Filthy Rich Clients

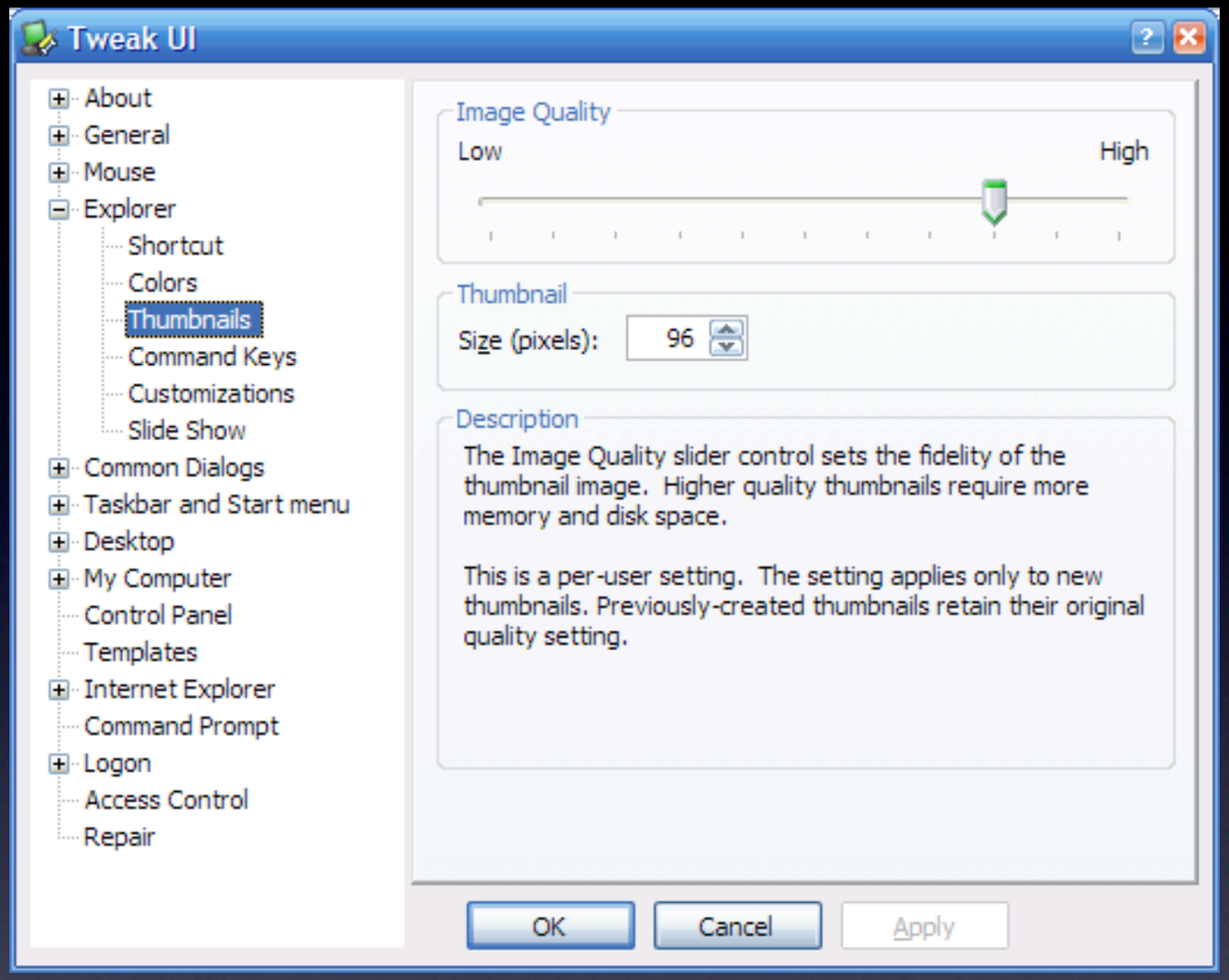
Chet Haase

by ~~Romain Guy~~

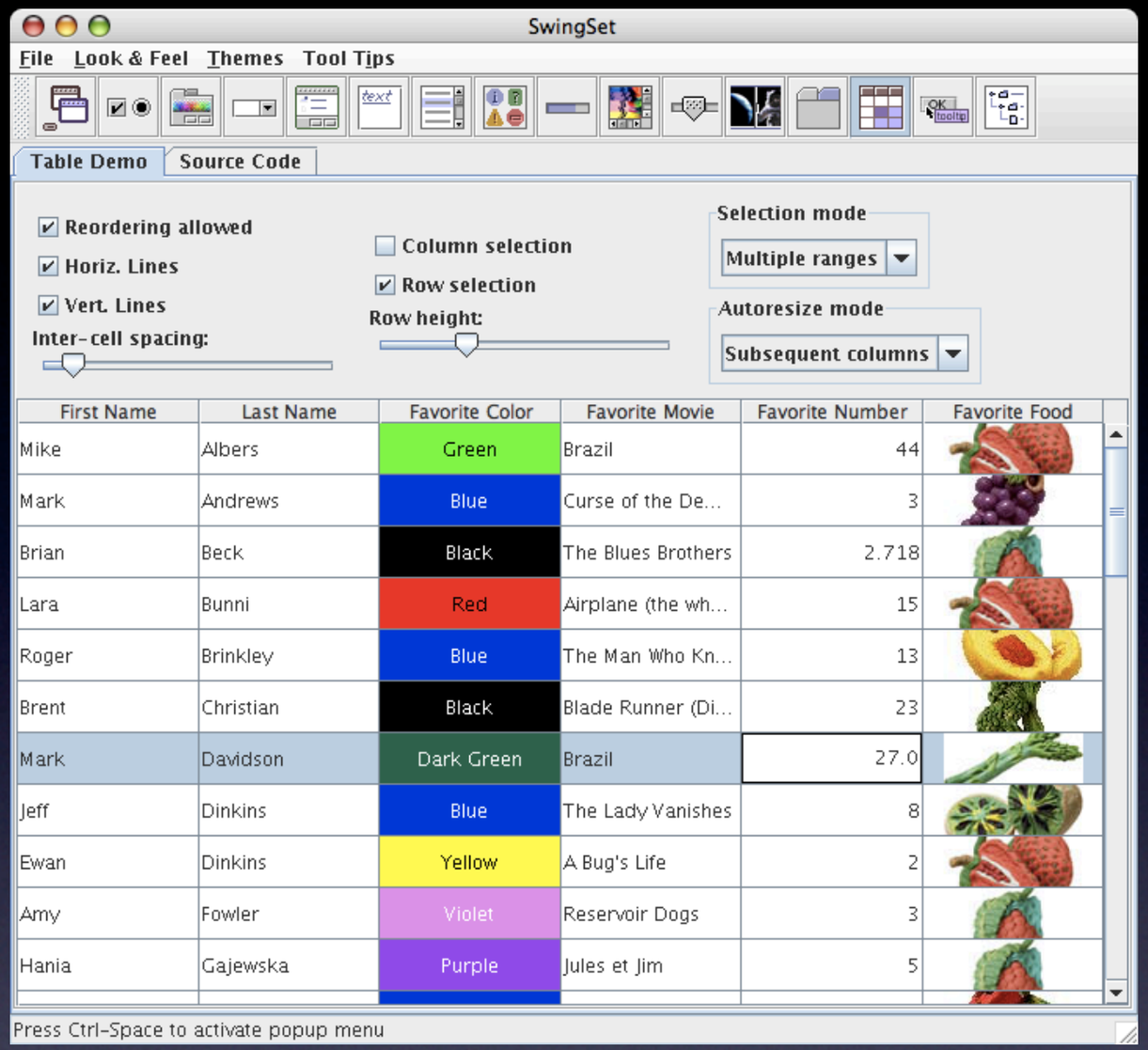

Java

Filthy Rich Clients

