Java SE on the Desktop: Extreme GUI Makeover

Sang Shin
Technology Evangelist
Sun Microsystems, Inc.
Learn to take advantage of advanced Java™ Foundation Classes (JFC/Swing) API and Java 2D™ API to solve problems and create visually-stunning applications
Agenda

• Chat Application
• Login Window
• Buddy List
• Chat Window
Chat Client
GAIM
MSN Messenger

A new version of MSN Messenger is now available. Click here for more information.

Salut Laure

salut romain

fais un beau sourire tu es prise en capture d'écran pour une conférence

ca va?
Login Window
Demo: Bad-Looking Login Window
Problems

• Practical
  > What is this?
  > Where is my account?
  > I am a frustrated power user @#!
  > Je suis un étranger laissé pour compte

• Visual
  > Poor alignment, inconsistent sizes
  > Windows XP? Not quite...

• And fun!
Practical: What is this?

• This is the first screen, so:
  > Recall the name of the application
  > Recall the purpose of the application
Practical: Where is my account?

• Sort accounts by name
• Use a ListCellRenderer:
  > Different colors for odd and even rows
  > Show the avatar of each account
  > While you’re at it... look nice
Practical: I am a power user!

• “I know how to use a keyboard”:
  > Provide button mnemonics
    ```java
    accountsButton.setMnemonic('A');
    ```
  > Make the login button the default one
    ```java
    frame.getRootPane().setDefaultButton(loginButton);
    ```
  > Give focus to the password field
    ```java
    passwordField.requestFocusInWindow();
    ```
Practical: I don't speak English!

• Use ResourceBundle and locales:

```java
// load MediumLogin_fr.properties,
MediumLogin_de.properties...

ResourceBundle.getBundle
("com/sun/java/chat/ui/login/MediumLogin")
new JButton(bundle.getString("accounts.button"))
```

• Think BIDI:
  > Do not use WEST, use LINE_START
  > Test your UI:

```
applyComponentOrientation
(ComponentOrientation.RIGHT_TO_LEFT)
```
Visual: Poor alignment, Inconsistent sizes

- Layout for alignment, insets for spacing
- Consistent sizes is a problem:
  - Set preferred sizes?
    - Difficult with L10N
  - Use filling layouts
    - BorderLayout
    - GridBagConstraints.fill
  - Use empty, borderless buttons
    - button.setBorderPainted(false);
    - button.setContentAreaFilled(false);
Visual: Windows XP? Not Quite...

• Use a custom PasswordView:

    passwordField.setUI(new BasicPasswordFieldUI() {
        public View create(Element elem) {
            return new PasswordViewXP(elem);
        }
    });

    private class PasswordViewXP extends PasswordView {
        protected int drawEchoCharacter(Graphics g, int x, int y, char c) {
            g.fillOval(x, y, width, width);
        }
    }
**Demo**: OK-Looking Login Window
Are We Done Yet?

• Targets most users
• Nice, clean and easy to use
• But... what about the **fun**?

Let’s Go Extreme!
Demo: Let's Go Extreme!
Demo: Let's Go Extreme!
How Can I Do This in Java Technology?

- Extensive use of the Java 2D API
  - Gradients
  - AlphaComposites
  - Hardware acceleration
- Custom JFC/Swing API-based components
- Custom layout
- Timers
Compositing Layers

• Introducing StackLayout
Compositing Layers: StackLayout

• Straightforward API

```java
JPanel pane = new JPanel();
pane.setLayout(new StackLayout());
pane.add(gradient, StackLayout.TOP);
pane.add(avatars, StackLayout.TOP);
pane.add(curves, StackLayout.TOP);
pane.add(buttons, StackLayout.TOP);
```
Drawing the Avatars

• Illusion of a 3-D scene
  > Reflected avatars on the “ground”

• Illusion of depth
  > The further, the smaller
  > The further, the less opaque
Drawing the Avatars: Illusion of a 3-D Scene

- Three steps
Drawing the Avatars: Illusion of a 3-D Scene

Image avatar = ...;
BufferedImage alphaMask = createGradientMask(avatarWidth, avatarHeight);
BufferedImage buffer = createReflection(avatar, avatarWidth, avatarHeight);
Graphics2D g2 = buffer.createGraphics();
g2.setComposite(AlphaComposite.DstOut);
g2.drawImage(alphaMask, null, 0, avatarHeight);
g2.dispose();
Drawing the Avatars: Illusion of Depth

