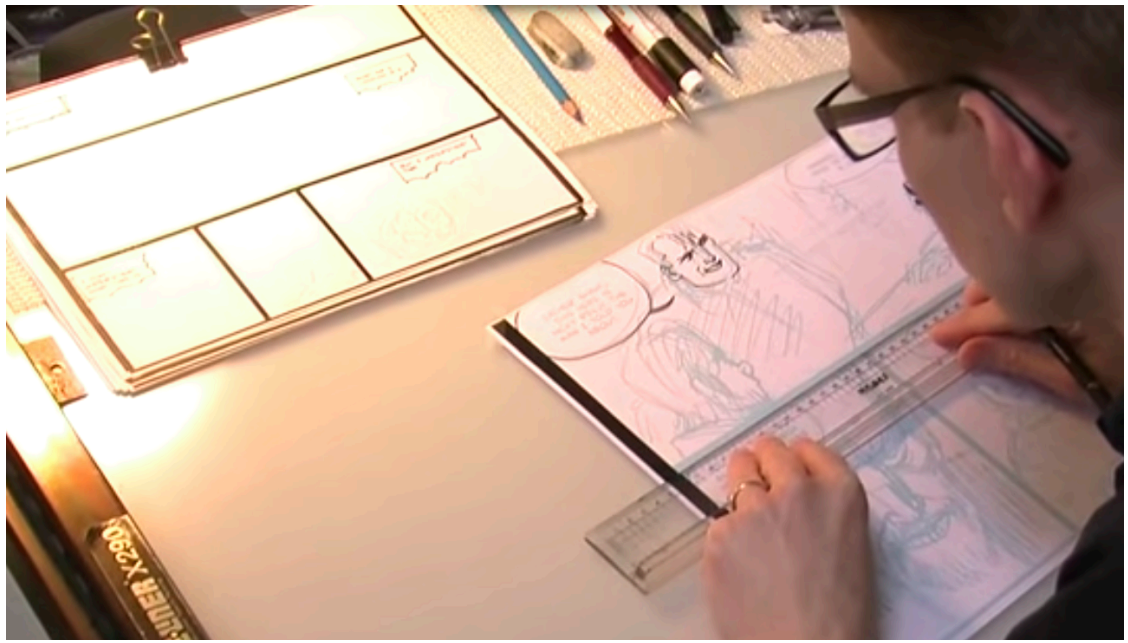
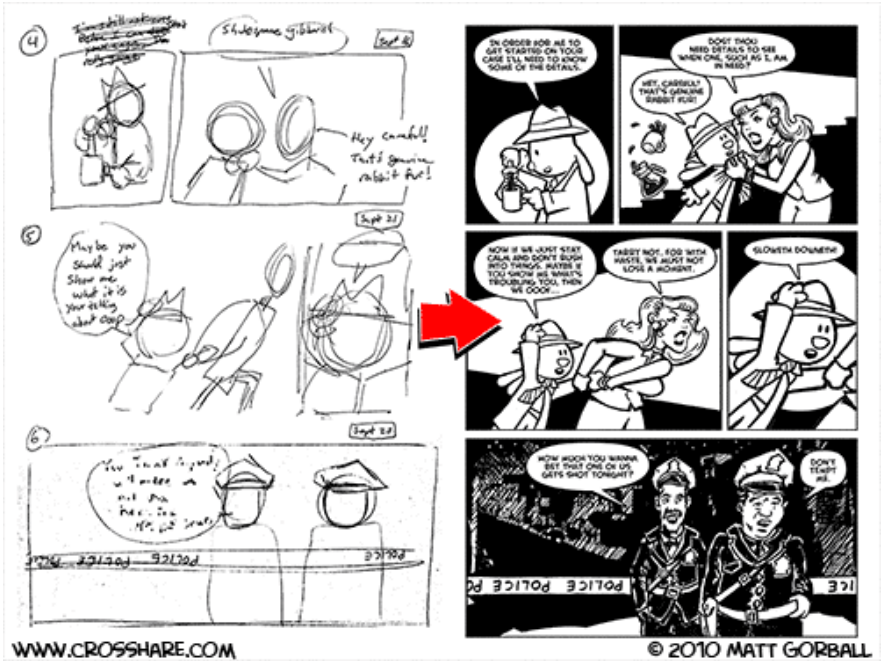
UI Programming

(part of this content is based on previous classes from Anastasia, S. Huot, M. Beaudouin-Lafon, N.Roussel, O.Chapuis)

Assignment 1 is out!

Design and implement an interactive tool for creating the layout of comic strips





https://www.lri.fr/~fanis/teaching/ISI2014/assignments/ass1/

Graphical interfaces

GUIs: input is specified w.r.t. output

Input peripherals specify commands at specific locations on the screen (*pointing*), where specific objects are drown by the system. Familiar behavior from physical world

WIMP interfaces

WIMP: Window, Icons, Menus and Pointing

Presentation

- Windows, icons and other graphical objects

Interaction

- Menus, dialog boxes, text input fields, etc

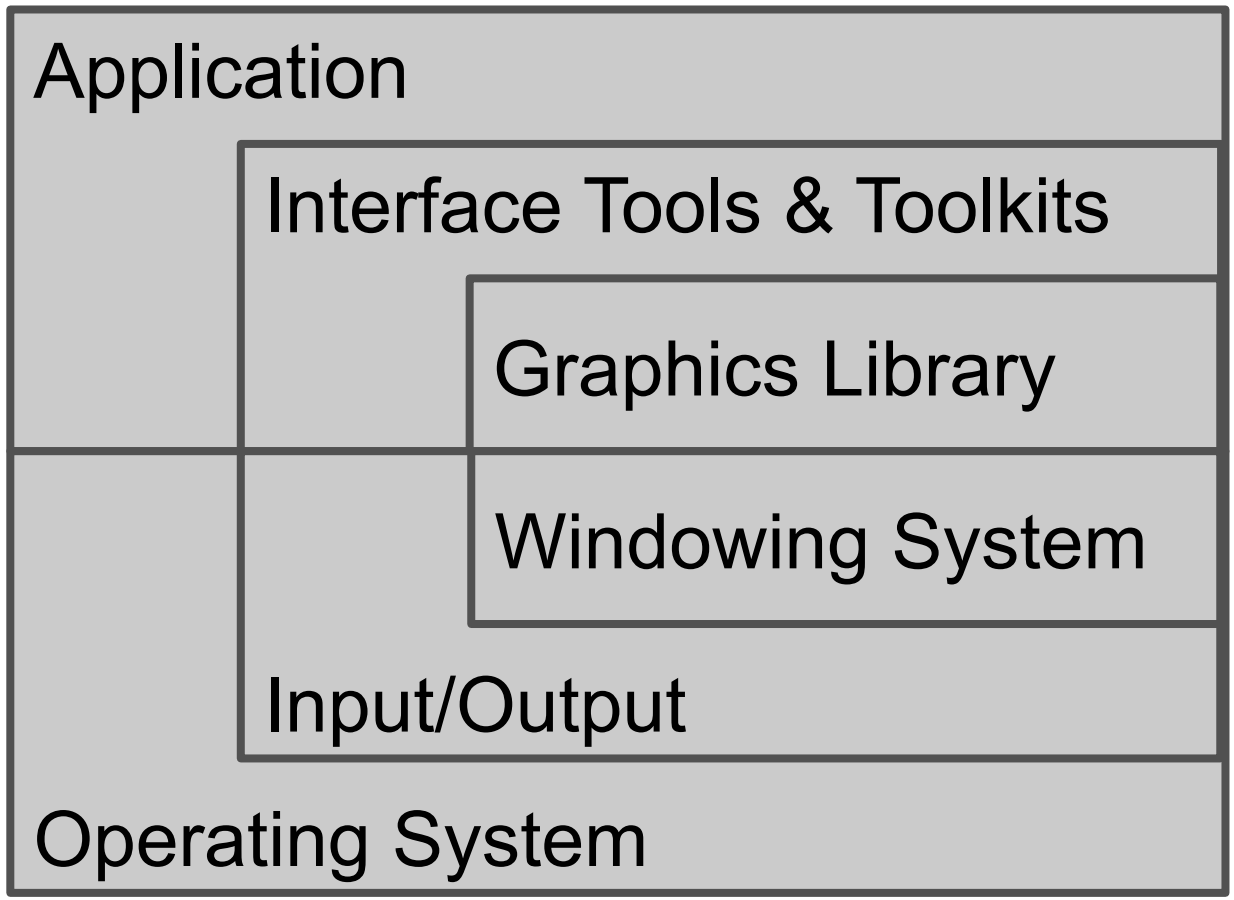
Input

- pointing, selection, ink/path

Perception-action loop

- feedback

Software layers



Applications/Communication (MacApp)

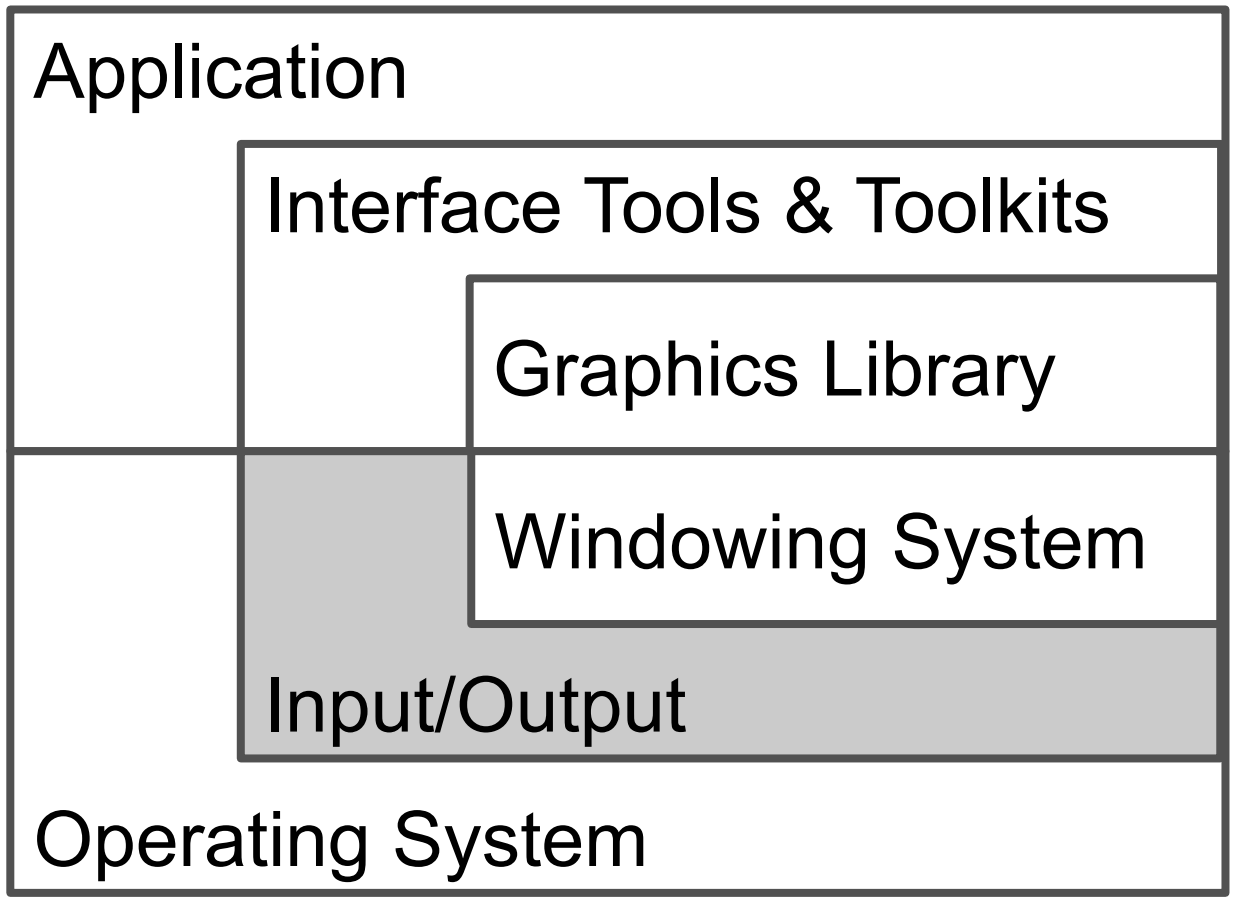
Builders, Java Swing, JavaFX, Qt (C++), GTK+, MFC, Cocoa

GDI+, Quartz, GTK+/Xlib, OpenGL

X Windows (+KDE or GNU)

Windows, Mac OS, Unix, Linux, Android, iOS, WindowsCE

Software layers



Input/output peripherals

Input: where we give commands





Output: where the system shows information & reveals its state



Interactivity vs. computing

Closed systems (computation):

- read input, compute, produce result
- final state (end of computation)

Open systems (interaction):

- events/changes caused by environment
- infinite loop, non-deterministic

Problem

We learn to program algorithms (computational)

Most languages (C/C++, Java, Lisp, Scheme, Pascal, Fortran, ...) designed for algorithmic computations, not interactive systems

Problem

Treating input/output during computation (interrupting computation) …

- write instructions (`print, put, send,`…) to send data to output peripherals
- read instructions (`read, get, receive,`…) to read the state or state changes of input peripherals

Problem

To program IS in algorithmic/computational form

```
two buttons B1 and B2
finish <- false
while not finish do
    button <- waitClick () //interruption, blocked comp.
    if button
            B1 : print « Hello World »
            B2 : finish <- true
    end
end
```

Managing input

Querying

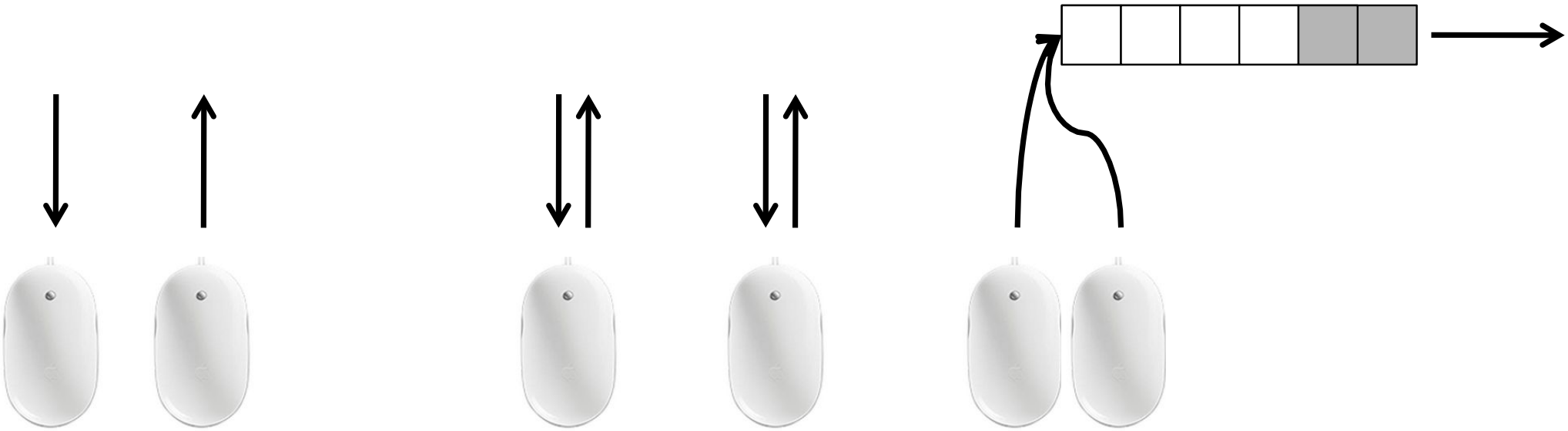
Query & wait
1 device at a time

Polling

Active wait
Polling in sequence
CPU cost

Events

Wait queue

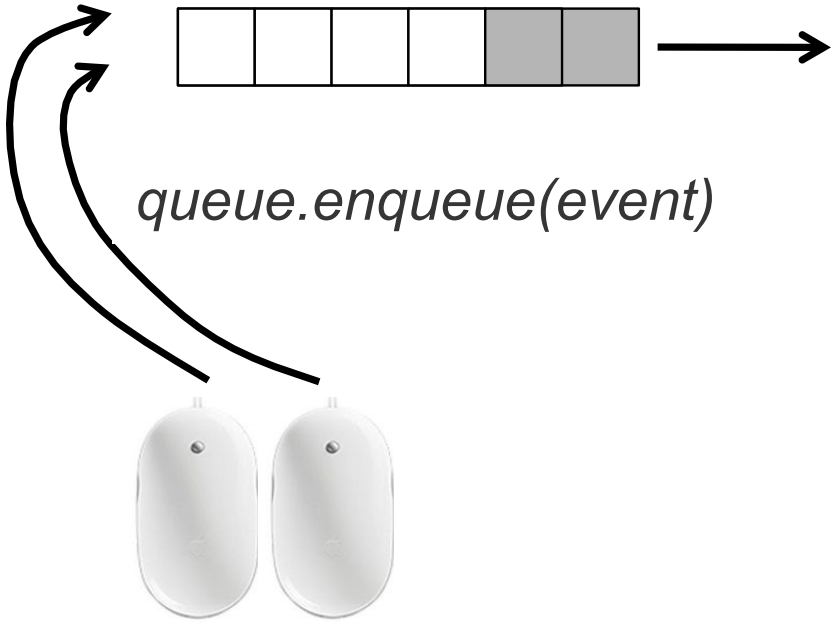


Event based (driven) programming

Source: Mouse Click

```
while active
   if queue is not empty
      event <- queue.dequeue()
      source <- findSource(event)
      source.processEvent(event)
   end if
end while
```

event (waiting) queue

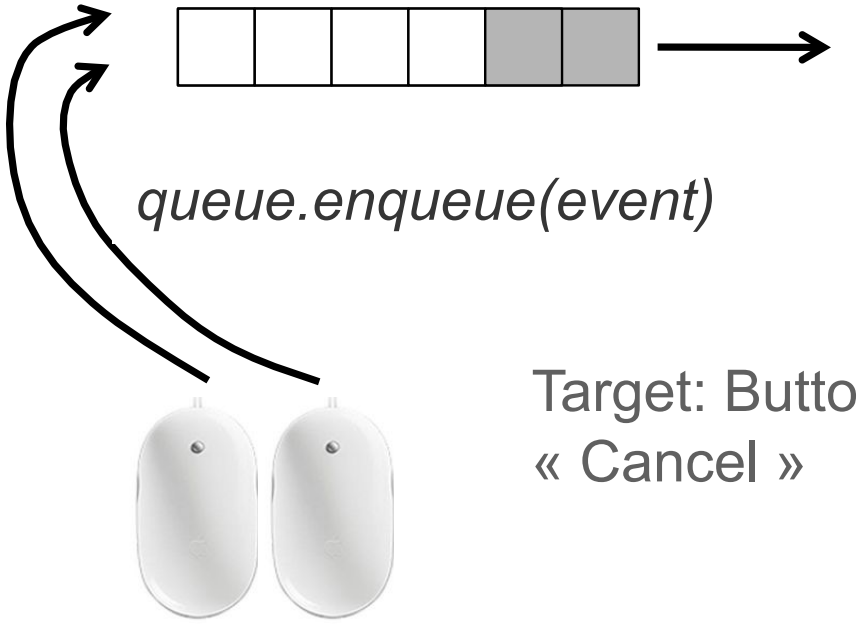


queue.enqueue(event)