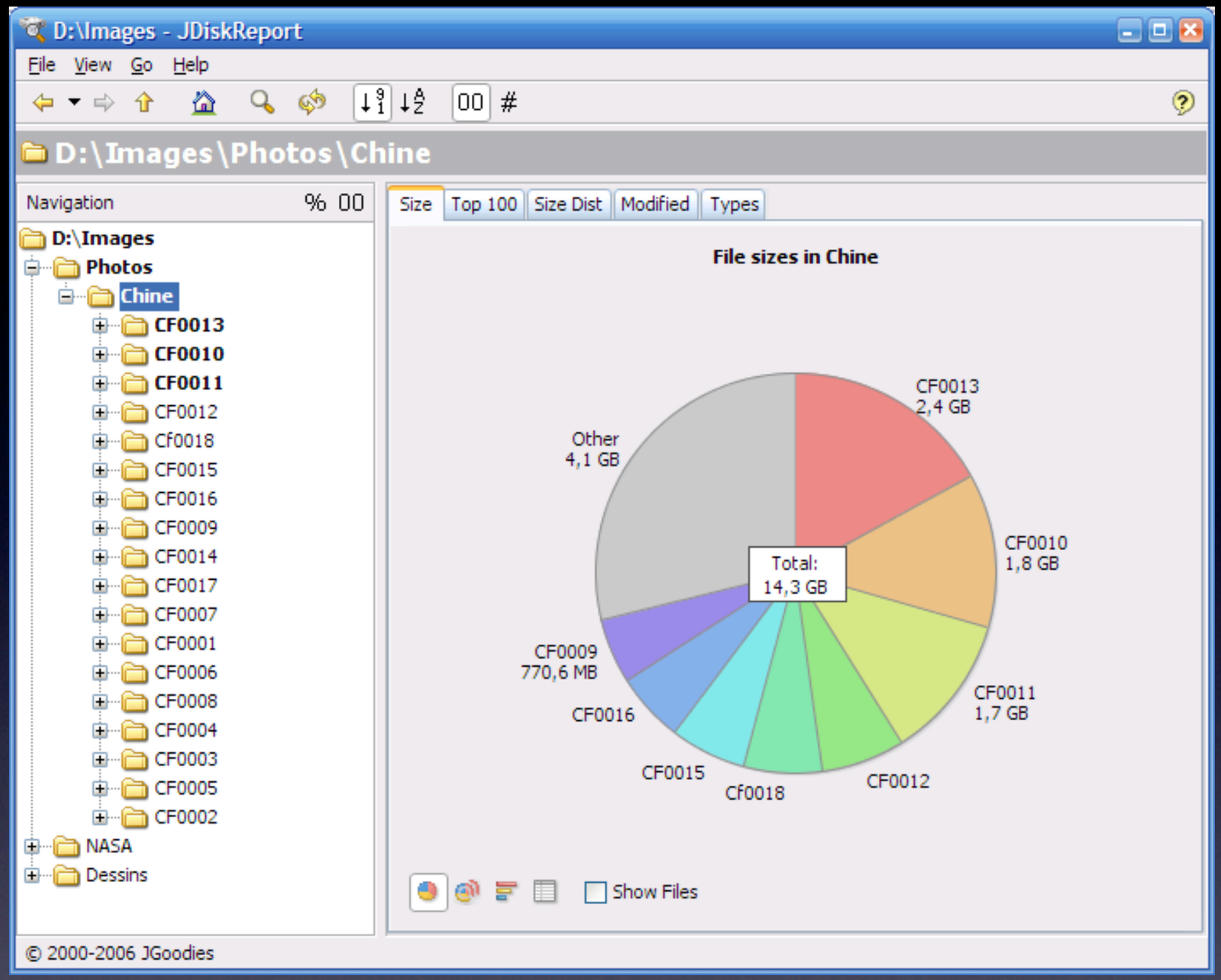
Applications that are so graphically rich that they ooze cool. They suck the user in from the outset and hang onto them with a death grip of excitement. They make the user tell their friends about the applications.
In short, they make the user actually enjoy their application experience.



