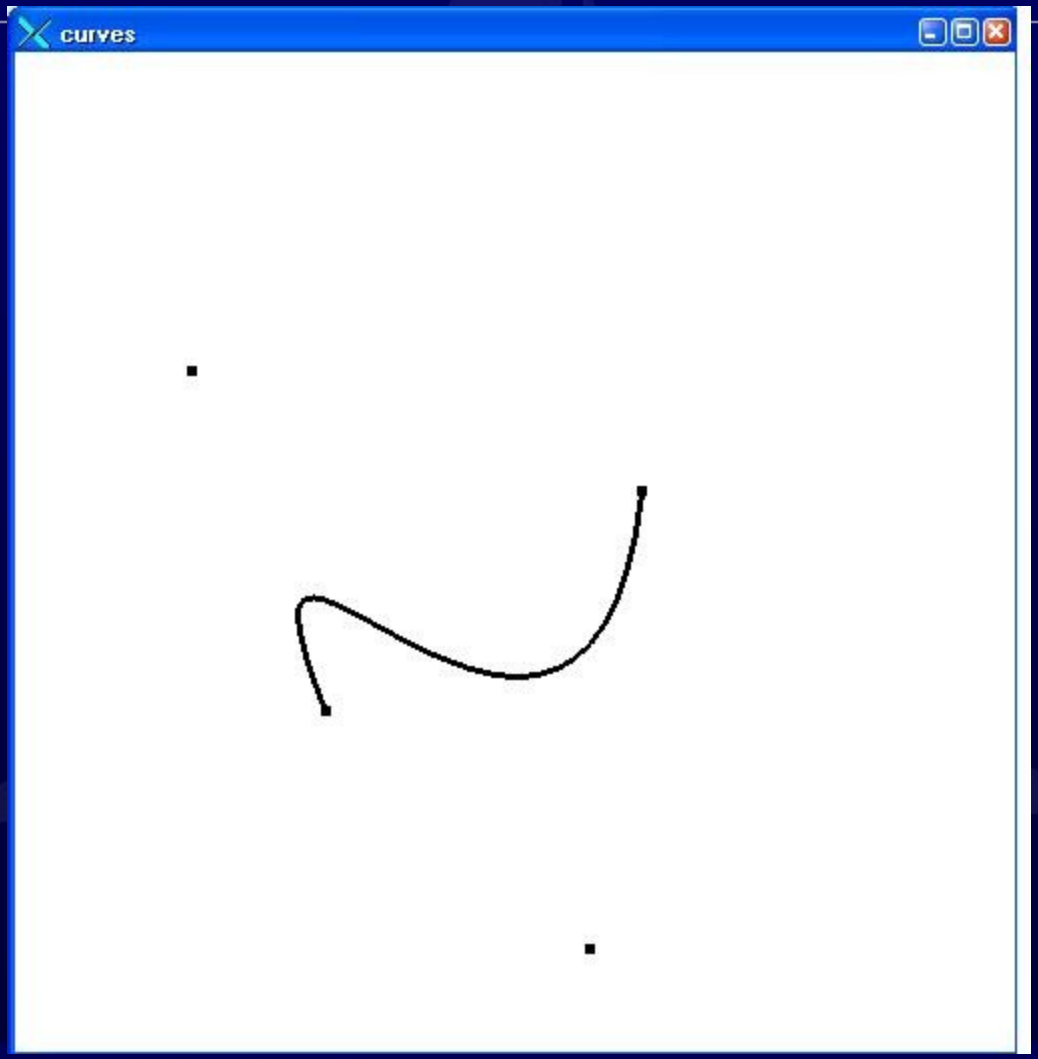
Event-Driven Programming

Appendix C

bézier



Objectives

- To understand how event-driven programming differs from request-response programming.
- To learn how to design and write event-driven programs using the Java graphics API.
- To learn about different kinds of events that can occur.
- To become familiar with the structure of the Java Swing API, including the class hierarchy, user interaction components, and ways to arrange the components.
- To learn about mouse events and how to draw figures with the mouse.

