

Graphical User Interfaces

The image illustrates the integration of graphical user interface design and data analysis. On the left, a design tool (likely Xcode) shows a 'System Browser' with various classes like 'Interval', 'LinkedList', and 'MappedCollection'. A code editor displays Objective-C code for a 'collect' method and a 'controlActivity' class. A diagram of a bolt is labeled 'Fig. 1'. On the right, a Microsoft Excel spreadsheet titled 'Contoso Expenses' features a pie chart showing the distribution of expenses across categories: Other, Taxes, Travel, Freelancers, Marketing, and Equipment. Below the chart is a table with columns for Category, 2008, 2009, 2010, 2011, and 2012, showing expense amounts for each category over time.

Category	2008	2009	2010	2011	2012
Rent and Utilities	\$ 18,840	\$ 17,628	\$ 16,368	\$ 18,000	\$ 19,000
Equipment	\$ 3,000	\$ 3,972	\$ 3,588	\$ 3,996	\$ 4,000
Marketing	\$ 5,556	\$ 5,424	\$ 5,784	\$ 5,904	\$ 6,000
Freelancers	\$ 5,604	\$ 5,556	\$ 5,700	\$ 5,568	\$ 5,600
Other	\$ 45,000	\$ 44,500	\$ 44,000	\$ 43,500	\$ 43,000



M. I. T.
LINCOLN LABORATORY
PRESENTS

Ivan Sutherland: "Sketchpad", <https://youtu.be/57wj8diYpgY>

The Xerox Star GUI, 1981

XEROX 6085 Workstation
User-Interface Design

To make it easy to compose text and graphics, to do electronic filing, printing, and making all at the one workstation, requires a revolutionary user interface design.

Bit-map display - Each of the pixels on the TV screen is mapped to a bit in memory; thus, arbitrarily complex images can be displayed. The 6085 displays all text and graphics as they will be printed. In addition, familiar office objects such as documents, folders, file drawers and lockers are portrayed as recognizable images.

The mouse - A unique pointing device that allows the user to quickly select any text, graphic or office object on the display.

See and Point
All functions are visible to the user on the keyboard or on the screen. The user does filing and retrieval by selecting them with the mouse and typing the MOVE, COPY, DELETE or PASTE command keys. Text and graphics are edited with the same keys.

Shorter Production Times
Experience at Xerox with prototype work stations has shown shorter production times and that lower costs, as a function of the percentage of use of the workstation. The following equation can be used to express that

Table 1: Percentages of use of the methods.

Year	See	Point
1978	65.2	15.0
1980	41.1	39.9
1982	85	35
1984	30	73
1985	10	93
1988	5	75

Figure 7: Data from Table 1 above

Workstation usage percentage Table 1 and illustrated in Figure 6085 users are likely to do the composition and layout, master piece or including printing and d

Text and Graphics
To replace typewriting, the 6085 offers a choice of type faces and sizes, from 6 point to 36 point:
Here is a sample of 6 point text.
Here is a sample of 12 point text.
18-point text.
24-point text.
36-point text.

DOC & Lotus data

NAME	EXTENSION	SIZE	DATE
COMMAND	COM	20677	15-11
APPS1	DVS	2556	18-11
ASSIGN	COM	864	28-11
ATTEND	EXE	15091	18-11
BACKUP	COM	17024	28-11
CHINDSK	COM	9476	24-11
CHIAOD	COM	6528	27-11
COMP	COM	3016	10-11
CEBUD	EXE	45364	15-11