What They Have


SwingSet
File Look & Feel Themes Tool Tips
Table Demo
Source Code
Reordering allowed
Horiz. Lines
Vert. Lines
Inter-cell spacing:
Column selection
Row selection
Row height:
Selection mode
Multiple ranges
Autoresize mode
Subsequent columns
First Name | Last Name | Favorite Color | Favorite Movie | Favorite Number | Favorite Food
Mike | Albers | Green | Brazil | 44
Mark | Andrews | Blue | Curse of the De... | 3
Brian | Beck | Black | The Blues Brothers | 2.718
Lara | Bunni | Red | Airplane (the wh... | 15
Roger | Brinkley | Blue | The Man Who Kn... | 13
Brent | Christian | Black | Blade Runner (Di... | 23
Mark | Davidson | Dark Green | Brazil | 27.0
Jeff | Dinkins | Blue | The Lady Vanishes | 8
Ewan | Dinkins | Yellow | A Bug's Life | 2
Amy | Fowler | Violet | Reservoir Dogs | 3
Hania | Gajewska | Purple | Jules et Jim | 5
Press Ctrl-Space to activate popup menu

What We Have



What You Want

What You Will Have

Demo

Agenda

Graphics

Effects

3D

Performance

Agenda

Graphics

Effects

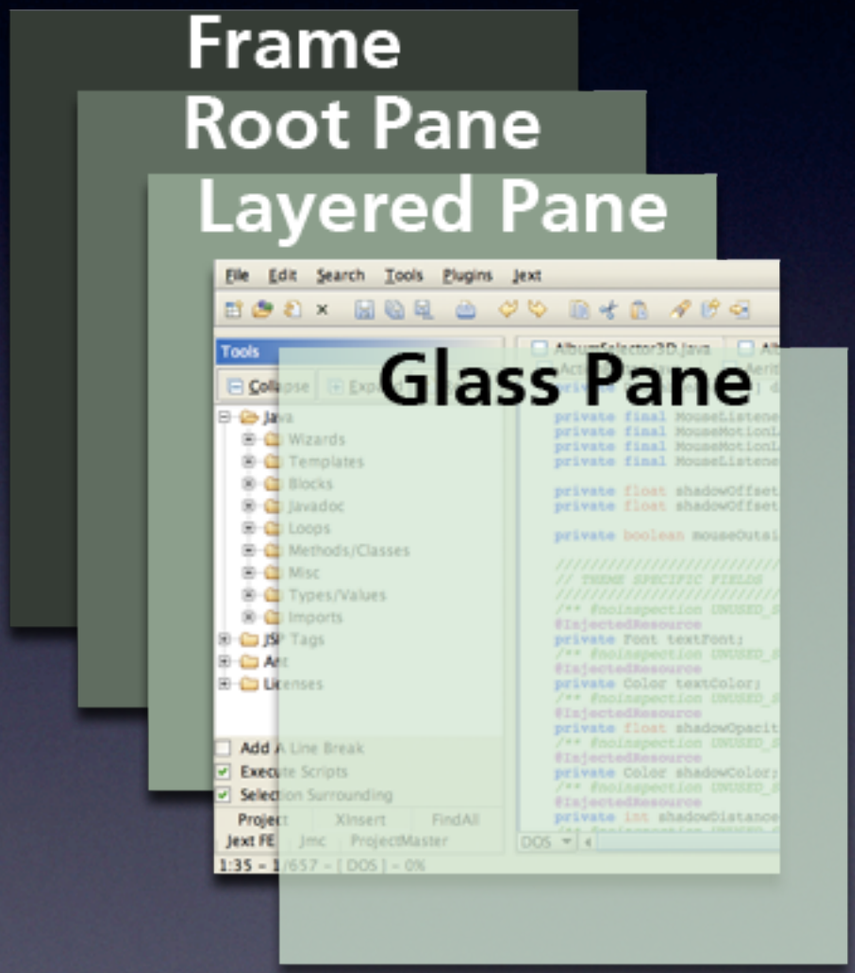
3D

Performance

Fundamentals

- You need:
  - Glass pane
  - Gradients
  - AlphaComposite

Glass Pane



Glass Pane

Glass Pane

```
class MyGlassPane extends JComponent {
  @Override
  protected void paintComponent(Graphics g) {
    // painting jobs
  }
}
```

Glass Pane

```
class MyGlassPane extends JComponent {
  @Override
  protected void paintComponent(Graphics g) {
    // painting jobs
  }
}

JFrame f = new JFrame();
f.setGlassPane(new MyGlassPane());
```

