



With apologies to Rene Magritte



About



**Ceci n'est pas une pipe**

# Review

Computer Graphics technology enables GUIs and computer gaming.

GUI's are a major enabling computer technology.

Without a GUI there would not be any, or much less:

- Computer based industry -- USA's major productivity advantage
  - Computer systems in the homes (40% broadband) and schools
- Internet, Web, E-commerce
  - before Mosaic -- telnet, ftp, gopher, lynx, WAIS
  - after Mosaic – Google, AOL, Amazon.com, Ebay
- PDAs, smart (cell) phones

How much software would you use / buy that does not have a GUI interface?

How many computer games without graphics would you play?

Where would Microsoft be without Windows? Apple w/o Macintosh?

# GUI Concepts

## Architecture

Model / View      UI is view-controller of application's model (data)  
Geometry          parent (container) / child (component)  
                         content hierarchy sets layout constraints

**Interface Components**    window, menu, dialog, label, button, scrollbar,  
list, combobox, toolbar, table, pane, image, icon, font, ....

## Events

event driven program architecture "*control flow*"

internal events: Timers, "event messages"

user events are associated w/ UI component

    implicitly events (resize, expose)

    explicitly events: user clicks, selects, drags on UI component

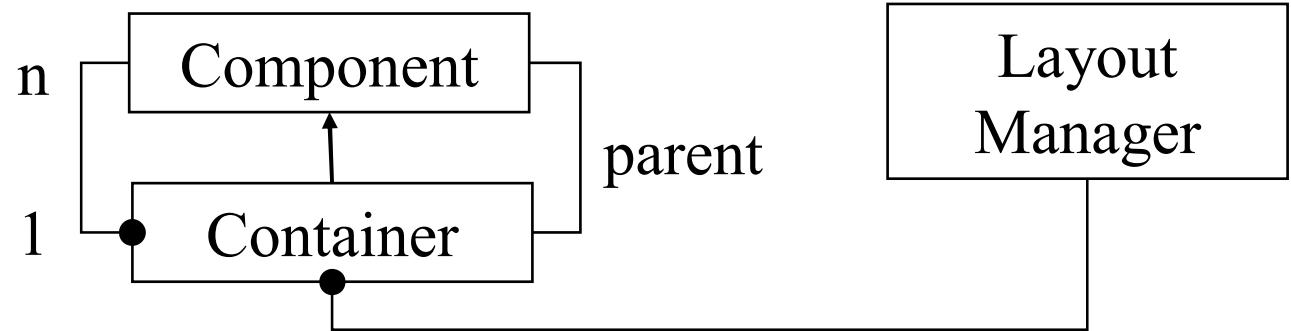
## Drawing

automatic (UI components), borders, fore / back, focus, hover

custom (application) draws: buttons, menus, cell renderers

# Swing / Java

OOP  
class, interface



Containers

JFrame, JApplet, JDialog,  
JOptionPane, JColorChooser, JFileChooser  
JMenu, JTabbedPane, JToolBar, JTree  
ContentPane, JPanel, JScrollPane,  
JList, JTable, JTextArea, JEditorPane

Geometry

Layout Managers: Border, Box, GridBag, Flow, Grid

Models are objects that manage collections: ListModel, TableModel  
abstract models, default models

Graphics

repaint & paintComponent

invoke with `repaint()` ;

Graphics class, `g.draw*(...)`, `g.fill*(...)`,

Color, Font, images, icons

# Event Handling

Events are classes :Action, Item, Adjustment, Window, Mouse, Timer  
Actions enable menuItems, buttons to share event handlers and labels.

Listeners & Adapters (for each listener w/ 2+ methods)

Action, Item, Mouse, Window, ...

