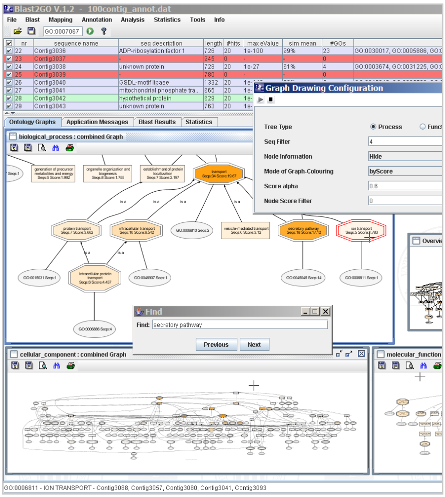
Week 1 :
c. UI Programming

Anastasia.Bezerianos@lri.fr

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

interactive systems



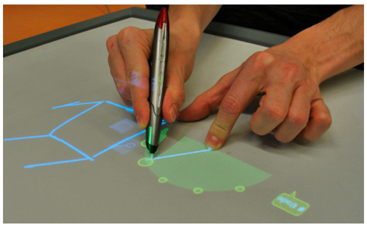
☑	nr	sequence name	seq description	length	#hits	max eValue	sim mean	#GOs	
☑	22	Contig3036	ADP-ribosylation factor 1	726	20	1e-100	99%	23	GO:0030017, GO:0005886, GO:0
☑	23	Contig3037	-	945	0	-	-	0	
☑	24	Contig3038	unknown protein	728	20	1e-27	61%	4	GO:0003674, GO:0031225, GO:0
☑	25	Contig3039	-	780	0	-	-	0	
☑	26	Contig3040	GSDL-motif lipase	1332	20	1e-			
☑	27	Contig3041	mitochondrial phosphate tra...	665	20	1e-			
☑	28	Contig3042	hypothetical protein	629	20	1e-			
☑	29	Contig3043	unknown protein	763	20	1e-			

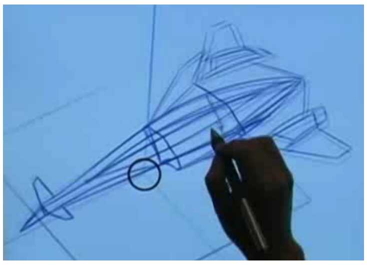
Graph Drawing Configuration

Tree Type ● Process ○ Funct
Seq Filter 4
Node Information Hide
Mode of Graph-Colouring byScore
Score alpha 0.6
Node Score Filter 0

Find
Find: secretory pathway

GO:0006811 - ION TRANSPORT - Contig3088, Contig3057, Contig3080, Contig3041, Contig3093

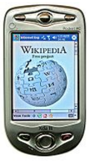




graphical interfaces

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

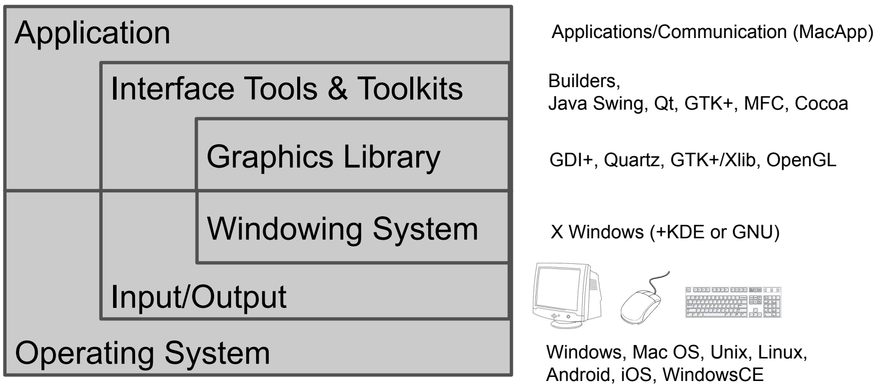


WIMP interfaces

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



Application

Interface Tools & Toolkits

Graphics Library

Windowing System

Input/Output

Operating System

Applications/Communication (MacApp)

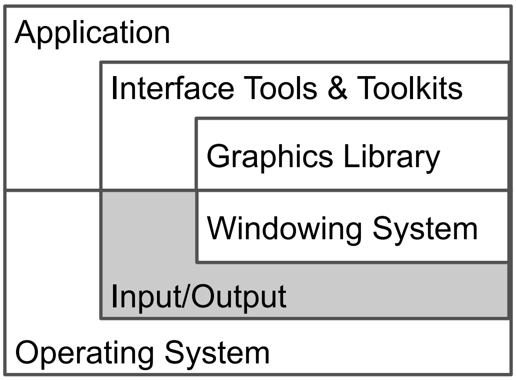
Builders,
Java Swing, Qt, GTK+, MFC, Cocoa

GDI+, Quartz, GTK+/Xlib, OpenGL

X Windows (+KDE or GNU)

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

software layers



Application

Interface Tools & Toolkits

Graphics Library

Windowing System

Input/Output

Operating System

input/output peripherals

input: where we give commands

output: where the system shows 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

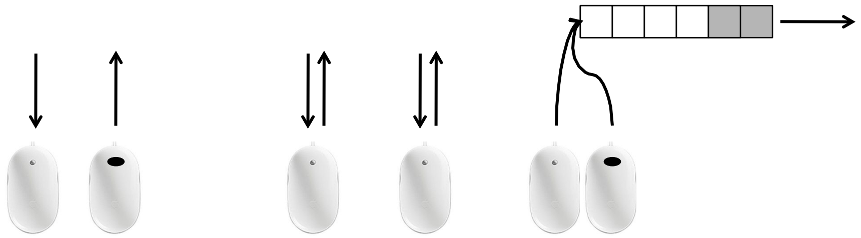
Query & wait
1 per. at a time

Polling

Active wait
Polling in sequence
CPU cost

Events

Wait queue

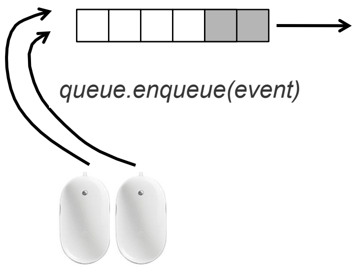


event based (driven) programming

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



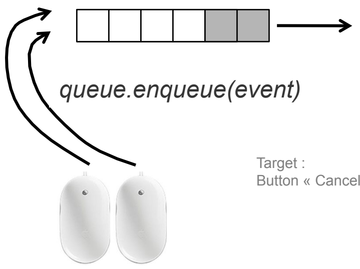
Animations : « clock » source of events
« tick » -> event -> animation progression

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



Target :
Button « Cancel »

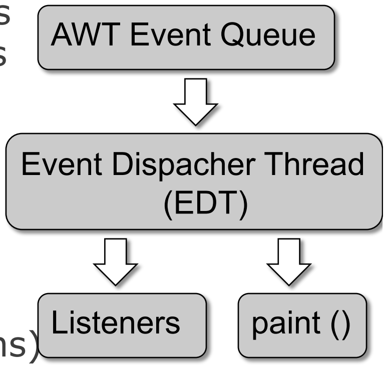
```
processEvent(event)
   target <- FindTarget (event)
   if (target ≠ NULL)
        target.processEvent(event)
```

Animations : « clock » source of events
« tick » -> event -> animation progression

e.g. Swing (and AWT)

3 threads in JVM:

- main ()
- toolkit thread that receives (from OS) events and puts them in a queue
- EDT manages the queue: sends events to *listeners* (functions dealing with events) and calls paint methods (drawing functions)



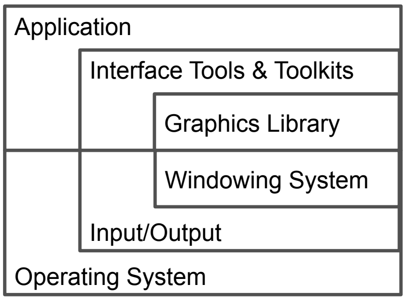
AWT Event Queue

Event Dispacher Thread (EDT)