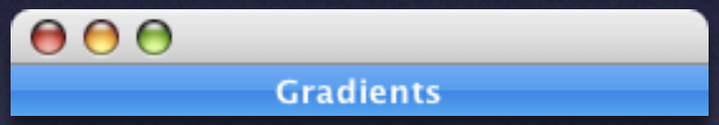
Glass Pane

```
class MyGlassPane extends JComponent {
  @Override
  protected void paintComponent(Graphics g) {
    // painting jobs
  }
}

JFrame f = new JFrame();
f.setGlassPane(new MyGlassPane());

f.getGlassPane().setVisible(true);
```

Gradients

- Up to J2SE 5.0:
  - java.awt.GradientPaint
    - Only linear gradients
    - Only two colors
- With Java SE 6:
  - LinearGradientPaint and RadialGradientPaint
    - Multiple stops, fraction based

Gradients



Gradients

```
GradientPaint p;

p = new GradientPaint(0, 0, new Color(0x63a5f7),
  0, 10, new Color(0x3799f4));
g2.setPaint(p);
g2.fillRect(0, 0, getWidth(), 10);

p = new GradientPaint(0, 10, new Color(0x2d7eeb),
  0, 20, new Color(0x30a5f9));
g2.setPaint(p);
g2.fillRect(0, 10, getWidth(), 10);
```

Gradients

```
LinearGradientPaint p;

p = new LinearGradientPaint(0.0f, 0.0f, 0.0f, 20.0f,
  new float[] { 0.0f, 0.499f, 0.50f, 1.0f },
  new Color[] { new Color(0x63a5f7),
                new Color(0x3799f4),
                new Color(0x2d7eeb),
                new Color(0x30a5f9) });
g2.setPaint(p);
g2.fillRect(0, 0, getWidth(), 20);
```

AlphaComposite

- Basic alpha compositing rules:
  - Porter and Duff equations
  - Source is the currently painted object
  - Destination is the current Graphics
- Only two are important:
  - AlphaComposite.SRC_OVER
  - AlphaComposite.DST_IN

SrcOver

- “Source Over” is used for translucency

SrcOver

- “Source Over” is used for translucency

SrcOver

- “Source Over” is used for translucency

```
c = AlphaComposite.getInstance(
  AlphaComposite.SRC_OVER, 0.5f);
g2.setComposite(c);
g2.drawImage(picture, x, y, null);
```

DstIn

- “Destination In” for reflections and fade-out

Subject Mirrored Mask Result

DstIn

DstIn

```
Image subject = ...;
BufferedImage alphaMask = createGradientMask(
  subjectWidth, subjectHeight);
BufferedImage buffer = createReflection(
  subjectWidth, subjectHeight);
```

DstIn

```
Image subject = ...;
BufferedImage alphaMask = createGradientMask(
  subjectWidth, subjectHeight);
BufferedImage buffer = createReflection(
  subjectWidth, subjectHeight);

Graphics2D g2 = buffer.createGraphics();
g2.setComposite(AlphaComposite.DstIn);
g2.drawImage(alphaMask, null, 0, subjectHeight);
g2.dispose();
```

Demo

Agenda

Graphics

Effects

3D

Performance

Life is rich and restless

and your applications?

Coolness Matters

- Fade
- Pulse
- Spring
- Morphing

Fade

- For gradual changes in UI state
- When a value is changed
- Fade in/out
  - Fade from/to a color
  - Opacity change
- Cross-fade
  - Current value fades out
  - New value fades in

Fade to Black

Fade to Black

```
Animator animator = new Animator(1000);
animator.addTarget(
    new PropertySetter(this, "fadeOut", 1.0f));
animator.setAcceleration(0.2f);
animator.setDeceleration(0.4f);
animator.start();
```

Fade to Black

```
public void setFadeOut(float fadeOut) {
  this.fadeOut = fadeOut;
  repaint();
}
```

```
protected void paintComponent(Graphics g) {
  g.setColor(
      new Color(0.0f, 0.0f, 0.0f, fadeOut));
  Rectangle r = g.getClipBounds();
  g.fillRect(r.x, r.y, r.width, r.height);
  // ...
}
```

