Interaction Styles

- choice of the interaction style: important decision in the design of the interface
- styles may be mainly based on keys, objects manipulations or linguistics:
  - menu
  - form fill-in
  - command language
  - natural language
  - direct manipulation
- each style has its own way of
  - presenting information
  - organising the system's functionality
  - managing user's input
- a good task analysis can help choosing the most appropriate style for a given task

Menu

- the user reads a list of items, selecting the most appropriate one
- advantages:
  - easy to learn
  - reduces keystrokes
  - permits use of dialogue-management tools
  - allows easy support of error handling
- disadvantages:
  - presents danger of too many menus
  - may slow frequent (expert) users
  - consumes screen space
  - requires rapid display rate

Form fill-in

- (or *fill in the blanks*) the user sees a display of related fields, moves a cursor among the fields, and enters data
- advantages:
  - simplify data entry
  - requires modest training
  - gives convenient assistance
- disadvantages:
  - consumes screen space
  - requires a keyboard
  - requires handling typing errors

Command Language

- the user types in commands following a given syntax
- advantages:
  - is flexible
  - support user initiative
- disadvantages:
  - has poor error handling
  - requires substantial training and memorization

Natural Language

- the user types in commands in English (or any other human language)
- advantages:
  - relieves burden of learning syntax
  - is more natural
  - automatically adapts to expert/novice users
- disadvantages:
  - hard to program (AI techniques)
  - requires clarification dialogues
  - may require more keystrokes
  - may not show context
  - is unpredictable

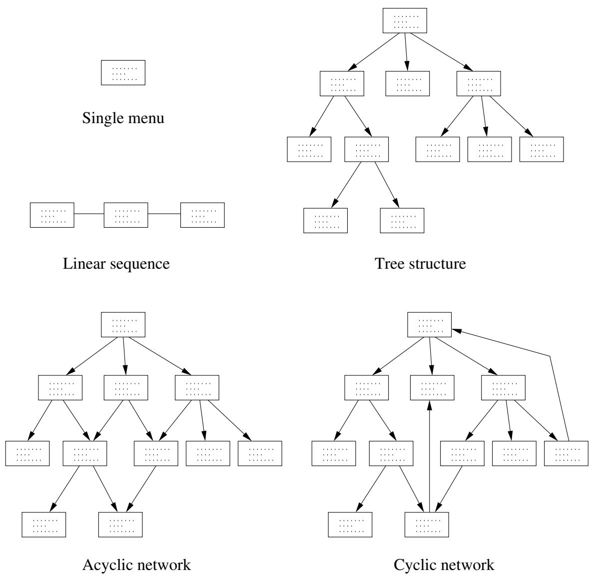
Direct Manipulation

- involves a visual representation of the world on which the user can operate directly
- advantages:
  - tasks are represented visually
  - easy learning and use
  - better control over (some) errors
  - high subjective satisfaction
  - keyboards can be eliminated and other devices used (e.g. touch-screens)
- disadvantages:
  - hard to program
  - require graphic displays
  - require pointing devices

Menu Facts

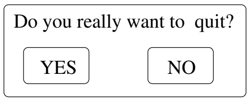
- require the user only to understand and recognize the options (little learning and memory effort)
- guide the user step by step suggesting viable options and hiding inappropriate options, so prevent some mistakes (e.g. using commands that cannot be used in a context)
- better for beginners (experts prefer commands, or shortcuts)
- need to be organized into structures (this can limit their flexibility, compared to, for instance, commands)
- can be very demanding in terms of screen space, and may require the user to navigate through several panels

Menu Structures

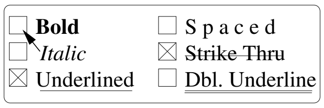


Single Menus

- may have two or more items, or allow multiple selections
- simple case: **binary menu** (yes/no, true/false, etc. kind of choices)