Event based (driven) programming

Source: Mouse Click

event (waiting) queue



queue.enqueue(event)

Target: Button
« Cancel »

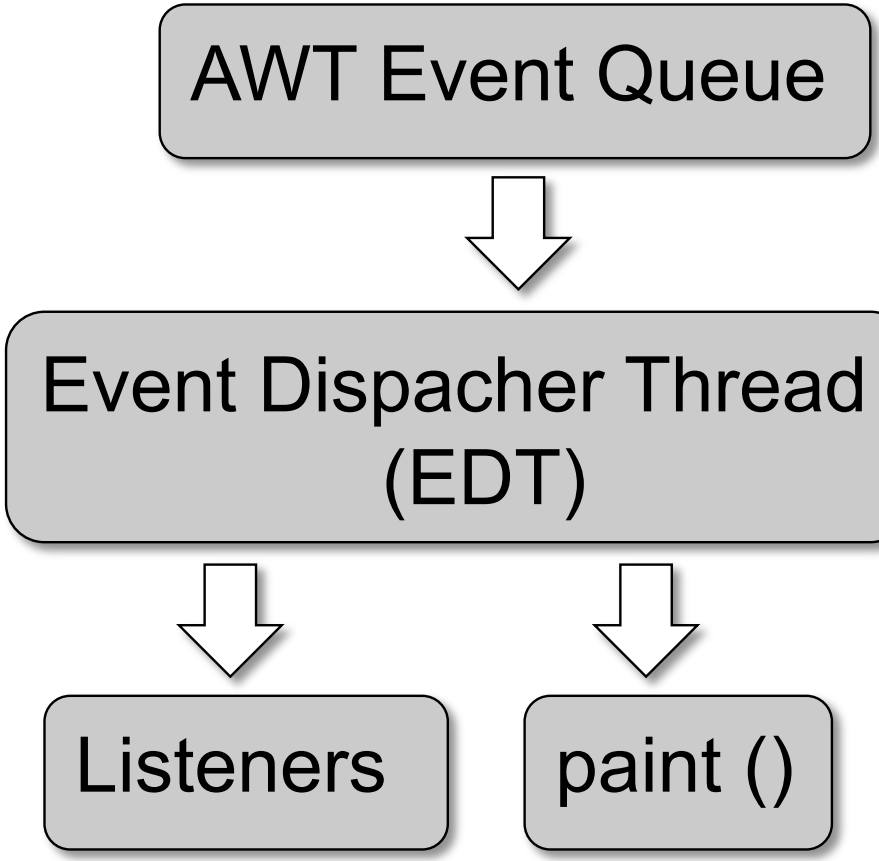
```
while active
   if queue is not empty
      event <- queue.dequeue()
      source <- findSource(event)
      source.processEvent(event)
   end if
end while
```

```
processEvent(event)
    target <- FindTarget (event)
    if (target ≠ NULL)
         target.processEvent(event)
```

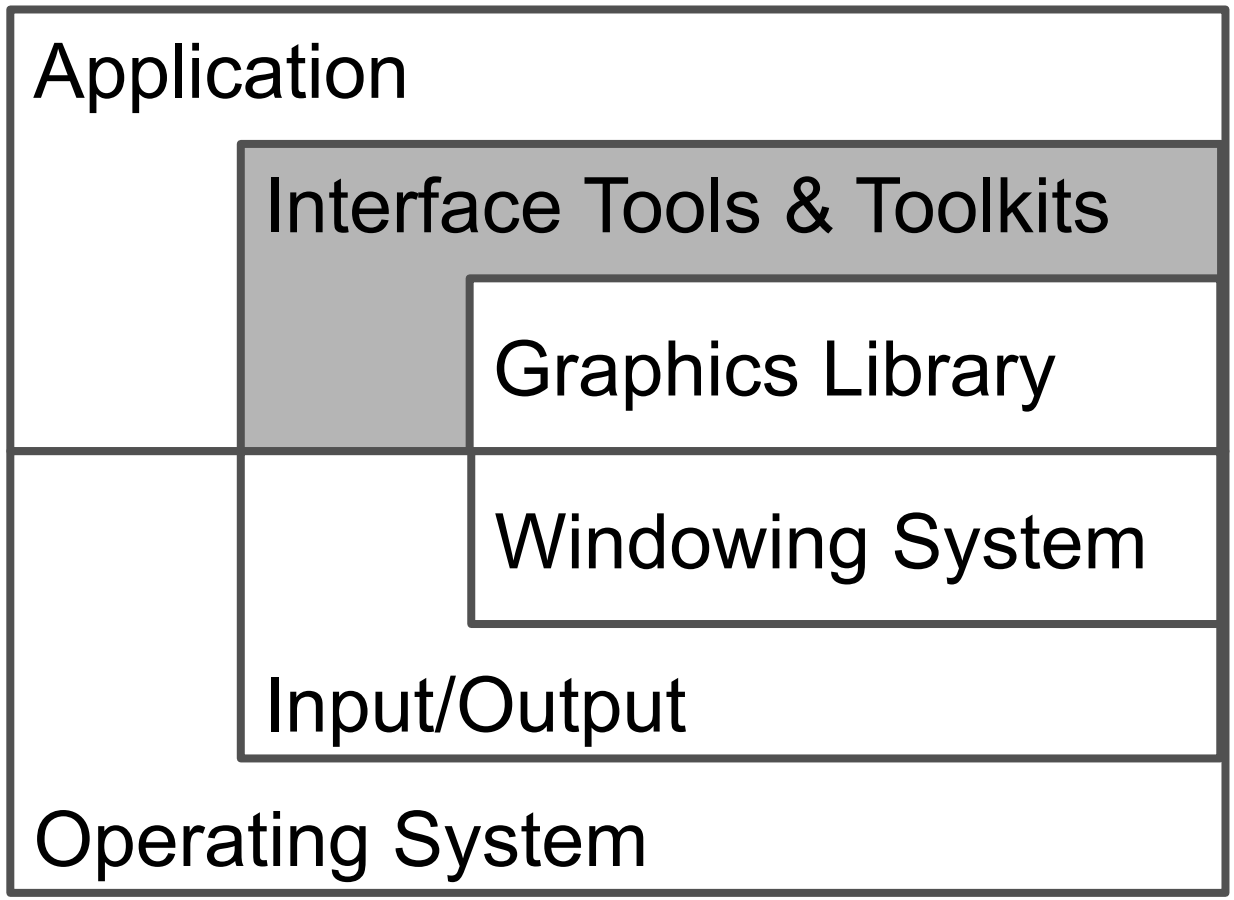
Example: Swing (and AWT)

3 threads

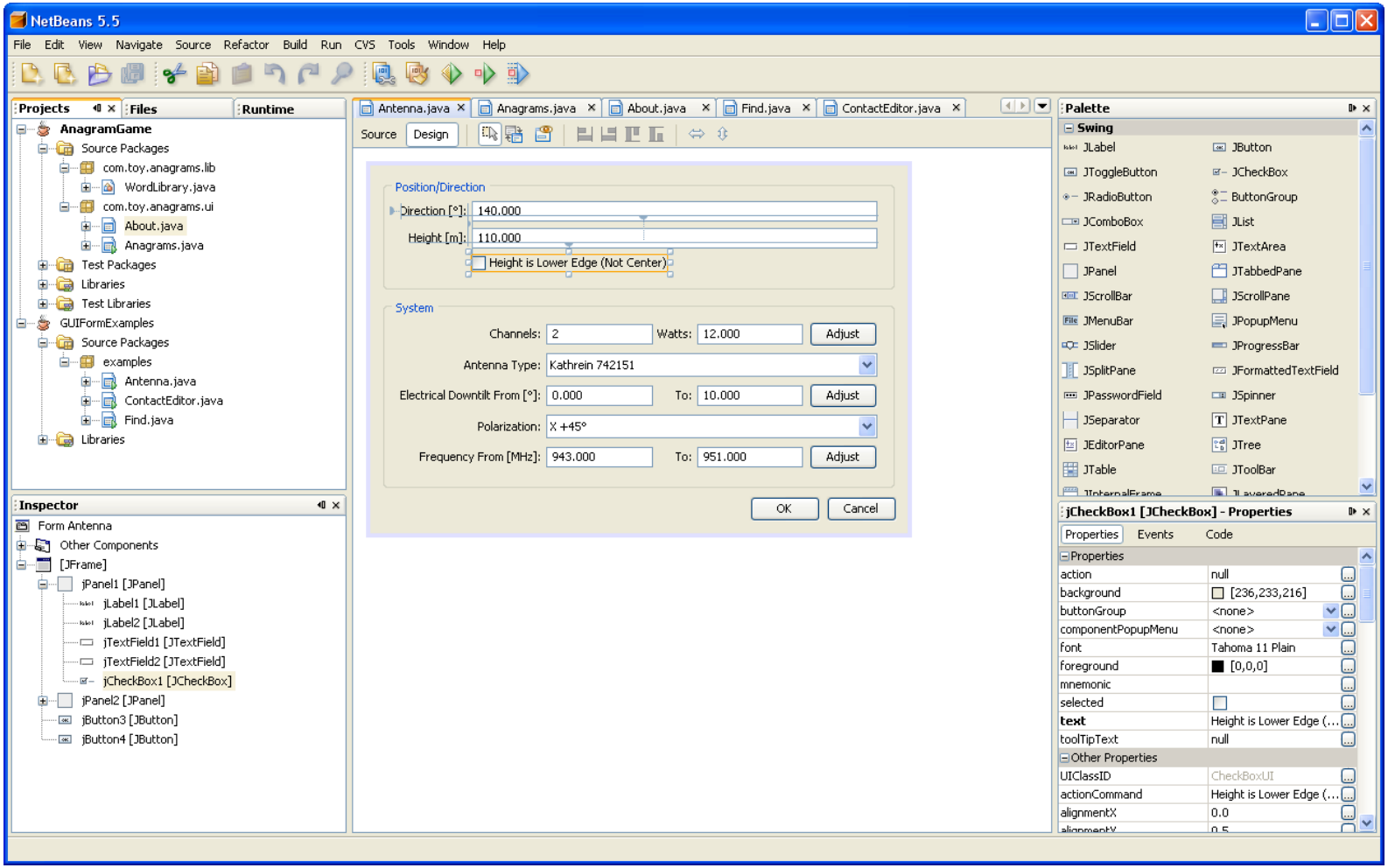
- Initial thread: main ()
- EDT manages the events queue: sends events to *listeners* (functions dealing with events) and calls paint methods (drawing functions)
- Worker (or background) threads, where time-consuming tasks are executed



Software layers



Interface builders



Examples : MS Visual Studio (C++, C#, etc.), NetBeans (Java),
Interface Builder (ObjectiveC), Android Layout Editor

Interface builders

Can be used to

- create prototypes (but attention it looks real)
- get the « look » right
- be part of final product

- design is fast
- modest technical training needed
- can write user manuals from it

But: still need to program (and clean code …)

Interface toolkits

Libraries of interactive objects (« widgets », e.g., buttons) that we use to construct interfaces

Functions to help programming of GUIs

...usually also handle input events (later)

Interface toolkits

Toolkit	Platform	Language
Qt	multiplatform	C++
GTK+	multiplatform	C
MFC later WTL	Windows	C++
WPF (subset of WTL)	Windows	(any .Net language)
FLTK	multiplatform	C++
AWT / Swing	multiplatform	Java
Cocoa	MacOs	Objective C
Gnustep	Linux, Windows	Objective C
Motif	Linux	C
JQuery UI	Web	javascript

Problem with toolkits? ….

Why Java Swing?

Based on Java (any platform, plenty of libraries)

A lot of online resources and examples

Why Java Swing?

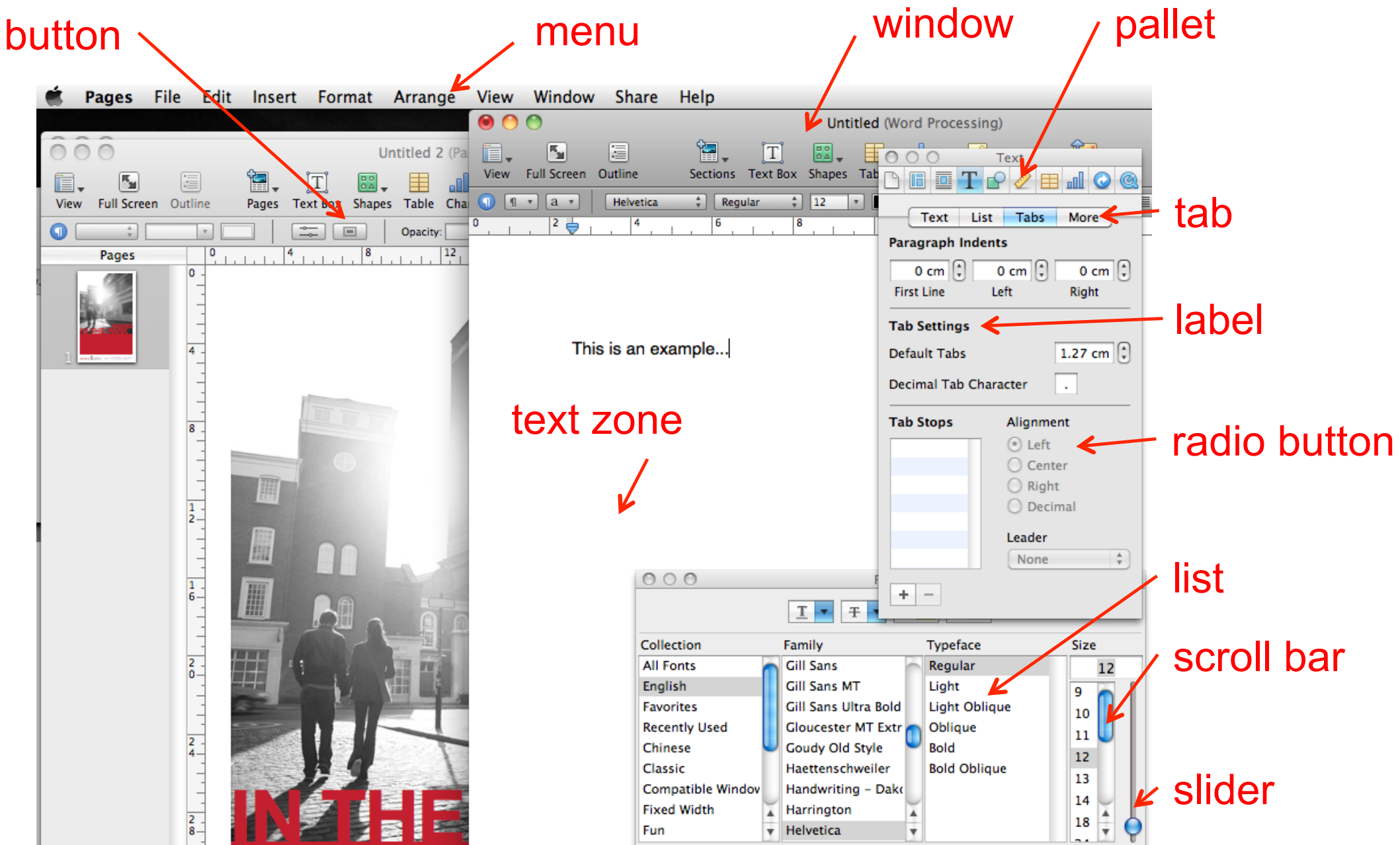
Based on Java (any platform, plenty of libraries)

A lot of online resources and examples

Other alternatives for Java?

