" value="Example Text"/>
Scroll Bars				
Progress Bars				
Progress Bars				
Spinners	<input type="text" value="123,456"/>	<input type="text" value="123,456"/>	<input type="text" value="123,456"/>	<input type="text" value="123,456"/>
Sliders				

Color Themes

```
UIManager.put("nimbusBase", new Color(...));
UIManager.put("nimbusBlueGrey", new Color(...));
UIManager.put("control", new Color(...));

for (LookAndFeelInfo laf :
    UIManager.getInstalledLookAndFeels() {
    if ("Nimbus".equals(laf.getName())) {
        UIManager.setLookAndFeel(laf.getClassName());
    }
}
```

Nimbus Colors

Name	Color	Web Color	Red	Green	Blue
desktop		#FFFFFF	255	255	255
activeCaption		#B8CFE5	184	207	229
inactiveCaption		#8E8F91	142	143	145
text		#000000	0	0	0
textHighlight		#B8CFE5	184	207	229
textHighlightText		#333333	51	51	51
textInactiveText		#B8CFE5	184	207	229
control		#D6D9DF	214	217	223
controlText		#000000	0	0	0
controlHighlight		#FFFFFF	255	255	255
controlLHighlight		#000000	0	0	0
controlShadow		#B8CFE5	184	207	229
controlDkShadow		#7A8A99	122	138	153
nimbusBase		#33628C	51	98	140
nimbusBlueGrey		#A9B0BE	169	176	190
nimbusOrange		#BF6204	191	98	4
nimbusGreen		#B0B332	176	179	50
nimbusRed		#A92E22	169	46	34
nimbusBorder		#9297A1	146	151	161
nimbusSelection		#39698A	57	105	138
nimbusInfoBlue		#2F5CB4	47	92	180
nimbusAlertYellow		#FFDC23	255	220	35
nimbusFocus		#73A4D1	115	164	209
nimbusSelectedText		#FFFFFF	255	255	255
nimbusSelectionBackground		#818181	129	129	129
nimbusFocusedSelectionBackground		#39698A	57	105	138
nimbusDisabledText		#8E8F91	142	143	145
nimbusLightBackground		#FFFFFF	255	255	255

Color Themes

DEMO

Agenda

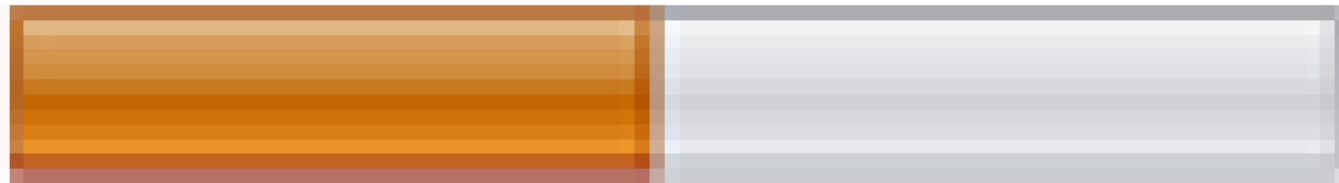
- How it looks
- Use it
- How it works
- Q & A

Vector Based

- All Nimbus components are purely vector based
 - Scalable: larger or smaller
 - Many aspects of painting can be customized
- Heuristic based image caching
 - Cache is invalidated if the sizes change dramatically
 - Or colors, fonts, etc change
- 9-Square technique resizes vector shapes
 - Simply move control points based on location in grid
- Could use images, but we didn't
 - Allows us to participate in solution for hi-DPI
 - Allows color theming for branding