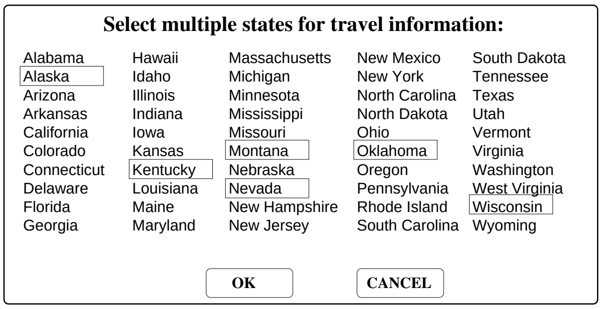


- a bit more complex: **multiple-selection menus** (check boxes)

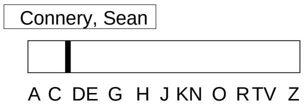


- single menus can be constantly available on the screen or appear in response to a mouse click (**pop-up menus**)

- when the list of menu items is vast (more than 20 lines), but the constraint to have a single conceptual menu is strong, a scrolling list, or a "fast and vast" organization can be used:



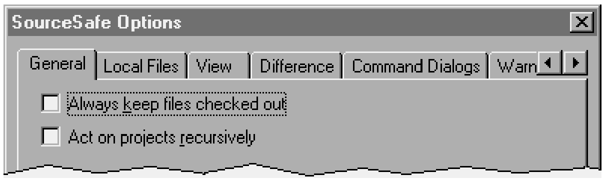
- if the menu items are too numerous to show on the screen at once, more compact strategies can be used, e.g. *alphasliders*:



- another example of single menu is.. a hypertext!

Linear Sequence Menus

- a series of interdependent menus guides the user through a series of choices
- several menus can be put on a single dialogue box, in order to simplify the selection: the user can enter choices in any order

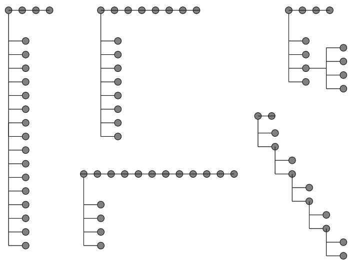


Tree Menus

- when choices become too many to be handled in one single screen, categories can be formed of similar items, creating a structure
- this way, huge numbers of items can be represented: a menu with 4 levels, each having 30 items, presents a total of 810,000 choices! (that is, 30^4)
- if grouping is done sensibly, menu traversal can be done very quickly
- but, if you have to decide where to go, and the menu options are vague, you can get lost for hours
- categorization and grouping are not easy tasks: often there can be several acceptable classifications

Deep or broad?

- *Depth*: number of levels
- *Breadth*: number of items per level
- several ways to group, e.g. for 64 menu items:
  - 8 items on 2 levels
  - 4 items on 3 levels
  - 2 items on 6 levels
  - a 4 item menu followed by a 16 item menu
  - a 16 item menu followed by a 4 item menu



- several studies were made, and there is strong evidence that breadth should be preferred over depth
- rule of thumb: limit menu trees to 3-4 levels

Network Menus

- in some cases, network are more appropriate than trees
- networks allow reaching an option from many different starting points
- an example: a Web site!
- problem: disorientation
- creating a mental model of a network is more difficult than creating a mental model of a tree
- problem of the backward: a stack of visited menus must be kept (you cannot rely on the parent-child relationship)
- the notion of "level" may be useful, that is letting the users know how far they are from the main menu

Menu-specific guidelines

- use task semantics to organize menus (e.g. linear sequence, tree structure, acyclic or cyclic network structure etc).
- prefer broad-shallow to narrow-deep
- show position by graphic, number or titles
- use items as titles for sub-trees
- group items meaningfully
- sequence items meaningfully
- use brief items, begin with the keyword
- use consistent grammar, layout, terminology
- allow shortcuts
- enable jumps to previous and main menus
- consider online help, optimal response time, display rate, screen size

