

# Human-Computer interaction

---

## Termin 5: Interaction

So far, you have learnt about

- capabilities & limitations of humans
- input/output devices & limitations of computers

Now, how can the user use the computer as a tool to perform, simplify or support a task?



# What is interaction?

- communication between two complex systems
  - need an *interface* to translate between them
- models of interaction
  - what is going on in the interaction
- styles of interaction
  - the nature of user/system dialog



# models of interaction

---

overview  
terms of interaction  
stages of action model  
interaction framework

# Overview

Models for different purposes and with different aims  
(not disjunctive!)

- *descriptive*: define terminology, model cognitive and interactive processes
- *predictive*: predict performance w.r.t. interface complexity, efficiency, etc.; enables comparison
- *prescriptive*: derive general design rules

Models with different focus

- *perceptual*: on perception and interpretation
- *cognitive*: on „internal processing“
- *motoric*: on physical user actions



# Some terms of interaction

- domain* – the area of work under study, has concepts  
e.g. graphic design
- tasks* – operations to manipulate domain concepts  
e.g. create shapes, colour things red, ...
- goal* – what the user wants to achieve  
e.g. have a solid red triangle
- intention* – what the user is going to do about it  
e.g. create the triangle himself with system
- actions* – specific plan of actions to fulfil the intention  
e.g. 1. select fill tool, 2. click over triangle, ...
- context* – situational conditions (physical, social, organizational)  
e.g. bright sunshine, boss wants it yesterday

Note ...

- use of terms differs a lot, especially task/goal



# Some more terms

*Task analysis* - identification of problem space for the user in terms of domain, goals, intention, tasks

*Core language* - language to describe computational attributes of domain relevant to the *system state*

e.g. polygon of color #FF0000

*Task language* - language to describe psych. attributes of the domain relevant to the *user state*

e.g. „red triangle“



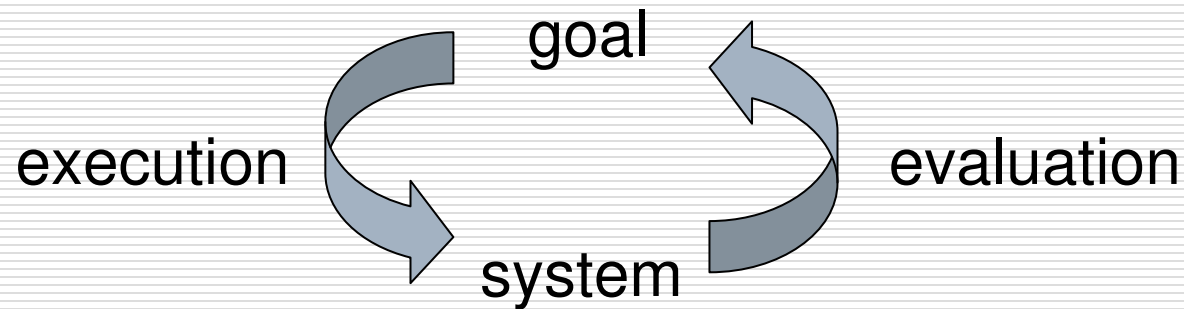
# Stages of action model

- Donald Norman (1988)
- the user...
  1. establishes the goal
  2. formulates intention (and plan of action)
  3. specifies actions at interface
  4. executes action
  5. perceives system state
  6. interprets system state
  7. evaluates system state with respect to goal
- each stage is an activity on its own
- concentrates on *user's view* of the interface





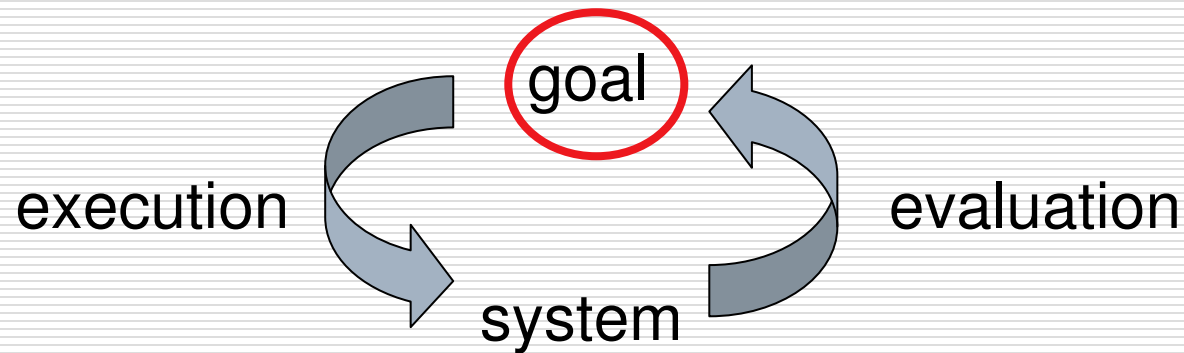
# execution/evaluation loop



1. user establishes the goal
2. formulates intention
3. specifies actions at interface
4. executes action
5. perceives system state
6. interprets system state
7. evaluates system state with respect to goal

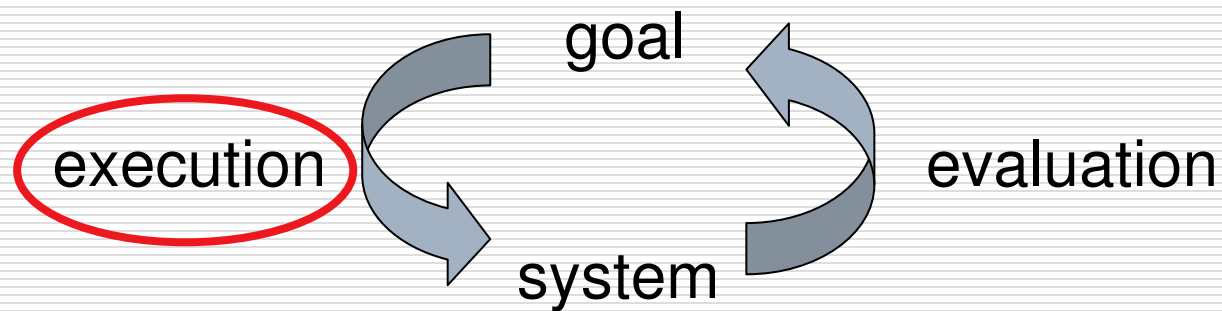


# execution/evaluation loop



1. user establishes the goal: imprecise, in task language
2. formulates intention
3. specifies actions at interface
4. executes action
5. perceives system state
6. interprets system state
7. evaluates system state with respect to goal

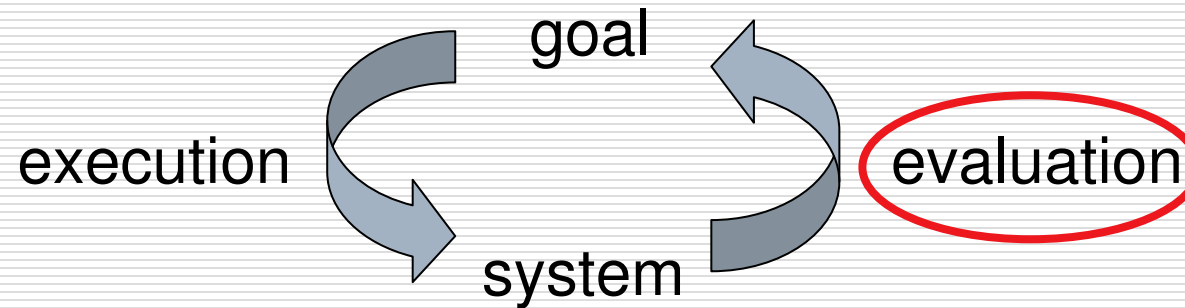
# execution/evaluation loop



1. user establishes the goal
2. formulates intention: more specific actions to reach the goal
3. specifies actions at interface: translate task lang. → core lang.
4. executes action
5. perceives system state
6. interprets system state
7. evaluates system state with respect to goal



# execution/evaluation loop



1. user establishes the goal
2. formulates intention
3. specifies actions at interface
4. executes action
5. perceives system state
6. interprets system state (in terms of expectations)
7. evaluates system state with respect to goal



# Example

You are sitting reading as evening falls

- Goal - need more light
- Intention - switch on desk lamp or ask for it or...
- Actions - reach over, press lamp switch
- Result - light is either on or off
- Interpret - light is off? Maybe bulb has blown
- goals - change bulb
- Evaluate - light is on? Is it enough?
- goals - switch on main ceiling light too



# Using Norman's model

Some interfaces are harder to use than others. Why?

