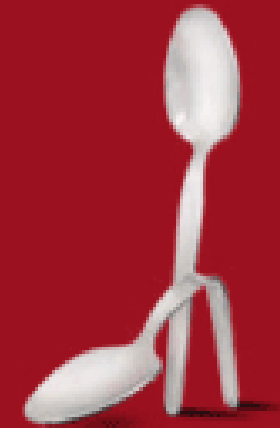


**IMPROVE YOUR
PATTERNS AT
JAVAPOLIS 2005.**

**METROPOLIS ANTWERP,
DECEMBER 12 UNTIL 16.**





Extreme Swing



Romain Guy
Sun Microsystems





Overall Presentation Goal

Learn how to create modern user interfaces with Swing





Agenda

- About user interfaces
- Add a new dimension
 - Swing and Java2D
- Going further
 - Swing and OpenGL
- Treats
- Q&A





Agenda

- About user interfaces
- Add a new dimension
 - Swing and Java2D
- Going further
 - Swing and OpenGL
- Treats
- Q&A





Today's User Interfaces

- Slow evolution
 - Same windowing concept
 - Same widgets
 - Same interaction
- 2D space
- Boring...
- ...or difficult to use





Modern User Interfaces

- Can be found in:
 - MacOS X
 - Windows Vista
 - And games!
- Not only special effects
- Add a new dimension
 - Moving toward the 3D space





The Third Dimension

- It looks cool
- Many shortcomings
 - Awkward user interaction
 - Mouse and keyboard?
 - Difficult to blend in today's UIs
 - Requires extra skills
 - Requires extra hardware





Enter the 2.5D World

- Pseudo-3D
- Depth faked with 2D techniques
 - Parallax
 - Resizing
 - Fade out
- Extended meaning
 - Full blown 3D rendering
 - Input restrained in 2D space





Agenda

- About user interfaces
- Add a new dimension
 - Swing and Java2D
- Going further
 - Swing and OpenGL
- Treats
- Q&A





Add a Third Dimension

- Fake 3D world
 - No projection
 - No frustum, FOV and all that jazz
- Use plain old Java2D
 - Gradients
 - AlphaComposite
 - Hardware acceleration





DEMO

Music Shelf

www.javapolis.com





How does it work?