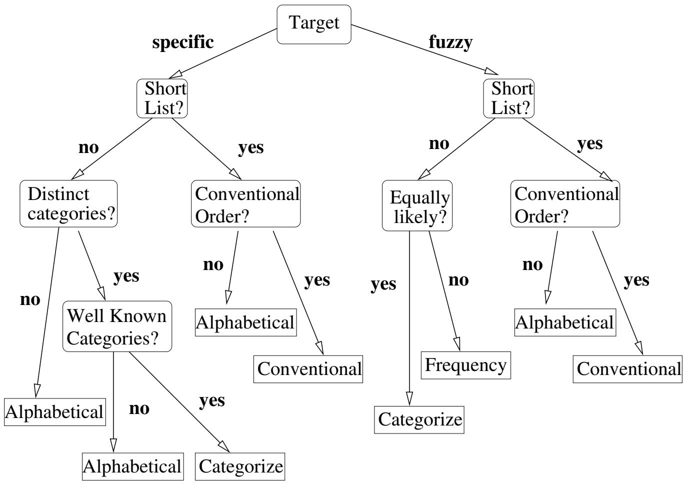
How to group?

- there is no single way, and what is OK for a user can be bad for another (how do you put your desk in order?)
- guidelines for grouping:
  1. group logically similar items, e.g. all actions related to file management (save, copy, open etc.)
  2. form groups that cover all possibilities, e.g. a menu with age ranges $0-9$, $0-19$, $20-29$, and > 30
  3. make sure items are not overlapping, e.g. Entertainment/Events is poorer than Concerts/Sports
  4. use familiar terminology, but ensure that items are distinct from one another (Day/Night)
- possible solution: use task analysis!

Item Presentation Sequence

- once the grouping has been chosen, there is still the problem of ordering the items in the same sub-menu
- if the items have a natural order, things are easy:
  - Time (e.g. years, months, etc)
  - Numeric ordering (e.g. ranges of values)
  - Physical properties (increasing/decreasing length, area, velocity etc.)
- when there is no task-related orderings, the designer must choose from such possibilities as:
  - Alphabetic sequence of terms
  - Grouping of related items, using blank lines or other demarcation between groups
  - Most frequently used items first
  - Most important items first (important to whom?)

Guidelines for Ordering Items



Menu Layout Issues

- Titles:
  - For single menus, use a simple descriptive title.
  - For tree-structured menus, use the exact same words in the higher-level menu items as in the titles for the next lower-level menu
- Phrasing of menu items
  - Use familiar and consistent terminology
  - Ensure that items are distinct from one another
  - Use consistent and concise phrasing
  - Bring the keyword to the left, so that the first word is the most significative (e.g. Save File rather than File Saving)
- Graphic layout and design:
  - screen width and length
  - display rate
  - character set (small/wide)
  - highlighting techniques (graphic/text only)

- Consistency of layout of at least:
  - Titles (centered, left justified, etc)
  - Item placement (typically left justified)
  - Instructions (identical in each menu, and in the same position)
  - Error messages position
  - Status reports (if you indicate which portion of the menu structure is currently being searched)
- Techniques to show the depth of the currently seen sub-menu:
  - Indentation
  - Upper/lower case characters
  - Symbols such as $*$ or $-$ to create separators or outlines
  - Position markers
  - Cascading menus or walking menus (such as the one starting from the Start button in Windows '95). Consider that "walking" can be difficult for some users

Menu Shortcuts: Typeaheads

- a natural way to permit frequent users to speed through the menus
- the user do not have to wait to see the menu before choosing the item, but can type a string of letters when presented with the main menu

Find

- if the menu items are identified with single letters, then the concatenation of menu selections generates a command name

Print → Black and White → Landscape

= PBL

- the strategy is known as the *BLT approach*
- Problems:
  - requires a more elaborate parser for input
  - nonexistent menu choices difficult to handle
  - have to ensure distinct 1st letters for items within each menu

Menu Shortcuts: Accelerators

- menus often offer *keyboard accelerators*
- these are key combinations that have the same effect as selecting the menu item
- while typeaheads are available only for the current level of menus, keyboard accelerators can be defined for any menu option, at any depth
- accelerators allow expert users to invoke an option without moving off the keyboard
- the accelerators should be displayed alongside the menu items, so that frequent use makes them familiar
- for example:

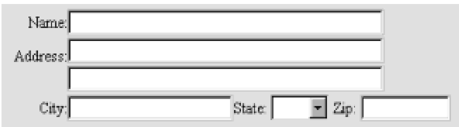
Find F3

- PROBLEM: most systems allow you to use the accelerators only if the menu is NOT displayed!
  - in the example, you could only click on the menu option or use the typeahead F, but pressing F3 would not work