→ JavaFX: soon becomes the new standard for Java UI programming, supporting a variety of different devices

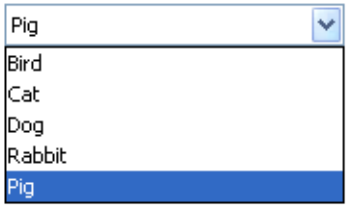
« widgets » (window gadgets)



Swing widgets

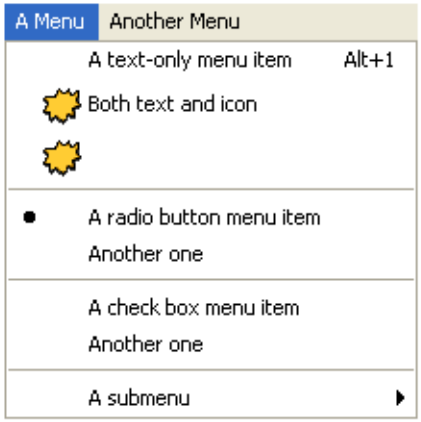
JButton

JCheckBox



JComboBox

JList



JMenu

JRadioButton

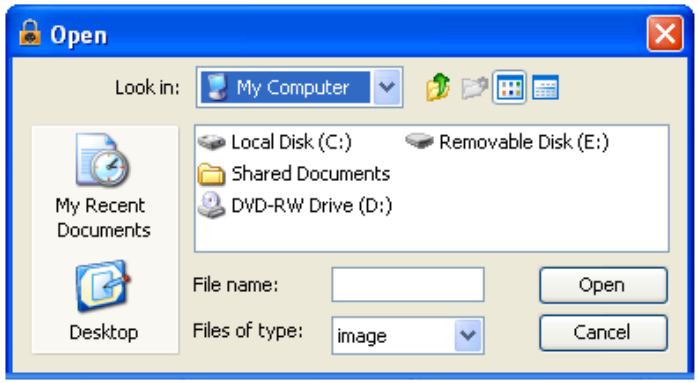
JSlider

JSpinner

JTextField

JPasswordField

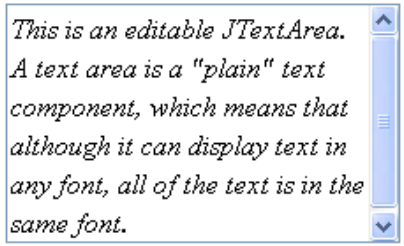
Swing widgets



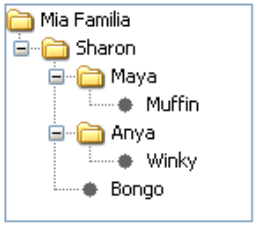
JFileChooser

Host	User	Password	Last Modified
Biocca Games	Freddy	!#asf6Awwzb	Mar 16, 2006
zabble	ichabod	Tazb!34$fZ	Mar 6, 2006
Sun Developer	fraz@hotmail.com	AasW541!fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail.com	bkz[ADF78!	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.com	vbAf124%z	Feb 22, 2006

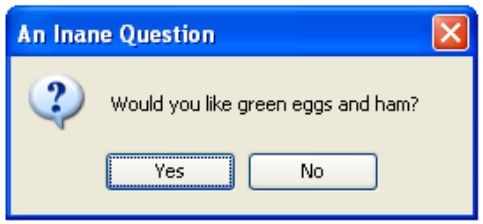
JTable



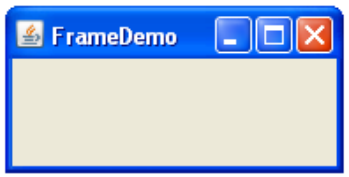
JTextArea



JTree

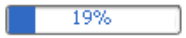


JDialog



JFrame

JLabel



JProgressBar

JSeparator

JToolTip

Widget complexity

Simple widgets

- buttons, scroll bars, labels, …

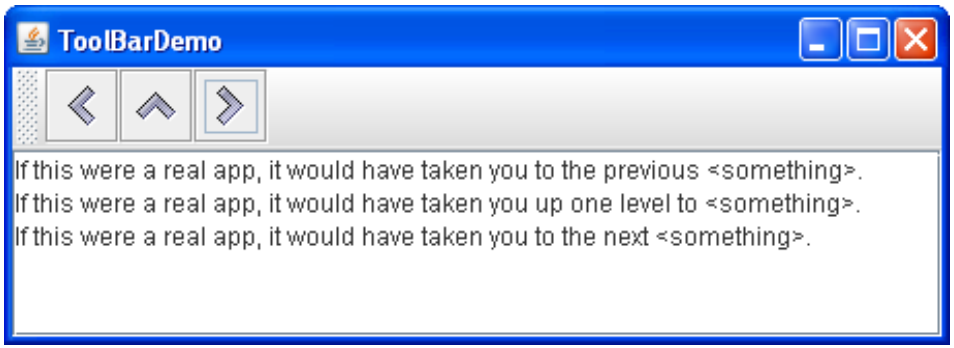
Composite/complex widgets

- contain other widgets (simple or complex)
- dialog boxes, menus, color pickers, …

Widget tree

Hierarchical representation of the widget structure

- a widget can belong to only one « container »

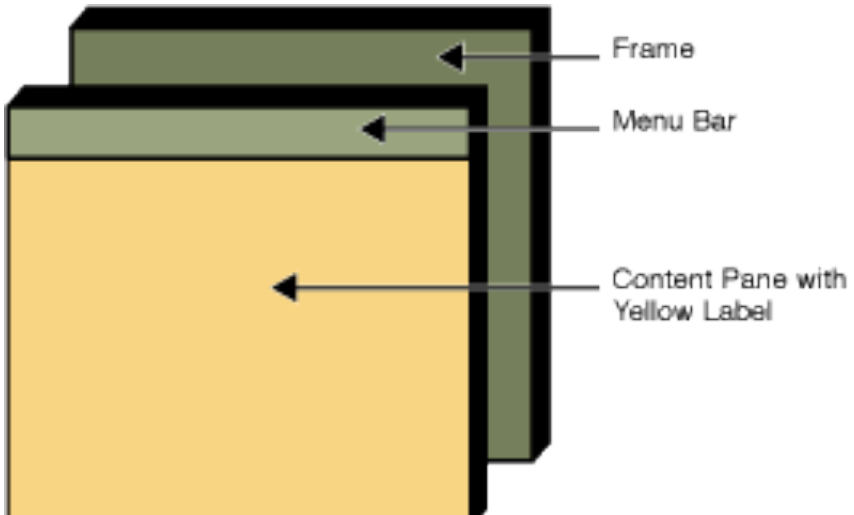


Swing widget classes

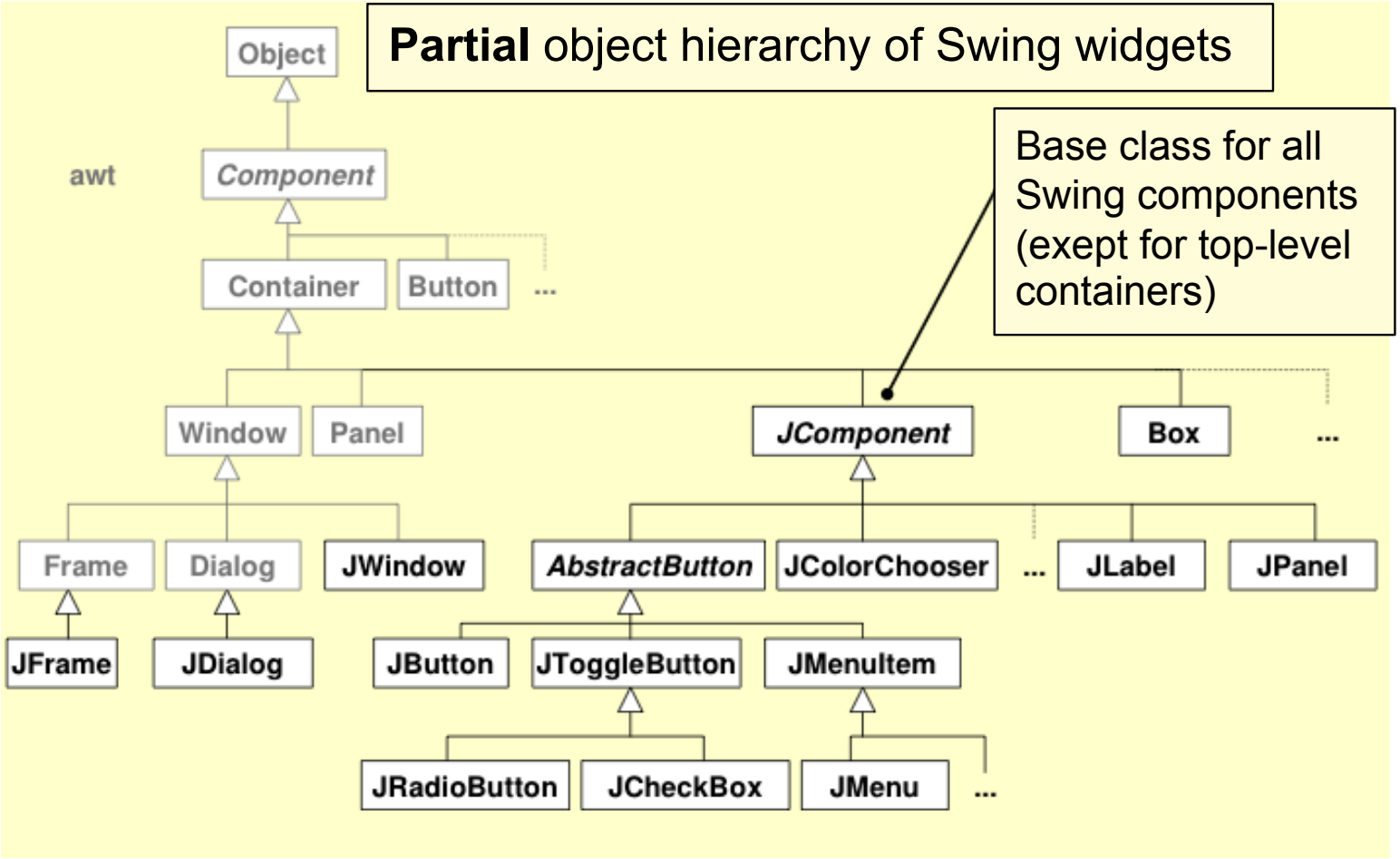
A GUI application has a top-level (container) widget that includes all others

In Swing there are 3 types: JFrame, JDialog and JApplet

They all contain other widgets (simple or complex), that are declared in the field **content pane**



Swing widget classes



http://docs.oracle.com/javase/tutorial/ui/features/components.html

Swing JFrame

a window with a basic bar

```
public static void main(String[] args) {
    JFrame jf = new JFrame("Ta ta!");
    jf.setVisible(true);
    jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    System.out.println("finished ? ! ?");
    System.out.println("no, still running …");
}
```

Useful functions

public JFrame();
public JFrame(String name);
public Container getContentPane();
public void setJMenuBar(JMenuBar menu);
public void setTitle(String title);
public void setIconImage(Image image);

This program does not terminate after "no, still running …"

Swing JDialog

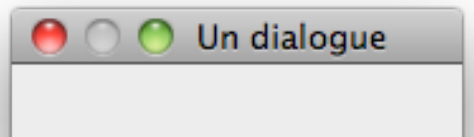
a message window (dialog) can be “modal” (blocks interaction)

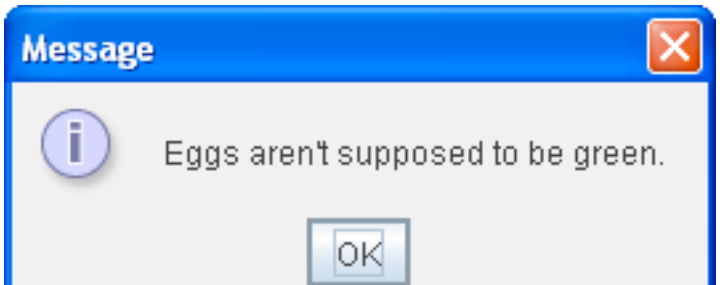
usually attached to another window (when that closes, so does the dialog)

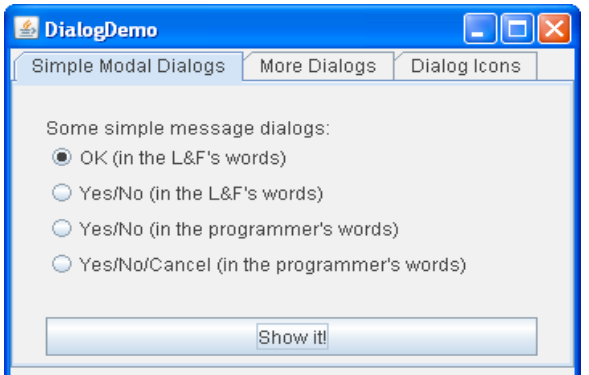
```
public static void main(String[] args) {
    JFrame jf = new JFrame("ta ta!");
    jf.setVisible(true);
    jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JDialog jd = new JDialog(jf,"A dialog",true);
    jd.setVisible(true);
}
```

modal

attached to



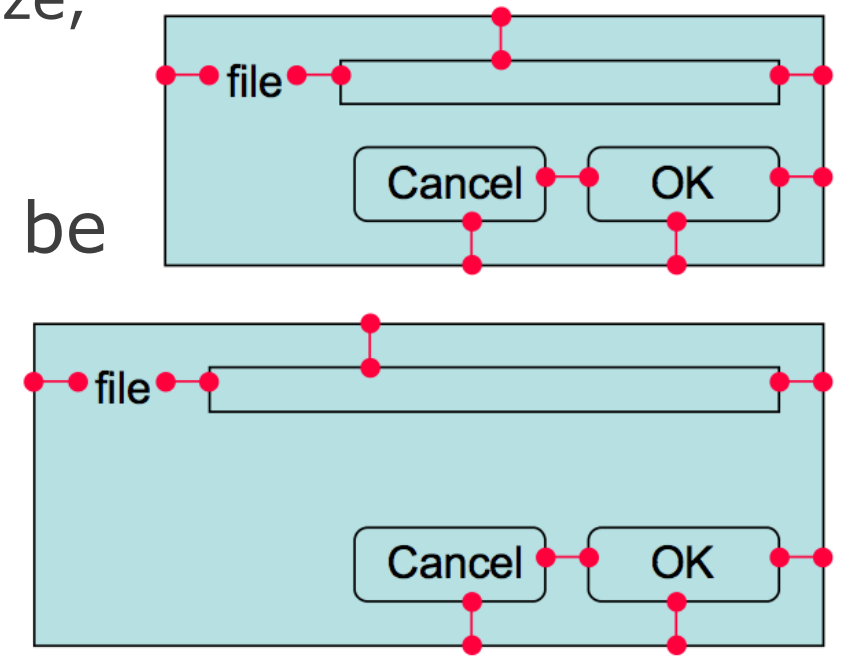




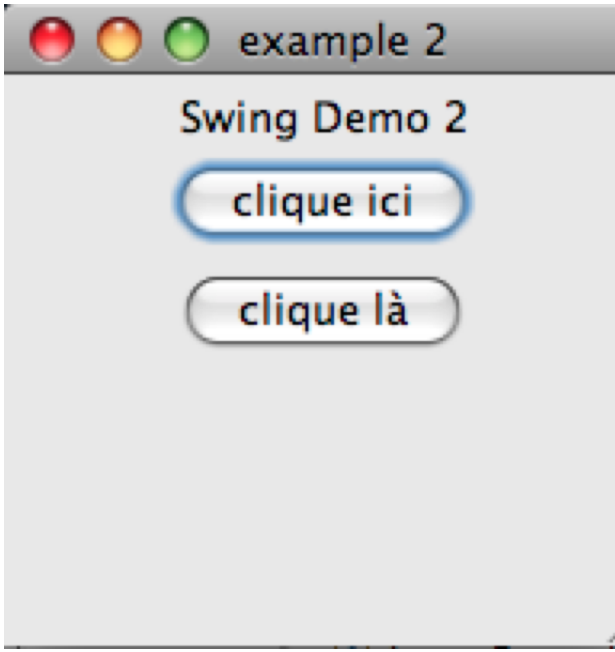
Widget placement

UI toolkits control widget placement:

- should be independent of widget size
  (menu at least as big as its largest item,
  change of scrollbar size with document size,
  adjusting text flow)
- done in *layout managers* that can be added to container widgets



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

public class SwingDemo2 extends JFrame {

        public void init()
        {
            this.setTitle("example 2");

            getContentPane().add(new JLabel("Swing Demo 2"));

