Programming Languages and Techniques (CIS120)

Lecture 32
April 9, 2018

Swing I: Drawing and Event Handling
Chapter 29
Announcements

• HW8: Spellchecker
  – Available on the web site
  – Due: Tuesday!
  – Parsing, working with I/O, more practice with collections

• Final exam: May 2, 3-5PM
  – If you have two finals at the same time, you can postpone one
  – If you have 3 finals on one day, you may postpone the middle one
  – If either case applies to you, add your name to the form on the website
  – Must discuss all other exam conflicts with course instructors
Game project

- **Game Design Proposal Milestone Due:** (12 points)
  - Friday April 13\(^{\text{th}}\) at NOON!!!!
  - (Should take about 1 hour)
  - Submit on GRADESCOPE
  - TAs will give you feedback over the weekend

- **Final Program Due:** (88 points)
  - Wednesday, April 25\(^{\text{th}}\) at 11:59pm
  - Submit zipfile online, submission *only* checks if your code compiles
  - Eclipse is STRONGLY recommended for this project
  - May distribute your game (after the deadline) if you do not use any of our code

- **Grade based on demo with your TA during reading days**
  - Grading rubric on the assignment website
  - Recommendation: don’t be too ambitious.

- **NO LATE SUBMISSIONS PERMITTED**
HW9: Game Project
Swing

Java's GUI library
Quiz

Have you ever used the Swing library to build a Java app before?

1. Nope
2. No, but I've used a different GUI library in Java
3. Yes, but I didn't really understand how it worked
4. Yes, I'm an expert
Why study GUlIs (yet again)?

- Most common example of *event-based programming*
- Heavy and effective use of OO inheritance
- Case study in library organization
  - and some advanced Java features
- Ideas applicable everywhere:
  - Web apps
  - Mobile apps
  - Desktop apps
- Fun!
### Terminology overview

<table>
<thead>
<tr>
<th></th>
<th>GUI (OCaml)</th>
<th>Swing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics Context</td>
<td>Gctx.gctx</td>
<td>Graphics</td>
</tr>
<tr>
<td>Widget type</td>
<td>Widget.widget</td>
<td>JComponent</td>
</tr>
<tr>
<td>Basic Widgets</td>
<td>button, label, checkbox</td>
<td>JButton, JLabel, JCheckBox</td>
</tr>
<tr>
<td>Container Widgets</td>
<td>hpair, vpair</td>
<td>JPanel, Layouts</td>
</tr>
<tr>
<td>Events</td>
<td>event</td>
<td>MouseEvent, KeyEvent</td>
</tr>
<tr>
<td>Event Listener</td>
<td>mouse_listener, mouseclick_listener (any function of type event -&gt; unit)</td>
<td>ActionListener, MouseListener, KeyListener</td>
</tr>
</tbody>
</table>
Swing practicalities

• Java library for GUI development
  – javax.swing.*

• Built on existing library: AWT
  – java.awt.*
  – When there are two versions of something, use Swing’s.
    (e.g., java.awt.Button vs. javax.swing.JButton)
    • The “Jxxx” version is usually the one you want, rather than “xxx”.

• Portable
  – Communicates with underlying OS's native window system
  – Same Java program looks appropriately different when run on PC, Linux, and Mac
Simple Drawing

DrawingCanvas.java
DrawingCanvasMain.java
Fractal Drawing Demo

[Image of a fractal drawing]
Recursive function for drawing

private static void fractal(Graphics gc, int x, int y,
    double angle, double len) {

    if (len > 1) {
        double af = (angle * Math.PI) / 180.0;
        int nx = x + (int)(len * Math.cos(af));
        int ny = y + (int)(len * Math.sin(af));

        gc.drawLine(x, y, nx, ny);

        fractal(gc, nx, ny, angle + 20, len - 8);
        fractal(gc, nx, ny, angle - 10, len - 8);
    }
}
How do we draw a picture?

- In the OCaml GUI HW, we created widgets whose `repaInt` function used the graphics context to draw an image

```ocaml
let w_draw : widget = 
{
  repaint = (fun (gc:gctx) ->
             fractal (with_color gc green)
              200 450 270 80);

  size    = (fun () -> (200,200));

  handle  = (fun () -> ())
}
```

- In Swing, we *extend* from class JComponent ...
Fundamental class: JComponent

• Analogue to widget type from GUI project
  – (Terminology: widget == JComponent)

• Subclasses override methods
  – paintComponent (like repaint, displays the component)
  – getPreferredSize (like size, calculates the size of the component)
  – Events handled by listeners (don't need to use overriding...)

• Much more functionality available
  – minimum/maximum size
  – font
  – foreground/background color
  – borders
  – what is visible
  – many more...
public class DrawingCanvas extends JComponent {

    public void paintComponent(Graphics gc) {
        super.paintComponent(gc);

        // set the pen color
        gc.setColor(Color.GREEN);

        // draw a fractal tree
        fractal(gc, 200, 450, 270, 80);
    }

    // get the size of the drawing panel
    public Dimension getPreferredSize() {
        return new Dimension(200, 200);
    }
}

How to display this component?
JFrame

• Represents a top-level window
  – Displayed directly by OS (looks different on Mac, PC, etc.)
• Contains JComponents
• Can be moved, resized, iconified, closed

```java
public void run() {
    JFrame frame = new JFrame("Tree");

    // set the content of the window to be the drawing
    frame.getContentPane().add(new DrawingCanvas());

    // make sure the application exits when the frame closes
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // resize the frame based on the size of the panel
    frame.pack();

    // show the frame
    frame.setVisible(true);
}
```
User Interaction
Task: Program an application that displays a button. When the button is pressed, it toggles a “lightbulb” on and off.

Key idea: use a ButtonListener to toggle the state of the "lightbulb"
OnOffDemo

The Lightswitch GUI program in Swing.
class LightBulb extends JComponent {
    private boolean isOn = false;

    public void flip() {
        isOn = !isOn;
    }

    public void paintComponent(Graphics gc) {
        if (isOn) {
            gc.setColor(Color.YELLOW);
        } else {
            gc.setColor(Color.BLACK);
        }
        gc.fillRect(0, 0, 100, 100);
    }

    public Dimension getPreferredSize() {
        return new Dimension(100, 100);
    }
}

Remember the private state of the lightbulb

Draw the Light bulb here using the graphics context

Set the size of the window
public class OnOff implements Runnable {
    public void run() {
        JFrame frame = new JFrame("On/Off Switch");
        JPanel panel = new JPanel();
        frame.getContentPane().add(panel);
        LightBulb bulb = new LightBulb();
        panel.add(bulb);
        JButton button = new JButton("On/Off");
        panel.add(button);
        button.addActionListener(new ButtonListener(bulb));
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new OnOff());
    }
}
class ButtonListener implements ActionListener {
    private LightBulb bulb;
    public ButtonListener (LightBulb b) {
        bulb = b;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        bulb.flip();
        bulb.repaint();
    }
}