Xerox and Apple

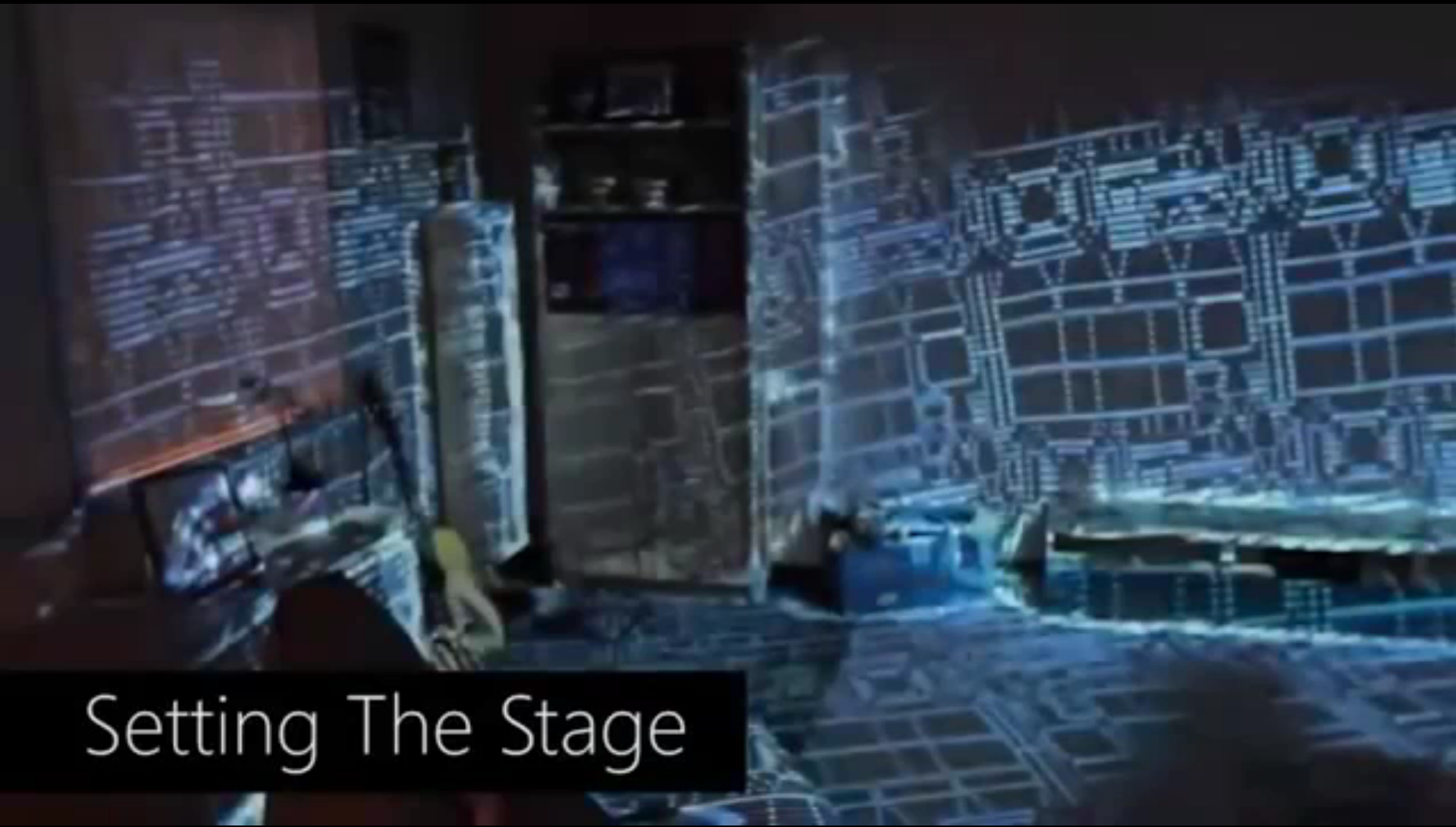


“Steve was working on a new secret project... and [we] were asked to go over to Xerox PARC and take a look at a new computer. We weren't told why.”

“We got a demonstration of the Star, which had a graphical user interface, a laser printer, and a mouse.”

“Xerox had done research to find out what the best input system for a computer was... After ten PhD-years of research they had concluded that the mouse was the best input device.”