Listeners

paint ()

event handling



Application

Interface Tools & Toolkits

Graphics Library

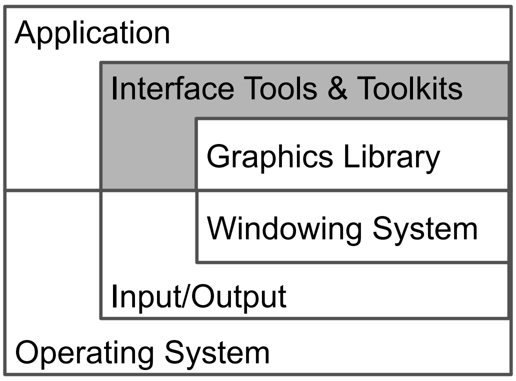
Windowing System

Input/Output

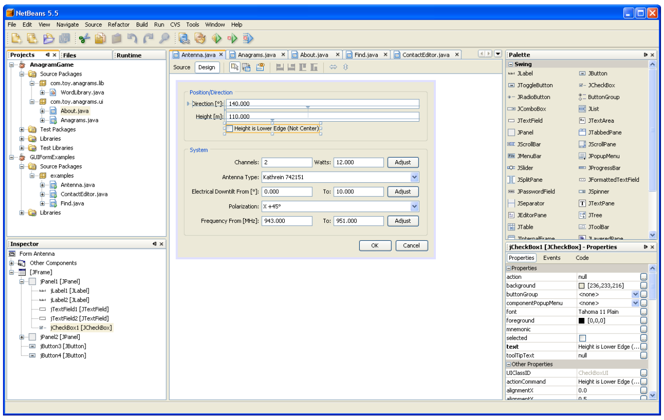
Operating System

Lower layers fill-up the queue
Upper layers de-queue and treat events

software layers



interface builders


NetBeans 5.5
File Edit View Navigate Source Refactor Build Run CVS Tools Window Help
Projects Files Runtime
Antenna.java Anagrams.java About.java Find.java ContactEditor.java
Source Design
Palette
Inspector

Position/Direction

Direction [°]: 140.000

Height [m]: 110.000

□ Height is Lower Edge (Not Center)

System

Channels: 2 Watts: 12.000 Adjust

Antenna Type: Kathrein 742151

Electrical Downtilt From [°]: 0.000 To: 10.000 Adjust

Polarization: X +45°

Frequency From [MHz]: 943.000 To: 951.000 Adjust

OK Cancel

jCheckBox1 [JCheckBox] - Properties

Properties	
action	null
background	[236,233,216]
buttonGroup	<none>
componentPopupMenu	<none>
font	Tahoma 11 Plain
foreground	[0,0,0]
mnemonic	
selected	□
text	Height is Lower Edge (...
toolTipText	null
Other Properties	
UIClassID	CheckBoxUI
actionCommand	Height is Lower Edge (...
alignmentX	0.0

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

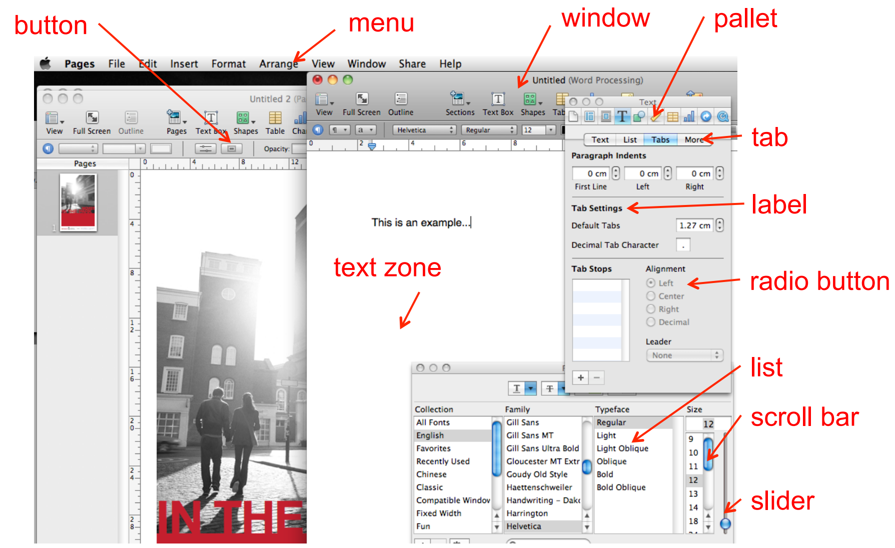
usually also handle input events (later)

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? ….

« widgets » (window gadget)



button

menu

window

pallet

tab

label

radio button

text zone

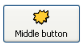
list

scroll bar

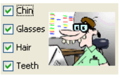
slider

This is an example...

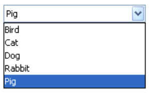
Swing widgets



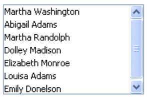
JButton



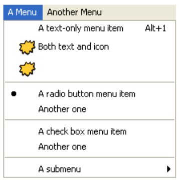
JCheckBox



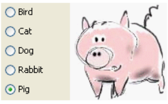
JComboBox



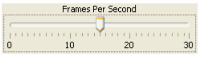
JList



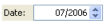
JMenu



JRadioButton



JSlider

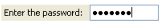


Date: 07/2006

JSpinner

City: Santa Rosa

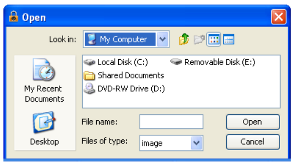
JTextField



Enter the password: •••••••

JPasswordField

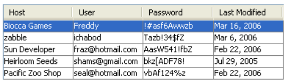
Swing widgets



File name:

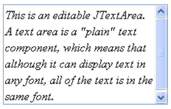
Files of type: image

JFileChooser



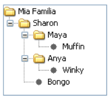
Host	User	Password	Last Modified
Biocca Games	Freddy	!#asf6Awwzb	Mar 16, 2006
zabble	ichabod	Tazb!34$fZ	Mar 6, 2006
Sun Developer	fraz@hotmail.com	AasW541!fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail.com	bkz[ADF78!	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.com	vbAf124%z	Feb 22, 2006

JTable

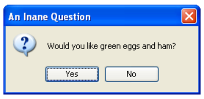


This is an editable JTextArea. A text area is a "plain" text component, which means that although it can display text in any font, all of the text is in the same font.

JTextArea



JTree

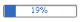


Would you like green eggs and ham?

JDialog

JFrame

JLabel



JProgressBar

JSeparator

JToolTip

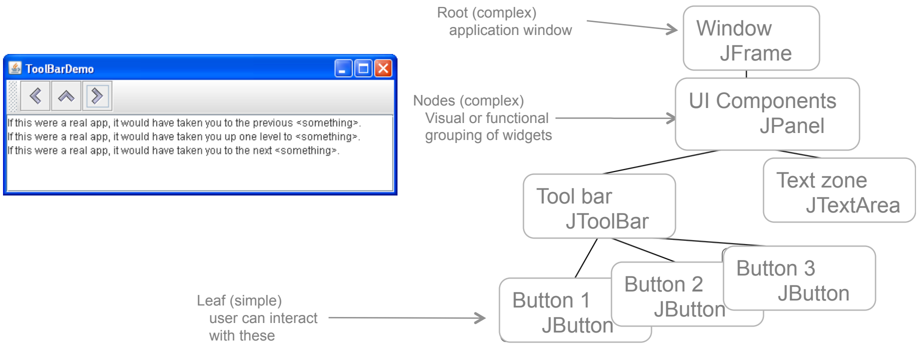
widget complexity

- 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 »



If this were a real app, it would have taken you to the previous <something>.
If this were a real app, it would have taken you up one level to <something>.
If this were a real app, it would have taken you to the next <something>.

Root (complex)
application window

Window
JFrame

Nodes (complex)
Visual or functional
grouping of widgets

UI Components
JPanel

Tool bar
JToolBar

Text zone
JTextArea

Leaf (simple)
user can interact
with these

Button 1
JButton

Button 2
JButton

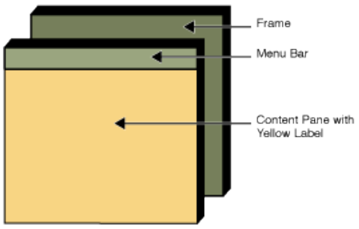
Button 3
JButton

Swing widget classes

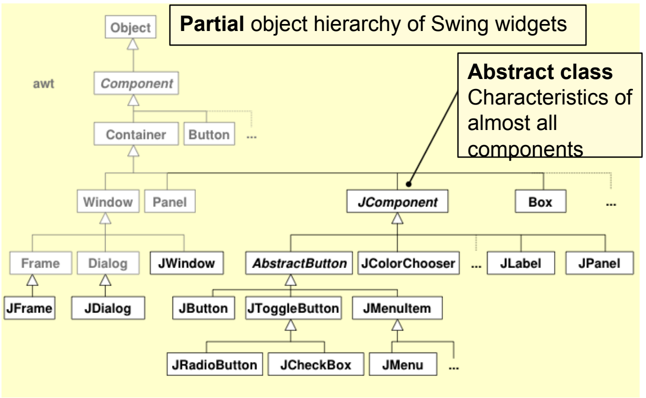
A GUI application has a top-level (container) widget that includes all others

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



Partial object hierarchy of Swing widgets

Abstract class Characteristics of almost all components

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

AWT (older) is more connected to the graphics system. Later extended with Swing (less use of the graphics system).