• The middle avatar is the nearest
  > At position 0.0
• Avatars on the edges are the furthest
  > At positions –1.0 and +1.0
• Given the position we need
  > A size factor
  > An opacity factor
• \( f([-1.0, 1.0]) \rightarrow [0.0, 1.0] \) with \( f(0)=1.0 \)
Drawing the Avatars: Illusion of Depth

- We use a Gaussian distribution
  \[
  \frac{1}{\sigma} \cdot \sqrt{2\pi} \cdot e^{-\frac{x^2}{(2\sigma)^2}}
  \]
- We distribute the avatars on the curve
Buddy List
Demo: Bad-Looking Buddy List Window
Problems

- **Practical standpoint**
  - All buddies look the same as each other and groups
  - Can’t determine the status of each buddy
  - Difficult to rearrange the buddy list

- **Fun aspect**
  - It’s boring—we could make it much more fun!
Practical: Use Custom TreeCellRenderer

• A custom `TreeCellRenderer` can
  > Give groups a different look, to separate from buddies
  > Paint the appropriate buddy icon for each buddy
  > Show a second icon to indicate status
MediumTreeCellRenderer: Extends DefaultTreeCellRenderer

private User user = null;

public Component getTreeCellRendererComponent(...) {
    super.getTreeCellRendererComponent(...);
    if (value instanceof User) {
        user = (User)user;
    }
    setIcon(user != null ? user.getBuddyIcon() : null);
    return this;
}
public void paint(Graphics g, JComponent c) {
    if (user == null) {
        GradientPaint gp =
            new GradientPaint(0, 0, c1,
            0, getHeight(), c2, true);
        ((Graphics2D)g).setPaint(gp);
        g.fillRect(0, 0, getWidth(), getHeight());
    }
    super.paint(g, c);
    if (user != null) {
        Icon statusIcon = getStatusIcon(user.getStatus());
        icon.paintIcon(
            this, g,
            getWidth() - icon.getIconWidth(),
            (getHeight() - icon.getIconHeight()) / 2);
    }
}
Practical: Make It Easy To Rearrange Buddies via Drag & Drop

• Easily supported via JFC/Swing API Drag and Drop
  > Turn on dragging
    ```java
tree.setDragEnabled(true);
```
  > Set the drop mode (new in 6.0)
    ```java
tree.setDropMode(DropMode.INSERT);
```
  > Install a custom TransferHandler to manage nodes
    ```java
tree.setTransferHandler(new UserTransferHandler());
```
private DataFlavor USER_FLAVOR =
    new DataFlavor(DataFlavor.javaJVMLocalObjectMimeType + ";class=com.sun.java.chat.model.User");

public int getSourceActions(JComponent c) {return MOVE;}

public boolean canImport(TransferInfo info) {
    if (!info.isDataFlavorSupported(USER_FLAVOR)) {
        return false;
    }
    JTree.DropLocation loc =
        (JTree.DropLocation)info.getDropLocation();
    return loc.getPath().getLastPathComponent() != tree.getModel().getRoot();
}
public Transferable createTransferable(JComponent c) {
    TreePath path = tree.getSelectionPath();
    if (path == null) {
        return null;
    }

    Object obj = path.getLastPathComponent();
    if (obj instanceof User) {
        return new UserTransferable((User)obj);
    }

    return null;
}
public boolean importData(TransferInfo info) {
    if (!canImport(info)) {return false;}
    User u = (User)
        info.getTransferable().getTransferData(USER_FLAVOR);

    Group oldGroup = u.getParent();
    int oldIndex = oldGroup.indexOf(u);
    JTree.DropLocation loc = info.getDropLocation();
    TreePath path = loc.getPath();
    Group newGroup = (Group)path.getLastPathComponent();
    int newIndex = loc.getChildIndex();
    if (oldGroup == newGroup && oldIndex < newIndex) {
        newIndex--;
    }
    getGroupModel().remove(oldGroup, oldIndex);
    getGroupModel().add(newGroup, newIndex, u);
    return true;
}
Demo: OK-Looking Buddy List Now
The Fun Stuff: Make Buddy List Extreme!

- Round animated buttons
- Images to represent buddy entries
- More curves, less lines
- Custom DnD feedback
Demo: Let's Go Extreme!
Round Animated Buttons