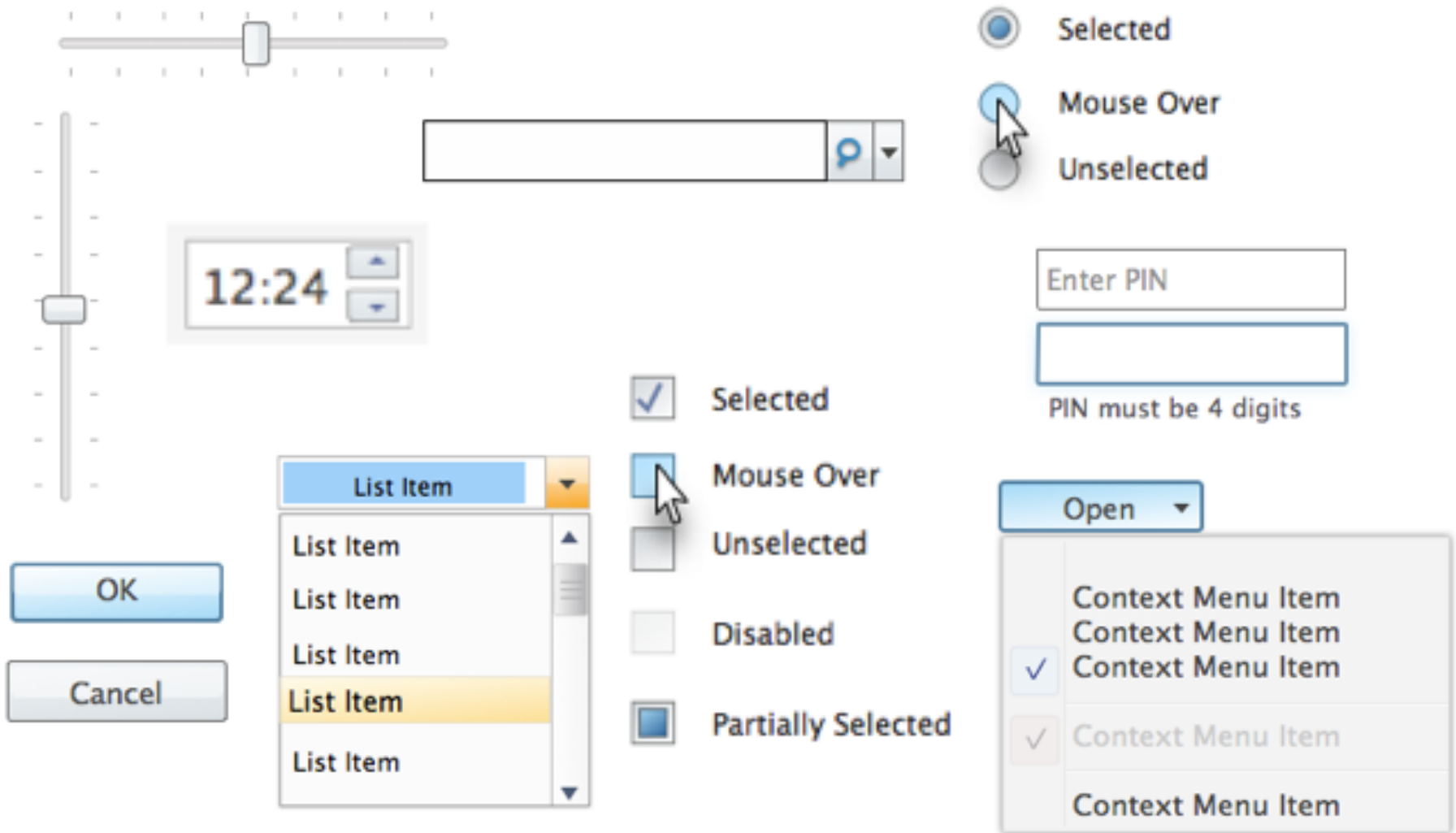
Jim Sachs on Apple Lisa, the first commercial computer with a GUI (\$10k in 1982)