## □ *Gulf of Execution*

- user's intention & formulation of actions in task lang.  
≠ actions allowed by the system
- interaction effective if allowed actions correspond to those intended by the user

## □ *Gulf of Evaluation*

- user's expectation of changed system state  
≠ actual physical presentation of system state
- interaction effective if user can readily interpret and evaluate this presentation in terms of his goals



# Gulf of execution - Example

Intention -  
I don't want to see this warning anymore, and I don't want cookies to be stored at all!

No suitable action offered!



# Human error - slips and mistakes

## *slip*

- 😊 understand system and goal
- 😊 correct formulation of action
- 😞 incorrect action (e.g. typo, didn't hit right icon)

## *mistake*

- 😞 don't understand system, may not even have right goal or formulation of action (e.g. misinterpretation of icon)

## Fixing things?

slip – better interface design

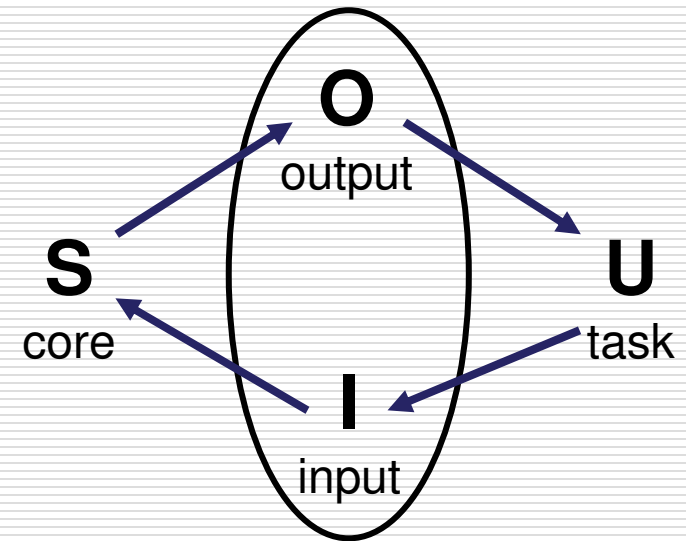
mistake – create better understanding of system





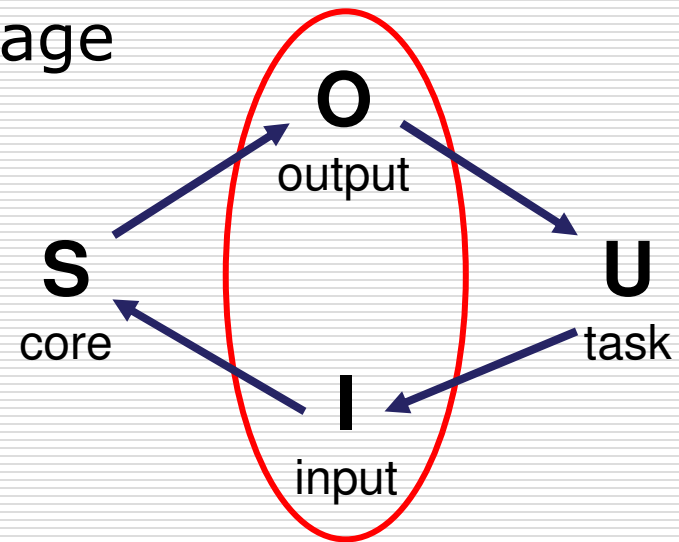
# Interaction framework (Abowd & Beale)

- extension of Norman's model
- interaction framework with 4 parts
  - user
  - input
  - system
  - output
- each component has its own unique language for its stimuli
  - user: task language
  - system: core language
  - input language
  - output language
- interaction  $\Rightarrow$  translation between languages
  - problems in interaction = problems in translation



# (User) Interface

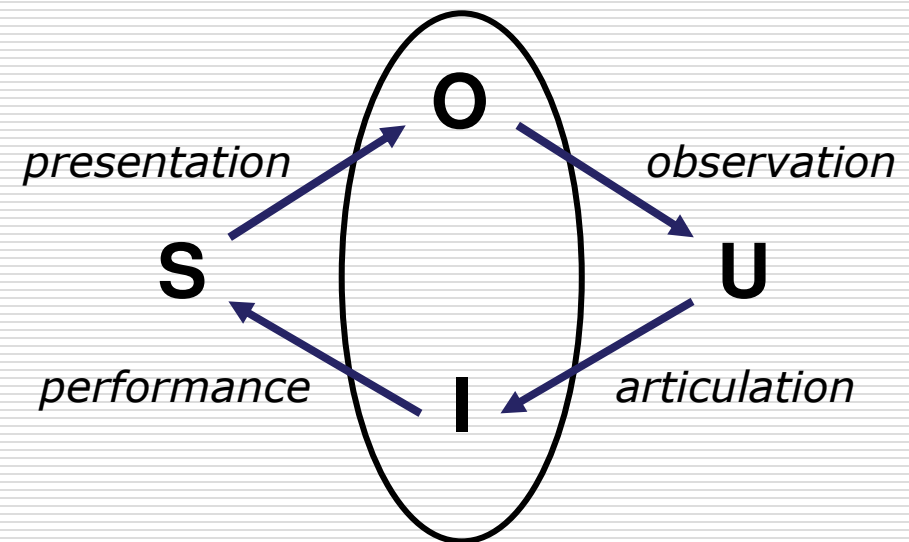
- *interface*  $\Rightarrow$  input & output
- All communication between user and system goes through
- Mediates between user/task and system/core
- Requires input in input language
- Provides output capabilities



# Using Abowd & Beale's model

- Interactive cycle: 4 steps, corresponding to translations
- user starts with intentions within task language

1. *articulates* actions (within input lang.) to the interface
2. translated into operations (core lang.) *performed* by system
3. new system state *presented* as output (output lang.)
4. *observed* and interpreted by the user and assessed w.r.t. expectations (task lang.)



# Using Abowd & Beale's model

allows comparative assessment of systems

- *Articulation* – do domain concepts that are important to the user map clearly onto input language?
  - e.g. direct manipulation vs. sliders for moving
- *Performance* – can translated input reach as many system states as possible with direct system stimuli?
  - e.g. remote control doesn't turn player off
- *Presentation* – are the relevant system attributes preserved in the output lang. expression?
  - e.g. narrow view at vs. outline of word document
- *Observation* – how easy and complete can the system state be observed?
  - e.g. HTML editing without seeing browser output



# Abowd & Beale's model - example

You want to program your VCR from the remote, but you are not sure the VCR is set to record properly.

Possible problems in translation

- Keys pressed in wrong order? – articulation
- Remote lacks ability to select channel? – performance
- VCR display doesn't indicate setting – presentation
- You don't interpret display correctly - observation



# Linguistic models

- interaction
  - = exchange of symbols according to some rules
  - = comparable to words of a formal language
- Definable by *automata* or *grammars*
- Complexity of grammar measure for complexity of interface
- Exampe: BNF definition of line drawing

```
draw-line      ::=      select-line      + choose-points
                                     + last-point
select-line    ::=      position-mouse + CLICK-MOUSE
choose-points  ::=      choose-one |
                                     choose-one + choose-points
choose-one     ::=      position-mouse + CLICK-MOUSE
last-point     ::=      position-mouse + DOUBLE-CLICK-MOUSE
position-mouse ::=      empty   | MOVE-MOUSE + position-mouse
```

*non-terminal*    *TERMINAL*    *|*: Alternative    *+*: Konkatenation



# Consistency through grammars

- important aspect of interfaces: *consistency*
- action possibilities should be systematic and predictable

Example:

<i>consistent</i>	<i>inconsistent (1)</i>	<i>inconsistent (2)</i>
delete/insert character	delete/insert character	delete/insert character
delete/insert word	remove/bring word	remove/insert word
delete/insert line	destroy/create line	delete/insert line
delete/insert paragraph	kill/birth paragraph	delete/insert paragraph

- inconsistent actions...
  - are hard to memorize, need longer learning
  - can cause slips and even mistakes
  - slow down the interaction



# Consistency through grammars

- Problem with BNF definition: cannot represent consistency of language structure very well
- Example: UNIX file commands

```
copy ::= 'cp' + filename + filename |
      'cp' + filenames + directory
move ::= 'mv' + filename + filename |
      'mv' + filenames + directory
link ::= 'ln' + filename + filename |
      'ln' + filenames + directory
```

- „*task-action grammar*“ (TAG; Payne & Green, 1986): parametric rules to represent subsets of language

```
file-op[Op] := command[Op] + filename + filename |
              command[Op] + filenames + directory
command[Op=copy] := 'cp'
command[Op=move] := 'mv'
command[Op=link] := 'ln'
```





## ...more models

- predictive
- prescriptive

later...