MouseAdapter, WindowAdapter

Add a listener to component

```
aButton.addActionListener(new JButtonListener());
```

Implement the listener to define event handling

```
class JButtonListener implements ActionListener {  
    public void actionPerformed(ActionEvent event) {  
        /* define behavior */    }  
}
```

Use an anonymous inner class -- adapter

```
app.addWindowListener( new WindowAdapter() {  
    public void windowClosing(WindowEvent e) {  
        System.exit(0); }    } );
```

# WindowsForms / C#

C# class, interface, property, delegate (interface),  
enum (DialogResult, FormBorderStyle)

## Windows Forms containers

application form	<code>Application.Run(new <i>aForm</i>());</code>
modal dialog form	<code>ShowDialog() / Hide()    Visible = false</code>
modeless dialog form	<code>Show()    Visible = true / "</code>

## Convenience dialogs:

ColorDialog, OpenFileDialog, SaveFileDialog, FontDialog,  
PageSetupDialog, PrintDialog, MessageBox,

Menus: MainMenu, ContextMenu, MenuItem

Geometry: Location (x,y) // size ! change  
anchor to parent {top, left, bottom, right} // size usually ! change  
docking to parent {top, left, fill, right, bottom, none} // size changes

Models are properties encapsulating C# collection interfaces

# Events

Events are members of Control classes.

Events have a publisher / subscriber model

Publisher: sender object raises event that is sent to

Subscriber: receiver object that has attached (+=) an  
EventHandler delegate.

EventHandlers have types: eg. EventHandler – ScrollEventHandler,

EventHandler for most menu, button events.

Events w/ MouseEventArgs: MouseDown, \*Up, \*Move, \*Wheel  
drag is a mouse down, move, up sequence

Events w/ EventArgs: Click, DoubleClick, MouseEnter, \*Leave,  
\*Hover

# Event Handling

OnEventName OnPaint, OnResize,

in a Control's class override protected virtual method

```
protected void override OnPaint(PaintEventArgs, p) { ... }
```

eventHandlers

Control defines a PaintEventHandler delegate

```
public delegate void PaintEventHandler (object obj,  
    PaintEventArgs p);
```

In a Control's class add the handler to the event

```
aForm.Paint += new PaintEventHandler(aPainter);
```

write the handler method

```
static void aPainter(object obj, PaintEventArgs p)  
    { ... }
```

# Design

HCI human performance

visual perception and memory are superior

Error: intention (mistake), storage (lapse), action (slip)

GUIs Functional > Aesthetic Simplicity > realism

usage is a visual search – pattern recognition task

low syntactic demand (applications share "look && feel" syntax)

images have semiotic "iconic representation"

images / labels can embedded semantics into tasks.

control and closure – UI objects are manipulated in "*reversible*" steps

minimizes slip errors – application state show w/ enable / disable

graphic art principles: reduction, regulation, scale, contrast, color use,

activity / visual search / spatial logic , grouping – distinction,

figure / ground – negative space, grids,

Interface Builders: directional, layout ! design tool,

bind UI && events

# GUI trends

2D Variable magnification of display

Allow search or large information space a low magnification see more

Select objects for detailed evaluation and zoom in increase magnification  
e.g. datalens, fisheye menus, table lens, star tree

Micro GUIs – cell phones, pdas, wearable computers

3D Navigation incorporates zoom.

Volume > Area, depth == variable magnification  
e.g. data wall, cone trees, cubic eye

Are 2D constant magnification GUI environments the qwerty keyboard of GUIs ?

Is 2D good enough ?

Resistance to change ?

## What to do next wrt GUI?

Become more visually literate -- visual thinking is a critical skill for graphical user interface design! **ART principles are important!**

Always evaluate and appreciate good (and bad) design in the world around you. Design ideas can come from other media.

Do not limit design to what you currently know how to do!

Learn more about GUI APIs (design, controls, user interface, building)  
how are events really implemented (C# delegates, firing events)  
manipulate images, serialization, owner draw menuItems

One design means:

1. You can't think of another one.
2. If plan A doesn't work you are stuck
3. You will implement your first (not best) solution to the problem.

Design is an exercise ...

Design strengthens your creativity ...