Request-Response

- (This chapter refers back to chapter 1 in places.) See Figure 1.6 (not 1.4).
- From Figure C.1, page 763:

```
do{
    choice=JOtionPane.showOptionDialog(…);
    switch(choice) {
        case 0:  doAddChangeEntry( ); break;
        case 1:  doLookupEntry( ); break;
        … }; } while (choice < commands.length-1);
```

Comparison doLookupEntry

- Listing 1.6, page 47

```
private void doLookupEntry ( ) {
String theName =
JOptionPane.showInputDialog(...);
if(theName == null) { return }
String theNumber =
theDirectory.lookupEntry(theName);
String message = null;
if(theNumber != null {//store num
} else { /* store error */ }
JOptionPane.showMessageDialog(
   n    essage);
}
```

- Example C.1, page 765

```
private class DoLookupEntry
    implements ActionListener {
public void actionPerformed (
    ActionEvent e) {
String theName =
JOptionPane.showInputDialog( ... );
//check the name
String theNumber =
theDirectory.LookupEntry(theName);
// store message or error
// display message
```

What Changed?

- Changed method name from doLookupEntry to actionPerformed.
- Enclosed actionPerformed in a class named doLookupEntry.
- Class doLookupEntry implements the ActionListener interface (see page 17).
- Listing C.1 shows surrounding code.

Another Comparison

- Batch version

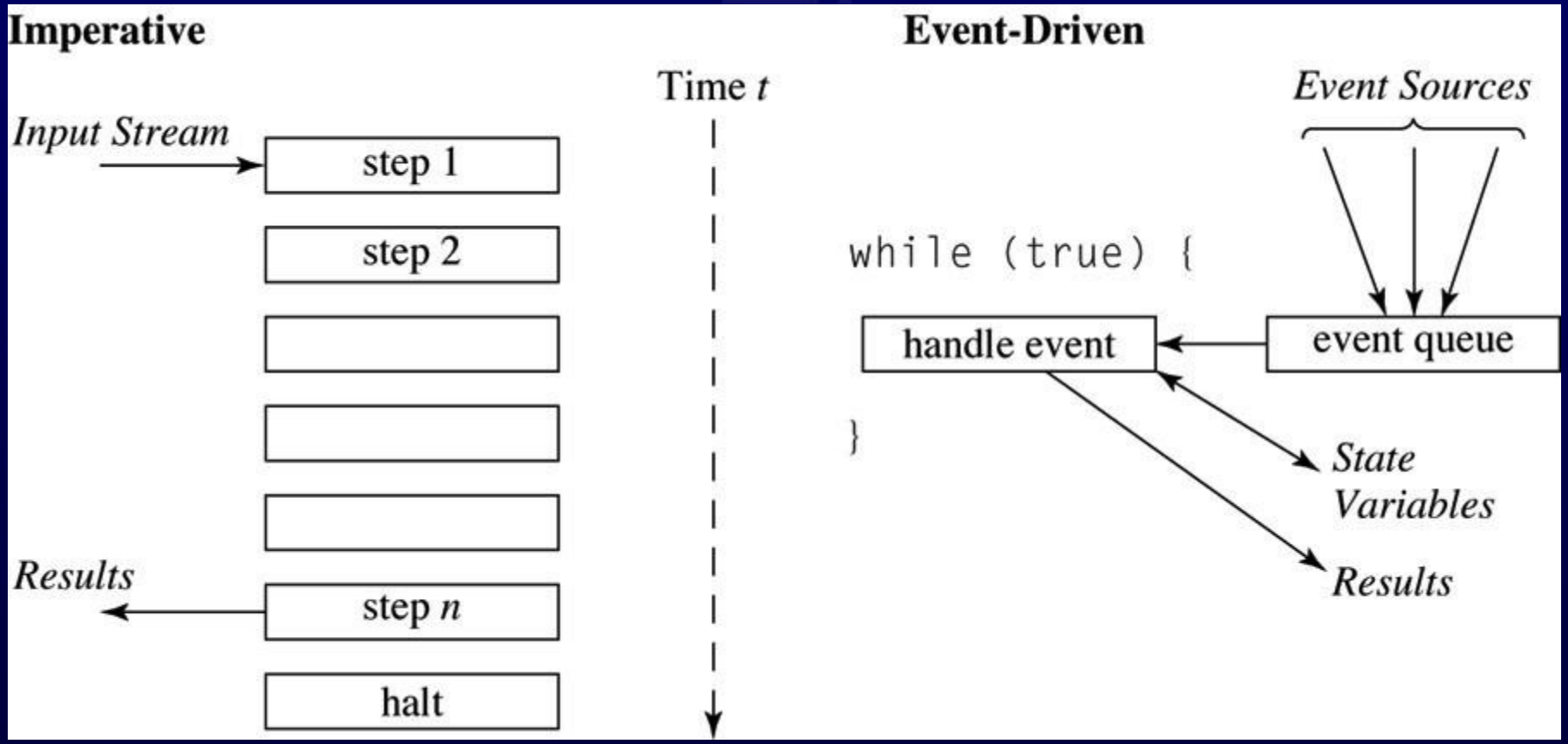
read a number (from the keyboard), store in A[0]
read a number (from the keyboard), store in A[1]
print A[0] + A[1]

