Programming Languages and Techniques (CIS120)

Lecture 33

April 16, 2014

Swing I: Drawing and Event Handling

```
Set<String> set = new TreeSet<String> ();
Map<String,Set<String>> map =
        new TreeMap<String,Set<String>> ();
set.add("1");
set.add("2");
map.put("a", set);
set.clear(); // remove all elements
set.add("3");
map.put("b",set);
System.out.println(map);
```

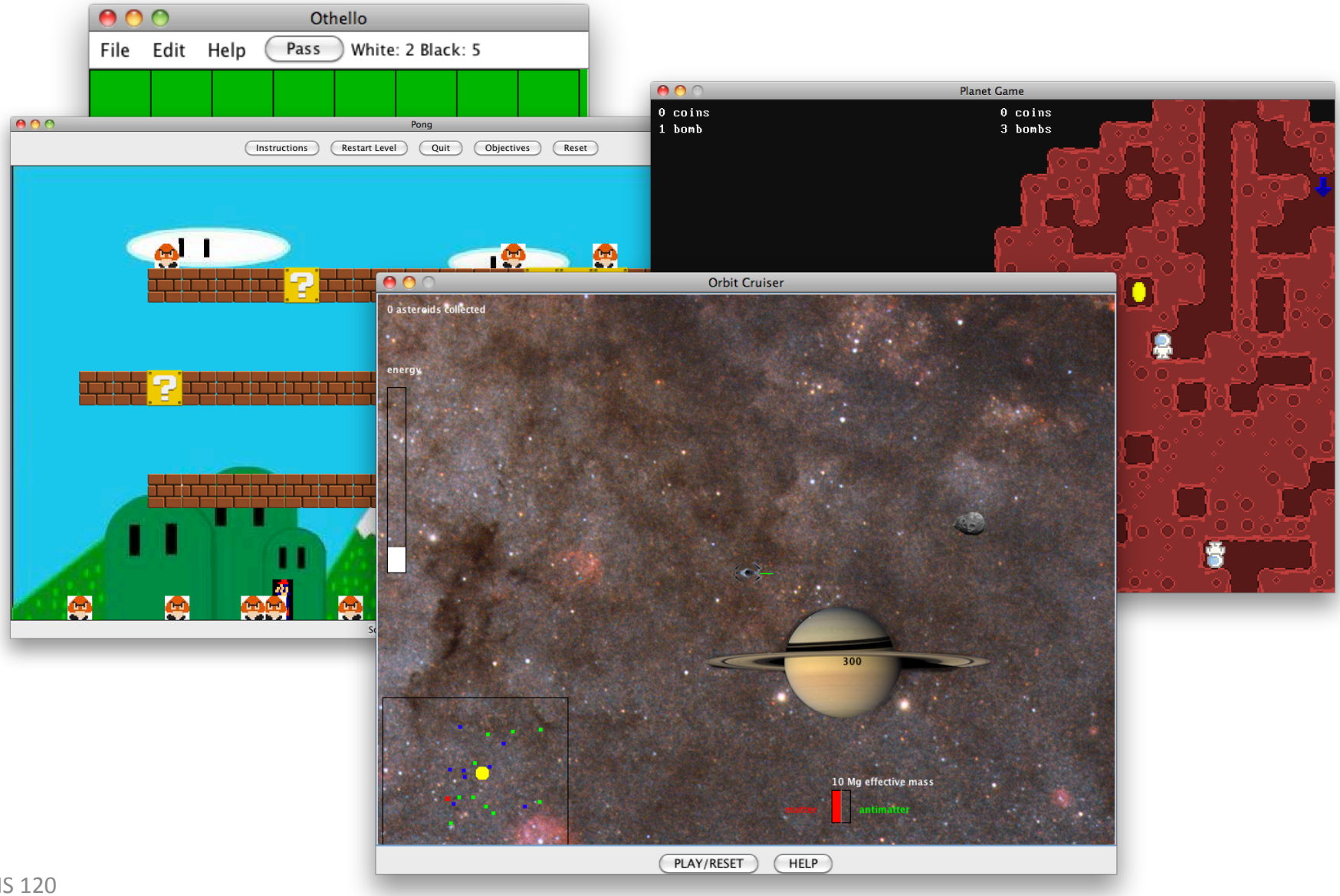
What result is printed?

1. { a=[1,2], b=[3] }
2. { a=[3], b=[3] }
3. { a=[1,2], b=[1,2] }
4. NullPointerException

HW 08 Feedback

- Good
  - I liked this homework.
  - I actually liked working on this assignment a lot, I'm definitely looking forward to more GUI-related assignments, makes the stuff we work on seem so much cooler than simply computing values.
  - Satisfying end result. Not nearly as difficult as paint in OCaml.
- Not so good
  - HELL WEEK. SLEEP DEPRIVATION. I DONT CARE ANYMORE. JSHDLKSFGKGFDHDFG
  - :(
- Brief
  - Fun
  - Wow
  - Cool!
  - ha
- Insightful
  - Found a bug when I was showing off to my friends that I finally finished, good thing I had one submission left.
  - Actually took the time to understand Gctx and Notifiers this time around instead of blindly testing things when we did GUI in OCaml... much easier and faster when you understand why you are doing the things you are doing!
- ????
  - Who needs a TI-83 when you have Calculator.java?!
  - please show this comment in class

HW 10: Game project



Announcements

- HW10 is now available. Due Wednesday, April 30th, at 11:59PM.