            Container contentPane = this.getContentPane();
            contentPane.setLayout(new FlowLayout());

            this.setDefaultCloseOperation(EXIT_ON_CLOSE);

            contentPane.add(new JButton("clique ici"));
            contentPane.add(new JButton("clique là"));
        }

        public static void main(String[] args)
        {
                SwingDemo2 frame = new SwingDemo2();

                frame.init();

                frame.setSize(200,200);
                frame.setVisible(true);
        }
}
```



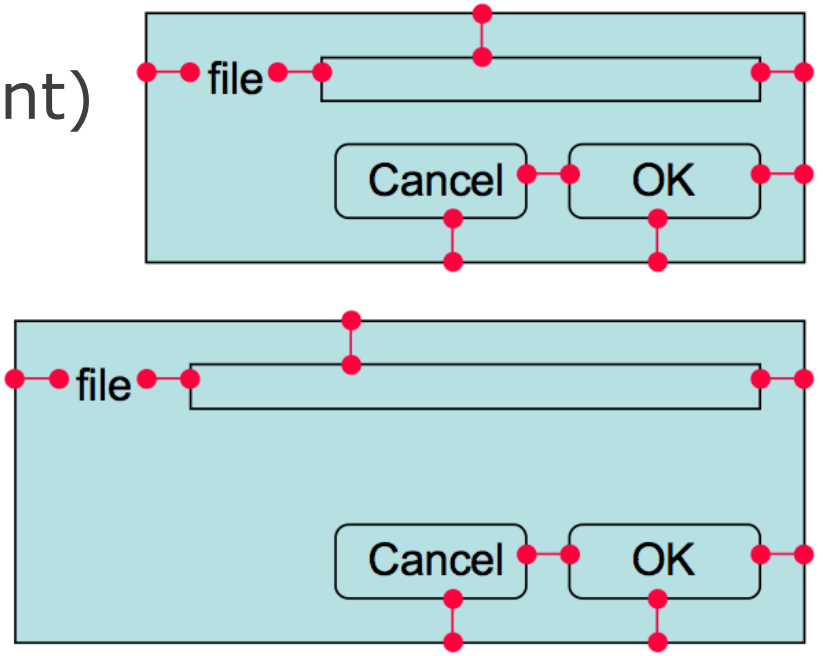
Widget placement

General guides

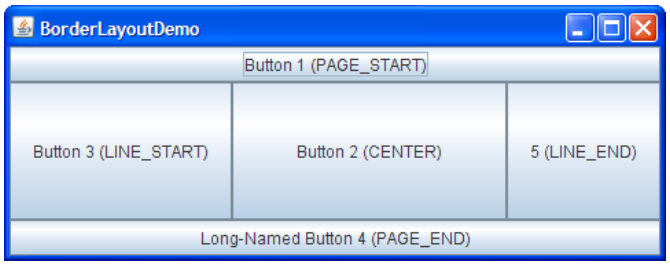
- embed geometry of a «child» widget to its parent
- parent controls the placement of its children

Layout algorithm

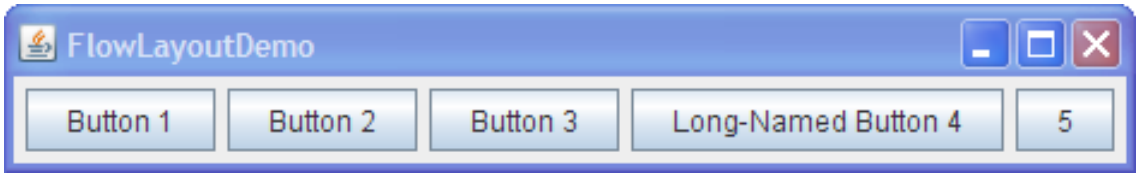
- natural size for each child (to fit content)
- size and position imposed by parent
- constraints: grid, form, etc.



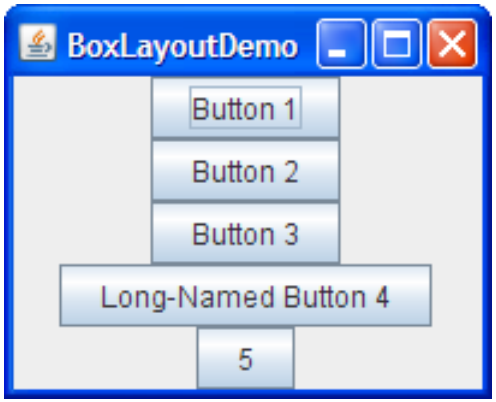
Layout managers (in Swing)



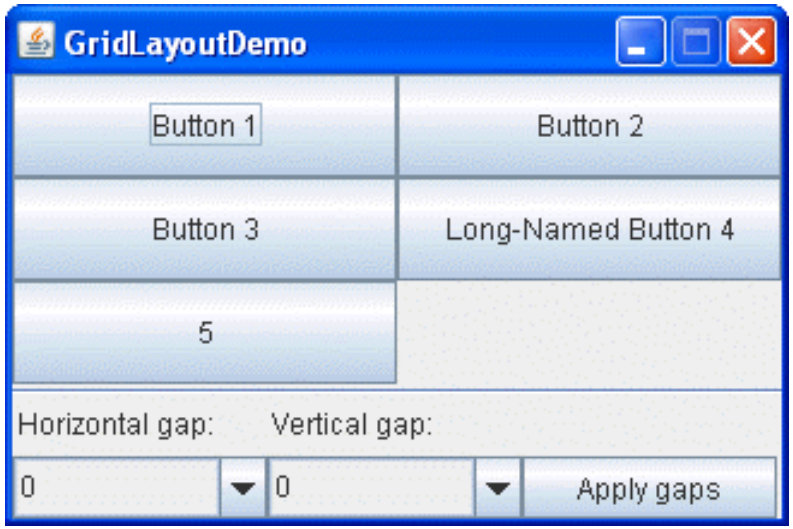
BorderLayout



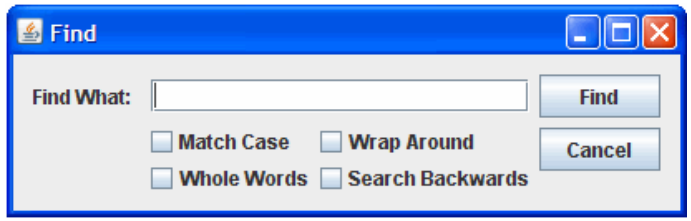
FlowLayout



BoxLayout

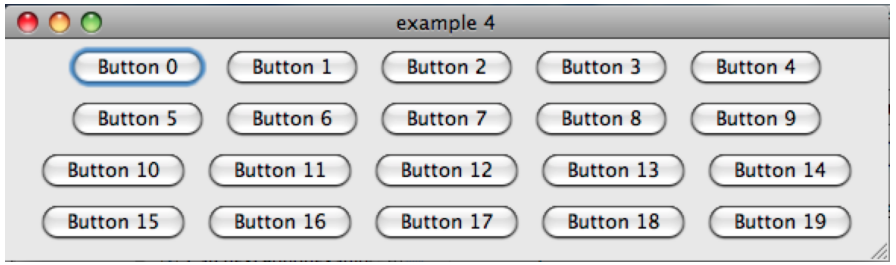


GridLayout



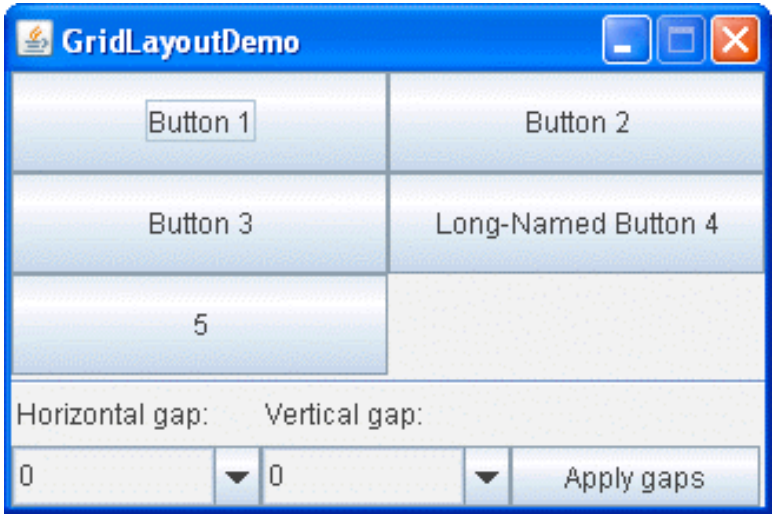
GroupLayout

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

public class SwingDemo4 extends JFrame {

    public void init()
    {
       Container cp = getContentPane();

       this.setTitle("example 4");
       this.setDefaultCloseOperation(EXIT_ON_CLOSE);

       cp.setLayout(new FlowLayout());
       for(int i = 0; i < 20; i++)
          cp.add(new JButton("Button " + i));
    }

    public static void main(String[] args)
    {
       SwingDemo4 frame = new SwingDemo4();

       frame.init();

       frame.setSize(200,700);
       frame.setVisible(true);
    }
}
```

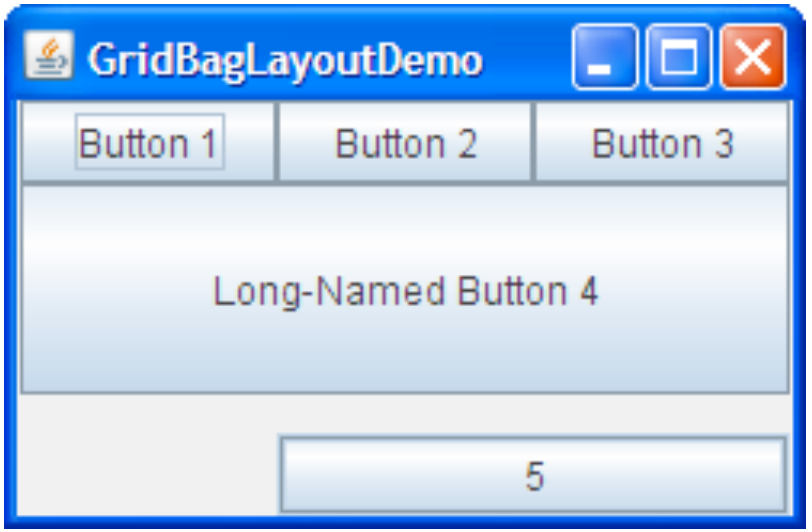


Layout managers (in Swing)

GridLayout: grid



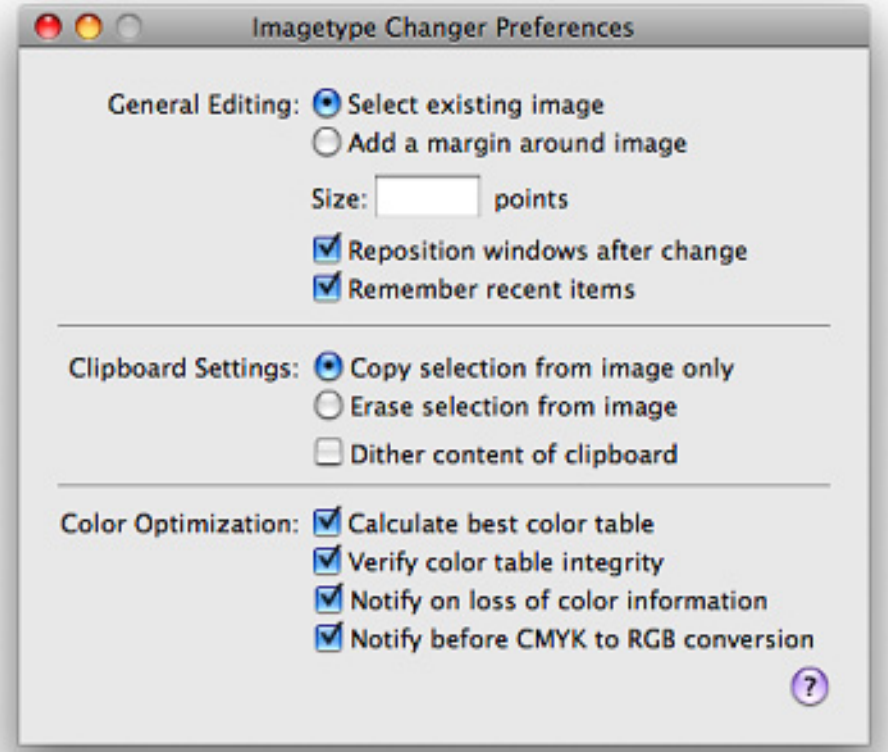
GridBagLayout: sophisticated grid



Layout managers (in Swing)

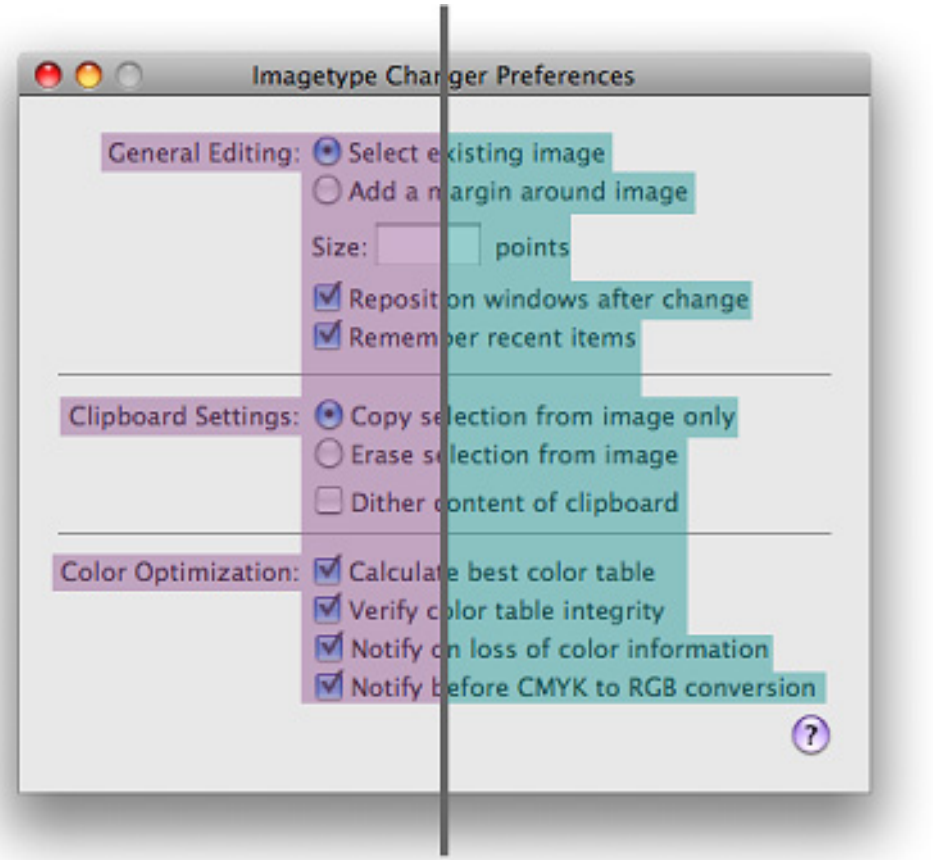
```
GridLayout gridLayout = new GridLayout(0,2);

JPanel gridPanel = new JPanel();
gridPanel.setLayout(gridLayout);

gridPanel.add(new JButton("Button 1"));
gridPanel.add(new JButton("Button 2"));
gridPanel.add(new JButton("Button 3"));
gridPanel.add(new JButton("Long-Named Button 4"));
gridPanel.add(new JButton("5"));
```

Placement guides (Mac OS X)