# interface styles



interaction styles

# common interaction styles

- command language/entry
- form filling
- menus
- point and click
- WIMP
- direct manipulation
- 3D interfaces

...often used in combination



# Command line interface

- way of expressing instructions to the computer directly (e.g. 438 commands in BSD Unis)

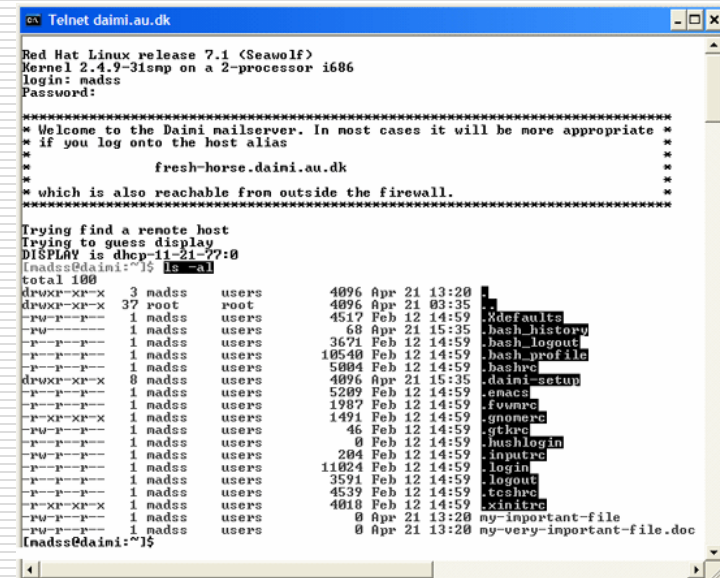
*commands* =

chars, abbreviations, words

*command Language* =

commands + syntax

→ grammars, TAGs, etc.



```
Telnet daini.au.dk
Red Hat Linux release 7.1 (Seawolf)
Kernel 2.4.9-3jisp on a 2-processor i686
login: madss
Password:
*****
* Welcome to the Daini mailserver. In most cases it will be more appropriate *
* if you log onto the host alias *
* *
*      fresh-horse.daini.au.dk *
* *
* which is also reachable from outside the firewall. *
*****
Trying find a remote host
Trying to guess display
DISPLAY is dhcp-11-21-27:0
[madss@daini:~]$ ls -a
total 100
drwxr-xr-x  3 madss  users      4096 Apr 21 13:20 .
drwxr-xr-x 37 root   root       4096 Apr 21 03:35 ..
-rw-r--r--  1 madss  users      4517 Feb 12 14:59 .Xdefaults
-rw-r--r--  1 madss  users       68 Apr 21 15:35 .bash_history
-rw-r--r--  1 madss  users      3671 Feb 12 14:59 .bash_logout
-rw-r--r--  1 madss  users     10540 Feb 12 14:59 .bash_profile
-rw-r--r--  1 madss  users     5004 Feb 12 14:59 .bashrc
drwxr-xr-x  0 madss  users      4096 Apr 21 15:35 .daini-setup
-rw-r--r--  1 madss  users     5209 Feb 12 14:59 .emacs
-rw-r--r--  1 madss  users     1987 Feb 12 14:59 .fvwmrc
-rw-r--r--  1 madss  users     1491 Feb 12 14:59 .gnome
-rw-r--r--  1 madss  users       46 Feb 12 14:59 .gtivc
-rw-r--r--  1 madss  users       60 Feb 12 14:59 .hushlogin
-rw-r--r--  1 madss  users      204 Feb 12 14:59 .inputrc
-rw-r--r--  1 madss  users    11024 Feb 12 14:59 .login
-rw-r--r--  1 madss  users     3591 Feb 12 14:59 .logout
-rw-r--r--  1 madss  users     4539 Feb 12 14:59 .rcshrc
-rw-r--r--  1 madss  users     4010 Feb 12 14:59 .xinitrc
-rw-r--r--  1 madss  users       60 Apr 21 13:20 ny-important-file
-rw-r--r--  1 madss  users       60 Apr 21 13:20 ny-very-important-file.doc
[madss@daini:~]$
```

- Cognitive burden: requires to recall names *and* syntax
  - "afmtodit" = create font files for use with "groff"
  - "bc" = arbitrary precision calculator language
  - "5" + "d" + "w" = delete five words in vi



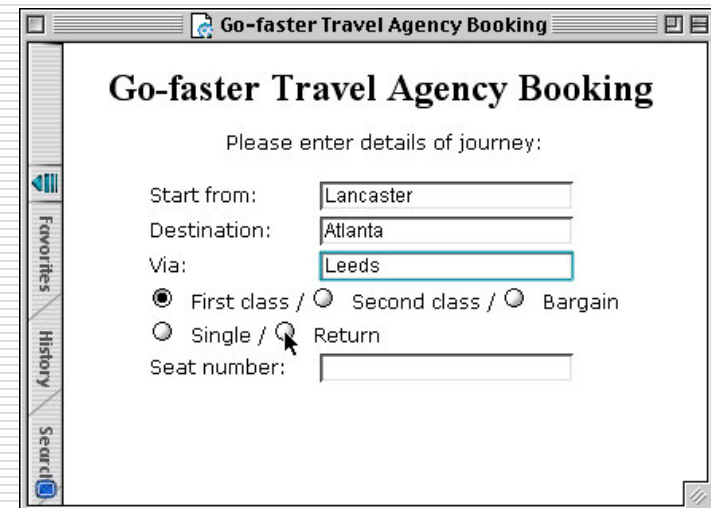
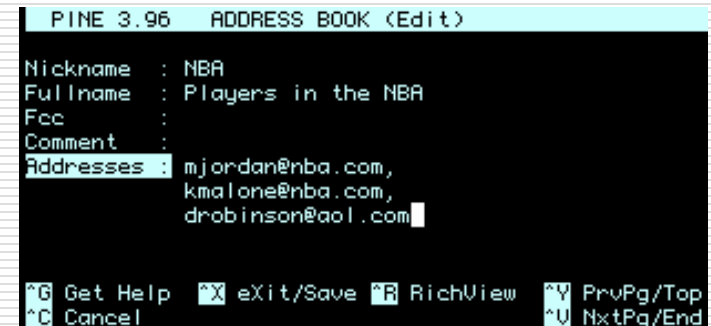
# Command entry

- Advantages:
  - direct access to functionality, flexible
  - appeals to expert users, they get fast
  - supports creation of user-defined "scripts" or macros
  - suitable for interacting with networked computers even with low bandwidth
- Disadvantages:
  - difficult to learn and to retain, requires a lot of practice
  - high error rates
  - complex mapping from tasks to input language
  - error messages and assistance hard to provide
  - not suitable for non-expert users
  - command names should be meaningful (but a lot of abbrev. to minimize typing)
- recommended for frequent users (and for work under time pressure)



# Form filling

- ❑ whole interface is form-based
- ❑ data entered into *fields*
- ❑ few keys to navigate through fields and conclude form
- ❑ advantages:
  - simplifies data entry
  - shortens learning in that the fields are predefined and need only be 'recognised'
  - good for non-expert users
- ❑ disadvantages:
  - limited in scope, useful only for structured information
  - consumes a lot of screen space
  - rigid, not very flexible