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

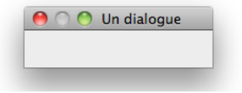
a message window (dialog) can be “modal” (blocks interaction)

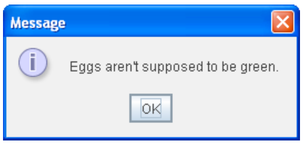
usually attached to another window (when that closes, so does the dialog)

```
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);
}
```

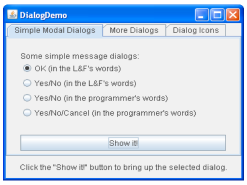
modal

attached to





Eggs aren't supposed to be green.

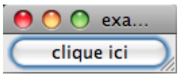


Some simple message dialogs:

● OK (in the L&F's words)
○ Yes/No (in the L&F's words)
○ Yes/No (in the programmer's words)
○ Yes/No/Cancel (in the programmer's words)

Click the "Show it!" button to bring up the selected dialog.

```
import javax.swing.*;

public class SwingDemo1 {

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

                frame.setTitle("example 1");

                frame.getContentPane().add(new JLabel("Swing Demo 1"));

                frame.setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);

                frame.getContentPane().add(new JButton("clique ici"));

                frame.setSize(100,50);

                frame.setVisible(true);
        }
}
```



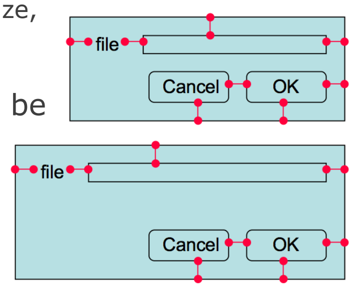
where is the label?

Bruce Eckel, Thinking in Java, 2nd edition

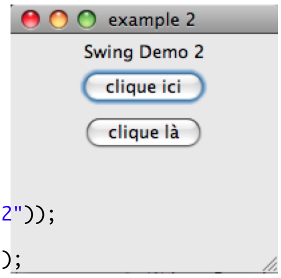
widget placement

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)
        {
                JFrame frame = new SwingDemo2();

                ((SwingDemo2)frame).init();

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



moved the "setup code" to init()

Bruce Eckel, Thinking in Java, 2nd edition

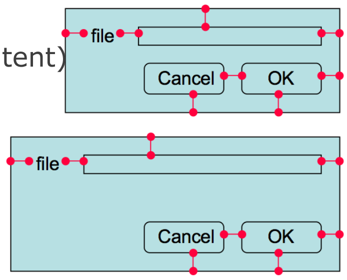
widget placement

general guides

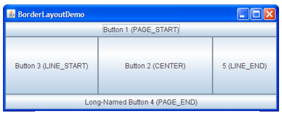
- embed geometry of a «child» widget to its parent
- parent controls the placement of its children

layout algorithm

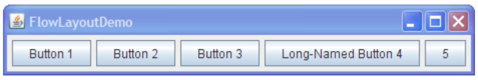
- natural size for each child (to fit content)
- size and position imposed by parent
- constraints: grid, form, etc.



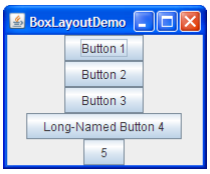
layout managers (in Swing)



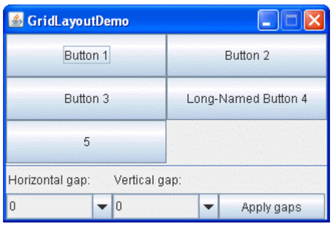
BorderLayout



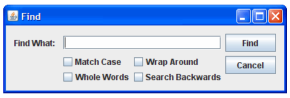
FlowLayout



BoxLayout



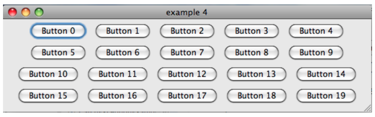
GridLayout

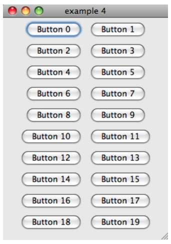


GroupLayout

http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

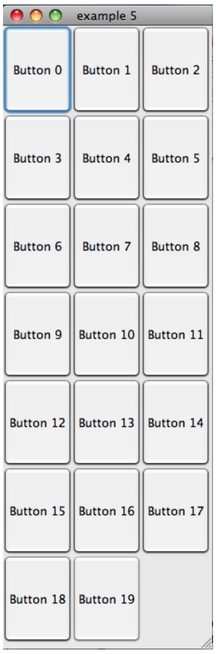
```
import javax.swing.*;
import java.awt.*;

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);
    }
}
```





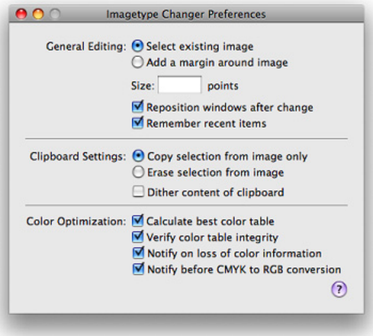
Bruce Eckel, Thinking in Java, 2nd edition

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

public class SwingDemo5 extends JFrame {

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

         this.setTitle("example 5");
         this.setDefaultCloseOperation(EXIT_ON_CLOSE);

         cp.setLayout(new GridLayout(7,3));
         for(int i = 0; i < 20; i++)
              cp.add(new JButton("Button " + i));
    }

    public static void main(String[] args)
    {
            SwingDemo5 frame = new SwingDemo5();

            frame.init();

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



Inspiré de: Bruce Eckel, Thinking in Java, 2e édition

placement guides (Mac OS X)



Imagetype Changer Preferences

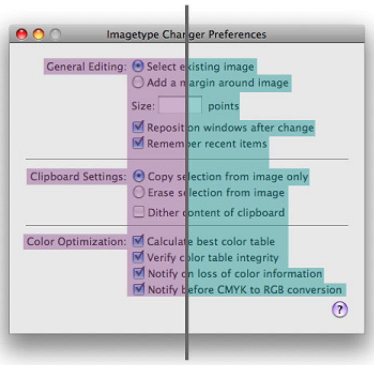
General Editing: ● Select existing image
○ Add a margin around image
Size: points
☑ Reposition windows after change
☑ Remember recent items

Clipboard Settings: ● Copy selection from image only
○ Erase selection from image
□ Dither content of clipboard

Color Optimization: ☑ Calculate best color table
☑ Verify color table integrity
☑ Notify on loss of color information
☑ Notify before CMYK to RGB conversion

placement guides (Mac OS X)

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



Imagetype Changer Preferences

General Editing: ● Select existing image
○ Add a margin around image
Size: ▭ points
☑ Reposition windows after change
☑ Remember recent items

Clipboard Settings: ● Copy selection from image only
○ Erase selection from image
□ Dither content of clipboard

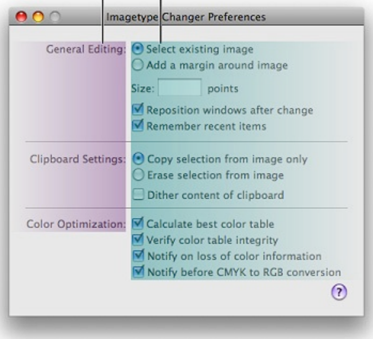
Color Optimization: ☑ Calculate best color table
☑ Verify color table integrity
☑ Notify on loss of color information
☑ Notify before CMYK to RGB conversion

placement guides (Mac OS X)