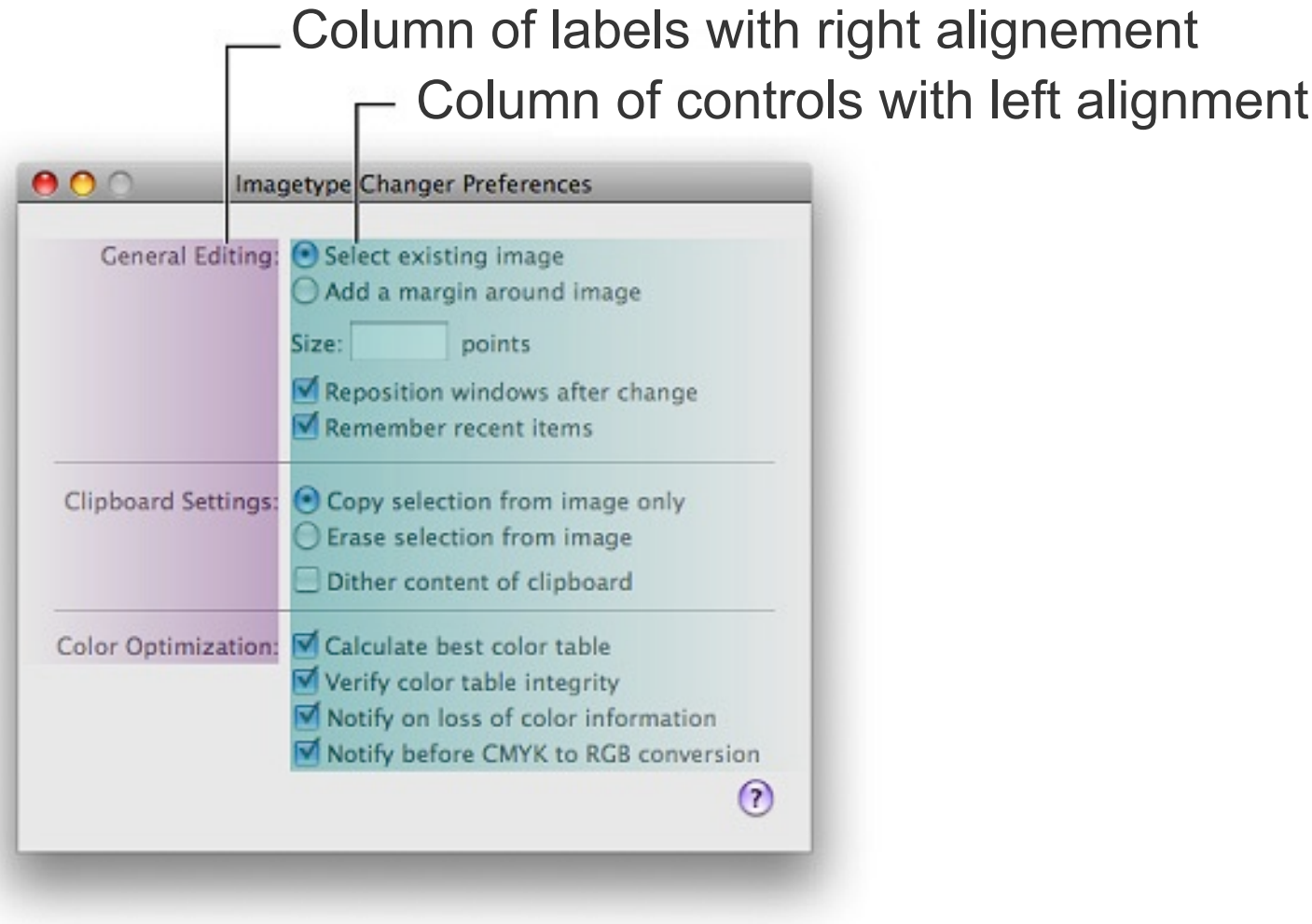
Placement guides (Mac OS X)

Center balance: visual balance of a container's content between the left and right parts



Placement guides (Mac OS X)

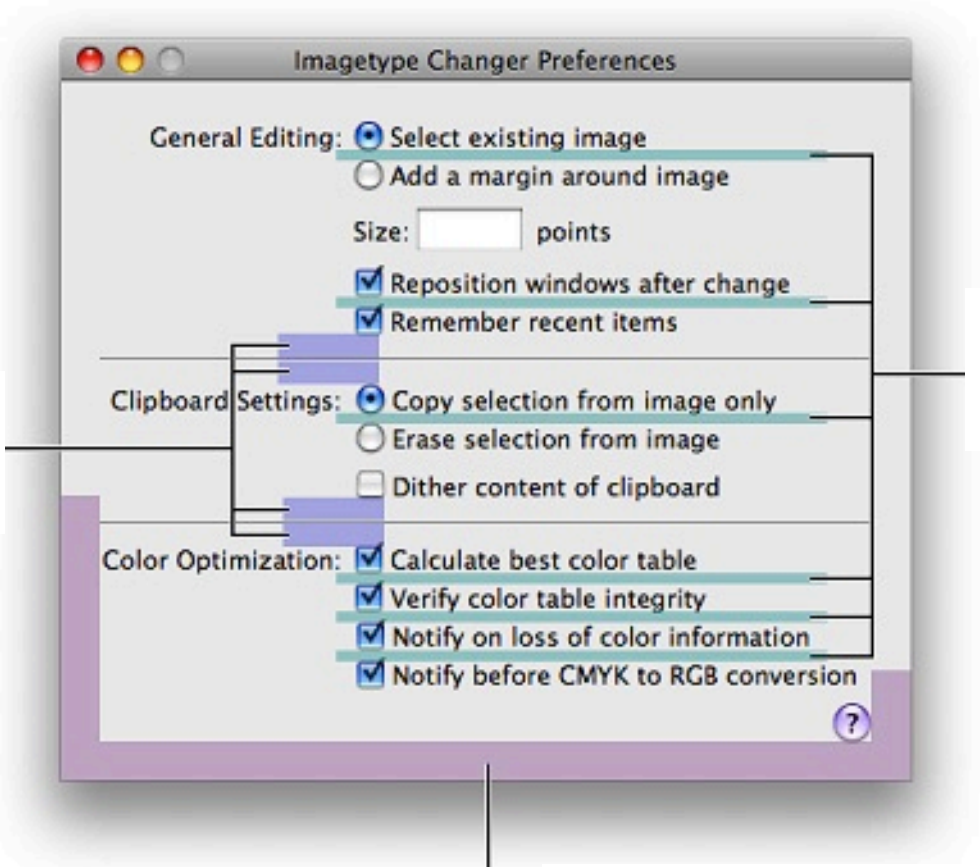
Alignement



Placement guides (Mac OS X)

Spacing

Same space before and after separator



Same space between controls

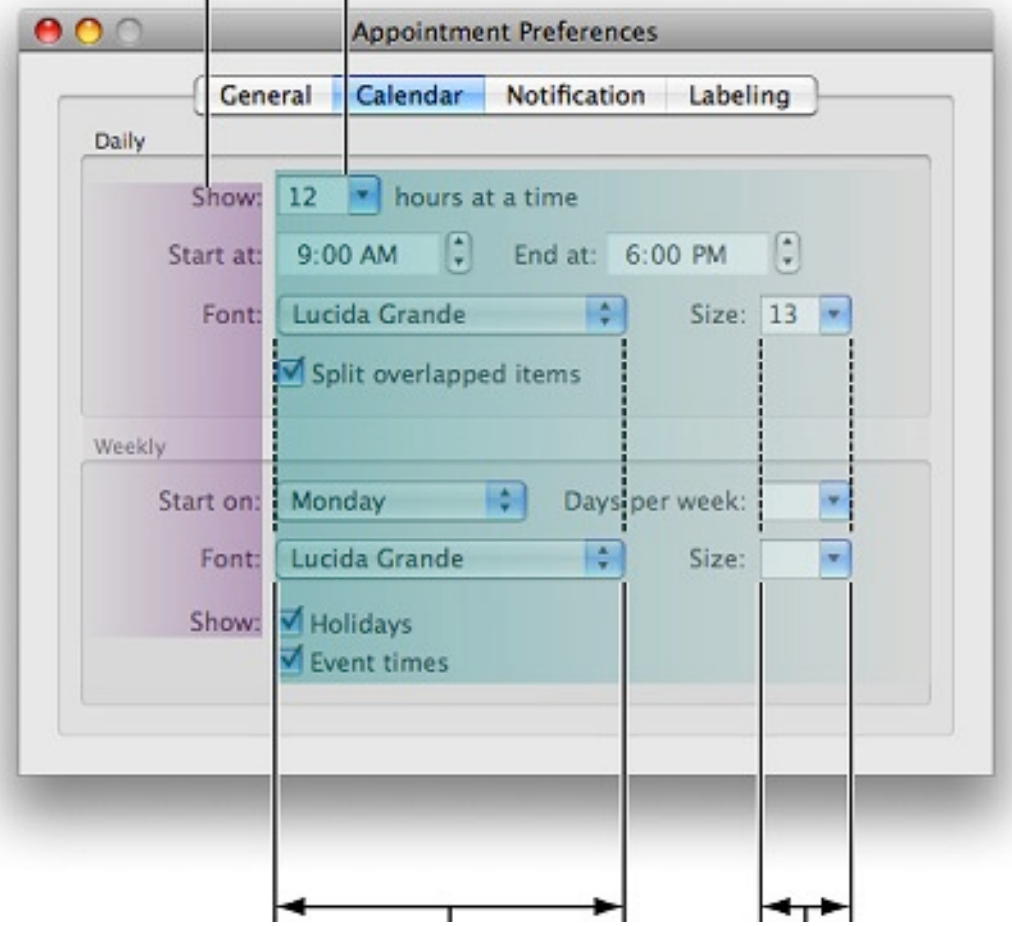
Same space on every side

Placement guides (Mac OS X)

Alignement and consistency

Column with labels with right alignement

Column of controls with left alignment



Consistency between controls of the same type

CRAP

contrast, repetition, alignment, proximity

Major sources: Designing Visual Interfaces, Mullet & Sano, Prentice Hall / Robin Williams Non-Designers Design Book, Peachpit Press

Good Design Is As Easy as 1-2-3

1. Learn the principles.
They're simpler than you might think.
2. Recognize when you're not using them.
Put it into words -- name the problem.
3. Apply the principles.
You'll be amazed.

Good design
is as easy as . . .

1 **Learn the principles.**
They're simpler than you might think.

2 **Recognize when you're not using them.**
Put it into words—name the problem.

3 **Apply the principles.**
You'll be amazed.

File Edit View Favorites Tools Help

Address C:\Documents and Settings\Saul Greenberg\My Documents\My Webs\~saul\hci_topics\powerpoint_presentations2003\graphical_design\index.h Go

A first lesson in Graphical Design

Contrast

Repetition

Alignment

Proximity

Example: this page,

home page

Original
Proximity 2
Alignment 3
Contrast 4
Repetition 5

File Edit View Favorites Tools Help

Address C:\Documents and Settings\Saul Greenberg\My Documents\My Webs\~saul\hci_topics\powerpoint_presentations2003\graphical_design\exampl Go

A First Lesson in Graphical Design

Contrast

Repetition

Alignment

Proximity

Examples

This page

Saul's Home Page
Proximity
Alignment
Contrast
Repetition

CRAP

Contrast

Repetition

Alignment

Proximity

CRAP

Contrast

make different things different
brings out dominant elements
mutes lesser elements
creates dynamism

Repetition
Alignment
Proximity

CRAP

Contrast

Repetition

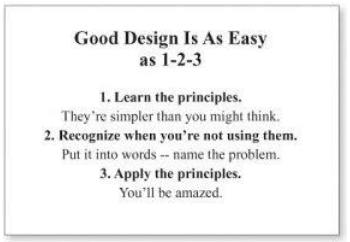
repeat design throughout the interface

consistency

creates unity

Alignment

Proximity



CRAP

Contrast

Repetition

Alignment

creates a visual flow

visually connects el.

Proximity

CRAP

Contrast

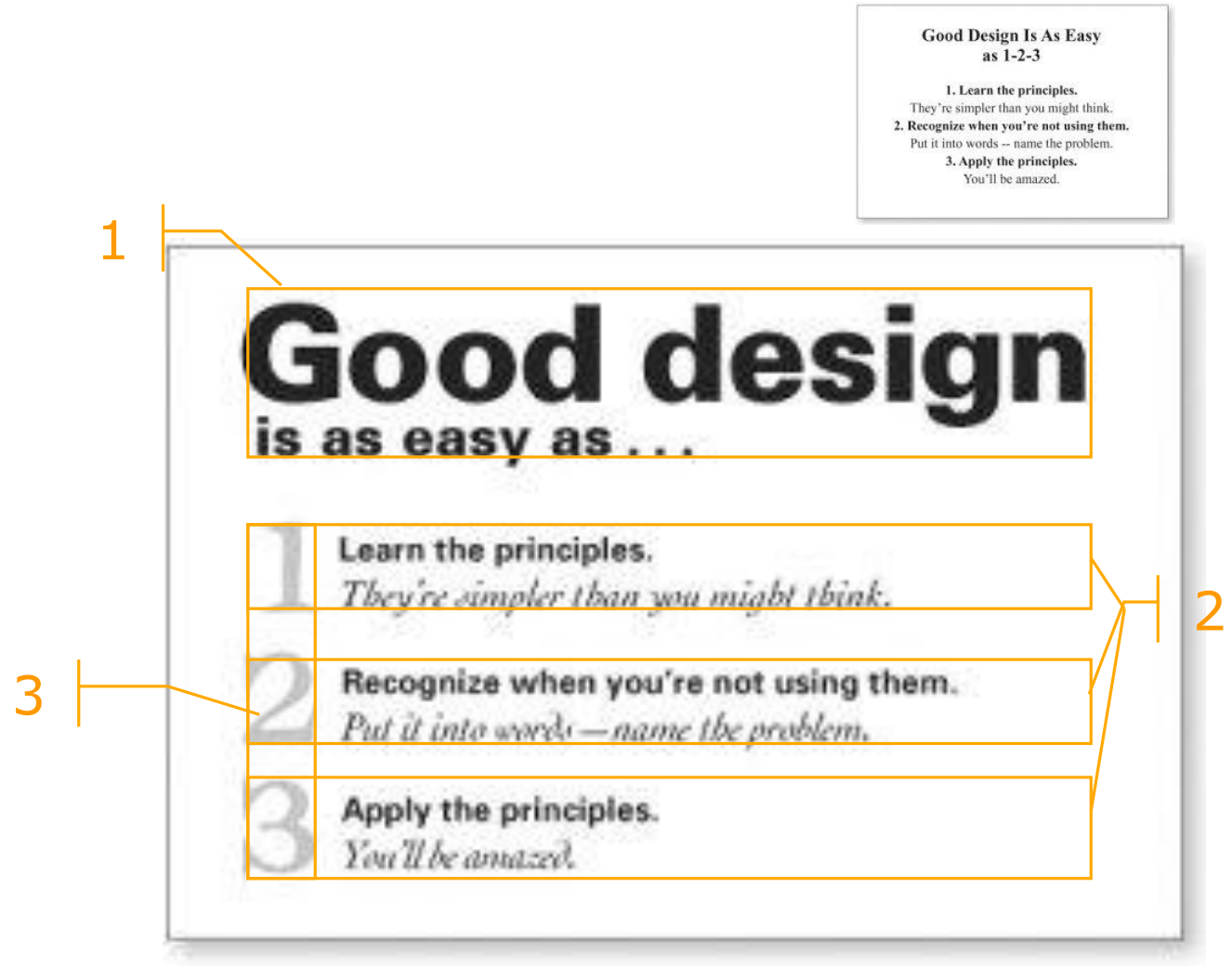
Repetition

Alignment

Proximity

groups related

separates unrelated



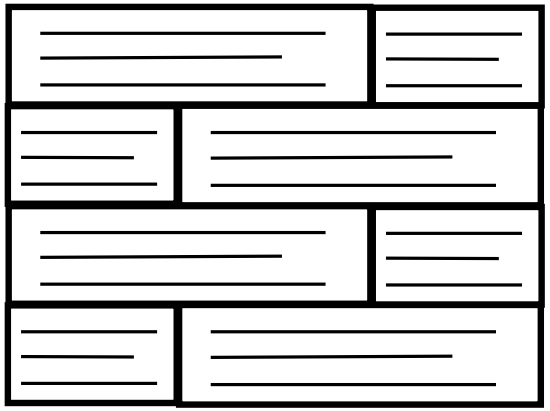
Where does your eye go?

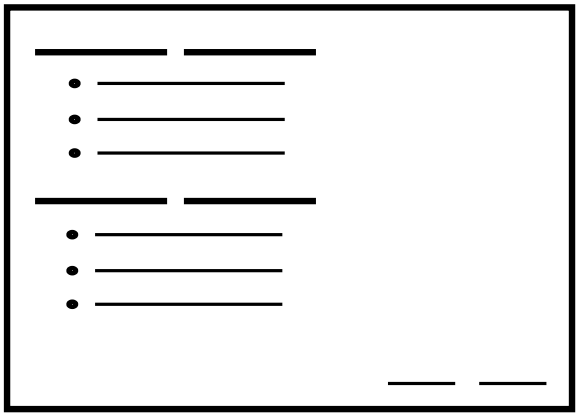
CRAP give you cues about how to read the graphic

Where does your eye go?