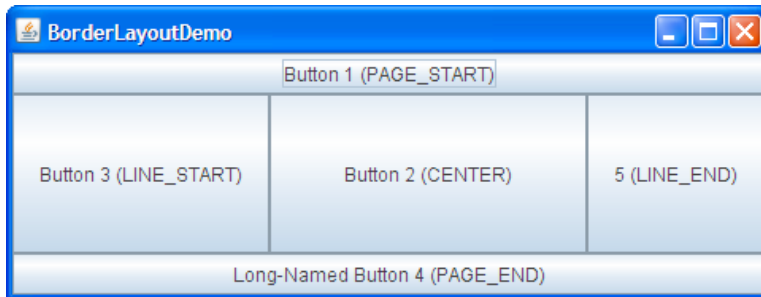
Setting The Stage

Microsoft Research: "Room Alive", <https://youtu.be/ILb5ExBzHqw>

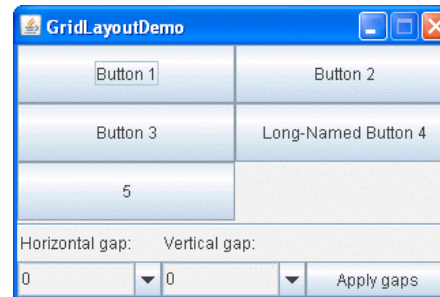
GUIs consist of Components/Widgets



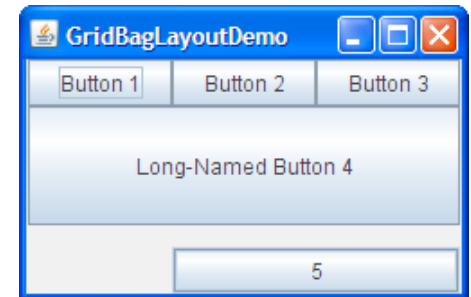
Components are arranged in Layouts



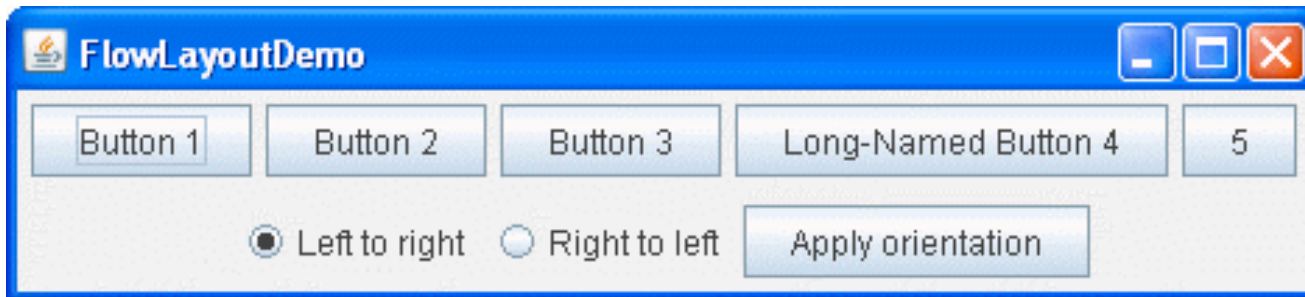
BorderLayout



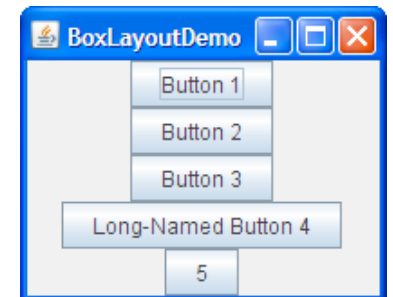
GridLayout



GridBagLayout

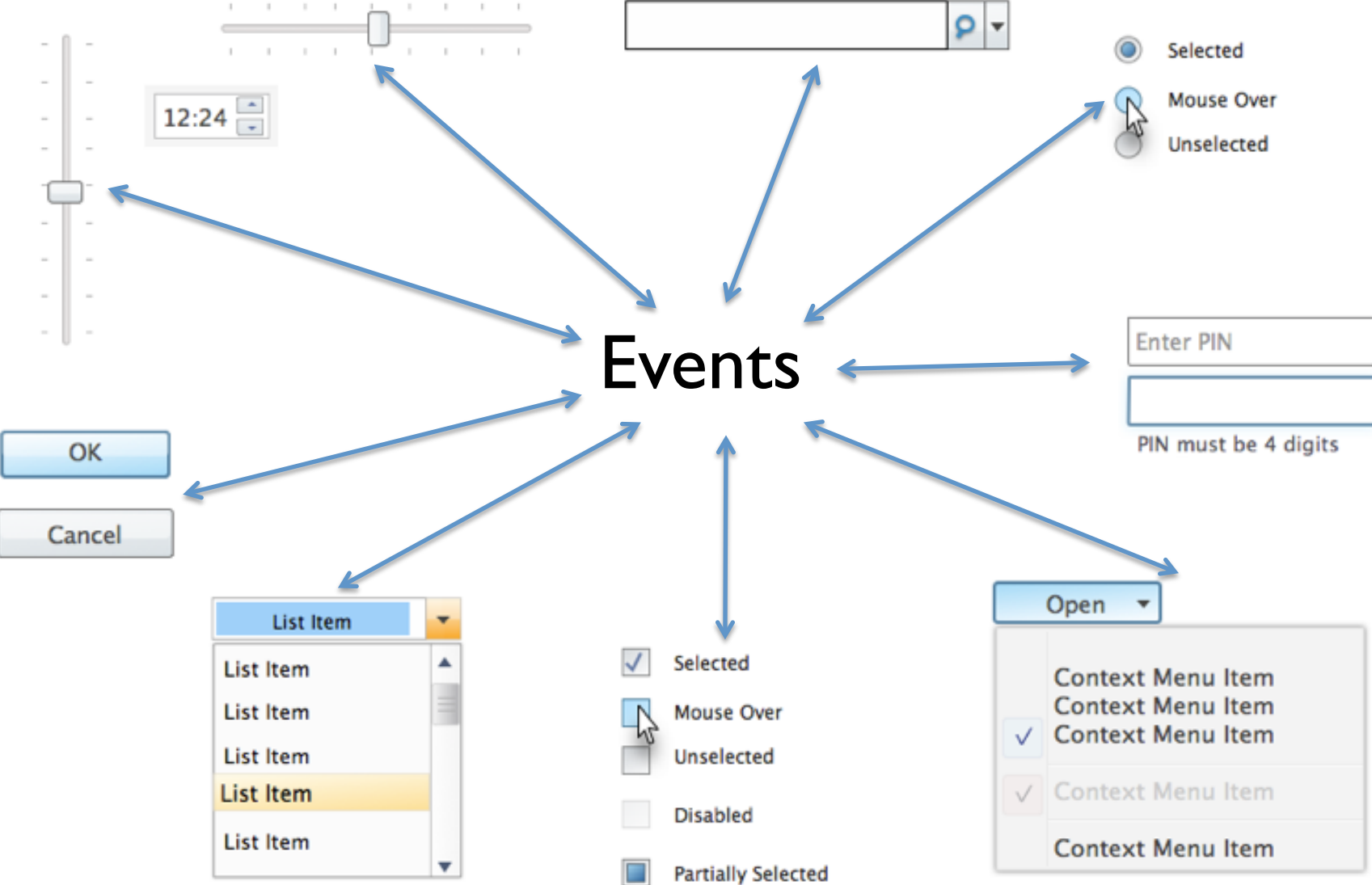


FlowLayout



BoxLayout

Components communicate via Events



GUI

- Provides a **friendly interface** between user and program
- Allows **event-driven** or reactive programming:
The program reacts to events such as button clicks, mouse movement, keyboard input
- Often is **multi-threaded**: Different threads of execution can be going on simultaneously

GUI

- Java provides two standard packages for making GUIs
 - **AWT** (Abstract or Awful Window Toolkit) – original one (`import java.awt.*`)
 - **Swing** – newer one, which builds on AWT as much as possible (`import javax.swing.*`)
- Two aspects to making a GUI:
 - Placing components (buttons, text...)
 - Listening/responding to events

TODAY

Next Lecture

Class JFrame

JFrame object: associated with a window on your monitor.

Generally, a GUI is a JFrame object with various components placed in it

Some methods in a JFrame object

hide()	show()	setVisible(boolean)
getX()	getY()	(coordinates of top-left point)
getWidth()	getHeight()	setLocation(int, int)
getTitle()	setTitle(String)	
getLocation()	setLocation(int, int)	

Over 100 methods in a
JFrame object!