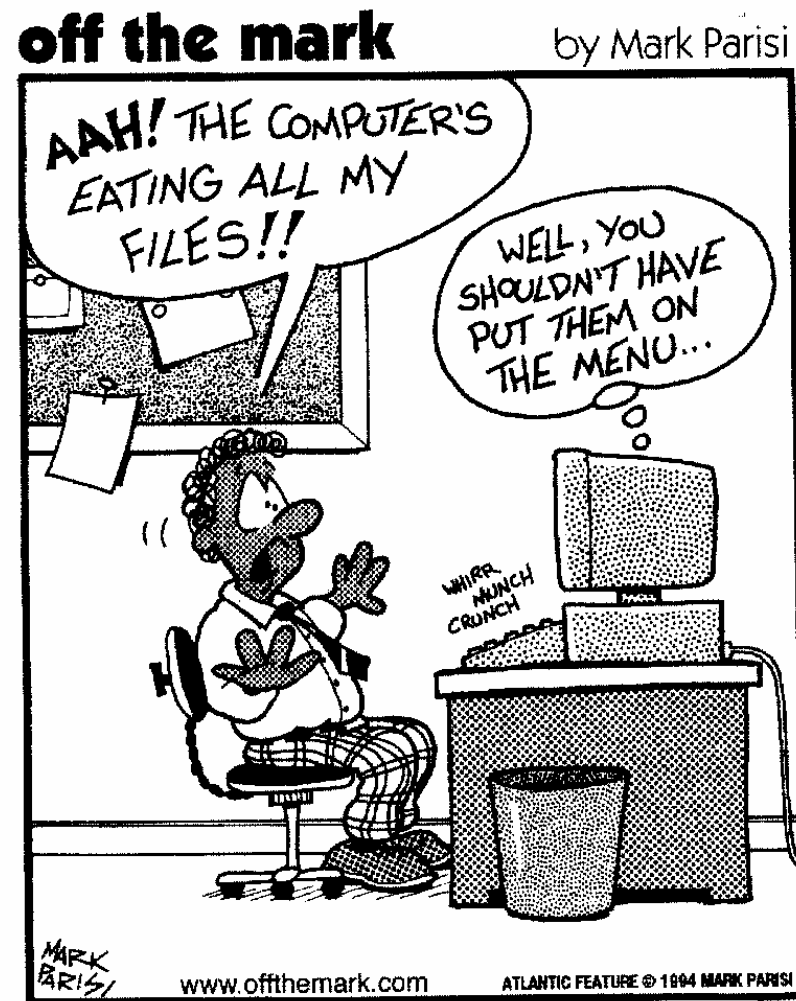
# Form filling

- requires
  - good design
  - obvious correction facilities
- *wizards*: interface leads user step-by-step through form
- sophisticated variant: spreadsheets
  - grid of cells for values or formulas
  - formulas can refer to other field values
  - user can enter and alter data *arbitrarily*, spreadsheet maintains consistency

	A	B	C	D	E
1	Amortization of a Loan				
2	Principal	Payment	Rate	Per Year	
3	10,000	200	0	4	
4	Principal	Amortized	Interest		
5	10,000	75	125		
6	9,925	76	124		
7	9,849	77	123		
8	9,772	78	122		
9	9,694	79	121		
10	9,616	80	120		
11	9,536	81	119		
12	9,455	82	118		
13	9,373	83	117		
14	9,290	84	116		



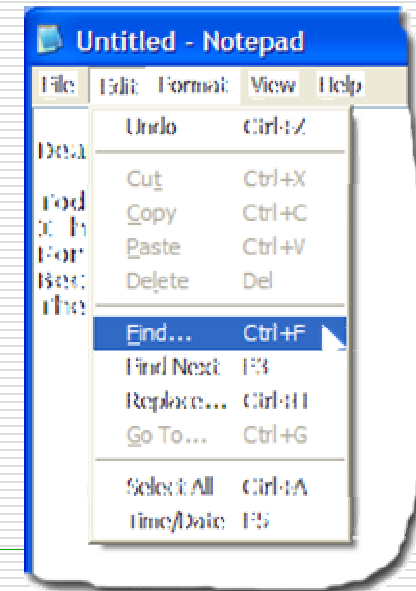
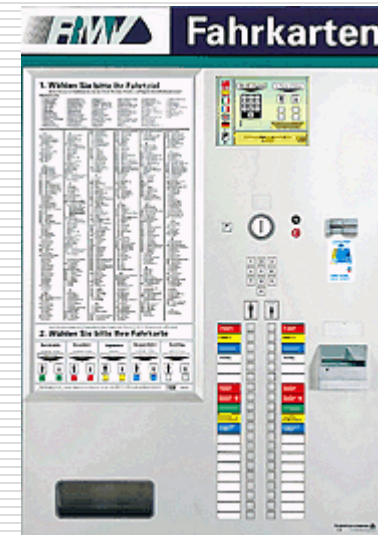
# Menus





# Menus

- menu =  
set of options displayed on screen, where the selection & execution of one (or more) of the options results in a state change of the interface (Paap & Roske-Hofstrand, 1989)
  
- user selects from *predefined* selection of operations *arranged* in menus
  
- selection by
  - Text input: numbers, keys/letters, speech (“shortcuts”)
  - Pointing: buttons, stylus, gesture
  - Positioning: arrow keys, mouse
  - Combination: mouse + “accelerator” key



# Menus

- items: understandable, meaningful and grouped well!
- Advantages
  - less learning as finding an item is recognition as opposed to recall
  - ideal for novice or intermittent users
  - can appeal to expert users if display and selection mechanisms are rapid and with appropriate "shortcuts"
  - affords exploration
  - structures decision making
  - allows easy support of error handling
- Disadvantages
  - too many menus may lead to information overload
  - may slow down experienced users
  - may not be suited for small graphic displays
- recommended for all users when complemented by menu commands or shortcuts



# Point & click interfaces

- just click something!

- icons, text links or location on map

- used in ..

- multimedia
- web pages and browsers
- hypertext

- minimal typing

- not tied to mouse, used for touch screens

- often combined with menu-based interfaces



# WIMP interface

GUIs out of basic building blocks

Windows

Icons

Menus

Pointers

(...and buttons, toolbars, etc.)



default style for majority of interactive computer systems,  
especially PCs and desktop machines

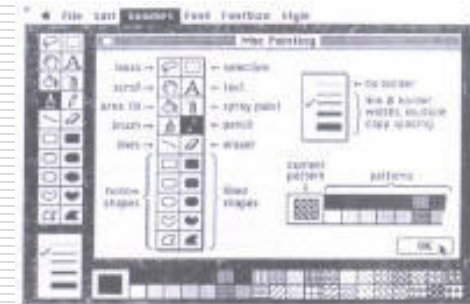
what does *wimp* mean literally?

a wimp is a weak, ineffectual or cowardly person

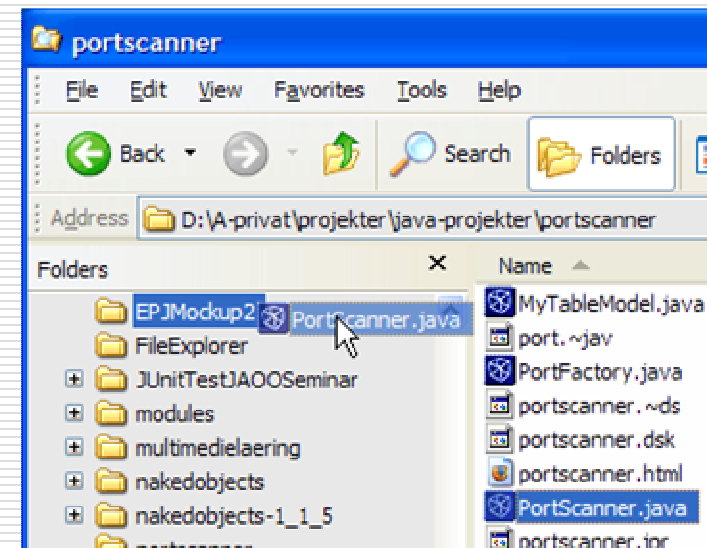


# Direct manipulation (Shneiderman, 1982)

- directly manipulate the *object of interest*
  - objects visible and distinguishable in the UI
  - rapid, reversible, incremental actions and feedback → can see results as you go
- Example: resizing a graphical shape, such as a rectangle, by dragging its corners or edges with a mouse
- Direct manipulation vs. WIMP/GUI interfaces
  - WIMP/GUI almost always incorporate direct manipulation, but need not
  - Direct manipulation does not imply use of windows or even graphical output (e.g. interfaces for blind users)