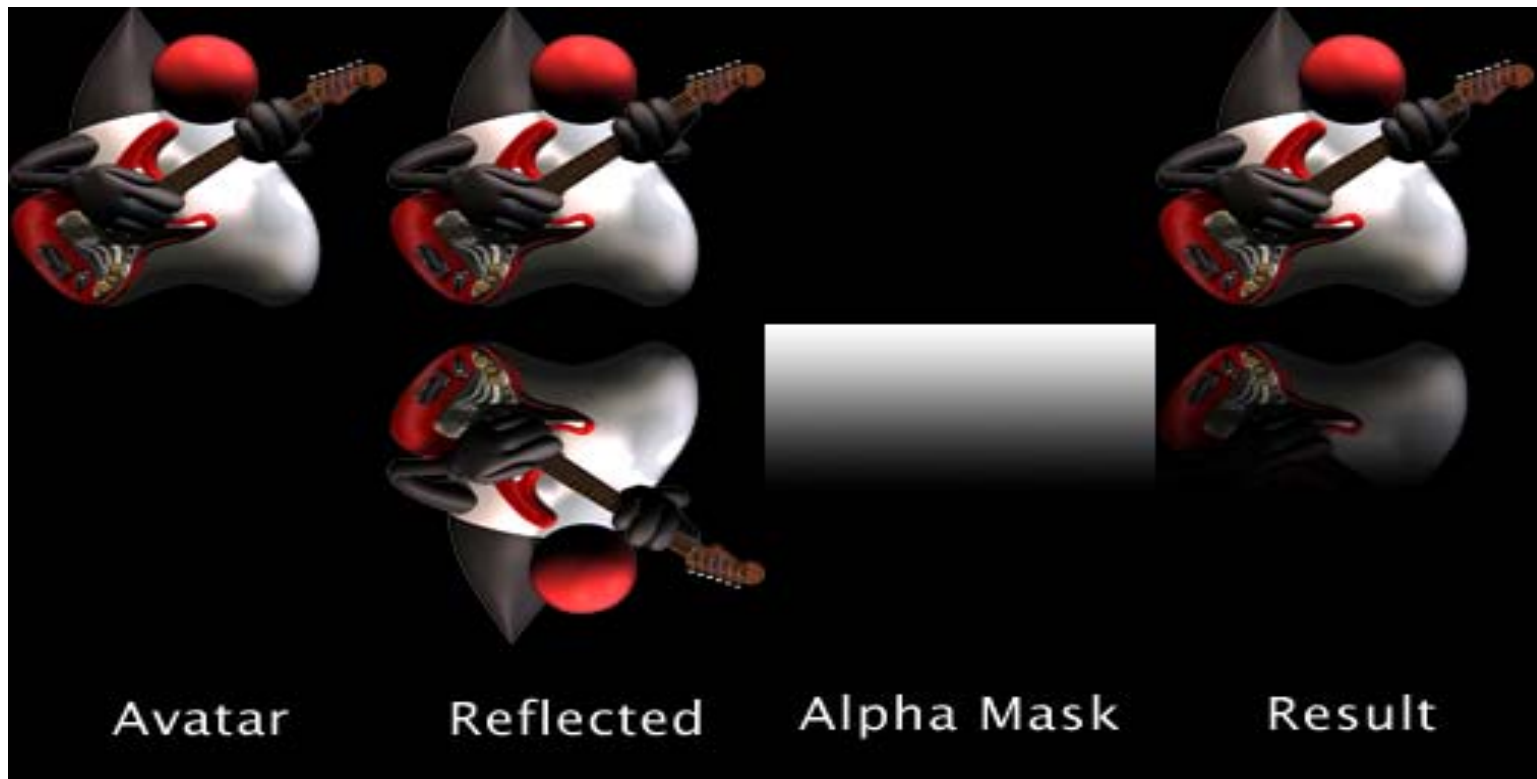
- Illusion of 3D environment
 - Reflection on the “ground”
- Illusion of depth
 - The further, the smaller
 - The further, the more translucent
 - Exercise: depth of field





Reflective Environment

- Three steps rendering





Reflective Environment

```
Image album = ...;
// gradient mask width is 1 pixel
BufferedImage alphaMask = createGradientMask(albumHeight);
BufferedImage buffer = createReflection(album,
                                       albumWidth,
                                       albumHeight);

Graphics2D g2 = buffer.createGraphics();
g2.setComposite(AlphaComposite.DstOut);
g2.drawImage(alphaMask, 0, albumHeight,
            albumWidth, albumHeight, null);
g2.dispose();
```





Illusion of Depth

- The middle picture is the nearest
 - Position = 0.0
- Pictures on the edges are the furthest
 - Position = -1.0 or +1.0
- Given a position we need
 - A resizing factor
 - An opacity factor

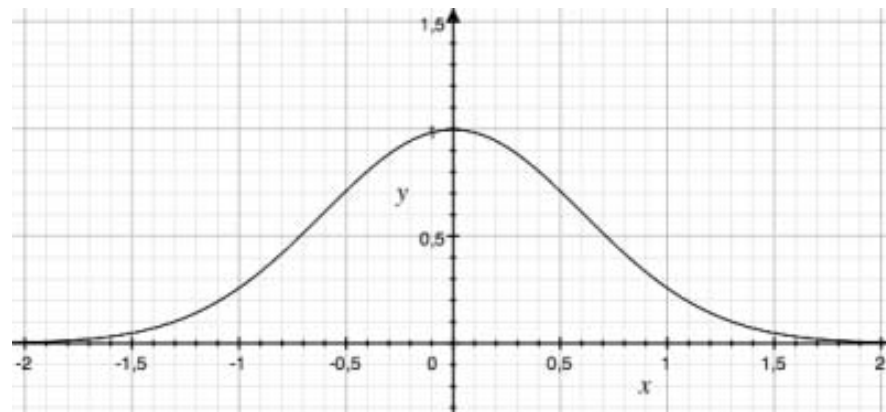




Illusion of Depth

- For crazy mathematicians
 - $f([-1.0, 1.0]) \rightarrow [0.0, 1.0]$
 - $f(0.0) \approx 1.0$
- Where f is a Gaussian distribution