Boxes do not create a strong structure

- CRAP fixes it

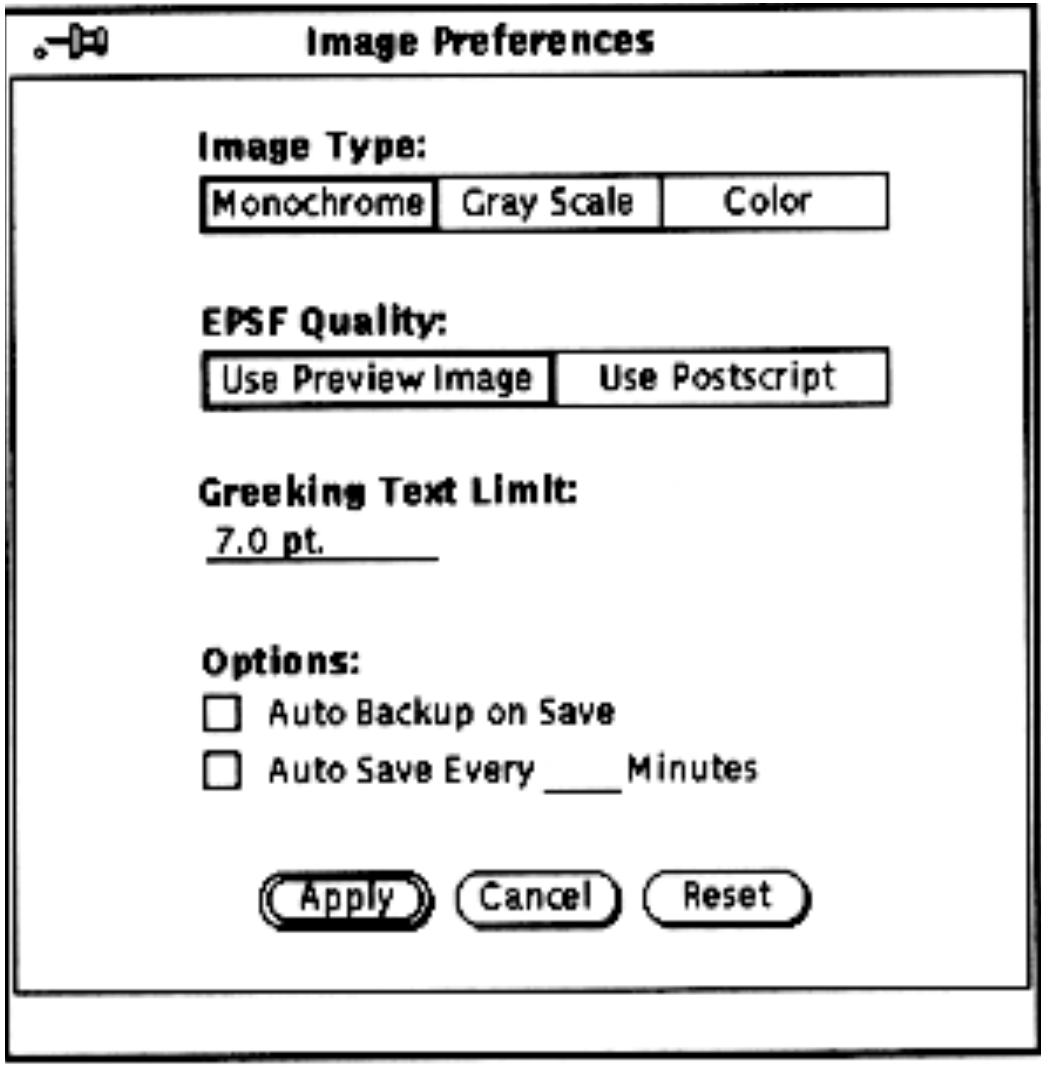




Where does your eye go?

Some contrast and weak proximity

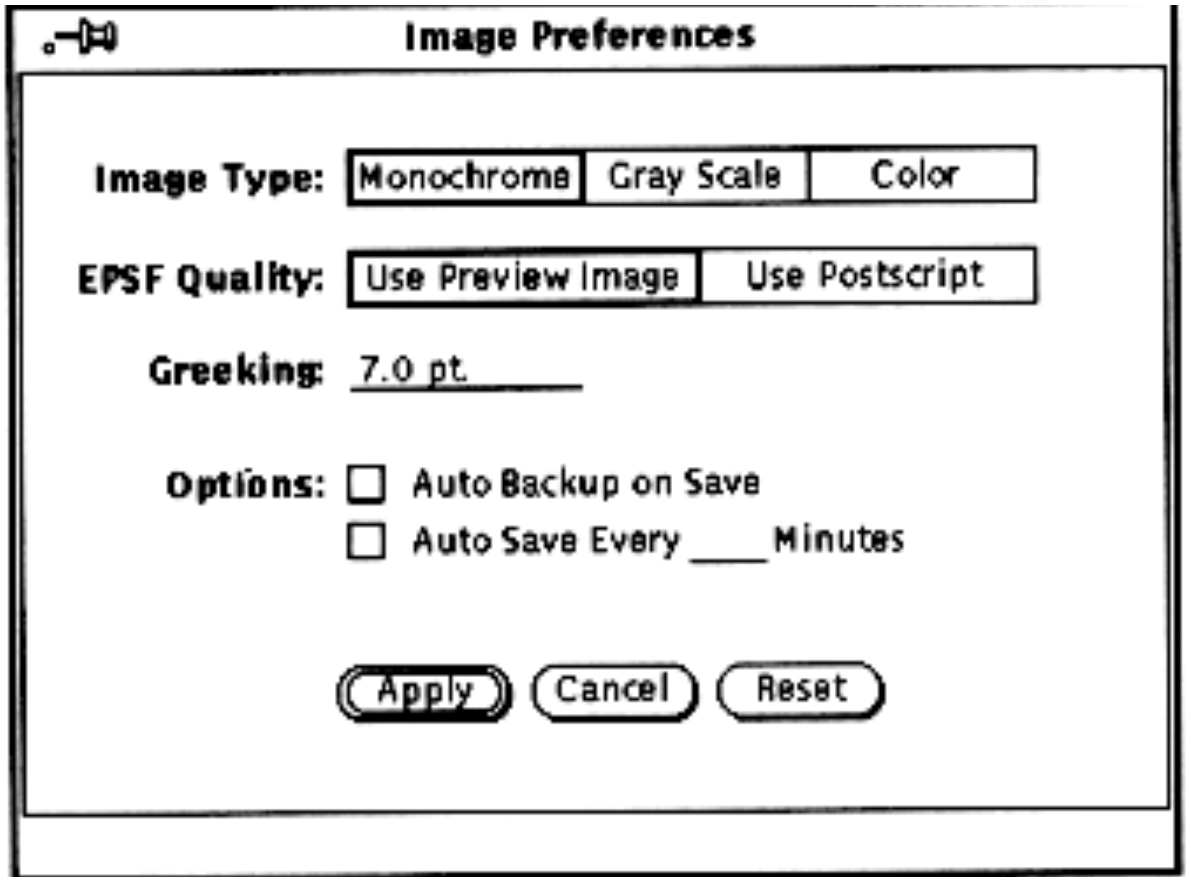
- ambiguous structure
- interleaved items



Where does your eye go?

Strong proximity (left/right split)

- unambiguous

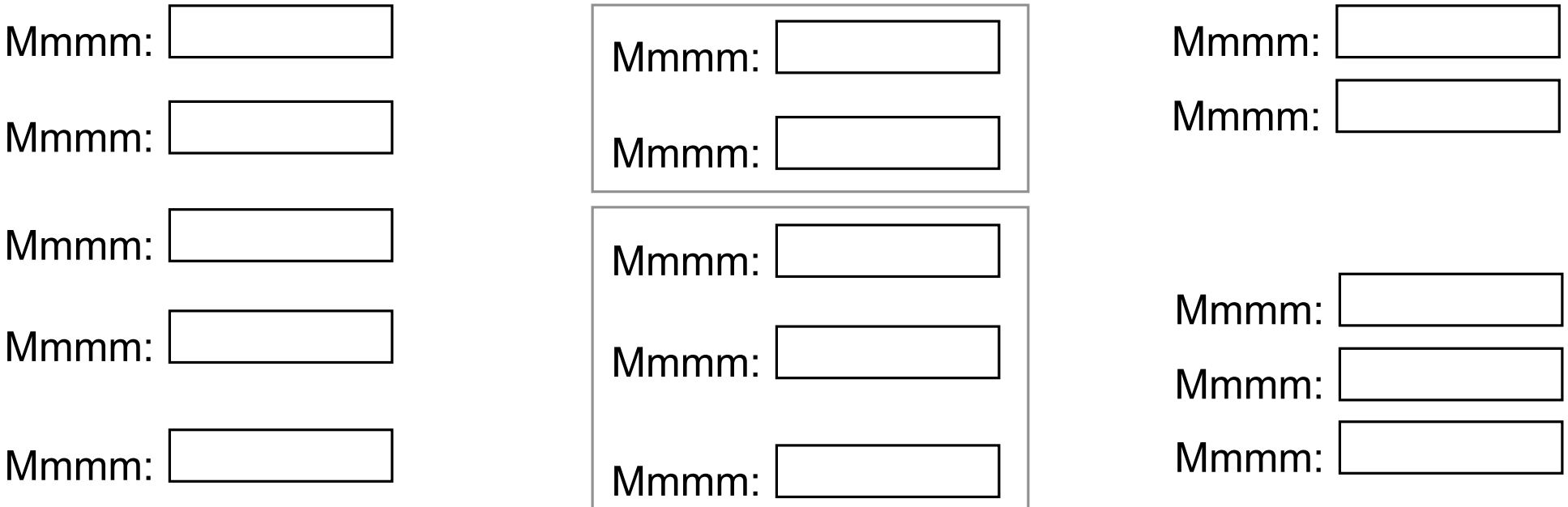


✓

Where does your eye go?

The strength of proximity

- alignment
- white (negative) space
- explicit structure a poor replacement



File Edit View Favorites Tools Help

Address C:\Documents and Settings\Saul Greenberg\My Documents\My Webs\~saul\hci_topics\powerpoint_presentations2003\graphical_design\index.h Go

A first lesson in Graphical Design

Contrast

Repetition

Alignment

Proximity

Example: this page,

home page

Original
Proximity 2
Alignment 3
Contrast 4
Repetition 5

File Edit View Favorites Tools Help

Address C:\Documents and Settings\Saul Greenberg\My Documents\My Webs\~saul\hci_topics\powerpoint_presentations2003\graphical_design\exampl Go

A First Lesson in Graphical Design

Contrast

Repetition

Alignment

Proximity

Examples

This page

Saul's Home Page
Proximity
Alignment
Contrast
Repetition

File Edit View Favorites Tools Help

Address C:\Documents and Settings\Saul Greenberg\My Documents\My Webs\~saul\hci_topics\powerpoint_presentations2003\graphical_design\versior Go

GroupLab
The University of Calgary

Saul Greenberg GroupLab Dept Computer Science University of Calgary

Saul Greenberg, Professor
Human-Computer Interaction &
Computer Supported Cooperative Work
Dept. of Computer Science
University of Calgary
Calgary, Alberta
CANADA T2N 1N4
Phone: +1 403 220-6087
Fax: +1 403 284-4707
Email: saul@cpsc.ucalgary.ca

Research

GroupLab project describes research by my group

Publications by our group; most available in HTML, PDF, and postscript

Project snapshots describes select projects done in Grouplab

Grouplab software repository

Grouplab people

Graduate Students

I have a few openings for MSc and PhD students who are interested in Human Computer Interaction and / or Computer Supported Cooperative Work. Some research and project ideas honors and graduate students

Courses offered this year

Original

File Edit View Favorites Tools Help

Address C:\Documents and Settings\Saul Greenberg\My Documents\My Webs\~saul\hci_topics\powerpoint_presentations2003\graphical_design\versior Go

GroupLab
The University of Calgary

Saul Greenberg GroupLab Dept Computer Science University of Calgary

Saul Greenberg, Professor
Human-Computer Interaction &
Computer Supported Cooperative Work

Dept. of Computer Science
University of Calgary
Calgary, Alberta
CANADA T2N 1N4

Phone: +1 403 220-6087
Fax: +1 403 284-4707
Email: saul@cpsc.ucalgary.ca

Research

GroupLab project describes research by my group
Publications by our group; most available in HTML, PDF, and postscript
Project snapshots describes select projects done in Grouplab
Grouplab software repository
Grouplab people

Graduate Students

I have a few openings for MSc and PhD students who are interested in Human Computer Interaction and / or Computer Supported Cooperative Work. Some research and project ideas honors and graduate students

Courses offered this year

CPSC 481: Foundations and Principles of Human Computer Intera
CPSC 581: Human Computer Interaction II: Interaction Desig

Proximity

File Edit View Favorites Tools Help

Address C:\Documents and Settings\Saul Greenberg\My Documents\My Webs\~saul\hci_topics\powerpoint_presentations2003\graphical_design\version Go

Saul Greenberg GroupLab Dept Computer Science University of Calgary

GroupLab
The University of Calgary

Saul Greenberg, Professor
Human-Computer Interaction &
Computer Supported Cooperative Work

Dept. of Computer Science
University of Calgary
Calgary, Alberta
CANADA T2N 1N4

Phone: +1 403 220-6087
Fax: +1 403 284-4707
Email: saul@cpsc.ucalgary.ca

Research
GroupLab project describes research by my group
Publications by our group; most available in HTML, PDF, and postscript
Project snapshots describes select projects done in Grouplab
Grouplab software repository
Grouplab people

Graduate Students
I have a few openings for MSc and PhD students who are interested in Human Computer Interaction and / or Computer Supported Cooperative Work. Some research and project ideas honors and graduate students

Courses offered this year
CPSC 481: Foundations and Principles of Human Computer Interaction
CPSC 581: Human Computer Interaction II: Interaction Design
CPSC 601.13: Computer Supported Cooperative Work

Previous Years:
CPSC 681: Research Methodologies in Human Computer Interaction

Alignment

File Edit View Favorites Tools Help

Address C:\Documents and Settings\Saul Greenberg\My Documents\My Webs\~saul\hci_topics\powerpoint_presentations2003\graphical_design\versior Go

Saul Greenberg GroupLab Dept Computer Science University of Calgary

GroupLab
The University of Calgary

Saul Greenberg
Professor

Human-Computer Interaction &
Computer Supported Cooperative Work

Dept. of Computer Science
University of Calgary
Calgary, Alberta
CANADA T2N 1N4

Phone: +1 403 220-6087
Fax: +1 403 284-4707
Email: saul@cpsc.ucalgary.ca

Graduate Students — **Research Ideas.** I have a few openings for MSc and PhD students who are interested in Human Computer Interaction and / or Computer Supported Cooperative Work.

Courses offered this year

CPSC 481: Foundations and Principles of Human Computer Interaction
CPSC 581: Human Computer Interaction II: Interaction Design
CPSC 601.13: Computer Supported Cooperative Work

Previous Years

CPSC 681: Research Methodologies in Human Computer Interaction
CPSC 699: Research Methodology for Computer Science (old!)
CPSC 601.48: Special Topics: Heuristic Evaluation
CPSC 601.56: Advanced Topics in HCI: Media Spaces and Casual Interaction
SENG 609.05: Graphical User Interfaces: Design and Usability
SENG 609.06: Special Topics in Human Computer Interaction
Ego alert: My entry on U Calgary's 'Great Teachers" Web Site

Administration — **Ethics Committee** for research with human subjects; I am the chair

Contrast

File Edit View Favorites Tools Help

Address C:\Documents and Settings\Saul Greenberg\My Documents\My Webs\~saul\hci_topics\powerpoint_presentations2003\graphical_design\versior Go

Saul Greenberg GroupLab Dept Computer Science University of Calgary

GroupLab
The University of Calgary

Saul Greenberg
Professor

Human-Computer Interaction &
Computer Supported Cooperative Work

Dept. of Computer Science
University of Calgary
Calgary, Alberta
CANADA T2N 1N4

Phone: +1 403 220-6087
Fax: +1 403 284-4707
Email: saul@cpsc.ucalgary.ca

Graduate Students — Research Ideas I have a few openings for MSc and PhD students who are interested in Human Computer Interaction and / or Computer Supported Cooperative Work.

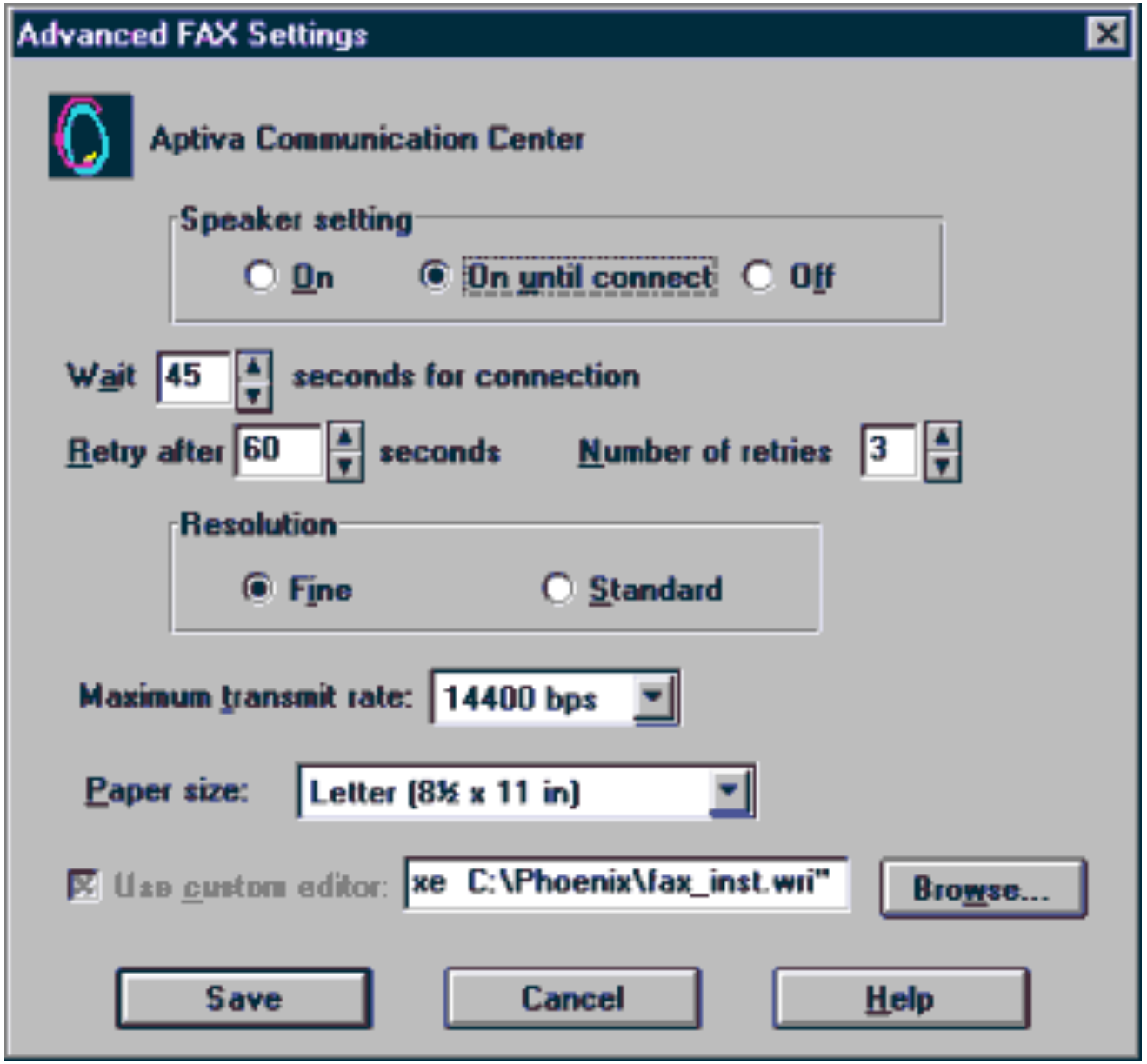
Courses offered this year
- CPSC 481 Foundations and Principles of Human Computer Interaction
- CPSC 581 Human Computer Interaction II: Interaction Design
- CPSC 601.13 Computer Supported Cooperative Work