Alignement

Column of labels with right alignement

Column of controls with left alignment



Imagetype Changer Preferences

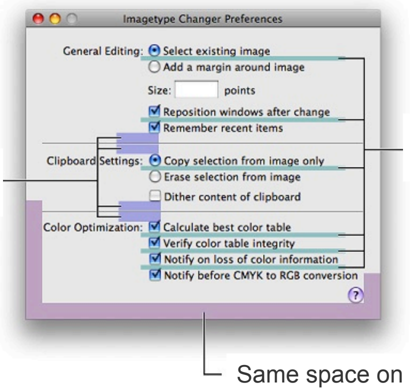
General Editing: ● Select existing image
○ Add a margin around image
Size: ▭ points
☑ Reposition windows after change
☑ Remember recent items

Clipboard Settings: ● Copy selection from image only
○ Erase selection from image
□ Dither content of clipboard

Color Optimization: ☑ Calculate best color table
☑ Verify color table integrity
☑ Notify on loss of color information
☑ Notify before CMYK to RGB conversion

placement guides (Mac OS X)

Spacing



Imagetype Changer Preferences

General Editing: ● Select existing image
○ Add a margin around image
Size: points
☑ Reposition windows after change
☑ Remember recent items

Clipboard Settings: ● Copy selection from image only
○ Erase selection from image
□ Dither content of clipboard

Color Optimization: ☑ Calculate best color table
☑ Verify color table integrity
☑ Notify on loss of color information
☑ Notify before CMYK to RGB conversion

Same space before and after separator

Same space between controls

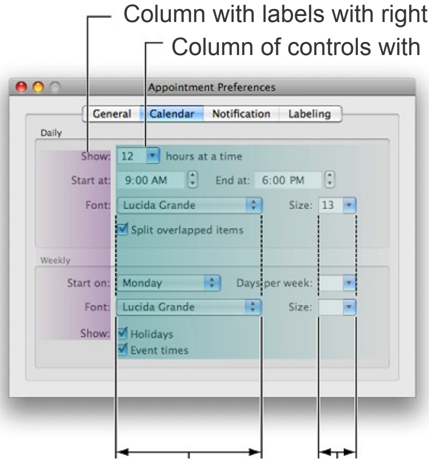
Same space on every side

placement guides (Mac OS X)

Alignement and consistency

Column with labels with right alignement

Column of controls with left alignment



Appointment Preferences

General | Calendar | Notification | Labeling

Daily
Show: 12 hours at a time
Start at: 9:00 AM End at: 6:00 PM
Font: Lucida Grande Size: 13
☑ Split overlapped items

Weekly
Start on: Monday Days per week:
Font: Lucida Grande Size:
Show: ☑ Holidays
☑ Event times

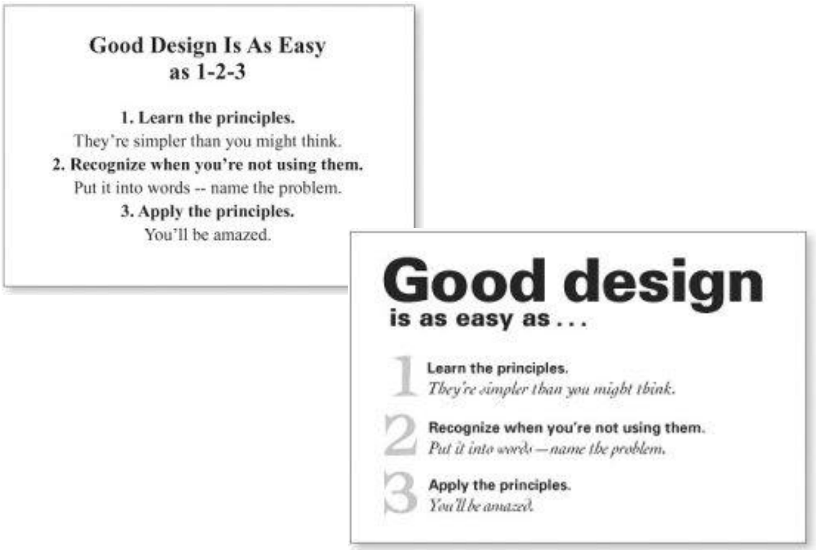
Consistency between controls of the same type

CRAP

contrast, repetition, alignment, proximity

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

Slide deck by Saul Greenberg.



Good Design Is As Easy
as 1-2-3

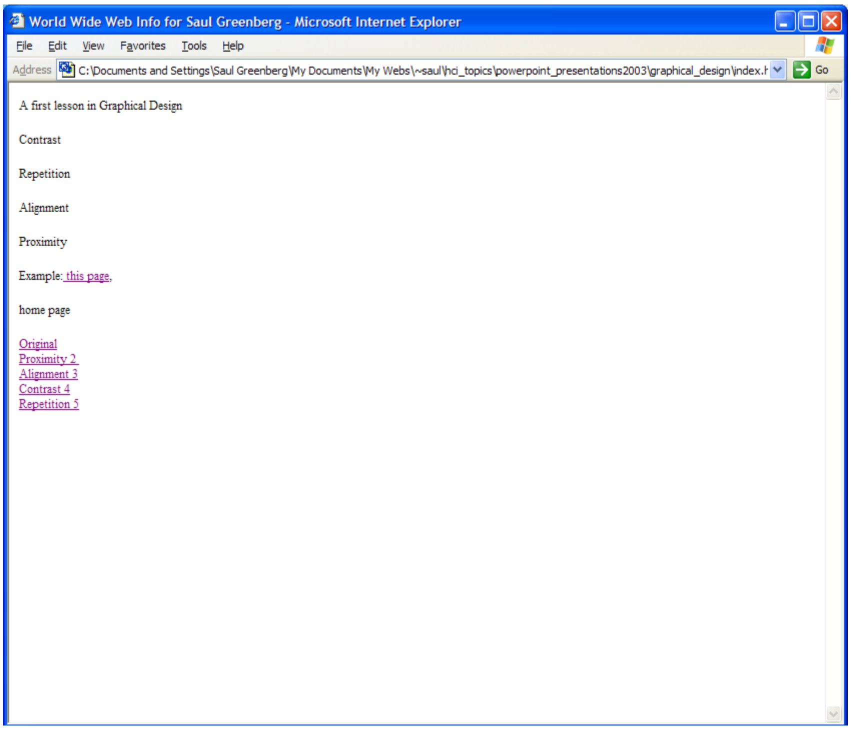
1. Learn the principles.
They're simpler than you might think.
2. Recognize when you're not using them.
Put it into words -- name the problem.
3. Apply the principles.
You'll be amazed.

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.


World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
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

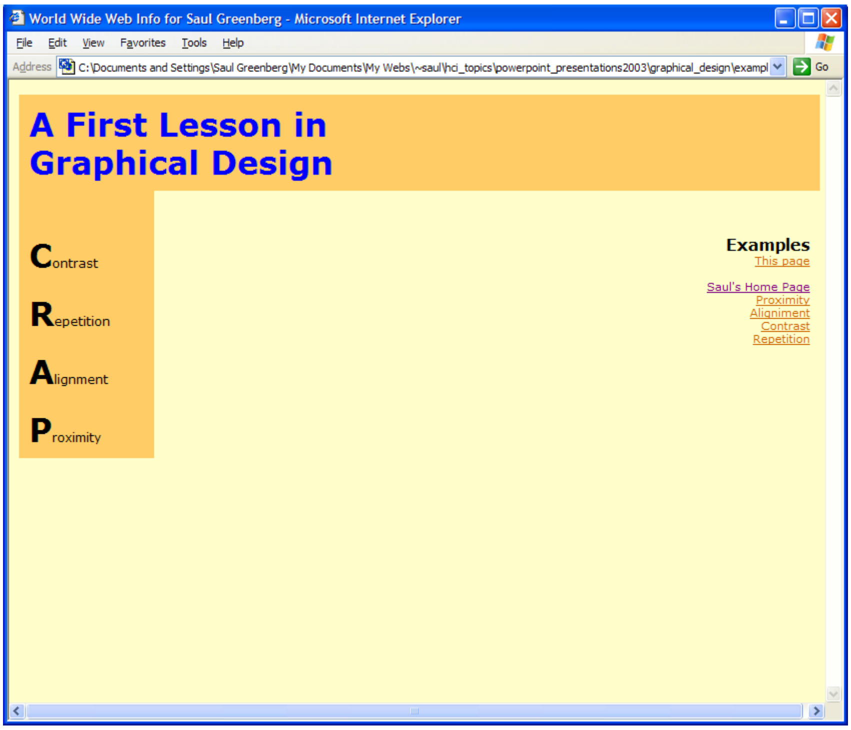
Alignment

Proximity

Example: this page,

home page

Original
Proximity 2
Alignment 3
Contrast 4
Repetition 5


World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
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