Demo

Pulse

- Shows that the UI is alive
  - The user might poke it with a stick
- Draws attention
- Indeterminate progress
- Short, repeated animation

Demo

Pulse

Pulse

```
glow = /* a BufferedImage */;
BufferedImageOp filter = getGaussianBlurFilter(24);
glow = filter.filter(glow, null);
filter = new ColorTintFilter(Color.WHITE, 1.0f);
glow = filter.filter(glow, null);
```

Pulse

```
g2.setComposite(
    AlphaComposite.SrcOver.derive(getAlpha()));
g2.drawImage(glow, x, y, null);
g2.setComposite(AlphaComposite.SrcOver);
g2.drawImage(image, x, y, null);
```

Pulse

```
PropertySetter setter = new PropertySetter(
    this, "alpha", 0.0f, 1.0f);
Animator animator = new Animator(
    600, Animator.INFINITE,
    Animator.RepeatBehavior.REVERSE, setter);
animator.start();
```

Spring

- Visual feedback
- For interactive elements
  - On click
  - On rollover
- Glasspane

Demo

Spring

Spring

```
int width = image.getWidth(this);
width += (int) (image.getWidth(this) *
    MAGNIFY_FACTOR * getZoom());

int height = image.getHeight(this);
height += (int) (image.getHeight(this) *
    MAGNIFY_FACTOR * getZoom());

int x = (bounds.width - width) / 2;
int y = (bounds.height - height) / 2;
```

Spring

```
Graphics2D g2 = (Graphics2D) g.create();
g2.setRenderingHint(
    RenderingHints.KEY_INTERPOLATION,
    RenderingHints.VALUE_INTERPOLATION_BILINEAR);

g2.setComposite(AlphaComposite.SrcOver.derive(
    1.0f - getZoom()));
g2.drawImage(image, x + bounds.x, y + bounds.y,
    width, height, null);
```

Morphing

- Seamless shapes transitions
  - “Shape tweening” in Flash
- Convey more information
- SwingLabs
  - http://www.swinglabs.org

Demo

Morphing

Morphing

```
Shape sourceShape = new RoundRectangle2D.Double(2.0, 2.0,
    getWidth() - 4.0, getHeight() - 4.0, 12.0, 12.0);

GeneralPath.Double destinationShape = new GeneralPath.Double();
destinationShape.moveTo(2.0, getHeight() / 2.0);
destinationShape.lineTo(22.0, 0.0);
destinationShape.lineTo(22.0, 5.0);
destinationShape.lineTo(getWidth() - 2.0, 5.0);
destinationShape.lineTo(getWidth() - 2.0, getHeight() - 5.0);
destinationShape.lineTo(22.0, getHeight() - 5.0);
destinationShape.lineTo(22.0, getHeight());
destinationShape.closePath();

return new Morphing2D(sourceShape, destinationShape);
```

Morphing

```
Morphing2D morph = createMorph();
morph.setMorphing(getMorphing());

Graphics2D g2 = (Graphics2D) g;
g2.setPaint(gradient);
g2.fill(morph);
```

Morphing

```
Animator animator =
PropertySetter.createAnimator(
    150, this, "morphing", 0.0f, 1.0f);
animator.setAcceleration(0.2f);
animator.setDeceleration(0.3f);
MouseTrigger.addTrigger(button, animator,
    MouseTriggerEvent.ENTER, true);
```