Previous Years
- CPSC 681 Research Methodologies in Human Computer Interaction
- CPSC 699 Research Methodology for Computer Science (old!)
- CPSC 601.48 Special Topics: Heuristic Evaluation
- CPSC 601.56 Advanced Topics in HCI: Media Spaces and Casual Interaction
- SENG 609.05 Graphical User Interfaces: Design and Usability
- SENG 609.06 Special Topics in Human Computer Interaction
- Ego alert My entry on U Calgary's 'Great Teachers" Web Site

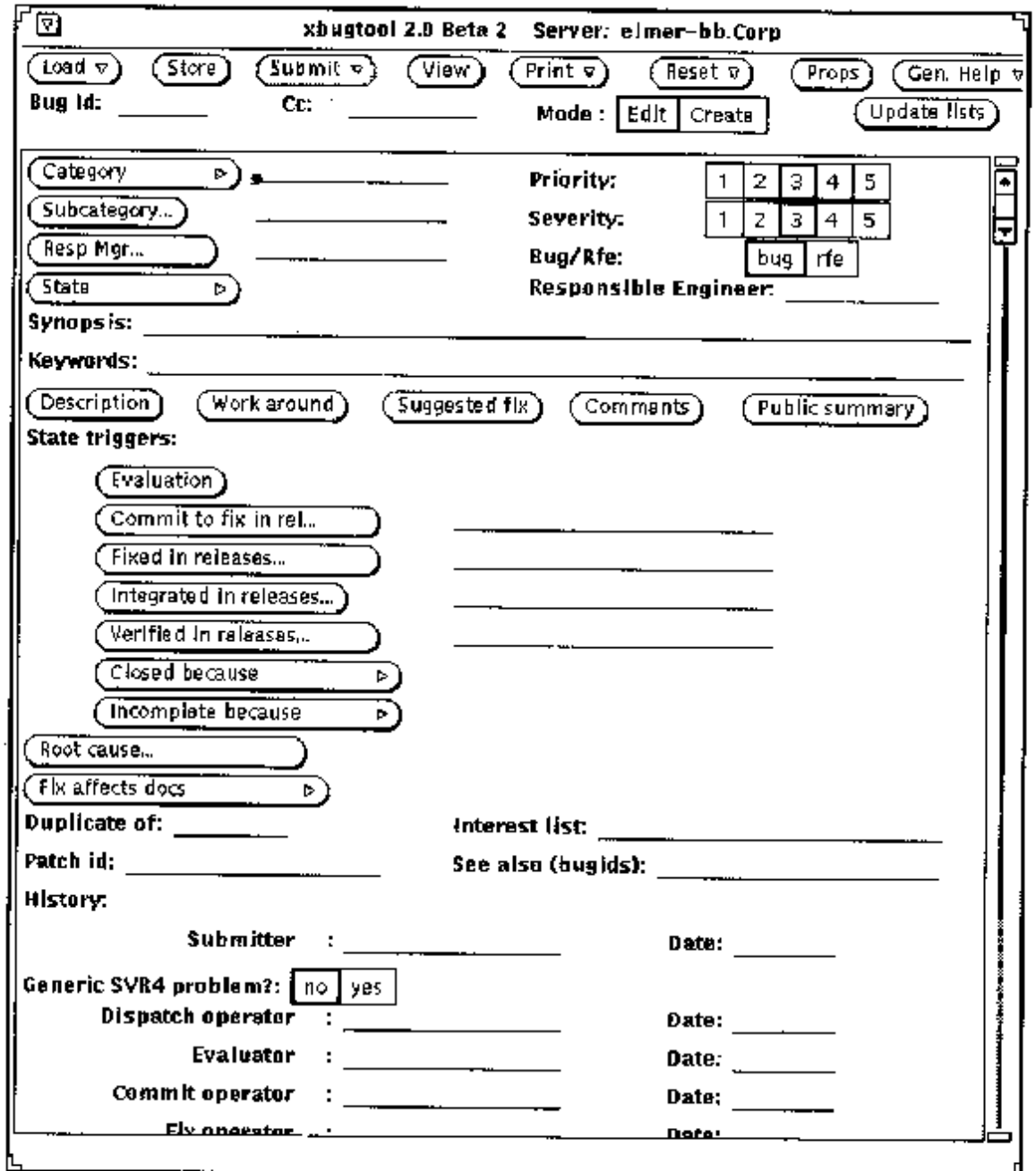
Administration — Ethics Committee for research with human subjec

Repetition

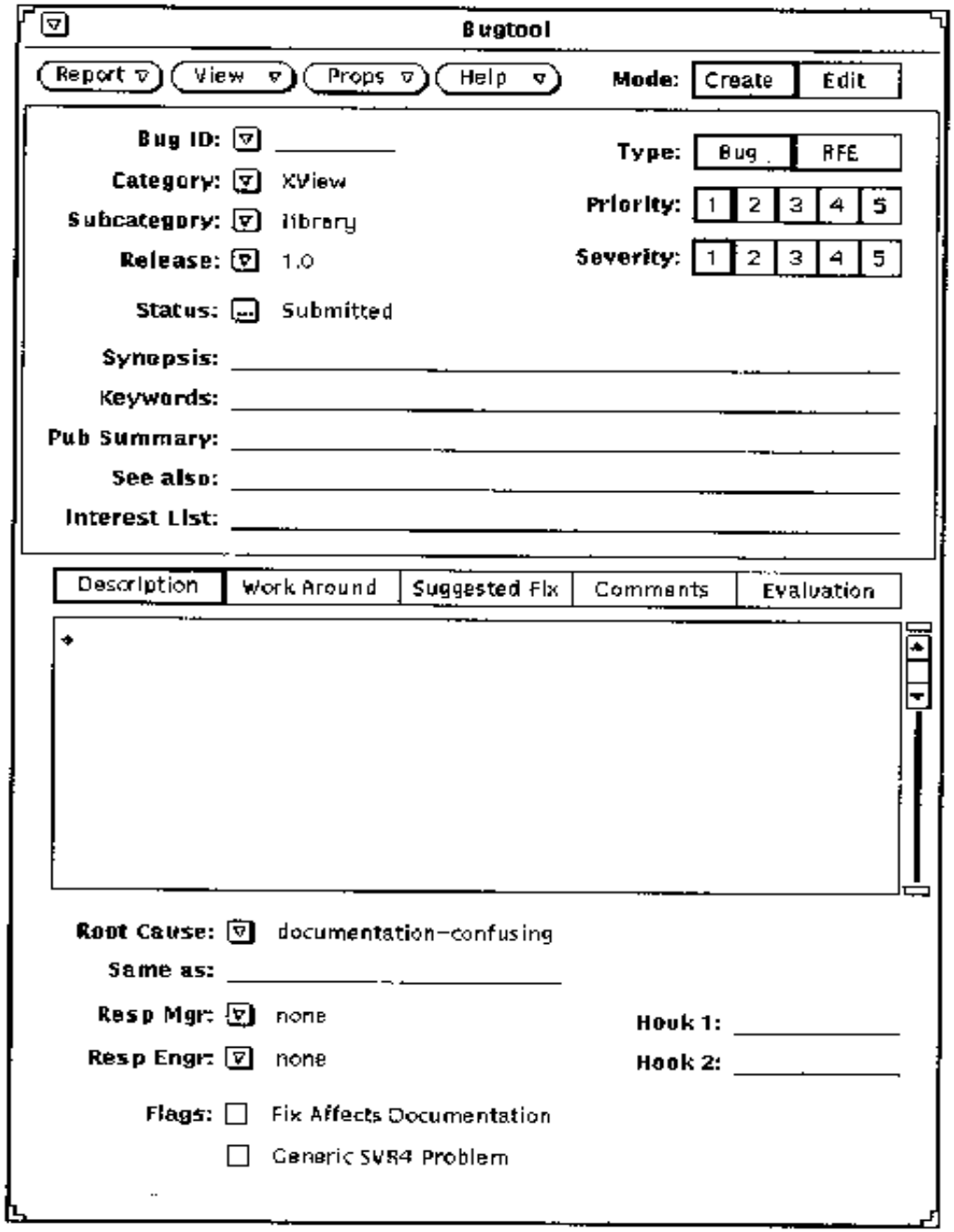
Example of bad design



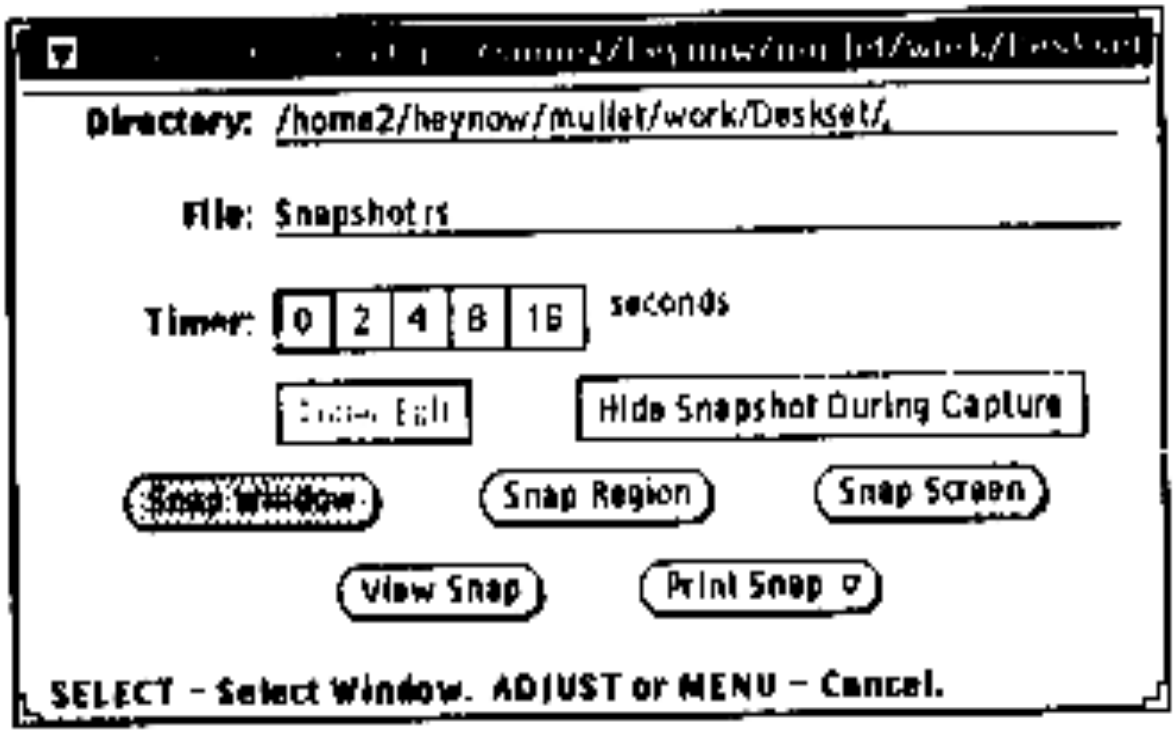
Example of bad design

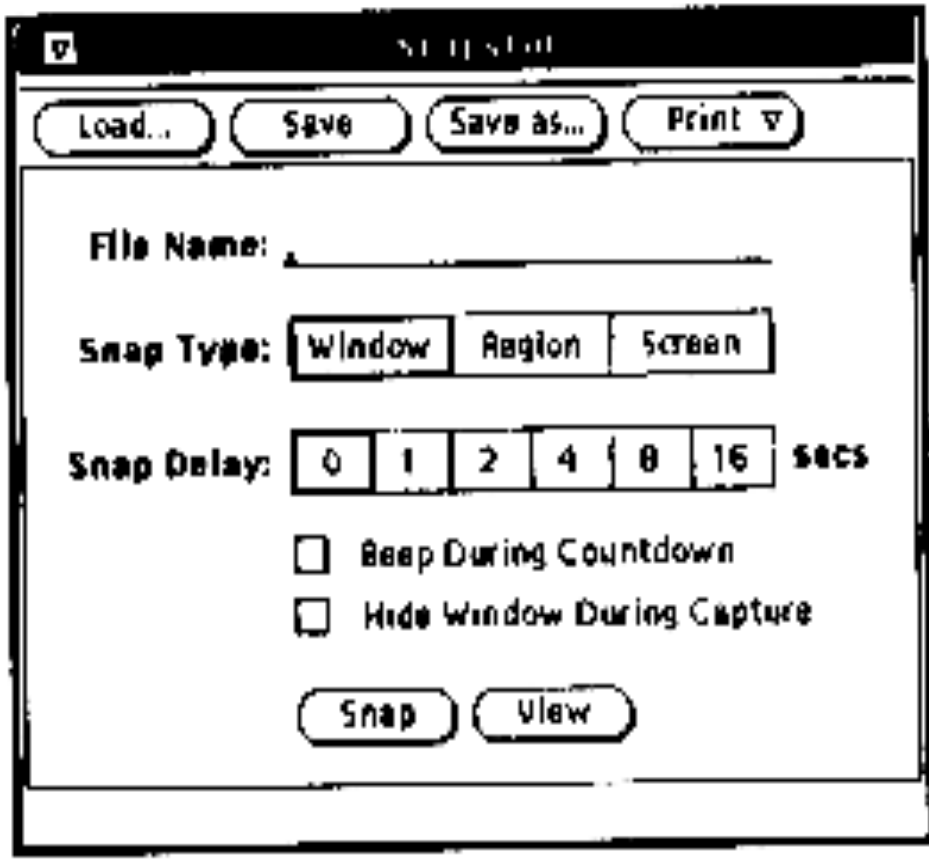


Reparing the layout



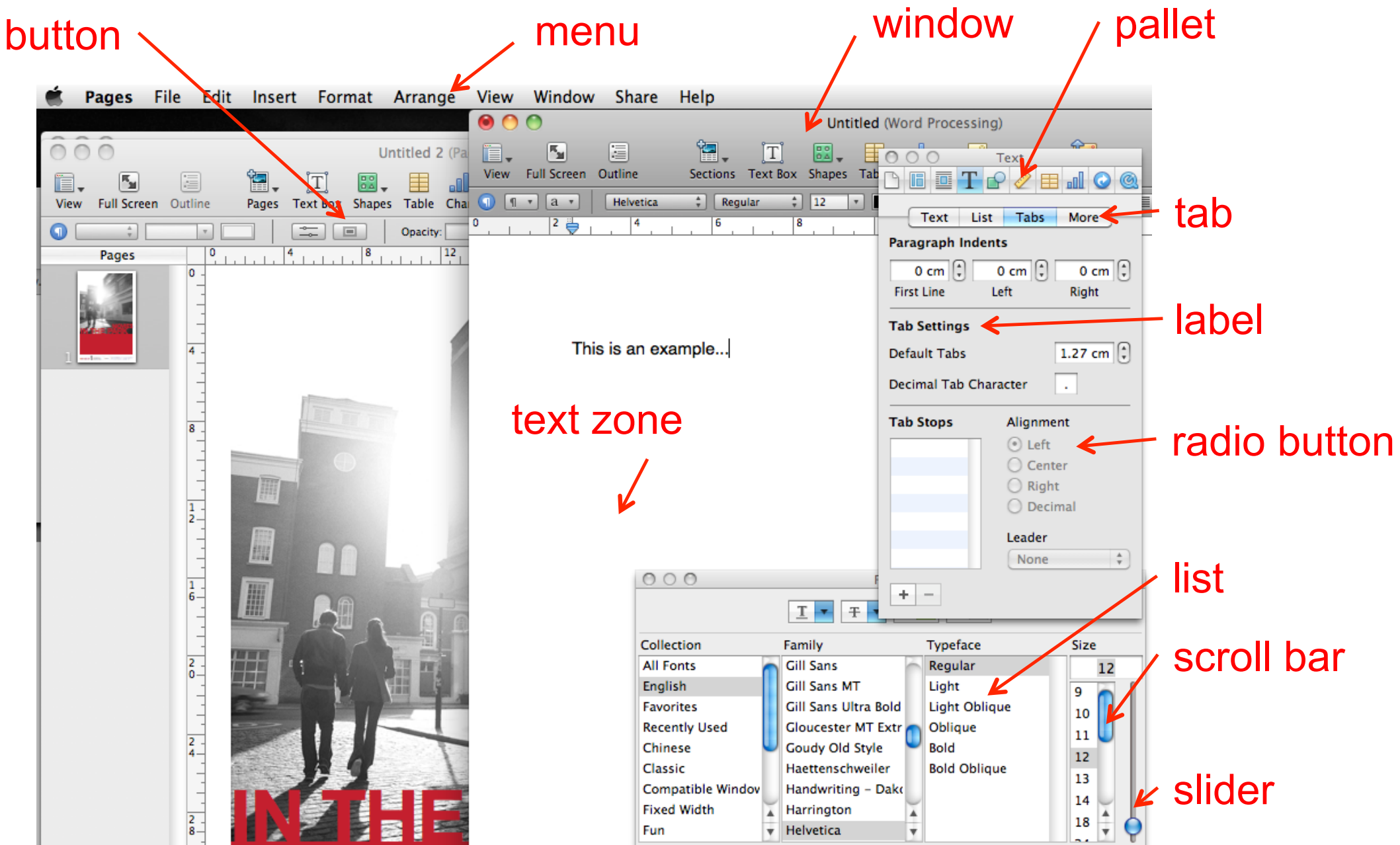
Reparing the layout





Facets of a widget

« widgets » (window gadgets)