Form Fill-in

- if data entry is required, menus can be very cumbersome:

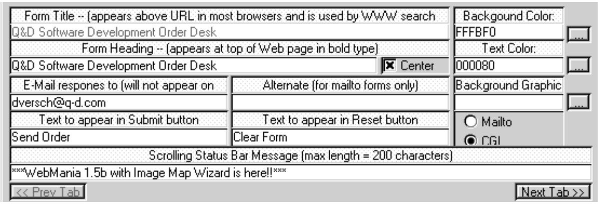
- when many fields of data are necessary, the appropriate interaction style is the form fill-in



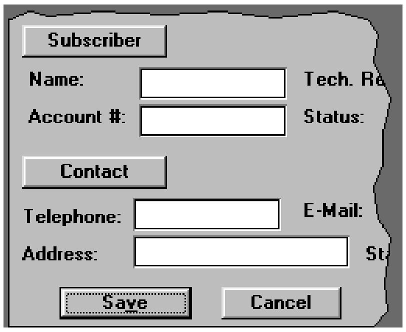
- in terms of locus of control, forms are very attractive, as the users feel they are in control of the dialogue
- on the other hand, users need to be familiar with keyboards (backspace, enter keys etc.), use of some way to move across the form (TAB key, cursor, mouse), field-labels and permissible field contents
- form fill-in is widely used on the World Wide Web

Form Fill-in Specific Guidelines

- meaningful title
- comprehensible instructions
- logical grouping and sequencing of fields (e.g. city field close to the postal code field)
- clear layout
- familiar field labels (Home Address is better than Domicile)
- consistent terminology and abbreviations
- visible space and boundaries for data entry fields
- error prevention, when possible
- error messages for unacceptable values
- marking of optional fields
- explanatory message for fields
- explicit closure: the user should know what to do when the data entry is finished



Labels not aligned to the fields. Difficult to read (eyes zig-zag around the screen)



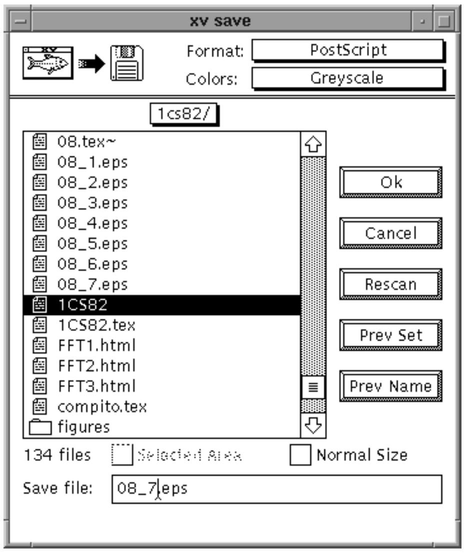
Section titles appear like buttons

Coded Fields

- some data may require a specific format
- a layout can be provided to help the user
- examples:
  - telephone numbers:
    `Telephone: (_ _ _) _ _ _ - _ _ _ _`
  - times:
    `_ _ : _ _  _ _  (09:45 AM or PM)`
  - dates:
    `_ _ /_ _ / _ _ _ _`
    `(03/04/1999 indicate 3rd April 1999)`
  - a currency amount:
    £ `_ _ _ . _ _`
- but: consider international issues!!

Dialogue Boxes

- sometimes users are requested to both make choices and enter data in order to perform a task
- dialogue boxes combine menu selection and form fill-in



Unix *xv* “Save” dialogue box

Dialogue Boxes Specific Guidelines

- Internal layout:
  - meaningful title, consistent style
  - top-left to bottom-right sequencing
  - clustering and emphasis
  - consistent layouts (margins, grid, white space, lines, boxes)
  - consistent terminology, fonts, capitalization, justification
  - standard buttons (OK, Cancel)
  - error prevention by direct manipulation
- External relationship:
  - smooth appearance and disappearance
  - distinguishable but small boundary
  - size small enough to reduce overlap problems
  - display close to appropriate items
  - no overlap of required items
  - easy to make disappear
  - clear how to complete/cancel