Bitmap Scaling

Nimbus L&F



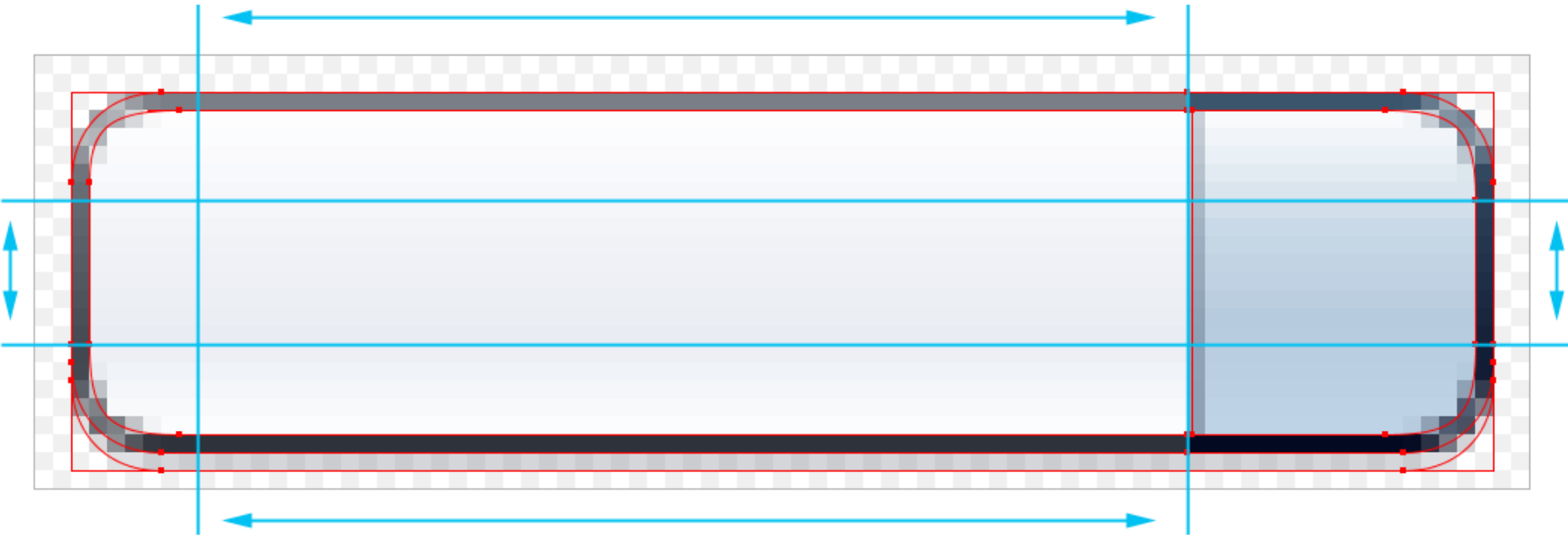
Vector Scaling

Nimbus L&F



9-Square

How vectors are resized dynamically



Painters

- Component rendering is delegated to Painters
 - Simple interface
 - Stateless
 - Extend `AbstractRegionPainter` to get scaling and caching

Painters

```
public interface Painter {  
    public void paint(  
        Graphics2D g,  
        JComponent c,  
        int width,  
        int height);  
}
```

Painters

- You can replace **any** painter for **any** region of **any** component for **any** state
- You can use **any** painter of **any** region of **any** component in **any** state

Everything is in UIDefaults

- All customizations are exposed via UIDefaults
- Global and per component instance UIDefaults
- Examples keys:
 - foreground
 - Button.foreground
 - Button[Disabled].foreground
 - Button[Default+Focused+MouseOver].backgroundPainter
 - CheckBoxMenuItem:MenuItemAccelerator.contentMargins
 - ComboBox:"ComboBox.arrowButton".Editable

Cascading Defaults

- Most specific default used
- Example:
 - foreground = Color.BLACK
 - Label.foreground = Color.BLUE
 - Label[Disabled].foreground = Color.GRAY
 - “SomeLabel”[Disabled].foreground = Color.WHITE

Customizations

- Visual states
- Colors and fonts
- Standard components
 - Replace painting code and change defaults
 - For all components of a single type (ie: JButton)
 - For a single component instance
 - For all components with a specific name
- Add support for 3rd Party components