MacPaint



# Direct manipulation

## □ Advantages

- visually presents task concepts
- needs recognition memory as opposed to recall memory
- easy to learn and to retain
- errors can be avoided more easily
- encourages exploration
- high subjective satisfaction

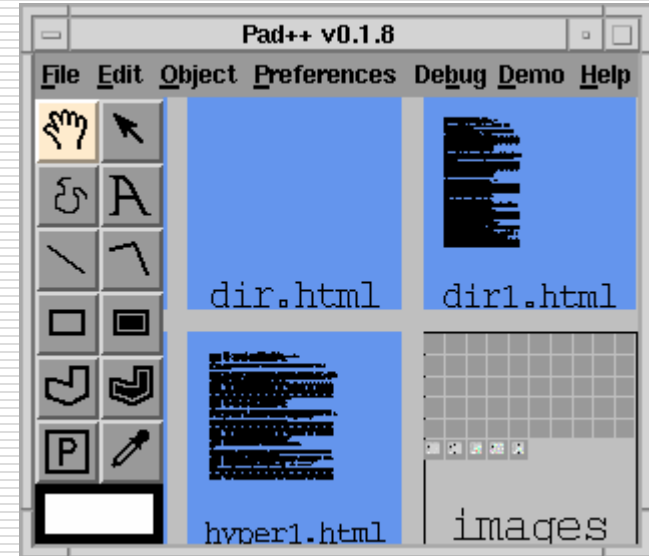
## □ Disadvantages

- may be more difficult to programme
- not suitable for small graphic displays
- spatial and visual representation is not always preferable
- may slow expert users down, compact notations may better suit them



# 3D interfaces

- in 'ordinary' window systems
  - highlighting (e.g. 3D buttons)
- *3D workspaces*
  - infinite virtual space
  - Light, size, and occlusion give depth
  - a lot like WIMP, but point & click in 3D (how does a 3D button look like?)
- *ZUI's: Zoomable UI's*
  - Navigation like panning a video camera
  - Zooming in on objects
- Virtual Reality



# Natural language

- Familiar to user
- speech recognition or typed natural language
- Problems
  - have to deal with syntax, semantics and pragmatics
  - vague
  - ambiguous
  - hard to do well!
- Solutions
  - try to understand a subset
  - pick on key words





# Speech-driven interfaces

- rapidly improving, but still inaccurate
- how to have robust interaction?
  - Dialogue, e.g. airline reservation: user with reliable “yes” and “no” + system reflects back its understanding (“you want a ticket from New York to Boston?”)
  - Multimodality

later more...



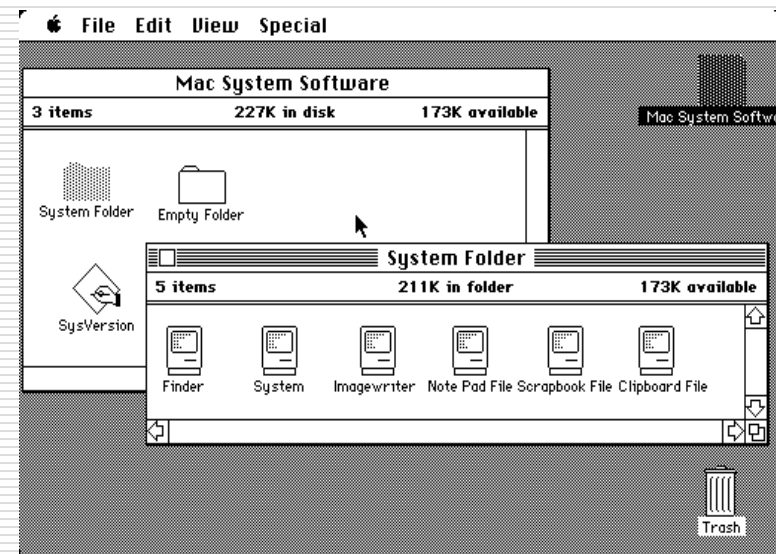
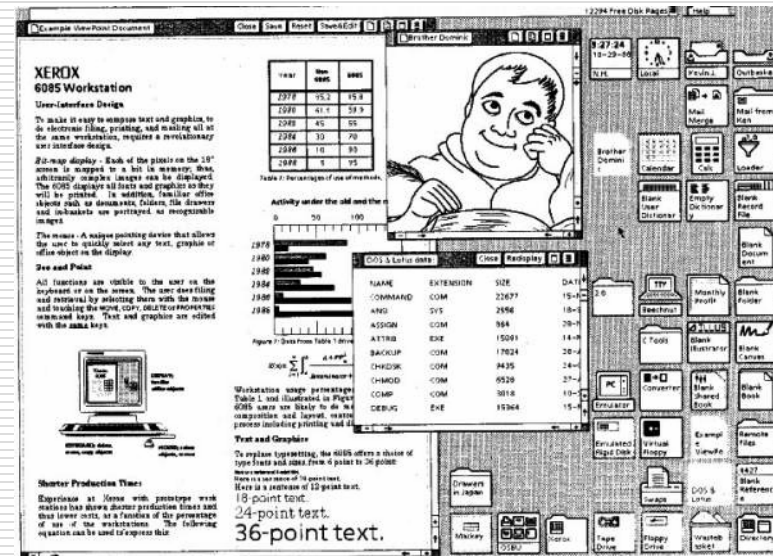
# WIMP/GUI interfaces

---

building blocks  
widgets  
look and feel

# history - review

- ❑ Xerox Alto (1973)
- ❑ Xerox Star (1981), desktop metaphor
- ❑ Apple Lisa (1979), document-centered view
- ❑ Apple Macintosh (1984), no more cursor keys
- ❑ MS Windows (1987-...)



# Windows

- Areas of the screen that behave as if they were independent
  - can contain text or graphics
  - can be moved or resized
  - can overlap and obscure each other, or can be laid out next to one another (tiled)
- scrollbars
  - allow the user to move the contents of the window up and down or from side to side
- title bars
  - describe the name of the window



# Windows

- pop up windows
  - take the user out of working context
  - user has to refocus attention
  - possibly obscure the working area
  - is the task in the pop up window really related to what the user is working on?
  - related tasks belong in the same window!



- tradeoff! mind!
  - time spent understanding & manipulating windows instead of on task



# Icons

- small picture or image
- represents some object in the interface
  - often a window, a folder, or action
- windows can be closed down (*iconifying*)
  - small representation if many accessible windows
- can take many forms
  - from highly stylized...
  - ...to realistic representations



# Excursion: „Icon“

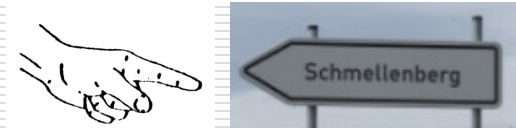
- Ferdinand de Saussure (*Cours de linguistique générale*, 1916): a *sign* („Zeichen“) consists of
  - *signifier* („Bezeichnendes“)
  - *signified* („Bezeichnetes“)

- Charles Sanders Peirce (*Grundbegriffe der Semiotik und der formalen Logik*, 1898-1902): three basic relations between signifier & signified object:

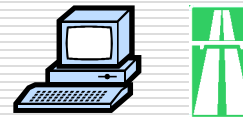
1. *symbol*: connected to object by convention