$$y = \frac{\frac{1}{\sigma} \cdot \sqrt{2 \cdot \pi} \cdot e^{-\frac{x^2}{(2 \cdot \sigma)^2}}}{Q}$$





DEMO

Gaussian Distribution

www.javapolis.com





Illusion of Depth

- World coordinates
 - $p.x = xToPixels(x)$
 - $p.y = center(screen, picture)$
 - $p.z = f(x)$
- Rendering components
 - $width = picture.width * p.z$
 - $height = picture.height * p.z$
 - $alpha = p.z$





Illusion of Depth

```
private void drawAvatars(Graphics2D g2,  
                        DrawableAvatar[] drawableAvatars) {  
    for (DrawableAvatar avatar: drawableAvatars) {  
        AlphaComposite c;  
        c = AlphaComposite.getInstance(AlphaComposite.SRC_OVER,  
                                       (float) avatar.getAlpha());  
        g2.setComposite(composite);  
        g2.drawImage(avatars.get(avatar.getIndex()),  
                    (int) avatar.getX(), (int) avatar.getY(),  
                    avatar.getWidth(), avatar.getHeight(), null);  
    }  
}
```





Input Support

- Expect the worst case scenario
 - Clueless user
 - Well, it's actually the regular scenario
- Support many input methods
 - Left/Right/Up/Down
 - Page Up/Down
 - Home/End
 - Mouse Wheel Up/Down
 - Space/Enter





Agenda

- About user interfaces
- Add a new dimension
 - Swing and Java2D
- **Going further**
 - **Swing and OpenGL**
- Treats
- Q&A





Going Further with OpenGL

- What else could we do?
- Think about what you could **not** do
- OpenGL is a 3D rendering API
 - Multi-platform
 - Easy to understand
 - Well defined and supported
 - A lot of great documentation





Why OpenGL?

- Java bindings
 - JOGL, LWGL, GL4Java...
- Lightweight/heavyweight issues
 - Very difficult to use with Swing
- JOGL solves this problem with GLJPanel
- Well, almost
 - Awful rendering performance





Blessed Be Mustang

- Mustang is cool
- Java2D OpenGL pipeline
 - Offers impressive Swing performance
- Java2D/OpenGL interoperability
 - JOGL hooks into Java2D
 - Excellent performance with GLJPanel
 - Non opaque GLJPanel
- **-Dsun.java2d.opengl=true**





DEMO

Twinkle

www.javapolis.com





Resources

- Mustang
 - <http://mustang.dev.java.net>
- JOGL
 - <http://jogl.dev.java.net>
- Official OpenGL Web Site
 - <http://www.opengl.org>
- NeHe Tutorials and Articles
 - <http://nehe.gamedev.net/>





Rendering Utilities

- org.progx.jogl
 - CompositeGLPanel
 - GLUtilities
 - Texture
- org.progx.jogl.rendering
 - Billboard, Quad, ReflectedQuad
 - Renderable
 - RenderableFactory





CompositeGLPanel

```
public class CompositeGLPanel extends GLJPanel
    implements GLEventListener {
    public CompositeGLPanel(boolean isOpaque,
        boolean hasDepth) {
        // ...
    }

    public void init(GLAutoDrawable drawable) {
        // ...
    }

    protected void render2DBackground(Graphics g) { }
    protected void render3DScene(GL gl, GLU glu) { }
    protected void render2DForeground(Graphics g) { }
}
```