- Suggested steps:
  - Get started early!
  - Download, run, and read the code we've provided
  - Play around with adding a few small features
  - Read over our suggestions and think about what game you'd like to try to code
  - Discuss with your TAs
  - Do it

Do you have an idea for a game already?

1. no, not yet ☺
2. I have a few ideas
3. I know what I want to do
4. I started coding back in January

GUIs, take 3…

OCaml GUI review

- Graphics Context
  - Provides drawing operations
  - Translates coordinates so that they are relative to each widget
  - Keeps track of state necessary for drawing (pen color, line thickness)
- Widgets
  - Abstract type for "things" on the screen
  - Something that can paint itself, handle events and calculate its size

```
type widget = {  repaint : gctx -> unit,
                 size    : gctx -> int,
                 handle  : event -> unit  }
```

  - basic widgets: buttons, canvas, scrollbars, labels, checkboxes, radiobuttons
  - container widgets: border, hpair, vpair, hlist, vlist, grid
- Event Listeners
  - Functions that execute when events happen
  - Update the state of the application
  - Widgets reflect changes when they are redrawn

Terminology overview

	HW06 (OCaml)	HW08 (Java)	Swing
Graphics Context	Gctx.gctx	Gctx	Graphics
Widget type	Widget.widget	Widget	JComponent
Basic Widgets	button label checkbox	Button Label …	JButton JLabel JCheckBox
Container Widgets	hpair, vpair	HPair, VPair	JPanel, Layouts
Events	event	Event	ActionEvent MouseEvent KeyEvent
Event Listener	mouse_listener mouseclick_listener (any function of type event -> unit)	EventListener	ActionListener MouseListener KeyListener

Simple Drawing

DrawingCanvas.java

DrawingCanvasMain.java

How do we draw a picture?

- In HW06/08, create a widget where the repaint function uses the graphics context to draw an image

```
let w_draw =
{
   repaint = (fun (gc:Gctx.t) ->
       Gctx.draw_line gc (0, 0) (100, 100);
       Gctx.draw_point gc (3,4)) ;

   size    = (fun (gc:Gctx.t) -> (200,200));

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

- In Swing, extend from class JComponent….

Fundamental class: JComponent

- Analogue to Widget.widget and Widget class
  - (*Terminology*: widget == component)
- Subclasses *override* methods
  - paintComponent (like repaint, displays the component)
  - getPreferredSize (like minSize, calculates the size of the component)
  - Events handled by subclasses
- Much more functionality available
  - minimum/maximum size
  - font
  - foreground/background color
  - borders
  - what is visible
  - many more…

Simple Drawing Component

```
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, 75, 100, 270, 15);
    }

    // get the size of the drawing panel
    public Dimension getPreferredSize() {
        return new Dimension(150,150);
    }
```

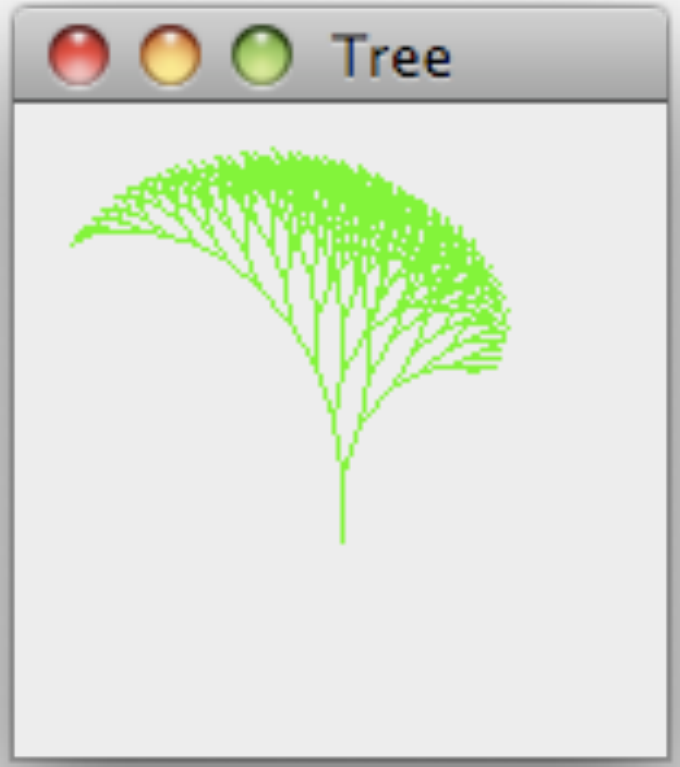
How to display this component?

JFrame

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

```
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);
}
```

Fractal Drawing Demo



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.

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
- Components as Containers
  - Use JPanel to organize and position other components (analogous to vpair, hpair)

OnOffDemo

The Lightswitch GUI program in Swing.