Named Component support

- All components of a given name can be styled specifically
 - Would like to extend this support to include a css-like “class” client property
- “Foo”.foreground would specify the color to use as the foreground for any component named “Foo”
 - Values within quotes are interpreted literally and have no special meaning

Customize Defaults per Instance

- Create a UIDefaults Map
- Load the map with all overrides of global values
- Set map on target JComponent subclass
 - `component.putClientProperty("Nimbus.Overrides", map);`
- `Nimbus.Overrides = UIDefaults map`
- `Nimbus.Overrides.InheritDefaults = Boolean`
 - Indicates whether to ignore defaults in UIManager

Specific component hierarchy support

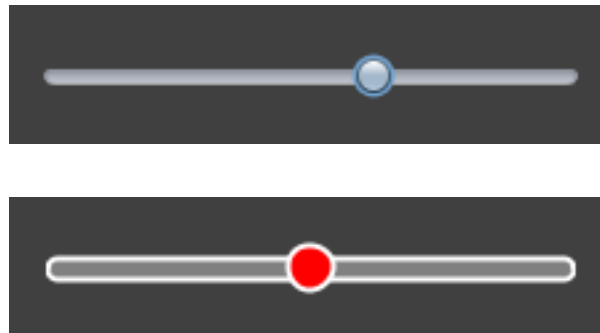
- Suppose: Style a JButton differently based on its parent
 - Foo:Bar:Baz
 - Baz.parent = Bar; Bar.parent = Foo
 - Only these Baz's are styled by this rule
 - All other Baz's use other styles

Installing Custom Painters

- `Button[Enabled].backgroundPainter`
 - Specifies the painter to use for JButtons when enabled
- `Button[MouseOver].borderPainter`
 - Specifies a border to use for the JButton when mouse over
- `Button.foregroundPainter`
 - Specifies a painter to use for rendering the foreground for all states (assuming a more specific state isn't registered)

Skinning

Easy as it looks



Skinning JSpinner

```
UIDefaults sliderDefaults = new UIDefaults();  
sliderDefaults.put("Slider.thumbWidth", 20);  
sliderDefaults.put("Slider.thumbHeight", 20);  
sliderDefaults.put(  
    "Slider:SliderThumb.backgroundPainter",  
    thumbBackgroundPainter);
```

Skinning JSpinner

```
thumbBackgroundPainter = new Painter() {
    public void paint(Graphics2D g, JComponent c,
                     int w, int h) {
        g.setRenderingHint(
            KEY_ANTIALIASING,
            VALUE_ANTIALIAS_ON);
        g.setStroke(new BasicStroke(2f));
        g.setColor(Color.RED);
        g.fillOval(1, 1, w-3, h-3);
        g.setColor(Color.WHITE);
        g.drawOval(1, 1, w-3, h-3);
    }
});
```

