Filthy Rich Clients: Talk Dirty to Me

(The presentation based on the book based on the presentation)

Chet Haase, Sun Microsystems, Inc.

Romain Guy, Google

TS-3165

Goal

Data Binding

Goal

~~Data Binding~~

Application Frameworks

Goal

~~Data Binding~~

~~Application Frameworks~~

Global, Cooperative, Multi-Tiered Business Logic Interoperability Models for Service Oriented Architectures

Goal

~~Data Binding~~

~~Application Frameworks~~

Global, Cooperative, Multi-Tiered Business Logic Interoperability Models for Service Oriented Architectures

Cooler Applications

Agenda

Graphics
Performance
Animation
Effects

Agenda

Graphics

Performance

Animation

Effects

`paint()` vs. `paintComponent()`

- Custom components should usually only override **`paintComponent()`**
  - Customizes the contents of a component
- **`JComponent.paint()`** does more
  - **`paintBorder()`**
  - **`paintChildren()`**
  - **`paintComponent()`**
- Overriding **`paint()`** may clobber painting that should really happen

Proper paint() Overriding

- Sometimes you really do want to override paint()
  - When you want to affect everything about a component's looks
    - The contents, the border, and the children
- Tip: Use the superclass

```
// Paints a translucent component
public void paint(Graphics g) {
    // Setup composite
    ((Graphics2D)g).setComposite(translucentComposite);
    // Ask superclass to do the painting
    super.paint(g);
}
```

Components as Images

- Useful trick in Filthy Rich Clients
  - Performance
  - Effects
- Create an image, draw the component into it

```
// Create an opaque image
compImage = getGraphicsConfiguration().
    createCompatibleImage(getWidth(), getHeight());
// get the Graphics for the Image
Graphics gImage = compImage.getGraphics();
// render component to the image
paint(gImage);
```

Component Image: Example

```
// Paints a translucent button ... correctly
public void paint(Graphics g) {
  // ...Assume compImage is created/cached as before...

  Graphics gImage = compImage.getGraphics();

  // Copy clip into image Graphics
  gImage.setClip(g.getClip());

  // Have the superclass render the component for us
  super.paint(gImage);

  // Setup translucent composite for g
  ((Graphics2D)g).setComposite(newComposite);

  // Copy component's image translucently
  g.drawImage(compImage, 0, 0, null);
}
```

Agenda

Graphics

Performance

Animation

Effects

Image Scaling

- Common operation
  - Zooming in/out, thumbnails, various effects…
- Two things matter
  - Quality
  - Performance
- Scaling approach affects both

Image Scaling

- [Too] many approaches, including

```
Image.getScaledInstance()

g.drawImage(img, x, y, w, h, null);

g.drawImage(img, dx1, dy1, dx2, dy2,
                         sx1, sy1, sx2, sy2, null);

g.translate(x, y,);
((Graphics2D)g).scale(sx, sy);
g.drawImage(img, 0, 0, null);

// etc., etc....
```

Image Scaling: Quality

```
Graphics2D.setRenderingHint(RenderingHints.
    KEY_INTERPOLATION, hint);
```

- Where ‘hint’ is one of
  - `VALUE_INTERPOLATION_NEAREST_NEIGHBOR`
  - `VALUE_INTERPOLATION_BILINEAR`
  - `VALUE_INTERPOLATION_BICUBIC`

```
Image.getScaledInstance(w, h, hint);
```

- Where ‘hint’ is one of
  - `SCALE_AREA_AVERAGING`
  - `SCALE_AREA_SMOOTH`
  - `SCALE_DEFAULT`
  - `SCALE_FAST`
  - `SCALE_REPLICATE`

Image Scaling: Performance

- In order of performance (best to worst)
  - drawImage() with default interpolation (NEAREST)
  - drawImage() with BILINEAR
  - drawImage() with BICUBIC
  - getScaledInstance() with SMOOTH/AREA_AVERAGING
- **Don't use Image.getScaledInstance()**
  - **Ever**
  - Highest quality, but horrid performance
- Instead, try Progressive Bilinear
  - Downscale by halves with BILINEAR

DEMO

Image Scaling

Use the Clip

Two sides to the equation

- Request repaints only for necessary areas
- Don’t paint anything outside clip bounds