Class JFrame is in package javax.swing

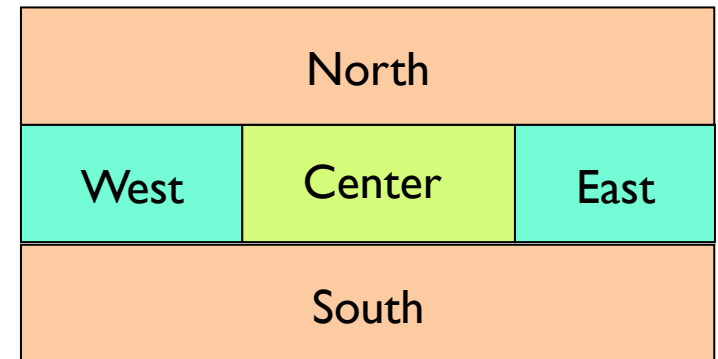
Placing components in a JFrame

Layout manager: Instance controls placement of components.

JFrame layout manager default: BorderLayout.

BorderLayout layout manager: Can place 5 components:

```
public class C extends JFrame {  
    public C() {  
        Container cp= getContentPane();  
        JButton jb= new JButton("Click here");  
        JLabel jl= new JLabel("label 2");  
        cp.add(jb, BorderLayout.EAST);  
        cp.add(jl, BorderLayout.WEST);  
        pack();  
        setVisible(true);  
    }  
}
```



JFrameDemo.java

Placing components in a JFrame

```
import java.awt.*;  import javax.swing.*;
/** Demonstrate placement of components in a JFrame.
    Places five components in 5 possible areas:
    (1) a JButton in the east,          (2) a JLabel in the west,
    (3) a JLabel in the south,         (4) a JTextField in the north
    (5) a JTextArea in the center. */
```

```
public class ComponentExample extends JFrame {
    /** Constructor: a window with title t and 5 components */
    public ComponentExample(String t) {
        super(t);
        Container cp= getContentPane();
        cp.add(new JButton("click me"), BorderLayout.EAST);
        cp.add(new JTextField("type here", 22), BorderLayout.NORTH);
        cp.add(new JCheckBox("I got up today"), BorderLayout.SOUTH);
        cp.add(new JLabel("still winter"), BorderLayout.WEST);
        cp.add(new JTextArea("type\nhere", 4, 10), BorderLayout.CENTER);
        pack();
    }
}
```

Add components to
its contentPane

ComponentExample.java

Packages – Components

Packages that contain classes that deal with GUIs:

java.awt: Old package. **javax.swing:** New package.

javax.swing has a better way of listening to buttons, text fields, etc. Components are more flexible.

Jxxxx: in
Swing, with
xxxx in awt

Component: Something that can be placed in a GUI window. They are instances of certain classes, e.g.

JButton, Button:	Clickable button
JLabel, Label:	Line of text
JTextField, TextField:	Field into which the user can type
JTextArea, TextArea:	Many-row field into which user can type
JPanel, Panel:	Used for graphics; to contain other components
JCheckBox:	Checkable box with a title
JComboBox:	Menu of items, one of which can be checked
JRadioButton:	Similar functionality as JCheckBox
Container:	Can contain other components
Box:	Can contain other components

Hierarchy of Basic Components

Component

Button, Canvas

Checkbox, Choice

Label, List, Scrollbar

TextComponent

 TextField, TextArea

Container

 JComponent

 AbstractButton

 JButton

 JToggleButton

 JCheckBox

 RadioButton

 JLabel, JList

 JOptionPane, JPanel

 JPopupMenu, JScrollBar, JSlider

 JTextComponent

 JTextField, JTextArea

Component: Something that can be placed in a GUI window. These are the basic ones used in Java GUIs

Note the use of subclasses to provide structure and efficiency. For example, there are two kinds of JToggleButtons, so that class has two subclasses.

Components that can contain other components

Component

Box

Container

JComponent

JPanel

Panel

Applet

Window

Frame

JFrame

JWindow

java.awt is the old GUI package.

javax.swing is the new GUI package.