Rendering Process

- Subclass CompositeGLPanel
- Override render methods
- In render3DScene()
 - Init and dispose objects
 - Position the camera
 - Render objects
- Use the Renderable abstract class
 - Hides the OpenGL implementation





Scene Setup

```
private void initScene(GL gl) {
    gl.glMatrixMode(GL.GL_MODELVIEW);
    gl.glLoadIdentity();
}

private void setupCamera(GL gl, GLU glu) {
    glu.gluLookAt(camPosX, camPosY, camPosZ,
                 camLookAtX, camLookAtY, camLookAtZ,
                 0.0f, 1.0f, 0.0f);
    gl.glRotatef(camRotationX, 1.0f, 0.0f, 0.0f);
    gl.glRotatef(camRotationY, 0.0f, 1.0f, 0.0f);
    gl.glRotatef(camRotationZ, 0.0f, 0.0f, 1.0f);
}
```



Renderable Class

- `dispose(GL)`
- `init(GL)`
- `render(GL)`
- `get/setRotation(...)`
- `get/setPosition(...)`
- `get/setScale(...)`
- `get/setName(...)`



Rendering Objects

```

private void setAndRender(GL gl, Renderable renderable) {
    Point3f pos = renderable.getPosition();
    Point3i rot = renderable.getRotation();
    Point3f scale = renderable.getScale();

    gl.glPushMatrix();
    gl.glScalef(scale.x, scale.y, scale.z);
    gl.glTranslatef(pos.x, pos.y, pos.z);
    gl.glRotatef(rot.x, 1.0f, 0.0f, 0.0f);
    gl.glRotatef(rot.y, 0.0f, 1.0f, 0.0f);
    gl.glRotatef(rot.z, 0.0f, 0.0f, 1.0f);

    renderable.render(gl);
    gl.glPopMatrix();
}

```





Creating Objects

- Quad
 - Simple texture mapped flat rectangle
- Reflected quad
 - A quad with a simulated reflection
- Billboards
 - A Renderable always facing the camera
 - **new Billboard**(aRenderable)



Creating Objects

- Use the RenderableFactory for
 - Quads
 - Reflected quads

```
BufferedImage image = ImageIO.read(new File(name));
```

```
Renderable quad;  
quad = RenderableFactory.createQuad(0.0f, 0.0f, 0.0f,  
    QUAD_WIDTH, height, image, null /* crop */, name);  
quad.setPosition(-7.0f, 0.0f, 0.0f);  
quad.setRotation(0, 30, 0);
```





Quad Init Implementation

- Create a Texture object
 - Convert BufferedImage to OpenGL
 - Generate texture ID

```
public void init(GL gl) {  
    texture = Texture.getInstance(gl, textureImage);  
}
```

```
public void dispose(GL gl) {  
    texture.dispose(gl);  
}
```





Quad Render Implementation

```
public void render(GL gl) {  
    float[] crop = texture.getSubImageTextureCoords(  
        textureCrop.x, textureCrop.y,  
        textureCrop.x + textureCrop.width,  
        textureCrop.y + textureCrop.height);  
    float tx1 = crop[0];  
    float ty1 = crop[1];  
    float tx2 = crop[2];  
    float ty2 = crop[3];  
  
    float x = -width / 2.0f;  
    float y = -height / 2.0f;  
    float z = 0.0f;  
  
    // to be continued  
}
```





Quad Render Implementation

```
gl.glEnable(GL.GL_TEXTURE_2D);
texture.bind(gl);
gl.glTexEnvf(GL.GL_TEXTURE_ENV,
             GL.GL_TEXTURE_ENV_MODE, GL.GL_MODULATE);
gl.glBegin(GL.GL_QUADS);
gl.glColor4f(alpha, alpha, alpha, alpha);
gl.glTexCoord2f(tx2, ty1);
gl.glVertex3f(x + width, y + height, z);
gl.glTexCoord2f(tx1, ty1);
gl.glVertex3f(x, y + height, z);
gl.glTexCoord2f(tx1, ty2);
gl.glVertex3f(x, y, z);
gl.glTexCoord2f(tx2, ty2);
gl.glVertex3f(x + width, y, z);
gl.glEnd();
gl.glDisable(GL.GL_TEXTURE_2D);
```





And ReflectedQuad?