- Event-driven version

set counter K to 0
repeat {
if a number has been entered (from the keyboard) { store in A[K] and increment K
if K equals 2 print A[0] + A[1] and reset K to 0}
}

- Why? Easier to generalize the event-driven version.

A Diagram



An Infinite Loop???

- Yes!
- An example from CS425 OpenGL programming.
  - Set Up Your Scene.
  - Define all your "callback" functions.
  - Clear the screen.
  - Force one expose event.
  - Go into wait loop.
- So, how does it ever quit?
  - Stop event. (button or key combination)

Event Listeners

- When an event occurs, the button clicked will call each of the *event listeners* that are registered for the event.
- The Hollywood Principle: "Don't call us; we'll call you." … You implement the interfaces, you get registered. You get called when the time is right. This requires a distinctly different way of thinking . Dafyd Rees
- The main flow of your program is not sequential from beginning to end. If you've never done GUI programming, this is one of the trickiest paradigm shifts. (Robin Dunn, OSCON 2004)

Event Listeners (cont.)

- The process by which an event listener is associated with an event is called registering the listener for the event.
- This is done by a method named *addEventTypeListener* where *EventType* is the type of the event.
  - E.g. Action listeners are registered by calling the method addActionListener.
- Firing an event is when the component determines that an event has occurred and calls the registered event listeners.

The ActionListener Interface

- The ActionListener interface (java.awt.event) declares just one method, actionPerformed.

```
Interface ActionListener extends EventListener
    { void actionPerformed(ActionEvent e); }
```

Registering an Event Listener

- A button is an object of the class JButton.
- This class has a method addActionListener through superclass AbstractButton (see fig C.3).
- You register the action listener by:

```
lookupEntryButton.addActionListener ( new
    DoLookupEntry ( ) );
```

Creating a User Interface

- First take a look at TwoCircles example.
- All guis are displayed within a window.
- Java GUI API defines 3:
  - frames
  - dialogs
  - applets
- Only frames can stand alone. Implemented by the Frame class (java.awt) or Jframe class (javax.swing)
- Text will use swing whenever possible.

PDButtonUI Example (listing C.1)

- Begin with

```
public class PDButtonUI extends JFrame
                        implements PhoneDirectoryUI {
```

- Constructor builds the contents of the frame.

```
super ("Phone Directory"); // title on top of frame
addWindowListener(new WindowClosing ( ) );
Jpanel panel=new JPanel( ); // add panel container
JButton lookupEntryButton=new JButton("Look Up
    Entry");
lookupEntryButton.addActionListener(new
    DoLookupEntry ( ) );
Panel.add(lookupEntryButton); // repeat 3 for each
    button
```

PDBuuttonUI (cont.)

- Panel is added to the content pane:

```
getContentPane( ).add(panel);
pack( ); //size the frame to hold the panel.
```

- processCommands method:

```
public void processCommands(PhoneDirectory
    thePhoneDirectory) {
    theDirectory = thePhoneDirectory;
    show ( );
   }
```

- Results are shown in Figure C.2, listing is C.1

AWT and Swing Hierarchy

- Refer to Figure C.3
- Figure C.4 shows an empty JFrame.
- Go over Two Circles code. (relate to Figure C.6 the containment hierarchy)
- Be sure to review and have ready as a reference:
  - JFrame
  - JPanel
  - All the methods of Graphics

JFrame

- Constant: EXIT_ON_CLOSE
- Connstructor(s)
  - JFrame( )
  - JFrame(String title)
- Methods
    - getContentPane ( )
    - setDefaultCloseOperation (…)
    - setContentPane (…)
    - setTitleString (…)
    - setMenuBar (…)
    - setSize (…)
    - pack ( ) and show ( )