2. *Index*: points to object in context



3. **Icon**: resembles object

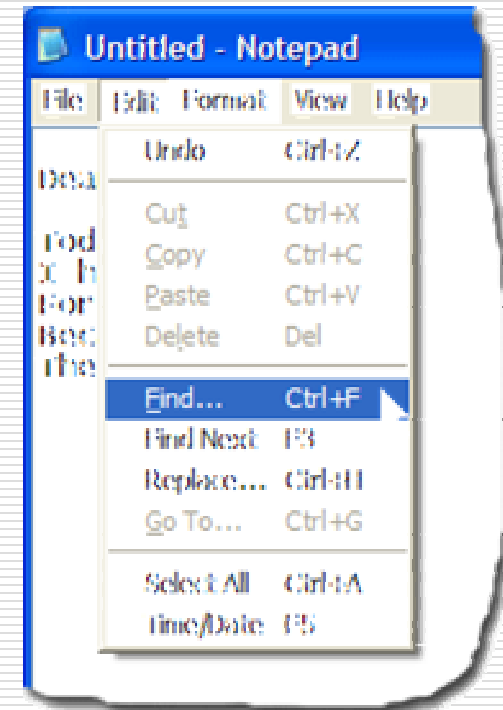


# Menus

- (see above) choice of operations arranged in a structured way
- selection with pointer
  
- Cascading menus
  - hierarchical menu structure
  - menu selection opens new menu
  - and so in ad infinitum

## problems

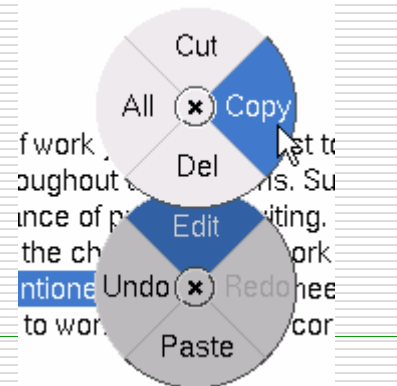
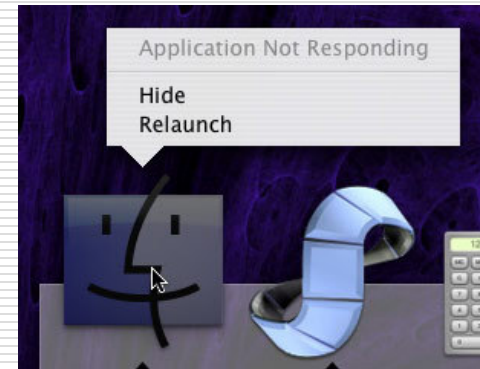
- take up a lot of screen space
- inefficient when too many items
- deep hierarchies are easy to create – but seldomly found in users' mental models





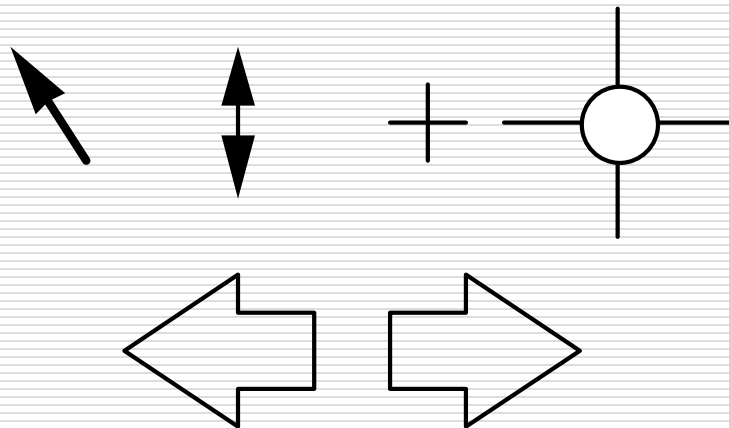
# Kinds of Menu

- Menu Bar at top of screen (normally), menu drags down
  - *pull-down menu* - mouse click to drag down menu
  - *pop-down menu* - stay as long as mouse button depressed
  - *fall-down menus* - mouse just moves over bar
  
- *Contextual* menu appears where you are
  - pop-up menus - menu appears when needed, offer actions for selected object
  - pie menus - arranged in a circle
    - easier to select item (larger target area)
    - quicker (same distance to any option)
    - comply with Fitt's law
    - not widely, but increasingly used



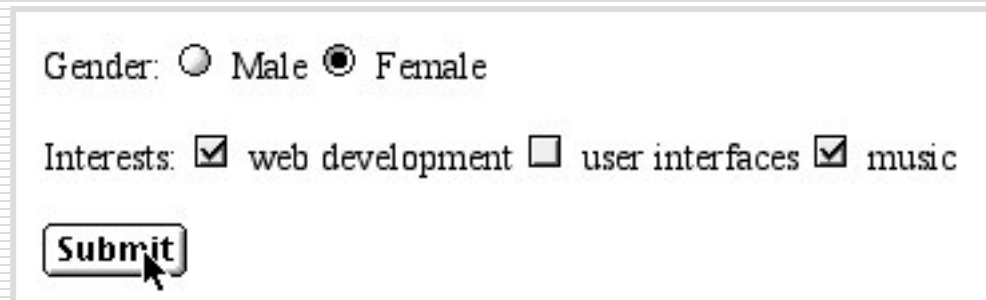
# Pointers

- important component - WIMP style relies on pointing and selecting things!
- uses mouse, touchpad, joystick, trackball, cursor keys or keyboard shortcuts
- wide variety of graphical representations



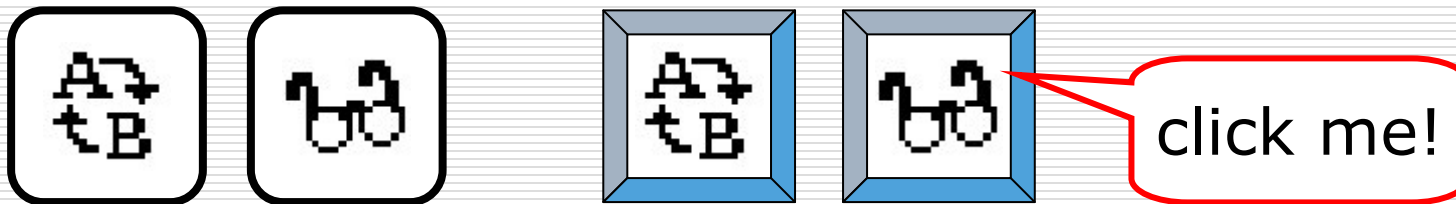
# Buttons

- individual and isolated regions within a display that can be selected to invoke an action
- Special kinds
  - radio buttons – exclusive choices
  - check boxes – non-exclusive choices
  - icon buttons
  - flat vs. sculptured



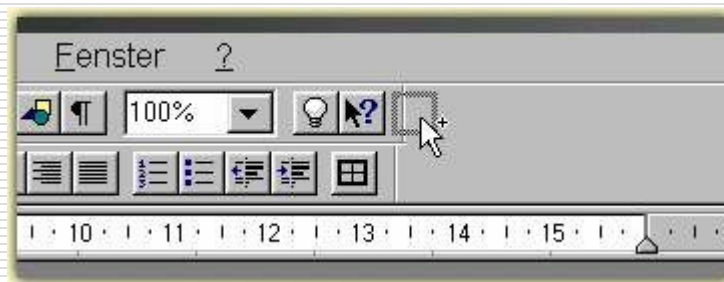
Gender:  Male  Female

Interests:  web development  user interfaces  music



# Toolbars

- long lines of buttons ... but what do they do?
- similar to menu bar, but
  - faster access
  - more can be displayed
- often customizable:
  - choose *which* toolbars to see