Skinning JSpinner

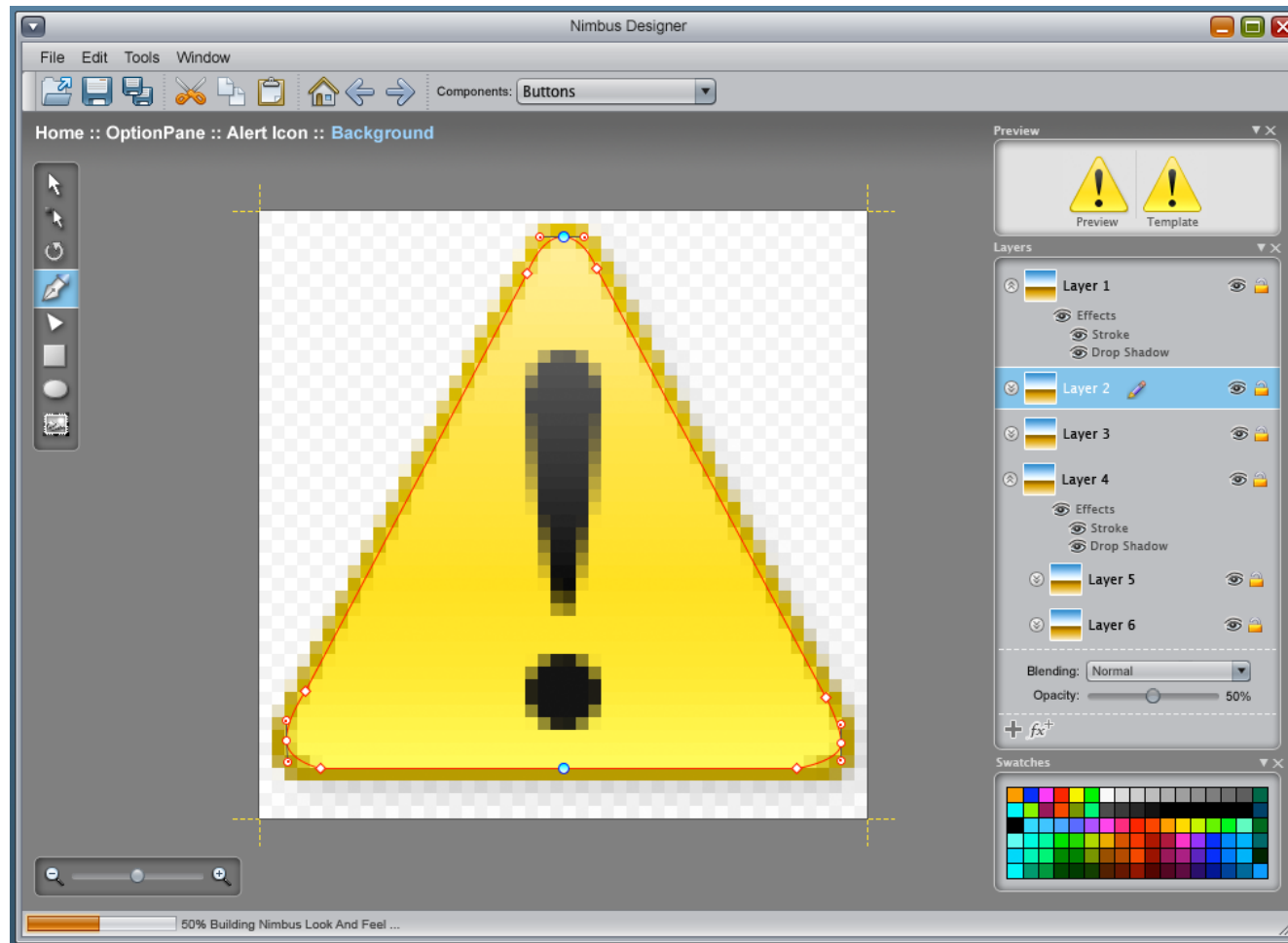
```
JSlider slider = new JSlider(0, 100, 50);  
slider.putClientProperty(  
    "Nimbus.Overrides", sliderDefaults);  
slider.putClientProperty(  
    "Nimbus.Overrides.InheritDefaults", false);
```

How it works

- Visual States
- Vector Based
- Customizations
- Tool Support

Nimbus Designer

For Designers



Nimbus Designer

- Used to create all the vector painters for Nimbus
- Pre-release form
- Open Source SwingLabs project

Nimbus Designer



DEMO

Summary

- Nimbus is a new cross platform Look and Feel
- It is available in Java 6 Update 10
- It is easily customizable, open for designers to play
- Nimbus is the default Look and Feel in JavaFX™ platform

For More Information

- <http://jdk6.dev.java.net>
- <http://laffy.dev.java.net>
- BOF 6101 – Nimbus Beyond the Basics (HE 135)

THANK YOU



Nimbus: The New Face of Swing
Richard Bair & Jasper Potts

TS-6096