```java
public CurvesButton(String text) {
    super(text);
    setContentAreaFilled(false);
    setBorderPainted(false);
    setBorder(BorderFactory.createCompoundBorder(
        getBorder(),
        BorderFactory.createEmptyBorder(2, 2, 2, 2)));
    addMouseListener(this);
}

public void mouseEntered(MouseEvent me) {
    startAnimation();
}

public void mouseExited(MouseEvent me) {
    stopAnimation();
}
```
public void paintComponent(Graphics g) {
    Graphics2D g2d = (Graphics2D)g.create();
    RoundRectangle2D.Float shape =
        new RoundRectangle2D.Float(0, 0, w - 1,
                                   h - 1, h - 1, h - 1);
    g2d.setClip(shape);

    // paint current step of animated background here,
    // clipped to the round shape

    super.paintComponent(g);
    g2d.setRenderingHint(KEY_ANTIALIASING,
                         VALUE_ANTIALIAS_ON);
    g2d.draw(shape);
    g2d.dispose();
}
Images to Represent Buddy Entries

- Same as before—use a custom `TreeCellRenderer`
- This time, paint the background with a small image, scaled to encapsulate the text (like GTK, Synth, etc.)
More Curves, Less Lines

- Nice curve painted on JMenuBar
  - Override paintComponent to paint the design
  - Ensure JMenus don't paint their background

- JTree shows connection between users and groups with curved lines
  - By way of a custom UI class
void drawCurve(Graphics2D g2d, Rectangle parBounds, 
    Rectangle chilBounds) {
    g2d.setRenderingHint(KEY_ANTIALIASING, 
        VALUE_ANTIALIAS_ON);
    int x1 = parBounds.x - tree.getInsets().left 
        - getRightChildIndent() + 1;
    int y1 = parBounds.y + parBounds.height;
    int x2 = chilBounds.x;
    int y2 = chilBounds.y + chilBounds.height / 2;
    int xc = x1, yc = y2;
    QuadCurve2D.Float curve = 
        new QuadCurve2D.Float(x1, y1, xc, yc, x2, y2);
    g2d.setPaint(new GradientPaint(x1, y1, col1, 
        x2, y1, col2, true));
    g2d.draw(curve);
}
Custom DnD Feedback

• TreeCellRenderer paints drag source with translucency
• TreeCellRenderer returns without painting target location—we want to paint it rotated
• CustomUI paints the item in the drop location rotated, as well as the item to be dropped
public void paint(Graphics g, JComponent c) {
    super.paint(g, c);

    // first paint the cell representing the item we're
    // dragging, offset to the right, with translucency

    // now find user at drop location and its bounds

    // paint it rotated
    g2d.rotate(Math.PI / 12.0, bounds.x,
               bounds.y + bounds.height / 2);
    Component comp = tree.getCellRenderer().
                    getTreeCellRendererComponent(...,user,...);
    rendererPane.paintComponent(g2d, comp, tree, bounds.x,
                                 bounds.y, bounds.width, bounds.height, true);
    g2d.dispose();
}
Chat Window
Demo: Bad-Looking Chat Window

![Image of a chat window with conversation history]

**Conversation History:**

- **(7:16:16 PM) Sang:** Hello
- **(7:16:17 PM) jackattack:** :) Hi. What seems to be your problem?
- **(7:16:24 PM) Sang:** What is going on?
- **(7:16:25 PM) jackattack:** ;) Why do you ask?
- **(7:16:35 PM) Sang:** Are you in Singapore?
- **(7:16:37 PM) jackattack:** Why are you interested in whether I am in Singapore or not?
Problems

- Hard to visually tell who said what
- Time stamp on each line often unnecessary
- Multiple chat sessions require lots of desktop space and continuous window switching
- Most users don’t use formatting buttons
Improving the Chat Window: Fun with JTextPane

- Only show time stamp for most recent message
- Align text to make each message’s origin clear
- Provide a cartoon-like bubble around each message
- Smoothly scroll new messages into view
Extreme Chat Window: Implementation Details

- Custom **JTextPane**
  - Custom element structure used to customize rendering
  - Custom views to render and align text and time-stamp components
- Component to render and animate time stamp
- Custom viewport for smooth scrolling
- Animation effects for showing and hiding formatting buttons
- Custom components for tabs provide status of each conversation
Rendering Time Stamp

• Custom JComponent
  > Draws background and text
  > Uses timing framework to determine frame to render