- A second quad is drawn
 - With an alpha gradient
 - With alpha blending mode
- OpenGL generates the gradient for us

```
// upper half of the second quad  
gl.glColor4f(alpha, alpha, alpha, alpha);  
gl.glTexCoord2f(...);  
gl.glVertex3f(...);  
// lower half  
gl.glColor4f(0.0f, 0.0f, 0.0f, 0.0f);  
// glTexCoord etc.
```



GLUtilities

- `drawLocalAxis(gl, axisLength)`
 - Shows orientation
- `renderAntiAliased(gl, renderable, aaLevel)`
 - Full scene anti aliasing is not always possible
 - Old, slow technique by accumulation
 - Precludes transparency on GLJPanel





DEMO

Twinkle, no AA and 2D background

www.javapolis.com





Agenda

- About user interfaces
- Add a new dimension
 - Swing and Java2D
- Going further
 - Swing and OpenGL
- **Treats**
- Q&A





Treat #1, 3D Icon

- Take a regular JButton
- Remove text and icon
- Insert a full OpenGL scene
- Animate it on mouse over
- It's a hack



DEMO

3D Icon in a JButton

www.javapolis.com





Treat #1, 3D Icon

- When I say “hack”, I mean it
- Seriously

```
 JButton button = new JButton();  
 button.setLayout(new FlowLayout());  
 button.add(build3dIcon(button));  
 button.add(new JLabel("Preferences"));  
 button.setMargin(new Insets(3, 3, 3, 3));
```





Treat #1, 3D Icon

```
private Component build3dIcon(JButton button) {
    GearsPanel panel = new GearsPanel();
    final Animator animator = new Animator(panel);
    button.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseExited(MouseEvent e) {
            animator.stop();
        }
        @Override
        public void mouseEntered(MouseEvent e) {
            animator.start();
        }
    });
    panel.setPreferredSize(new Dimension(64, 64));
    return panel;
}
```





Treat #2, Java3D and Opacity

- Take a Swing container
- Paint it onto an image
- Create a Java3D scene
- Use the image as a background
- You have `setOpaque(false)`
- It's also a hack



DEMO

Non Opaque Java3D Canvas

www.javapolis.com





Treat #2, Java3D and Opacity

- Some imagery tricks...

```
BufferedImage image = new BufferedImage(CANVAS3D_WIDTH,  
    CANVAS3D_HEIGHT, BufferedImage.TYPE_INT_RGB);  
Graphics2D g2 = image.getGraphics();  
g2.setClip(c3d.getBounds());  
getContentPane().paint(g2);  
g2.dispose();
```





Treat #2, Java3D and Opacity

- ...and oddly enough, it actually works

```
Background bg = new Background(new ImageComponent2D(  
    ImageComponent2D.FORMAT_RGB, image));
```

```
BoundingSphere bounds = new BoundingSphere();
```

```
bounds.setRadius(100.0);
```

```
bg.setApplicationBounds(bounds);
```

```
BranchGroup objRoot = new BranchGroup();
```

```
objRoot.addChild(bg);
```

```
SimpleUniverse u = new SimpleUniverse(c3d);
```

```
u.getViewingPlatform().setNominalViewingTransform();
```

```
u.addBranchGraph(objRoot);
```





Don't be shy, try anything wild you
have in mind*

* At least with graphical user interfaces





Q&A

Demos source code on
<http://jroller.com/page/gfx>

Help and information
romain.guy@mac.com

www.javapolis.com





**IMPROVE YOUR
PATTERNS AT
JAVAPOLIS 2005.**

**METROPOLIS ANTWERP,
DECEMBER 12 UNTIL 16.**