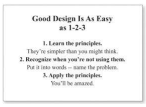
CRAP

- **C**ontrast
- **R**epetition
- **A**lignment
- **P**roximity

Robin Williams Non-Designers Design Book, Peachpit Press

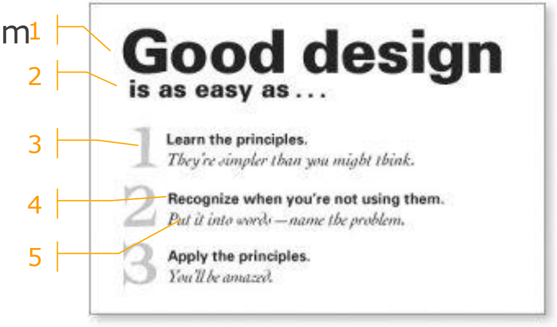
CRAP

- **Contrast**
  make different things different
  brings out dominant elements
  mutes lesser elements
  creates dynamism
- **R**epetition
- **A**lignment
- **P**roximity



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.

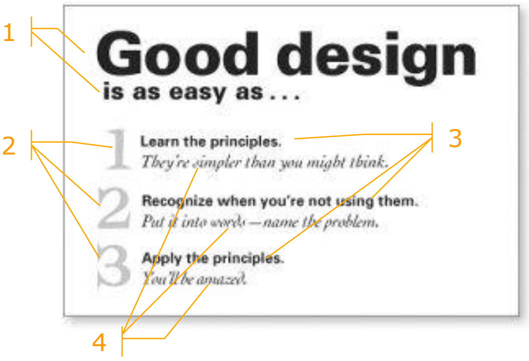
Robin Williams Non-Designers Design Book, Peachpit Press

CRAP

- **C**ontrast
- **Repetition**
  repeat design throughout the interface
  consistency
  creates unity
- **A**lignment
- **P**roximity

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.

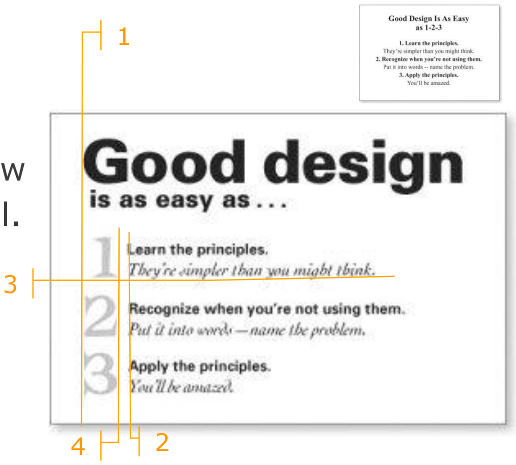
Robin Williams Non-Designers Design Book, Peachpit Press

CRAP

- **C**ontrast
- **R**epetition
- **Alignment**
  creates a visual flow
  visually connects el.
- **P**roximity

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.

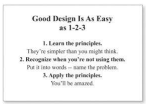
Robin Williams Non-Designers Design Book, Peachpit Press

CRAP

- **C**ontrast
- **R**epetition
- **A**lignment
- **P**roximity

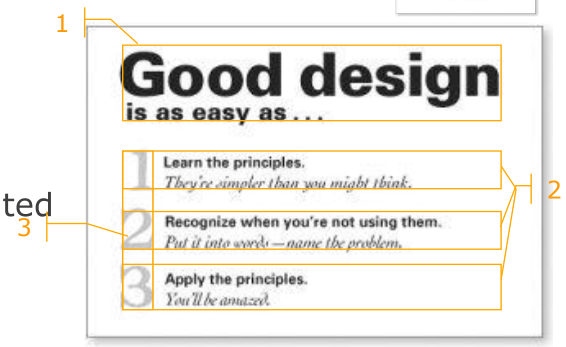
  groups related

  separates unrelated



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.

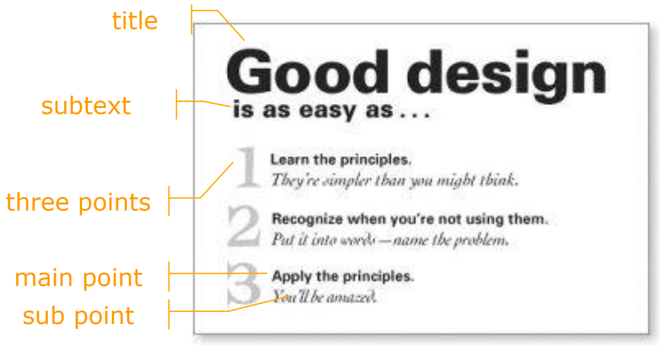
2 **Recognize when you're not using them.**
Put it into words—name the problem.

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

Robin Williams Non-Designers Design Book, Peachpit Press

Where does your eye go?

- CRAP combines to give you cues of how to read the graphic



Good design
is as easy as . . .

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

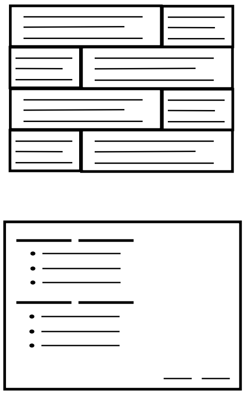
2 **Recognize when you're not using them.**
Put it into words—name the problem.

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

Robin Williams Non-Designers Design Book, Peachpit Press

Where does your eye go?

- Boxes do not create a strong structure
  - CRAP fixes it



Robin Williams Non-Designers Design Book, Peachpit Press

Where does your eye go?

- Some contrast and weak proximity
  - ambiguous structure
  - interleaved items

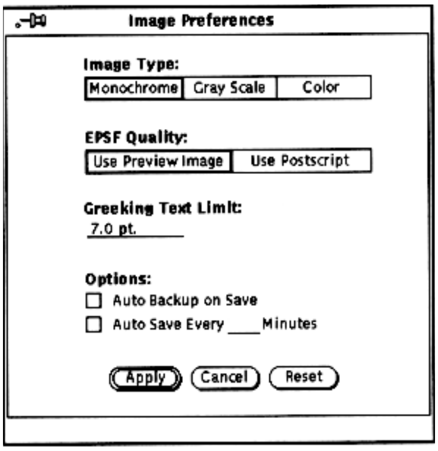


Image Preferences

Image Type:

Monochrome | Gray Scale | Color

EPSF Quality:

Use Preview Image | Use Postscript

Greeking Text Limit:

7.0 pt.

Options:

□ Auto Backup on Save

□ Auto Save Every ____ Minutes

Robin Williams Non-Designers Design Book, Peachpit Press

Where does your eye go?

- Strong proximity (left/right split)
  - unambiguous

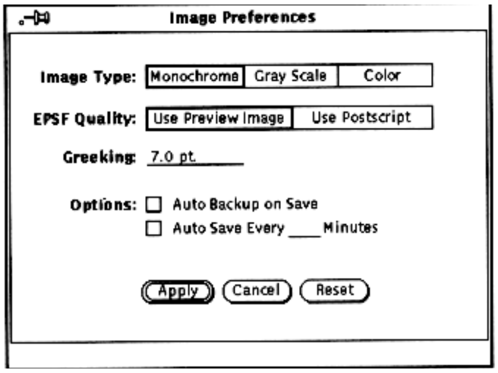


Image Type: Monochrome | Gray Scale | Color

EPSF Quality: Use Preview Image | Use Postscript

Greeking: 7.0 pt

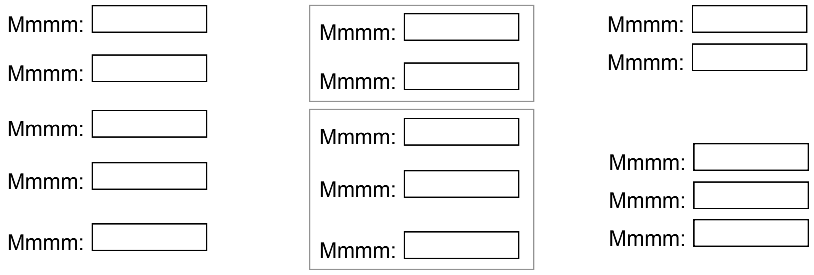
Options: □ Auto Backup on Save
□ Auto Save Every ___ Minutes