  > When timer updates, invokes repaint

• Embedded in JTextPane
  > Positioned by custom Views
Timing Framework

- Open source project on java.net
- Useful for building animations
- Calls back at specified frame rate
  - Reports percentage complete
  - Can run animation forward and backward
- http://timingframework.dev.java.net
public void timingEvent(long cycleTime, long totalTime, float percent) {
    if (totalTime > ANIMATE_TIME && percent < .5) {
        // Switch to next time
        this.timeString = nextTimeString;
    }
    this.percent = (1 - percent) * 2 - 1;
    repaint();
}
protected void paintComponent(Graphics g) {
    FontMetrics metrics = g.getFontMetrics();
    int timeWidth = metrics.stringWidth(timeString);
    int timeX = (getWidth() - timeWidth) / 2;
    Graphics2D g2d = (Graphics2D)g.create();
    g2d.translate(timeX, getInsets().top +
                   metrics.getHeight() / 2);
    g2d.scale(1, percent);
    g2d.drawString(timeString, 0, metrics.getAscent() -
                   metrics.getHeight() / 2);
    g2d.dispose();
}
Smooth Scrolling

- Determine how far to scroll
- Start timer using timing framework
- When timer updates call scrollRectToVisible
- Stops timer if user initiates scroll
- Stops timer if view size changes
public void smoothScrollRectToVisible(Rectangle rect) {
    initialWidth = component.getWidth();
    initialHeight = component.getHeight();
    deltaX = calculateHScroll(rect);
    deltaY = calculateYScroll(rect);
    stopScrolling();
    if (deltaX != 0 || deltaY != 0) {
        Point position = getViewPosition();
        initialX = position.x;
        initialY = position.y;
        startScrolling();
    }
}
public void timingEvent(long cycleTime, long totalTime,
        float percent) {
    smoothScroll(percent);
    if (percent == 1.0) {
        stopScrolling();
    }
}

private void smoothScroll(float percent) {
    int x = (int)(deltaX * percent) + initialX;
    int y = (int)(deltaY * percent) + initialY;
    super.setViewPosition(new Point(x, y));
}
Multiple Sessions in a Window

• New API for specifying component used to render a tab
  > setTabComponentAt()

• Component provides button that will close tab

• Visual indicator for status of buddy talking with

• Renders buddy icon
Demo: Let's Go Extreme!
Performance Tips
Giving the Wrong Impression

- Slow startup
  - No visual feedback
- Unresponsive GUI
  - Gray rectangle
- Slow graphics rendering
- Large memory footprint
- Wrong impression – Swing applications are slow
  - Nobody ever blames the developer!
Startup Improvements

• Splash screen
  > Continue loading in another thread
  > Use progress bar to indicate the state

• Preload data while waiting for user input
  > Preload classes
  > Load common resources eg. images
    > Avoid loading items that you don't immediately need

• Determine the heap size
  > Size your application and set -Xms and -Xmx

• Experiment with the various GC and VM options
GUI Responsiveness – 1

• Use models for Swing widgets eg. JTable, JTree, JComboBox, etc
  > NEVER use the default model
  > Better control over data that is displayed

• Paint less
  > Use clips
  > If possible, paint only the dirty region
  > Do not draw directly to the Graphics object in paint() method
GUI Responsiveness – 2

• Use proper scrolling mode
  > `JViewport.BLIT_SCROLL_MODE`

• Custom renderers must be quick and slim

• Dispatch event handling to an event processing thread
  > Use `SwingWorker`

• Provide visual feedback to long running task eg. downloads
  > Allow users to cancel the task
Rendering Performance

• Use newer image APIs
  > `BufferedImage` and `ImageIO`

• Beware of performance implications
  > Antialiasing and compositing may be slow
  > Consider pre rendered images
  > Cached hard to create/expensive images

• Batch Java2D primitives before flushing to screen
Summary
Summary

- Understand your applications and the target audience
- Take advantage of Swing and the powerful Java2D API to eliminate usability and visual problems
- Use your imagination. Make your application fun
- Understand what you are using could have tremendous impact on the performance of your application
Java SE on the Desktop: Extreme GUI Makeover

Sang Shin
Technology Evangelist
Sun Microsystems, Inc.