Use the Clip: Repaint Requests

- Repaint only what you need
- **`JComponent.repaint()`**
  - Sets clip to entire component
- **`JComponent.repaint(x, y, w, h)`**
  - Sets clip to specified area
  - Useful when only part of a component changes

Use the Clip: Paint Inside Clip Bounds

- Easy to check clip bounds:

```
Rectangle bounds = g.getClipBounds();
```

- Only paint overlapping primitives:

```
// assume xLeft,yTop,xRight,yBottom is
// rectangular bounds of primitive
int boundsR = bounds.getMaxX();
int boundsB = bounds.getMaxY();
if (xLeft > boundsR || xRight < bounds.x ||
    yTop > boundsB || yBottom < bounds.y) {
      // proceed with painting
}
```

- Note: Don’t go overboard

DEMO

Use the Clip

Performance Tips: Grab Bag

- Compatible Images
  - GraphicsDevice.createCompatibleImage(…)
- Intermediate Images
- Optimal primitive rendering
  - Simpler is faster

Agenda

Graphics

Performance

Animation

Effects

Timing Framework: Quick Course

- Timing engine for scheduling and running animations
- Takes
  - Duration
  - Repeat behavior
  - TimingTarget
- Advanced features
  - PropertySetter
  - Triggers
  - Multi-step animations

Timing Framework: Basics

```
class MyTarget implements TimingTarget {
    public void begin() {…}
    public void end() {…}
    public void repeat() {…}
    public void timingEvent(float fraction) {…}
}
TimingTarget target = new MyTarget();

// animate once for 5 seconds, then stop
Animator singleRun = new Animator(5000, target);
singleRun.start();

// animate for 5 cycles of 2 secs, reversing each time
Animator oscillator = new Animator(2000, 5,
                                   RepeatBehavior.REVERSE,
                                   target);
oscillator.start();
```

Timing Framework: Property Setters

Animate property automatically

```
// Move button between 'from' and 'to' in 2 seconds
Point from = (50, 50);
Point to = (100, 150);
PropertySetter ps = new PropertySetter(
    button, "location", from, to);
Animator mover = new Animator(2000, ps);
mover.start();

// Same thing, only easier
PropertySetter.createAnimator(2000,button,"location",
    from, to).start();
```

Triggers

Automatically start animations based on events

```
Animator anim1 = /* Animator setup */
Animator anim2 = /* Animator setup */

// Start anim1 when button is clicked
ActionTrigger.addTrigger(button, anim1);

// Start anim1 when mouse is over button
MouseTrigger.addTrigger(button, anim1,
    MouseTriggerEvent.ENTER);

// Start anim2 when anim1 stops
TimingTrigger.addTrigger(anim1, anim2,
    TimingTriggerEvent.STOP);
```

Non-Linear Interpolation

- Simple

```
setAcceleration(fraction)
setDeceleration(fraction)
```

- Complete
  - SplineInterpolator(...)
  - Custom Interpolator:

```
public class MyInterpolator implements
  Interpolator {
    public float interpolate(float f) {
       // return anything [0-1]
    }
}
```

Agenda

Graphics

Performance

Animation

Effects

And Your Questions?

Coolness Matters

- Blur
- Drop Shadows
- Spring
- Morphing
- Animated transitions

Blur

- Simulate natural vision
- Help focus user's attention

Various Blurs

- Simple blur
  - “Box blur”
  - ConvolveOp + Kernel
- Gaussian blur
  - org.jdesktop.swingx.image.GaussianBlurFilter
  - www.swinglabs.org
  - Looks better, more natural
- Expensive effect
  - Reduce image size, then blur

Box Blur

```
float[] data = new float[] {
    1.0f/9.0f, 1.0f/9.0f, 1.0f/9.0f,
    1.0f/9.0f, 1.0f/9.0f, 1.0f/9.0f,
    1.0f/9.0f, 1.0f/9.0f, 1.0f/9.0f };
Kernel k = new Kernel(3, 3, data);
ConvolveOp op = new ConvolveOp(k);

BufferedImage image = loadImage();
BufferedImage blurImage = op.filter(image, null);
```

Gaussian Blur

```
// changeImageWidth() uses
// Graphics.drawImage(image, x, y, w, h, null)
// and bilinear rendering hint

BufferedImage image = // ...
image = changeImageWidth(image, image.getWidth() / 2);

GaussianBlurFilter blur = new GaussianBlurFilter(12);
image = blur.filter(image, null);

changeImageWidth(image, image.getWidth() * 2);
```