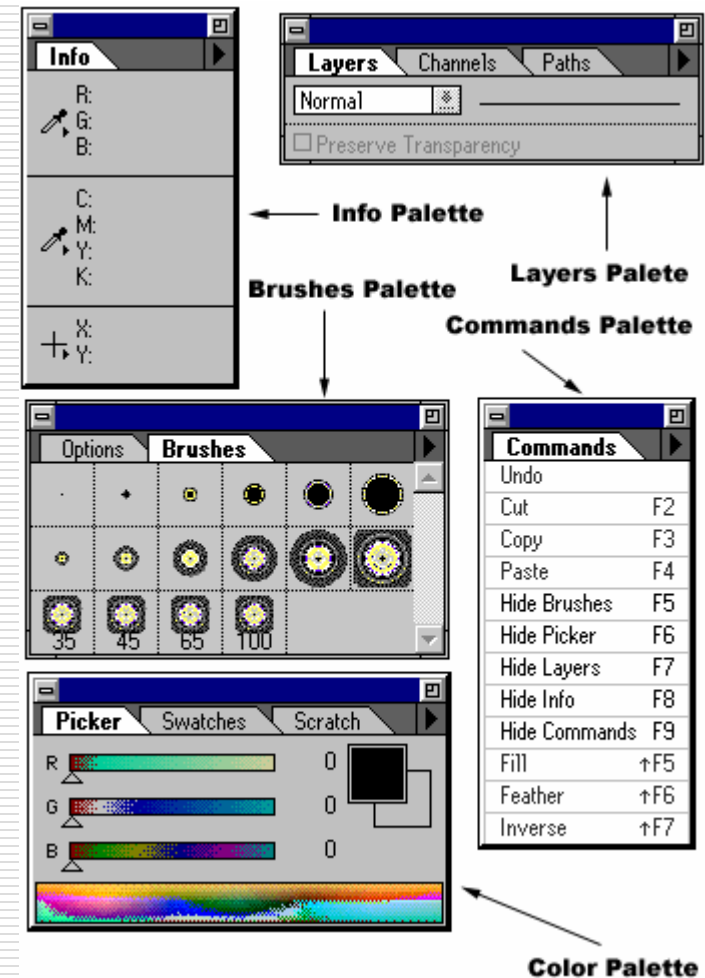


on it



# Palettes

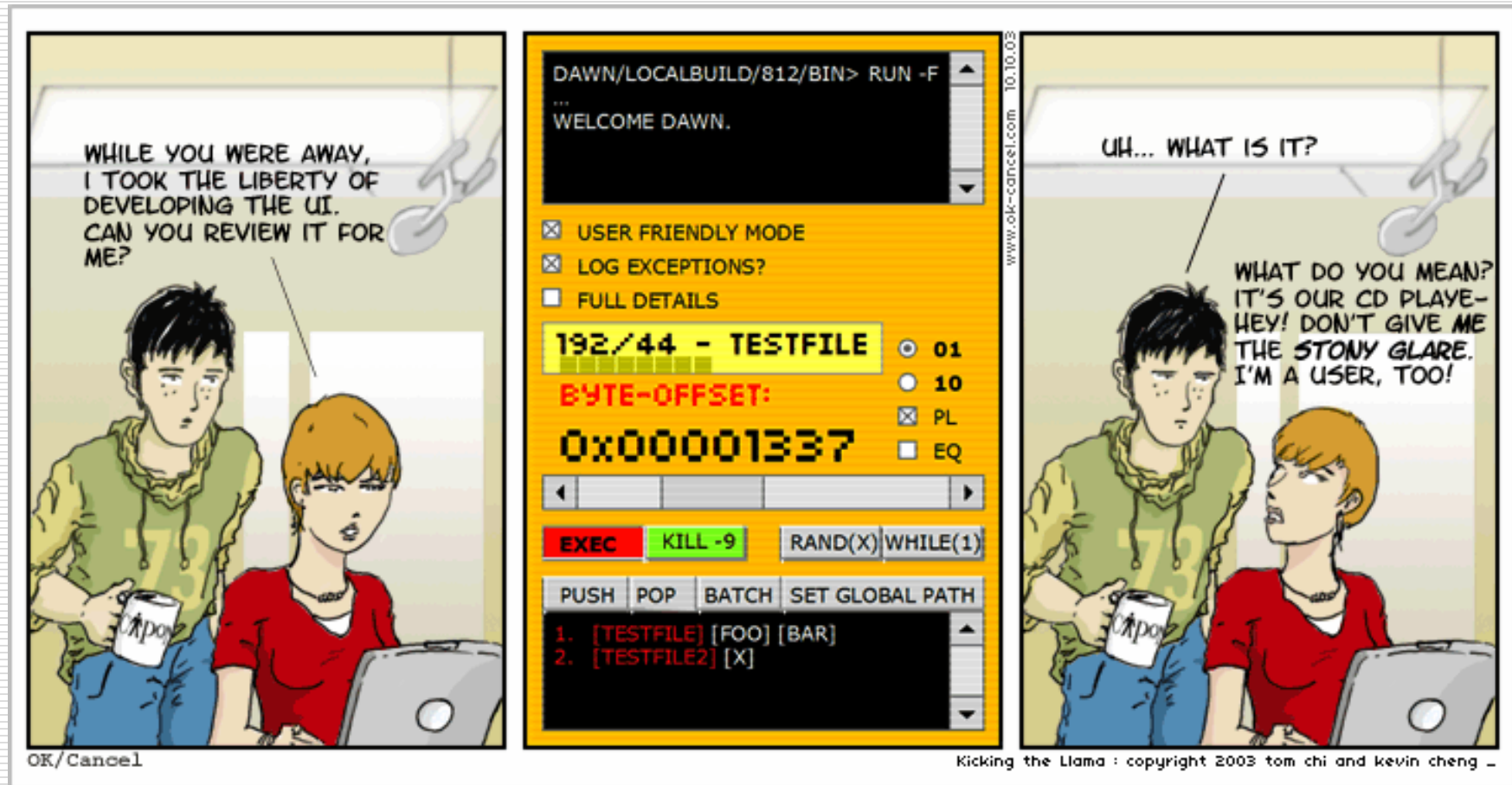
- Problem: menu not always there when you want it
- Solution: *palettes* – little windows of actions
  - shown/hidden via menu option e.g. available shapes in drawing package
  - tear-off and pin-up menus
  - menu 'tears off' to become palette
  - can be 'pinned on' screen to become palette



Adobe Photoshop



# handle with care...



# understanding and choosing *widgets*

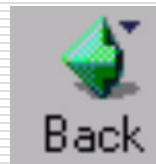
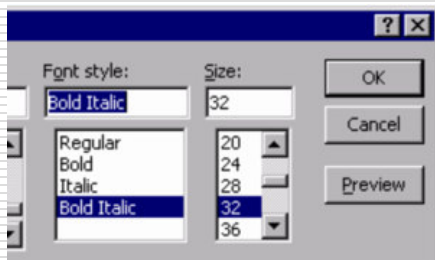
## □ Widgets