Agenda

Graphics

Effects

3D

Performance

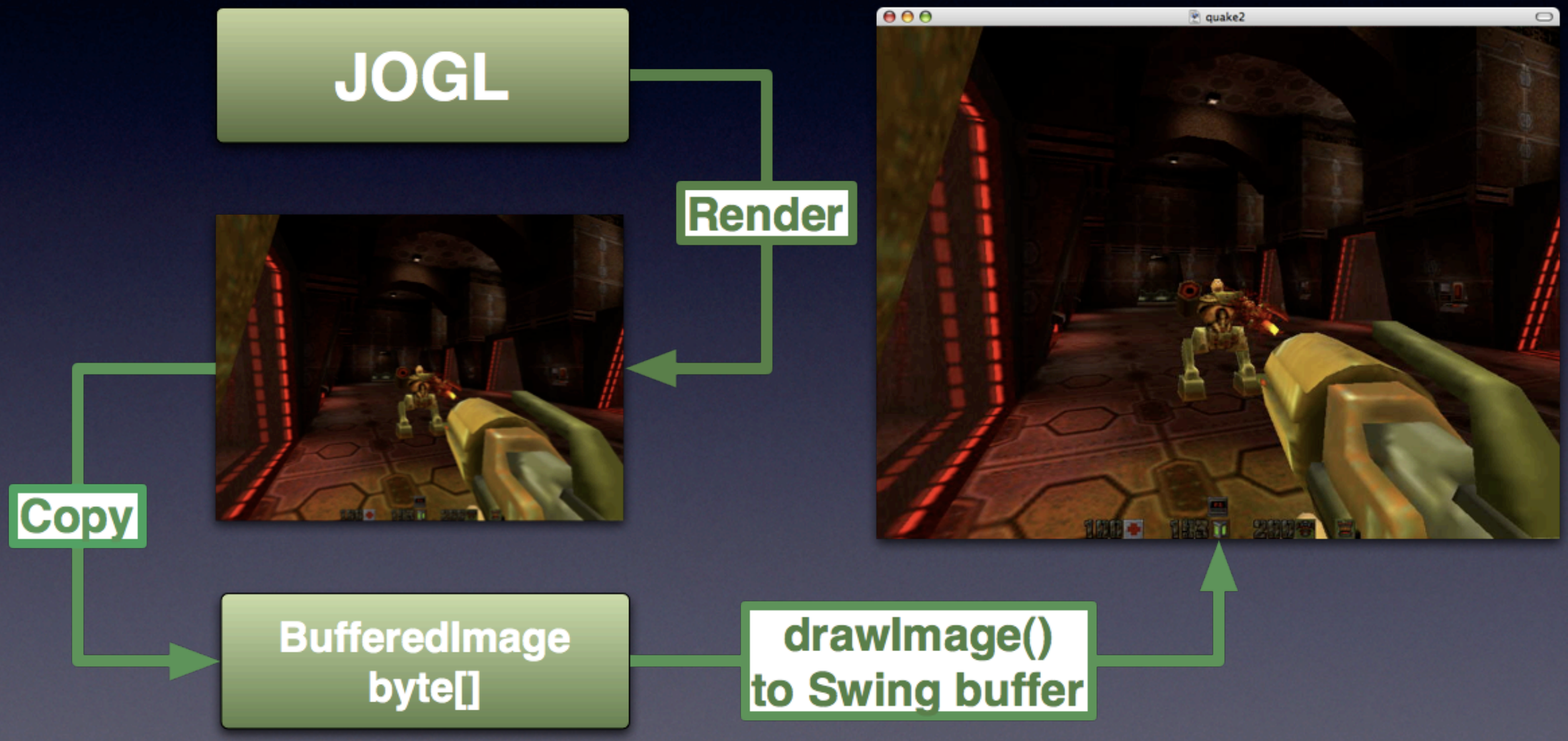
JOGL

- Development version of JSR-231
  - Java™ Bindings for the OpenGL API
  - http://jogl.dev.java.net
- Supplies GLJPanel
  - 100% correct Swing and 3D interaction

Mixing Swing and 3D

- Historical problems:
  - High 3D performance required heavyweight widget
  - Lightweight/heavyweight mixing issues
- 100% correct Swing integration expensive:
  - Render to an off-screen buffer (“pbuffer”)
  - Render back frame buffer into a byte array
  - Render BufferedImage using Java2D

Mixing Swing and 3D



Java SE 6

- Access to internals of Java2D/OpenGL pipeline
  - OpenGL drawable, context and rendering thread
- Highly experimental
- More work to be done in Java SE 7

Java2D/OpenGL Bridge

- GLJPanel bridged to the OpenGL pipeline
  - Much higher performance
    - Same speed as heavyweight components
    - 100% correct Swing integration
    - No application change
- Relies on experimental Java2D APIs
  - Interoperable with other OpenGL libraries

Java2D/OpenGL Bridge

JOGL

Demo

Agenda

Graphics

Effects

3D

Performance

Painting

- Swing IS double-buffered
- Swing coalesces repaint() calls
- Call repaint(x, y, w, h) whenever possible
  - Check out the clipping rectangle
  - Example: moving objects on the glass pane
- Cache large gradients in images
- Avoid primitive objects (Line2D...)