DEMO

Blur

Drop Shadows

- Simulate lighting
- Simulate depth
- Make GUI more realistic
  - Better, more natural feedback to the user
- Simple shadows are easy
- Realistic shadows requires blur
  - Use ShadowRenderer
  - www.swinglabs.org

Regular Shadow

```
public static BufferedImage
    createDropShadow(BufferedImage image, int size) {
  BufferedImage shadow = new BufferedImage(
      image.getWidth() + 4 * size,
      image.getHeight() + 4 * size,
      BufferedImage.TYPE_INT_ARGB);

  Graphics2D g2 = shadow.createGraphics();
  g2.drawImage(image, size * 2, size * 2, null);

  g2.setComposite(AlphaComposite.SrcIn);
  g2.setColor(Color.BLACK);
  g2.fillRect(0, 0, shadow.getWidth(), shadow.getHeight());

  g2.dispose();
  return shadow;
}
```

DEMO

Drop Shadows

Spring

- Attract user's attention
  - On mouse over
- Provide visual feedback
  - On click/double-click
- Creating a spring
  - Copy the element in an image
  - Grow the copy on top of the element
  - While growing, fade-out

Spring

```
Rectangle b = // position and size of original element

int width = image.getWidth();
width += (int) (width * MAGNIFY_FACTOR * zoom);

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

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

Graphics2D g2 = (Graphics2D) g.create();

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

DEMO

Spring

Morphing

- Seamless shapes transition
  - Shape “tweening” in Flash
- Convey more information
  - Buttons in Nintendo Wii's UI
- SwingLabs
  - www.swinglabs.org

Morphing

```
Morphing2D morph = new Morphing2D(
    sourceShape, destinationShape);

morph.setMorphing(value);

Graphics2D g2 = // ...
g2.fill(morph);

// animate value between 0.0 (source)
// and 1.0 (destination)
```

Triggering Morphing

```
Animator animator = PropertySetter.createAnimator(
    150, this, "value", 0.0f, 1.0f);
animator.setAcceleration(0.2f);
animator.setDeceleration(0.3f);

MouseTrigger.addTrigger(this, animator,
    MouseTriggerEvent.ENTER, true);
```

DEMO

Morph

Animated Transitions

- Don’t make your users work too hard
- When application state changes
  - ***Lead*** the user
  - Don’t ***leave*** them

Animated Transitions: The Project

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

DEMO

Search Transition

SearchTransition: The Code

```
// Configure Animator
Animator animator = new Animator(500);
animator.setAcceleration(.2f);
animator.setDeceleration(.4f);

// Setup ScreenTransition object
ScreenTransition transition = new ScreenTransition(
    searchContainer, this, // TransitionTarget
    animator);

// start transition on appropriate user action
transition.start();

// handle callback from transition
public void setupNextScreen() {
    // setup appropriate GUI state
}
```

DEMO

ImageBrowser

ImageBrowser: The Code

```
// Animator/ScreenTransition setup like SearchTransition
// ...

// start transition on slider state change
transition.start();

// handle callback from transition
public void setupNextScreen() {
    // New GUI derived by new Icons set on labels
    for (...) {
        labels[i].setIcon(...);
    }
    // Let the LayoutManager do its job
    revalidate();
}
```

AnimatedTransitions: Effects

- Standard effects
  - Disappearing: FadeOut
  - Appearing: FadeIn
  - Changing: Move/Scale
- Custom effects
  - Subclass Effect, override
    - init(): Sets up state of effect at start of transition
    - setup(Graphics2D): optional, sets graphics state per frame
    - paint(Graphics2D): optional, renders effect
  - CompositeEffect for combinations
  - EffectsManager.setEffect(effect, component,TransitionType)

SearchTransition: MoveIn Effect

```
class MoveIn extends Effect {
    private Point startLocation = new Point();
    public MoveIn(int x, int y) {
        startLocation.x = x;
        startLocation.y = y;
    }
    public void init(Animator animator, Effect parentEffect) {
        Effect targetEffect =
                (parentEffect == null) ? this : parentEffect;
        PropertySetter ps;
        ps = new PropertySetter(targetEffect, "location",
                startLocation,
                new Point(getEnd().getX(), getEnd().getY()));
        animator.addTarget(ps);
        super.init(animator, parentEffect);
    }
    // No setup/paint needed; handled by Effect superclass
}
```

SearchTransition: Setting the Effect