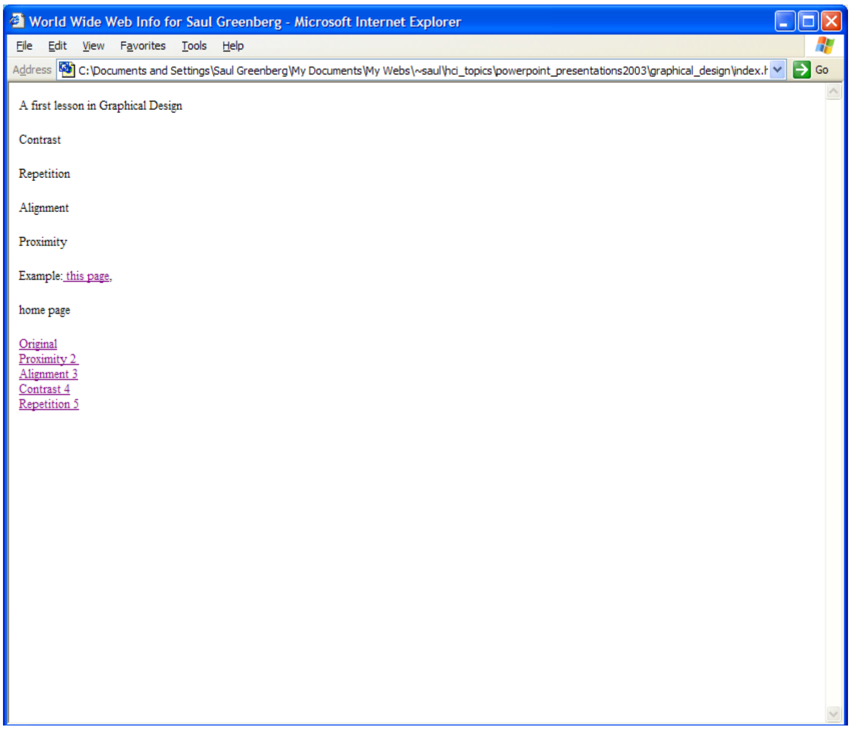
✓

Robin Williams Non-Designers Design Book, Peachpit Press

Where does your eye go?

- the strength of proximity
  - alignment
  - white (negative) space
  - explicit structure a poor replacement




World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
Go

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

A first lesson in Graphical Design

Contrast

Repetition

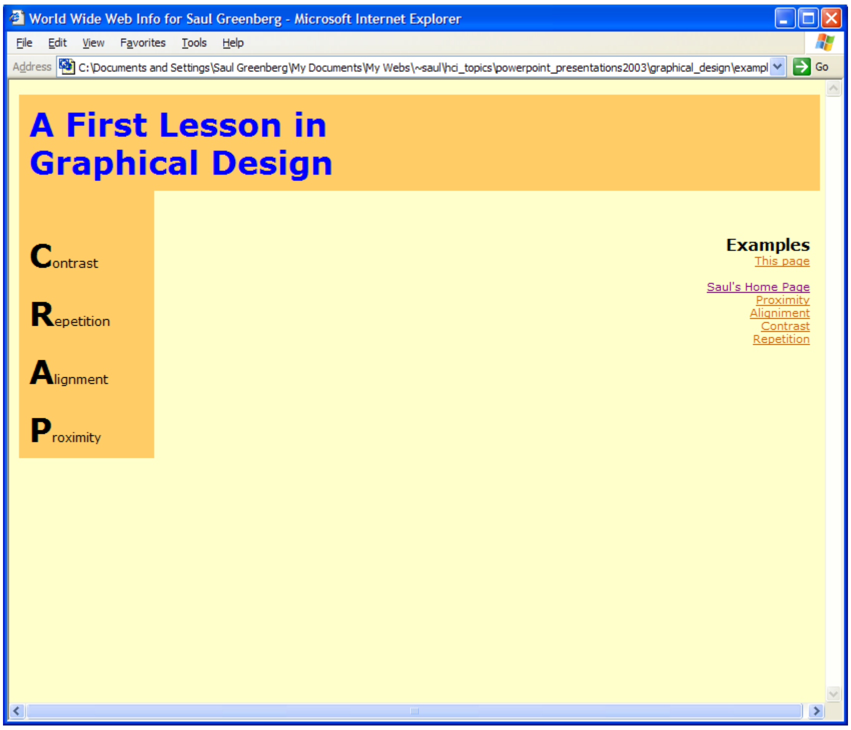
Alignment

Proximity

Example: this page,

home page

Original
Proximity 2
Alignment 3
Contrast 4
Repetition 5


World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
Go

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

A First Lesson in Graphical Design

Contrast

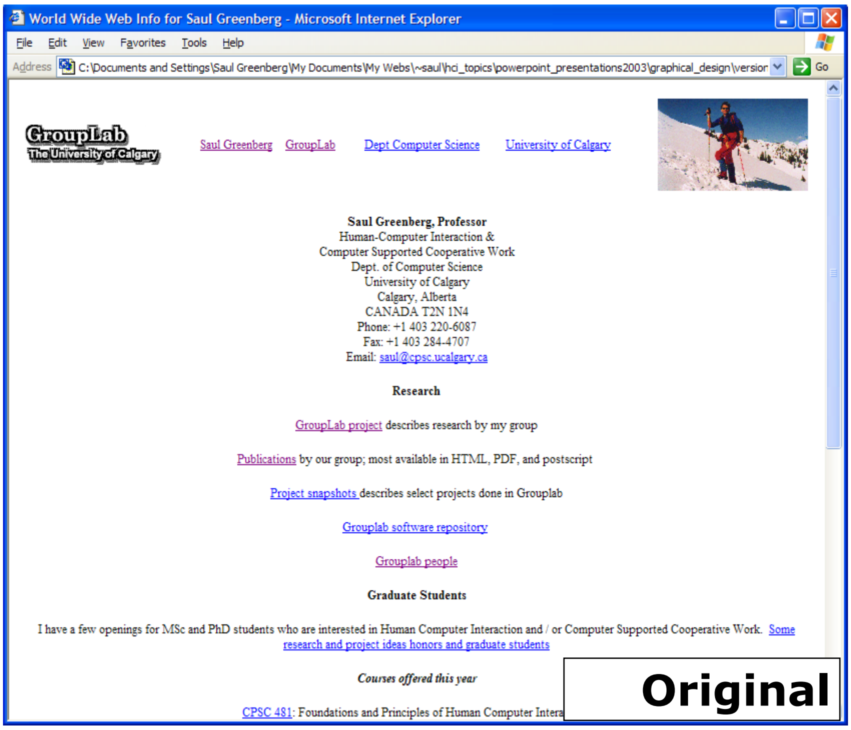
Repetition

Alignment

Proximity

Examples
This page

Saul's Home Page
Proximity
Alignment
Contrast
Repetition


World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Address
Go
GroupLab
The University of Calgary

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

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

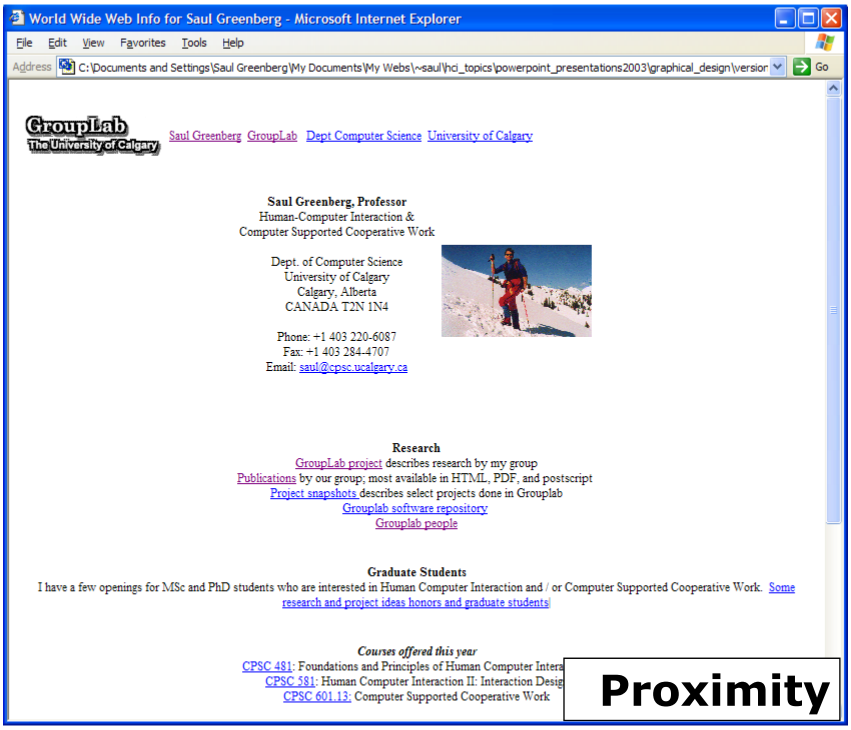
Graduate Students

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

Original


World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Address
Go
GroupLab
The University of Calgary