### ■ **window gadget**

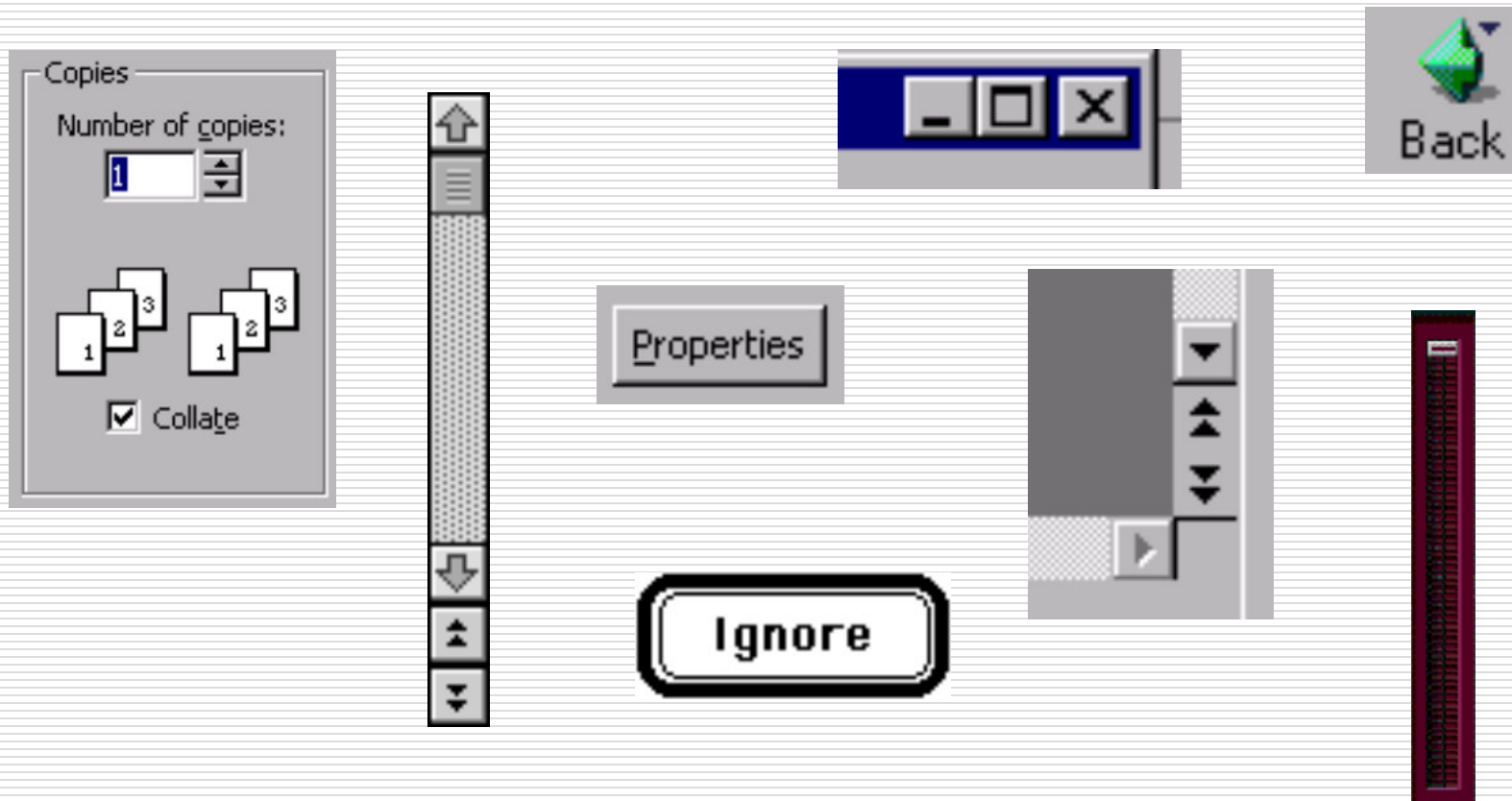
- bits that make the graphical user interface (GUI)
- checkboxes, menus, toolbars, buttons etc.

## □ three aspects:

- *appearance* - what they look like
- *behavior* - how they behave in interaction
- *semantics* - what they mean



# appearance



appearance = shape + color + position + symbols/icons + words

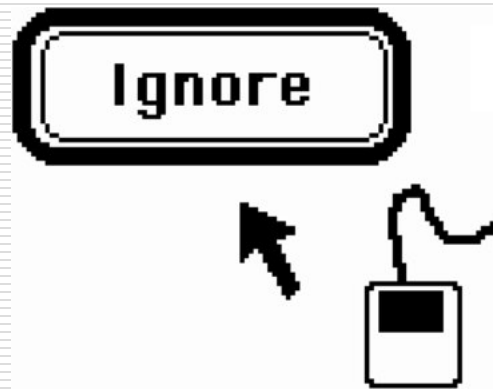




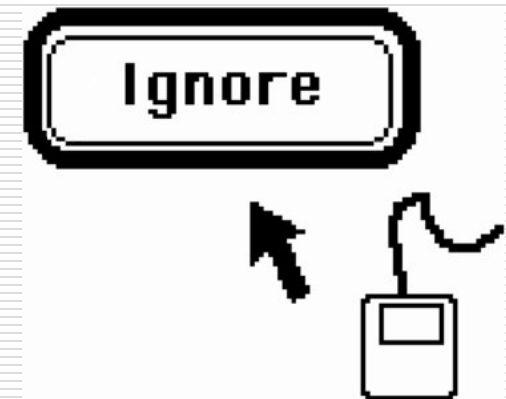
# behaviour



Move mouse over button  
– highlights



Move mouse off target with  
button still down  
– highlight removed



Release mouse  
– nothing happens



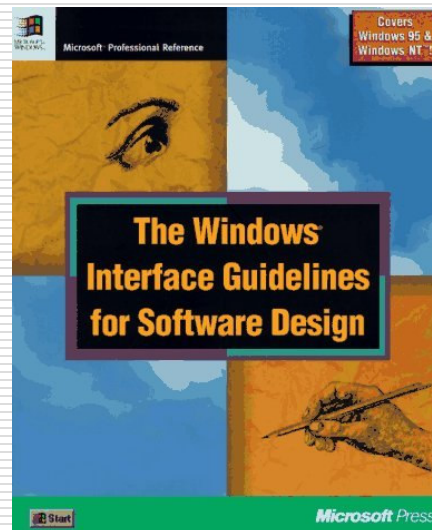
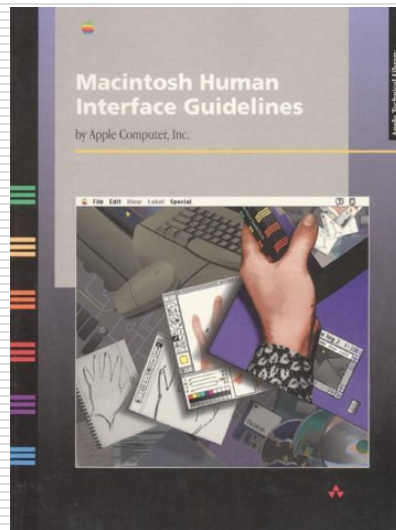
## ...at your fingertip - in *Java*:

- onMouseClick(Location point)
- onMouseEnter(Location point)
- onMouseExit(Location point)
- onMousePress(Location point)
- onMouseRelease(Location point)
- onMouseMove(Location point)
- onMouseDrag(Location point)



# look and ... feel

- All WIMP systems have the same elements (windows, icons., menus, pointers, buttons, etc.)
- but different GUIs *behave* differently!  
e.g. MacOS vs. Windows menus
- appearance + behaviour = “*look & feel*”



JAVA™  
LOOK AND FEEL  
DESIGN GUIDELINES  
SECOND EDITION



# semantics

- menus, buttons, etc. do things ...
  - “operational” semantics
  - semantics up to you!
1. think separately:
    - meaning first – what you want it to do
    - then appearance+ behavior – how you do it
  2. choose the widget for the job



... lets make it ***bold italic***



# what do you want?

## actions

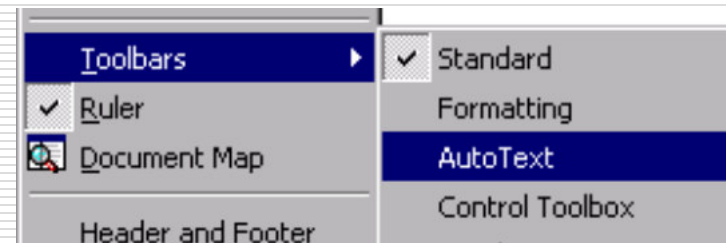
- usually menu, buttons, or toolbar

## setting state/options

- usually checkbox, radio button, combi-box

## but ...

- menus can be used to set state etc. ...



# setting options – how many?

- one of several options
  - radio buttons, selection menu
- zero, one or more options
  - checkbox, multi-choice menu
- free choice
  - offer recent/typical shortcuts
  - one line text boxes often terrible

favourite colour?  
 red  blue  orange

Word wrap text in autoshape  
 Resize autoshape to fit text  
 Rotate text within autoshape by 90°

Select the buttons you want to see in the toolbars

Bookmarks     Search  
 Go             Print  
 Home



## and more ...

### number of options

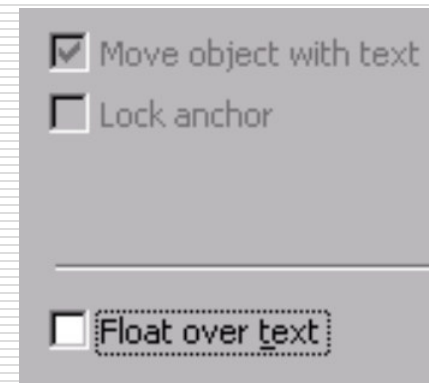
- fixed e.g. bold, italic, underline
- variable e.g. font list
- scrolling through telephone list ...

### "liveness" of options

- grey out inactive options

### dynamic interactions

- some choices dependent on others



## other important concepts

- metaphors
  - affordances
- principles & guidelines...

