Programming Languages and Techniques (CIS120)

Lecture 33

April 11, 2016

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
- HW9: Game project (details coming Wednesday!)
  - Strongly encouraged to design your **own** game
- If you need to reschedule the Final exam, see me

Swing

Java's GUI library

Quiz

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

1. No
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

Quiz

Do you remember how the OCaml GUI library from HW 5 worked?

1. What OCaml GUI library?
2. There was something about widgets and value_controllers, right?
3. I think I could remember how it works, given prompting
4. I could recreate it all right now

Why study GUIs (yet again)?

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

Terminology overview

	GUI (OCaml)	Swing
Graphics Context	Gctx.gctx	Graphics
Widget type	Widget.widget	JComponent
Basic Widgets	button label checkbox	JButton JLabel JCheckBox
Container Widgets	hpair, vpair	JPanel, Layouts
Events	event	ActionEvent MouseEvent KeyEvent
Event Listener	mouse_listener mouseclick_listener (any function of type event -> unit)	ActionListener MouseListener KeyListener

Swing practicalities

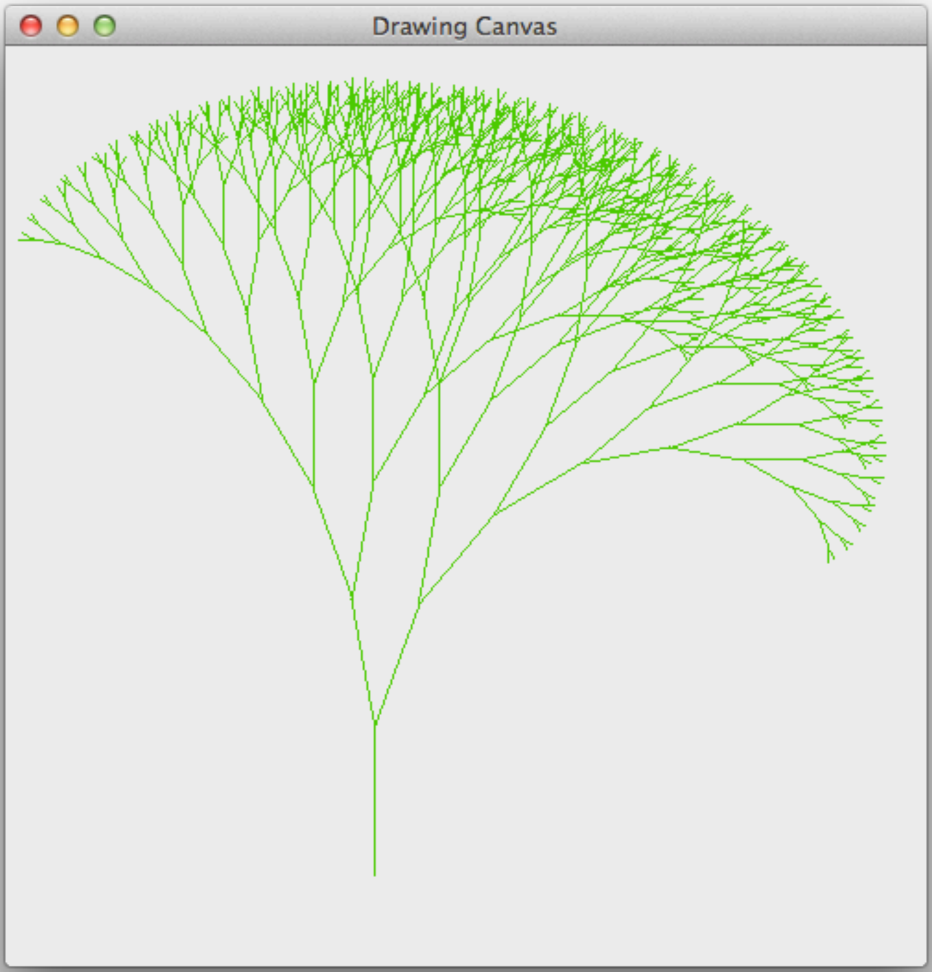
- Java library for GUI development
  - javax.swing.*
- Built on existing library: AWT
  - java.awt.*
  - If there are two versions of something, use Swing's. (e.g., java.awt.Button vs. javax.swing.JButton)
    - The "Jxxx" version is usually the one you want, rather than "xxx".
- Portable
  - Communicates with OS's native window system
  - Same Java program looks different when run on PC, Linux and Mac

Simple Drawing

DrawingCanvas.java

DrawingCanvasMain.java

Fractal Drawing Demo



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 OCaml GUI HW, create a widget where the repaint function uses the graphics context to draw an image

```
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, *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…

Simple Drawing Component

```java
public class DrawingCanvas extends JComponent {

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

        // set the pen color to green
        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

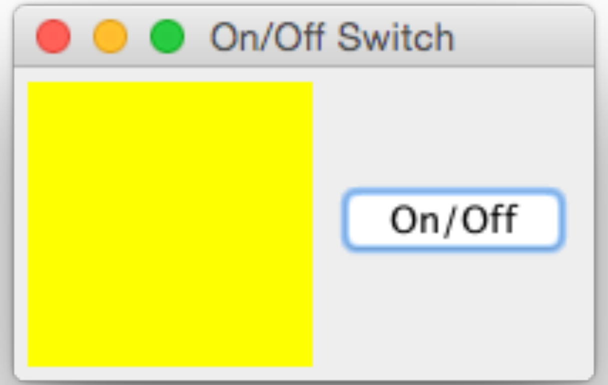
- Represents a top-level window
  - Displayed directly by OS (looks different on Mac, PC, etc.)
- Contains JComponents
- Can be moved, resized, iconified, closed

```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

Start Simple: Lightswitch Revisited

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".