```
// Create our custom effect
MoveIn mover = new MoveIn(RESULTS_X, getHeight());
// Create a standard FadeIn effect
FadeIn fader = new FadeIn();
// Create a CompositeEffect using our custom effect
moverFader = new CompositeEffect(mover);
// ... and the FadeIn effect
moverFader.addEffect(fader);

// Set our effect to take place on the results
// component when it is APPEARING
EffectsManager.setEffect(scroller, moverFader,
EffectsManager.TransitionType.APPEARING);
```

Faster Effects

- OpenGL®
  - Java™ Binding for OpenGL® (JOGL)
  - jogl.dev.java.net
- Pixel shaders
  - Implemented in GLSL
- Easy to implement with MSG
  - Mini scene-graph library for JOGL

Blur Shader

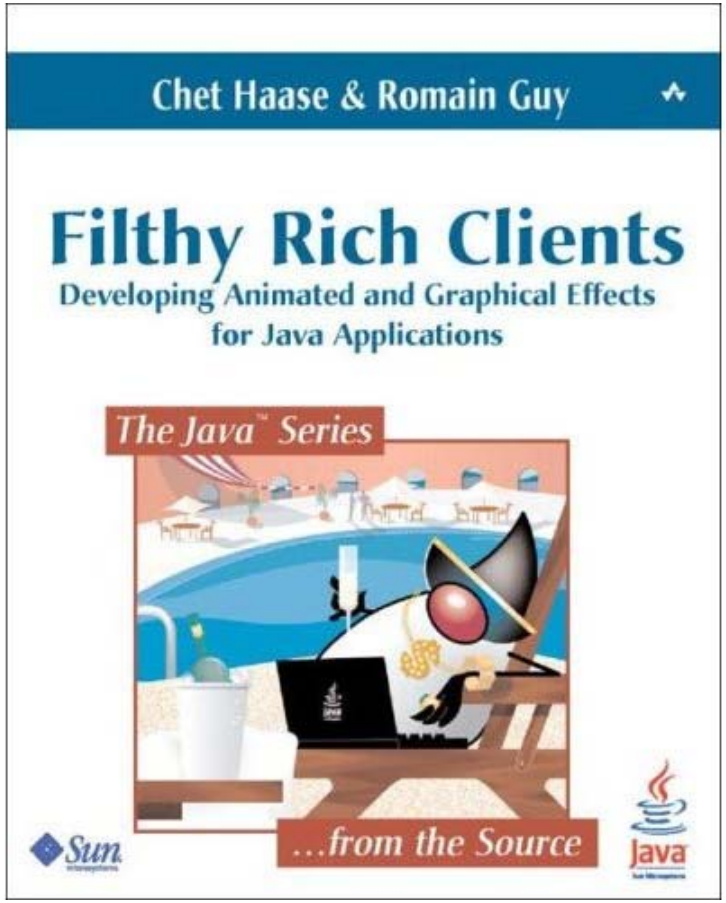
```
const int MAX_KERNEL_SIZE = 25;
uniform sampler2D baseImage;
uniform vec2 offsets[MAX_KERNEL_SIZE];
uniform float kernelVals[MAX_KERNEL_SIZE];

void main(void) {
  int i;
  vec4 sum = vec4(0.0);

  for (i = 0; i < MAX_KERNEL_SIZE; i++) {
    vec4 tmp = texture2D(baseImage,
        gl_TexCoord[0].st + offsets[i]);
    sum += tmp * kernelVals[i];
  }

  gl_FragColor = sum;
}
```

DEMO

Bloom

And Now: A Word From Our Sponsor

- The book is done!
- Final release in August
- “Rough cut” online now
- Deep dive into Swing graphics, animation, performance, and effects
- http://filthyrichclients.org
  - Demos, libraries, code



Q&A

chet.haase@sun.com

romain.guy@mac.com

Filthy Rich Clients: Talk Dirty to Me

(The presentation based on the book based on the presentation)

Chet Haase, Sun Microsystems, Inc.

Romain Guy, Google

TS-3165