When they wanted to use an old name, they put **J** in front of it.

(e.g. Frame and JFrame)

When constructing javax.swing, the attempt was made to rely on the old package as much as possible.

So, JFrame is a subclass of Frame.

But they couldn't do this with JPanel.

```

import java.awt.*;   import javax.swing.*;

/** Instance has labels in east /west, JPanel with four buttons in center. */
public class PanelDemo extends JFrame {
    JPanel p= new JPanel();

    /** Constructor: a frame with title "Panel demo", labels in east/west,
        blank label in south, JPanel of 4 buttons in the center */
    public PanelDemo() {
        super("Panel demo");
        p.add(new JButton("0"));  p.add(new JButton("1"));
        p.add(new JButton("2"));  p.add(new JButton("3"));

        Container cp= getContentPane();
        cp.add(new JLabel("east"), BorderLayout.EAST);
        cp.add(new JLabel("west"), BorderLayout.WEST);
        cp.add(new JLabel("  "), BorderLayout.SOUTH);
        cp.add(p, BorderLayout.CENTER);
        pack();
    }
}

```

**JPanel: a
container**

JPanel layout manager default: FlowLayout.

FlowLayout layout manager: Place any number of components. They appear in the order added, taking as many rows as necessary.

```

import javax.swing.*; import java.awt.*;
/** Demo class Box. Comment on constructor says how frame is laid out. */
public class BoxDemo extends JFrame {
    /** Constructor: frame with title "Box demo", labels in the east/west,
        blank label in south, horizontal Box with 4 buttons in center. */
    public BoxDemo() {
        super("Box demo");

        Box b= new Box(BoxLayout.X_AXIS);
        b.add(new JButton("0"));    b.add(new JButton("1"));
        b.add(new JButton("2"));    b.add(new JButton("3"));

        Container cp= getContentPane();
        cp.add(new JLabel("east"), BorderLayout.EAST);
        cp.add(new JLabel("west"), BorderLayout.WEST);
        cp.add(new JLabel(" "),    BorderLayout.SOUTH);
        cp.add(b,                  BorderLayout.CENTER);
        pack();
    }
}

```

**Box: a
container**

Box layout manager default: BorderLayout.

BoxLayout layout manager: Place any number of components.
They appear in the order added, taking only one row.

```

public class BoxDemo2 extends JFrame {
    /** Constructor: frame with title t and 3 columns with n, n+1, and n+2 buttons. */
    public BoxDemo2(String t, int n) {
        super(t);

        // Create Box b1 with n buttons.
        Box b1 = new Box(BoxLayout.Y_AXIS);
        for (int i = 0; i < n; i = i + 1)
            b1.add(new JButton("i " + i));

        // Create Box b2 with n+1 buttons.
        Box b2 = ...

        // Create Box b3 with n+2 buttons.
        Box b3 = ...

        // Create horizontal box b containing b1, b2, b3
        Box b = new Box(BoxLayout.X_AXIS);
        b.add(b1);
        b.add(b2);
        b.add(b3);

        Container cp = getContentPane();
        cp.add(b, BorderLayout.CENTER);
        pack(); show();
    }
}

```

Boxes within a Box

3 vertical boxes, each a column of buttons, are placed in a horizontal box

BoxLayout layout

manager: Place any number of components.

They appear in the order added, taking only one row.

Simulate BorderLayout Manager in a JFrame

To simulate using a BorderLayout manager for a JFrame, create a Box and place it as the sole component of the JFrame:

```
JFrame jf= new JFrame( "title" );  
Box b= new Box(BoxLayout.X_AXIS);  
Add components to b;  
jf.add(b, BorderLayout.CENTER);
```

1. **Start developing a GUI by changing an already existing one.** A lot of details. Hard to get all details right when one starts from scratch and has little idea about the Java GUI package
2. Showed how to place components in a GUI. Next time: how to “listen” to things like button clicks in a GUI
3. There are usually 5 different ways to achieve the same thing. Some are more elegant/efficient than others
4. To debug layouts, add borders to containers:
c.setBorder(BorderFactory.createLineBorder(Color.black));