JPanel

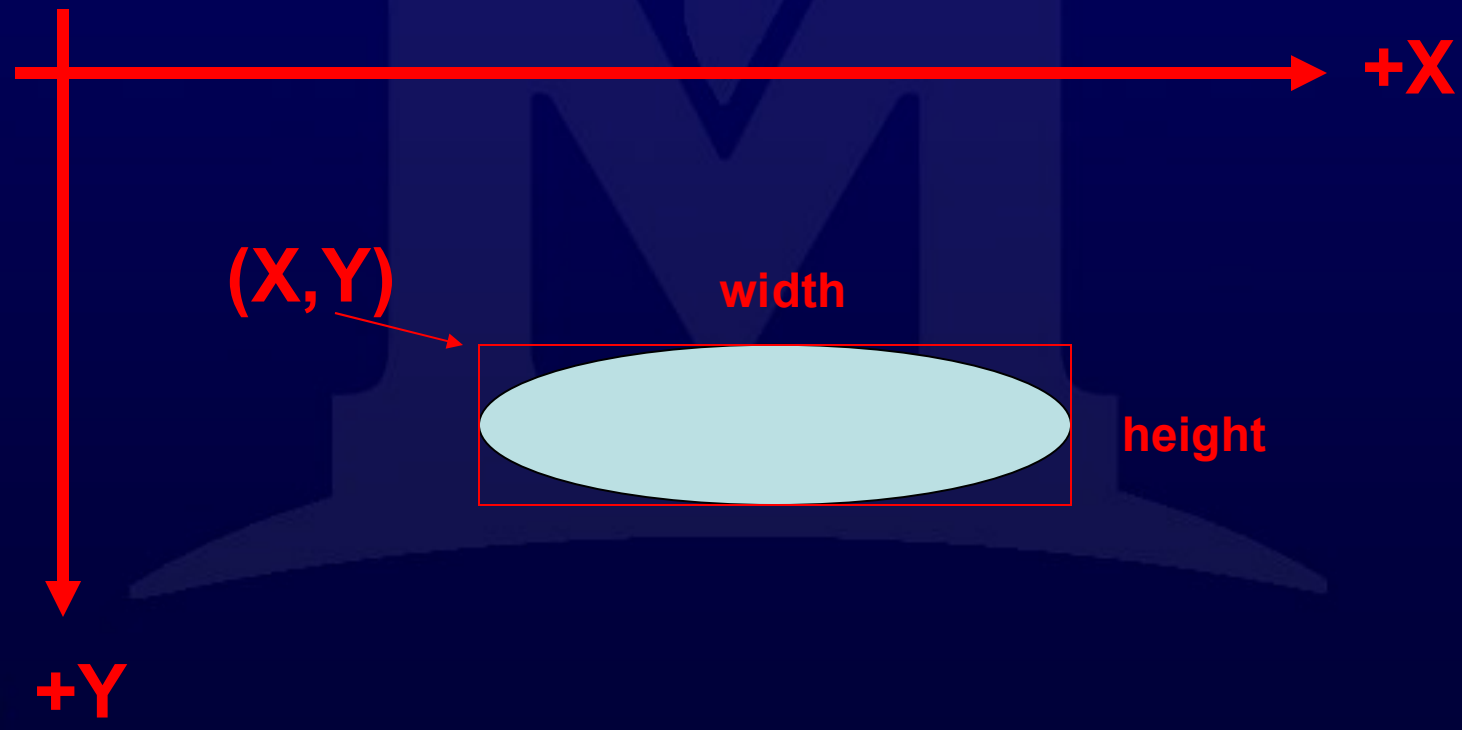
- Constructor(s)
  - JPanel ( )
  - JPanel (LayoutManager layout)
- Methods
  - add (Component c)
  - add (Component c, Object pos)
  - paintComponent(Graphics g)
  - repaint( )

Graphics

- drawLine
- drawOval
- drawPolygon
- drawRect
- fillOval
- fillPolygon
- fillRect
- getColor
- setColor
- setPaintMode
- setXORMode

fillOval(X,Y,width,height) See Figure C.7

- fillOval(x,y,width,height);



Two Circles (press On/Off)

Two Circles (Resized!)