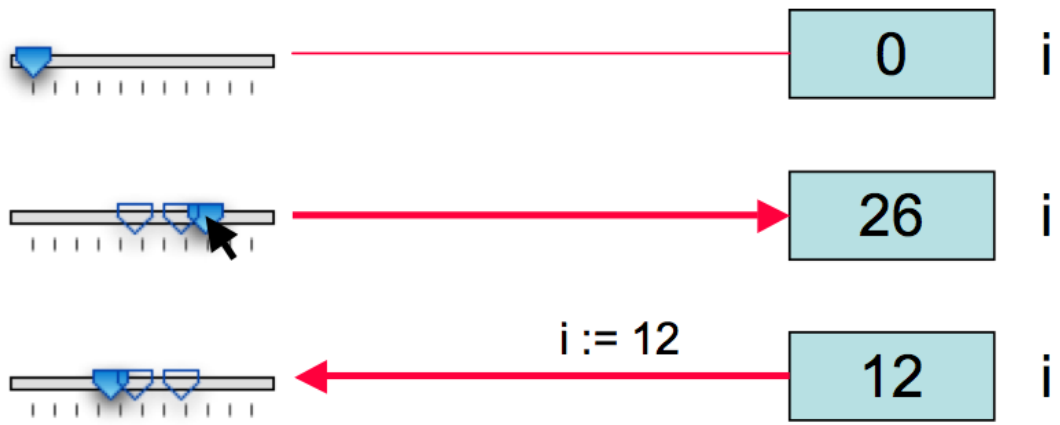


Facets of a widget

- Presentation
  - appearance
- Behavior
  - reaction to user actions
- Interface with the application
  - notification of state changes
- Example: Button
  - border with text inside
  - « pressing » or « releasing » animation when clicked
  - call function when the button is clicked

Variable wrappers (active variables)

two-way link between a state variable of a widget and another application variable
(in Tcl/Tk referred to as *tracing*)



problems

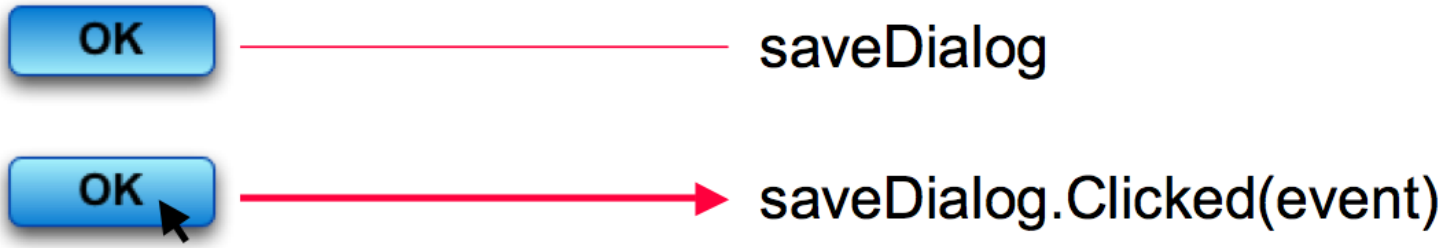
- limited to simple types
- return link can be costly if automatic
- errors when links are updated by programmers

Event dispatching

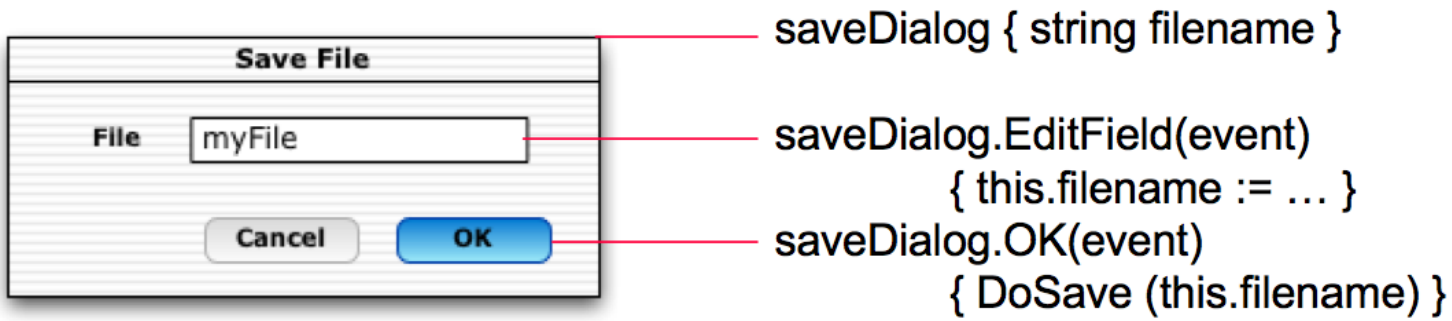
widgets act as input peripherals and send events when their state changes

a while loop reads and treats events

associate an object to a widget, and its methods to changes in the widget state



Event dispatching



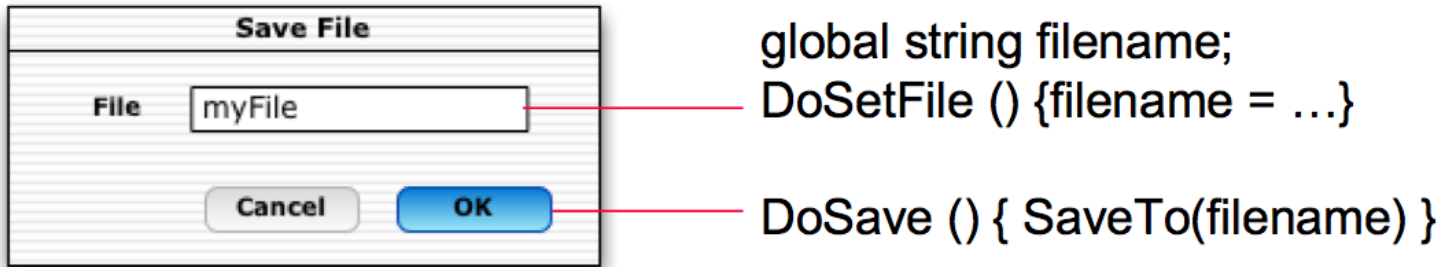
divide event sending and treatment

better encapsulation (inside widget class)

Callback functions

Registration at widget creation

Call at widget activation



Callback functions

Problem: spaghetti of callbacks

Sharing a state between multiple callbacks by

- global variables that widgets check:
  too many in real applications

- widget trees: callback functions are called with a reference to the widget that called it (visible in the same tree)
  Fragile if we change the structure of the UI, does not deal with other data not associated to widgets (e.g. filename)

- token passing: data passed with the callback function call

Callback functions

```
/* callback function */
void DoSave (Widget w, void* data) {
     /* retrieve file name */
      filename = (char**) data;
     /* call an application function */
     SaveTo (filename);
     /* close the dialog */
     CloseWindow (getParent(getParent(w)));
}
```

```
/* main program */
main () {
     /* variable with file name */
     char* filename = “”;
     …
     /* create a widget and assosiate a callback */
     ok = CreateButton (....);
     RegisterCallback (ok, DoSave, (void*) &filename);
     …
     /* event manager loop */
     MainLoop ();
}
```

Event listeners (Java)

a variation of callbacks in Java:

methods of type **AddListener** that do not specify a callback function but an object (the *listener*)

when a widget changes state, it triggers a predefined method of the *listener* object (e.g. *actionPerformed*)

Event listeners (Java)

```
public class ClickListener implements ActionListener
{
    public void actionPerformed(ActionEvent e){
        JButton button = (JButton)e.getSource();
        …
    }
}

…
ClickListener listener = new ClickListener();
JButton button = new JButton(''Click me'');
button.addActionListener(listener);
…
```

Event listeners (Java)

Anonymous Inner classes

```
…
button.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                    …
            }
});
…
panel.addMouseListener(new MouseAdapter(){
            public void mouseClicked(MouseEvent e){
                    …
            }
});
```

Methods and events are predefined

Event listeners (Java)

Anonymous Inner classes

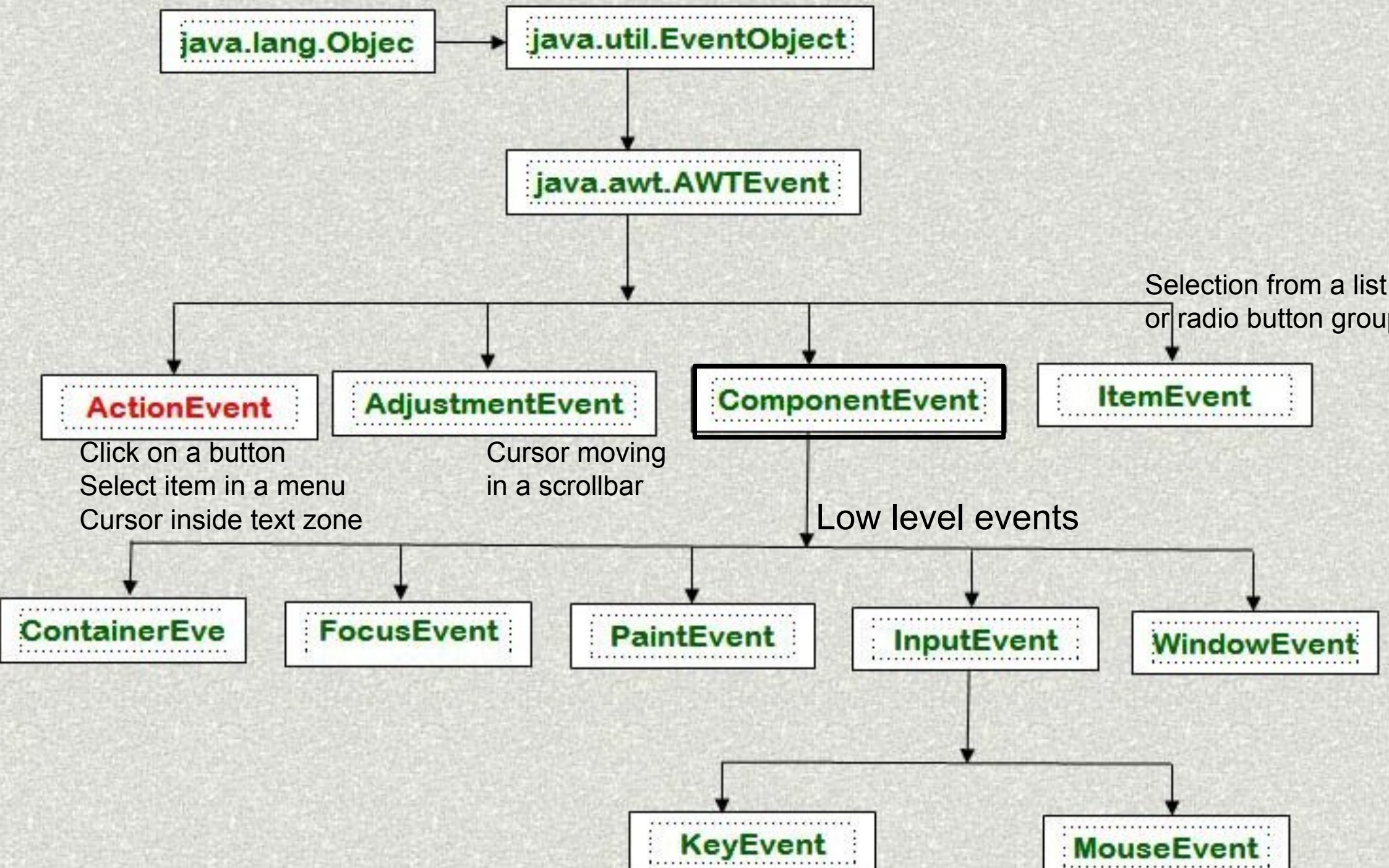
“new <class-name> () { <body> }”

this construction does 2 things:

- creates a new class without name, that is a subclass of <class-name> defined by <body>
- creates a (unique) instance of this new class and returns its value

this (inner) class has access to variables and methods of the class inside which it is defined

Events (Java)


java.lang.Objec
java.util.EventObject
java.awt.AWTEvent
Selection from a list
or radio button grou
ActionEvent
AdjustmentEvent
ComponentEvent
ItemEvent
Click on a button
Select item in a menu
Cursor inside text zone
Cursor moving
in a scrollbar
Low level events
ContainerEve
FocusEvent
PaintEvent
InputEvent
WindowEvent
KeyEvent
MouseEvent

Events and listeners (Java)

Each has a source (e.g. JButton, JRadioButton, JCheckBox, JToggleButton,JMenu, JRadioButtonMenuItem, JTextField)

Can get it with the function **getSource()**

(Listeners) need to implement the interface that corresponds to event
e.g. ActionEvent => ActionListener :

```
public interface ActionListener extends EventListener {
    /** Invoked when an action occurs.*/
    public void actionPerformed(ActionEvent e)
}
```

Events and listeners (Java)

all events inherit from the class EventObject

all listeners correspond to an interface that inherits from EventListener

a class receiving notification events of some type needs to implement the corresponding interface:

ActionEvent	ActionListener
MouseEvent	MouseListener
KeyEvent	KeyListener
...	

Events and listeners (Java)

listeners need to be registered (added) to widgets

a listener can be added to multiple widgets

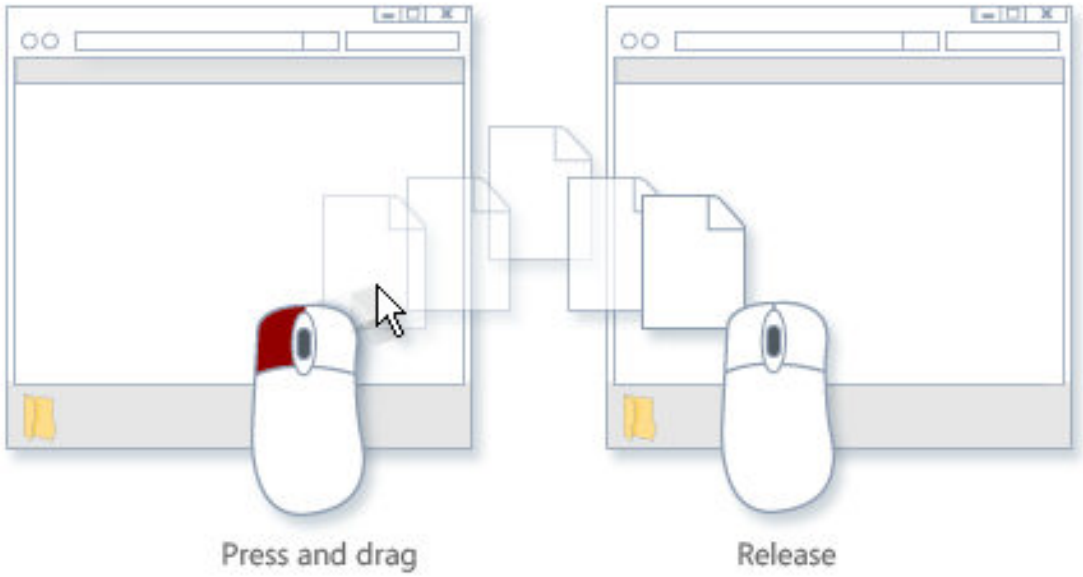
- e.g. one listener handles events from multiple buttons

a widget can have many listeners

- e.g. one for “click” events and for “enter” on button events

« drag-and-drop » to think about

What are the affected « widgets »?
What are the events?



How to describe this interaction with a « event listener » ?

Interface toolkits

Event-action model

- can lead to errors (e.g. forgotten events)
- difficult to extend (e.g. add hover events)
- complex code

Hard to do things the toolkit was not designed for

e.g., multi-device input, multi-screen applications, advanced interaction techniques (CrossY)