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

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

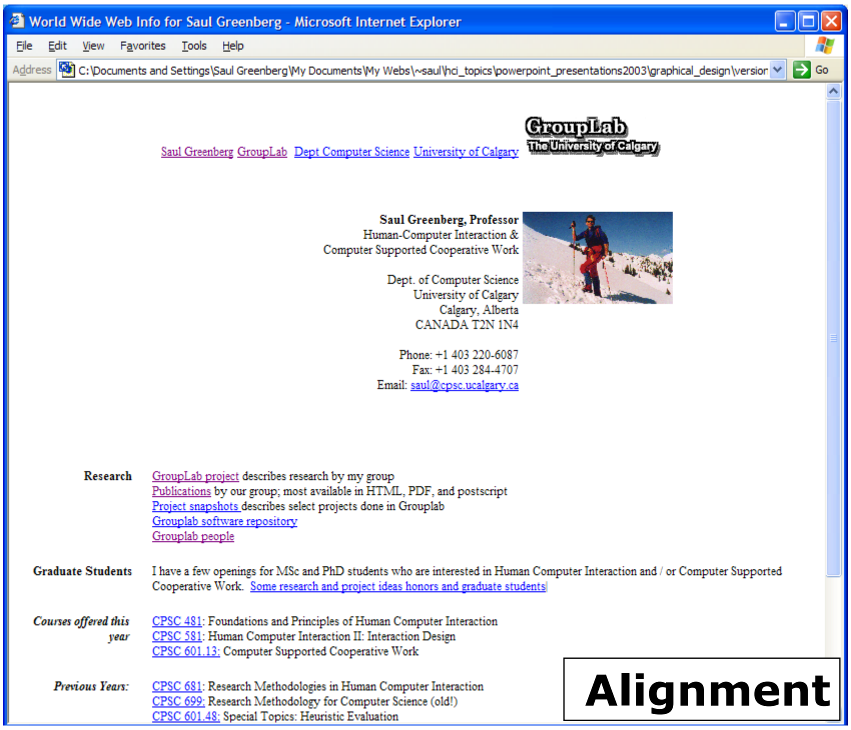
Phone: +1 403 220-6087
Fax: +1 403 284-4707
Email: saul@cpsc.ucalgary.ca

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
CPSC 601.13: Computer Supported Cooperative Work

Proximity


World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
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

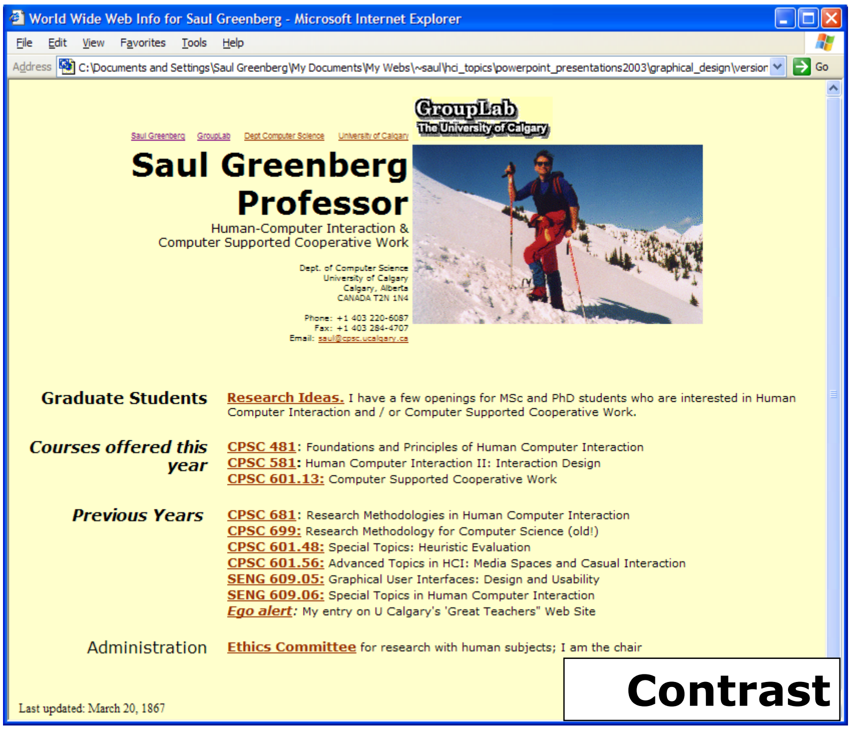
Research
GroupLab project describes research by my group
Publications by our group; most available in HTML, PDF, and postscript
Project snapshots describes select projects done in Grouplab
Grouplab software repository
Grouplab people

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 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

Alignment


World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
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

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.

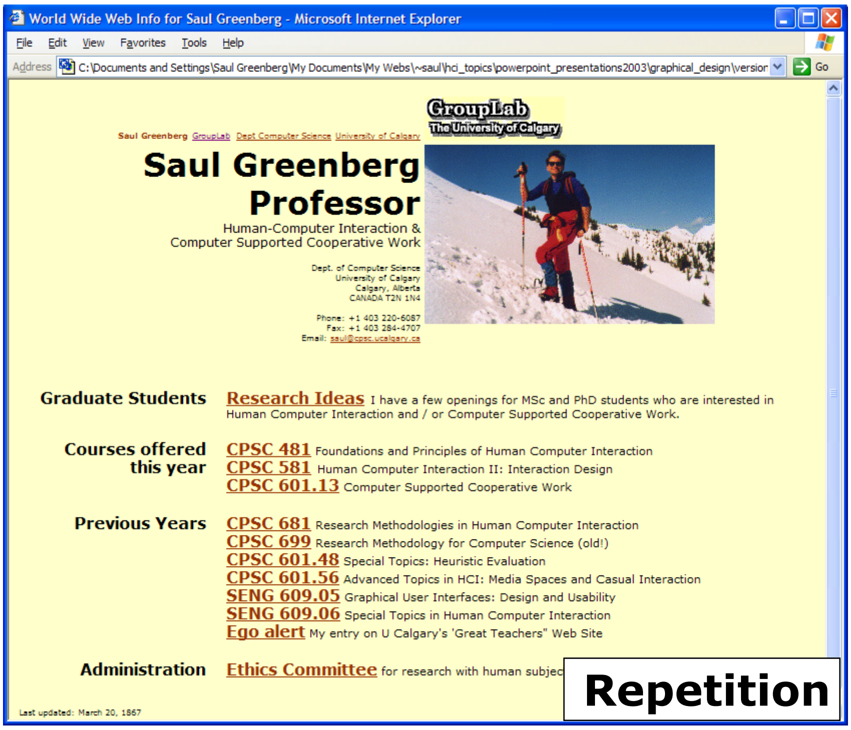
Courses offered this year
CPSC 481: Foundations and Principles of Human Computer Interaction
CPSC 581: Human Computer Interaction II: Interaction Design
CPSC 601.13: Computer Supported Cooperative Work

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

Last updated: March 20, 1867

Contrast


World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
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
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