Loading Images

- Hardware vs. software pixel formats
  - JPEG pictures are the most common case
- Use compatible images:
  - Easy to implement
  - The performance boost is impressive (20x)
- Don’t think about it, just do it!

Loading Images

```
public static BufferedImage toCompatibleImage(BufferedImage image) {
  GraphicsEnvironment e =
      GraphicsEnvironment.getLocalGraphicsEnvironment();
  GraphicsDevice d = e.getDefaultScreenDevice();
  GraphicsConfiguration c = d.getDefaultConfiguration();

  BufferedImage compatibleImage = c.createCompatibleImage(
      image.getWidth(), image.getHeight());

  Graphics g = compatibleImage.getGraphics();
  g.drawImage(image, 0, 0, null);
  g.dispose();

  return compatibleImage;
}
```

Loading Images

```
BufferedImage compatibleImage = c.createCompatibleImage(
    image.getWidth(), image.getHeight());

Graphics g = compatibleImage.getGraphics();
g.drawImage(image, 0, 0, null);
g.dispose();
```

Painting Images

- Well... Graphics.drawImage()
- Beware the fourth parameter!
  - java.awt.ImageObserver
  - Utterly useless with ImageIO
- Pass null instead
  - Easy performance gain

Resizing Images

- One very convenient way: Image.getScaledInstance()
- NEVER USE IT!
  - I mean it.
  - Really.
- Harness the power of Java2D instead

Resizing Images

```
@Override
protected void paintComponent(Graphics g) {
  g.drawImage(image, x, y, newWidth, newHeight, null);
}
```

Resizing Images

- Default resizing looks bad
  - Use the bilinear rendering hint
  - Bicubic resizing is too slow
- Bilinear is not perfect though
  - Dividing the size by 2+ is similar to default resize
  - Proceed step by step
  - Divide only by half the size at each step

Resizing Images

```
int width = image.getWidth();
float ratio = (float) width / (float) image.getHeight();
BufferedImage thumb = image;
do {
    width /= 2;
    // ...
    BufferedImage temp = new BufferedImage(width,
                                           (int) (width / ratio),
                                           BufferedImage.TYPE_INT_ARGB);
    Graphics2D g2 = temp.createGraphics();
    g2.setRenderingHint(...);
    g2.drawImage(thumb, 0, 0, temp.getWidth(), temp.getHeight(), null);
    g2.dispose();

    thumb = temp;
} while (width != thumbWidth);
```

Resizing Images

Step by step bilinear

Standard bilinear

Demo

Image Scaling

Animated Transitions

- Don’t make the user work to understand the GUI
- Lead them logically through state changes

Animated Transitions: The Project

- Built on top of Timing Framework
- Not available yet, released with The Book
  - http://filthyrichclients.org
- Idea:
  - Hand container to ScreenTransition
  - Configure Animator
  - start()
  - Handle callback to set up next screen
  - Transition animation just runs

Animated Transitions Sample Code

```
Animator animator = new Animator(1000);

ScreenTransition transition = new
    ScreenTransition(
            transitionContainer, this, animator);
animator.setDeceleration(.4f);
transition.start();

// TransitionTarget implementation
```

Demo

Animated Transitions

Resources

- http://aerith.dev.java.net
- http://timingframework.dev.java.net
- http://www.swinglabs.org
- http://www.curious-creature.org
- http://weblogs.java.net/blog/chet
- http://www.javadesktop.org
- http://developers.sun.com/learning/javaoneonline/

The Book

- Indeed! With interesting stuff and funny jokes!
- ETA: JavaOne 2007
- ISBN: 0132413930

Q&A

Complaints: romain.guy@mac.com

Fan mail: chet.haase@sun.com