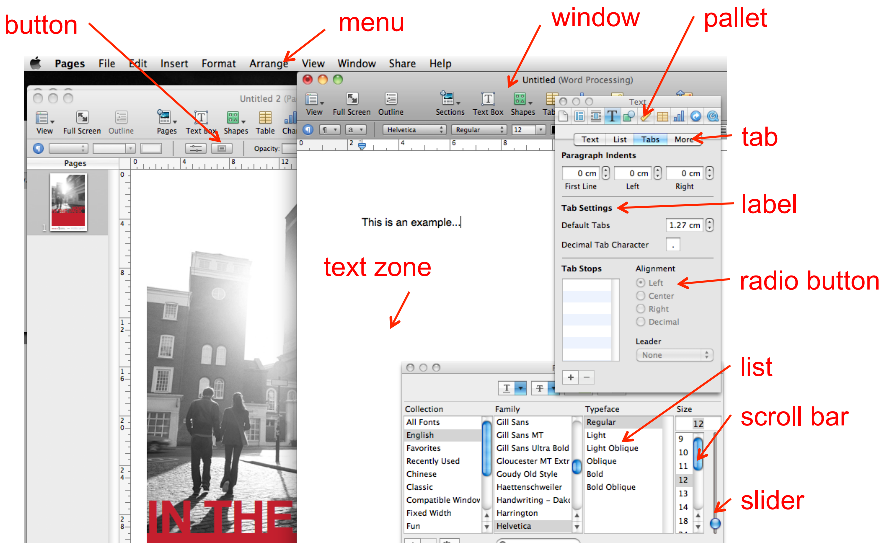
Graduate Students	**Research Ideas** I have a few openings for MSc and PhD students who are interested in Human Computer Interaction and / or Computer Supported Cooperative Work.
Courses offered this year	**CPSC 481** Foundations and Principles of Human Computer Interaction **CPSC 581** Human Computer 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

Last updated: March 20, 1867

Repetition

facets of a widget

« widgets » (window gadget)



This is an example...

facettes of a widget

presentation

- appearance

behavior

- reaction to user actions

interface with the application:
notification of state changes

Button:
border with text inside
« pressing » or « releasing » animation when clicked
call function when the button is clicked

facettes of a widget

presentation

- appearance

behavior

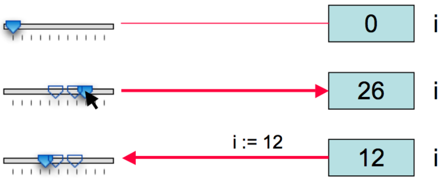
- reaction to user actions

interface with the application:
notification of state changes

- active/linked/wrapped variables (Tcl/Tk)
- event dispatching (Qt)
- callback functions (Swing)

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*)



```
main (){
   int i = 0;
   …
   /* widget */
   nc = CreateSlider (…);
   /* active var */
   SetIntegerActiveVariable (nc, &i);
   …
}
```

problems

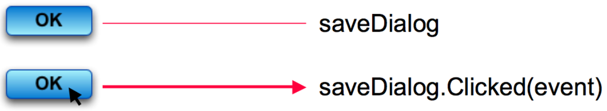
- limited to simple types
- return link can be costly if automatic
- errors when links are updated by programmers

event dispatching

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

a while loop reads and treats events

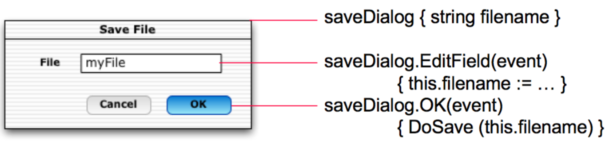
associate an object to a widget, and its methods to changes in the widget state



saveDialog

saveDialog.Clicked(event)

event dispatching



saveDialog { string filename }

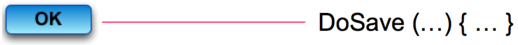
saveDialog.EditField(event)
{ this.filename := … }

saveDialog.OK(event)
{ DoSave (this.filename) }

- divide event sending and treatment
- better encapsulation (inside widget class)
- but when similar behaviors exist …

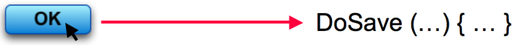
callback functions

Registration at widget creation

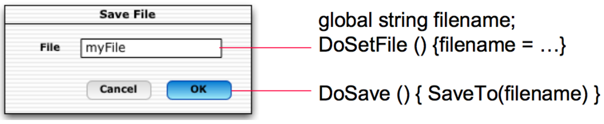


DoSave (…) { … }

Call at widget activation



DoSave (…) { … }



global string filename;
DoSetFile () {filename = …}

DoSave () { SaveTo(filename) }

callback functions

Problem: spaghetti of callbacks

Sharing a state between multiple callbacks by:

- global variables: widgets check them
  - 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
“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

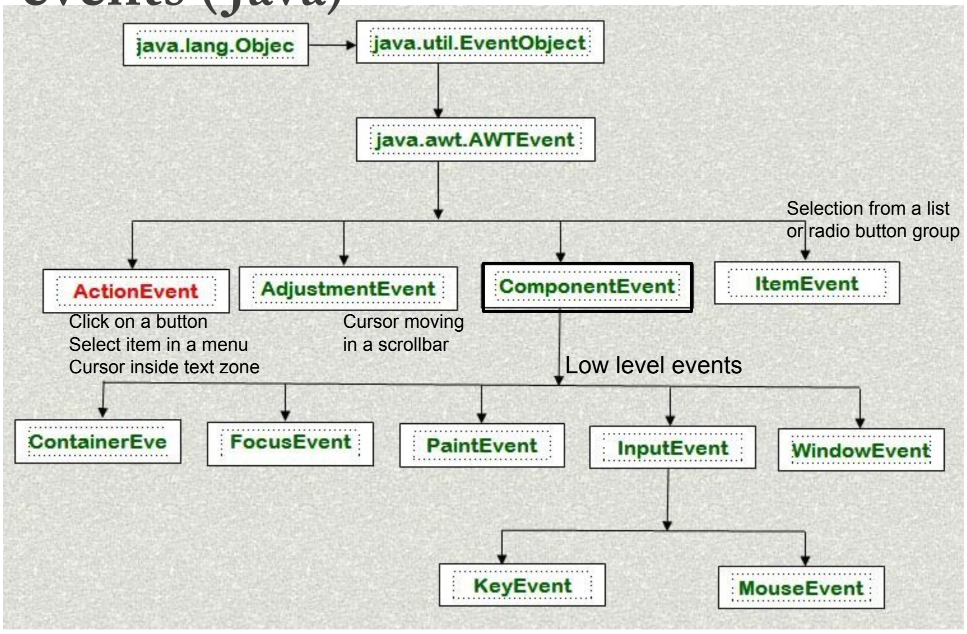
event listeners (Java)

Anonymous Inner classes

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

The functions and events are predefined

events (Java)



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

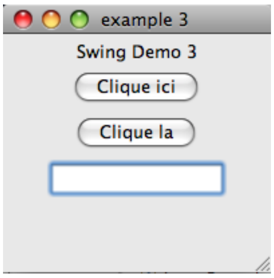
- e.g. one listener handles events from multiple buttons

a widget can have many listeners

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

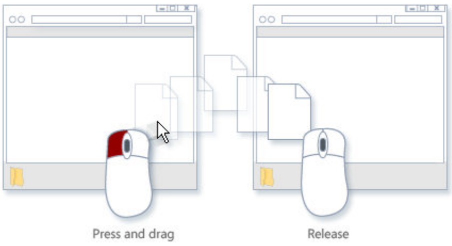
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SwingDemo3 extends JFrame {

    JButton b1 = new JButton("Clique ici");
    JButton b2 = new JButton("Clique la");
    JTextField txt = new JTextField(10);

    class ButtonListener implements ActionListener // INNER CLASS DEF.
    {
            public void actionPerformed(ActionEvent e) {
                        String name = ((JButton)e.getSource()).getText();
                        txt.setText(name);
            }
    } // END OF INNER CLASS DEFINITION

    ButtonListener bl = new ButtonListener();


    public void init() {

            b1.addActionListener(bl);
            b2.addActionListener(bl);

            Container cp = this.getContentPane();

            this.setTitle("example 3");

            cp.add(new JLabel("Swing Demo 3"));
            cp.setLayout(new FlowLayout());

            cp.add(b1);
            cp.add(b2);
            cp.add(txt);
   }

      public static void main(String[] args)
      {
                  SwingDemo3 frame = new SwingDemo3();

                  frame.init();

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

   } // end of SwingDemo3 class definition
```

inner class



« 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

=> Finite State Machine and Hierarchical SM (soon !)

hard to do things the toolkit was not designed for

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