

HUMAN COMPUTER INTERACTION

Styles for Interaction Design

Dr. Debasis Samanta



INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR

3.0 Interaction styles

A range of different terms has been used in HCI for describing the communication that occurs between the users and the computers. Until recently, most interactions with a computer involved strictly defined turn taking in which text was typed in and the responses were displayed on a screen. This is still a widespread and important way of interacting with a computer. For this kind of interaction, it makes sense to use the term dialogue to refer to the exchange of instructions and information that takes place between a user and a computer system. From a user's point of view, a typical if idealized dialogue can be broken down into the following stages

- 1) Instructions are specified in some form to the computer, this may involve speaking, pointing, typing, gesturing and so on. The user hopes that her intentions were correctly specified and that the computer has carried them out as required.
- 2) If it has been thoughtfully and well programmed, the system will provide some feedback. The user hopes for the desired result but failing this, a message explaining what has happened is displayed.

Richer styles of communication between a user and the computer have now become commonplace. Human computer communication often takes non verbal forms such as the manipulations of objects and tools, the indication of points, paths and areas and various types of gesturing. So, it is necessary to have a wider perspective. Many experts view the exchanges that occur between the users and computers more generally, and describe them as interactions rather than as dialogues...

So, **interaction styles** can be seen as generic term to include *all the ways in which users communicate or interact with computer systems*.

3.1 Interaction Design: An overview

Interaction Design is the creation of a representational dialogue among people and intelligent products, environments and communications encountered in their everyday experience.

As products become more complicated, often due to technology, designers are facing new challenges in gaining strong user acceptance. The number of people using and developing products powered by some sort of technology continues to grow; these new systems are more functional and robust than ever, providing more features, functionality, and capabilities.

However, with the added complexity, the new generation of products is becoming more difficult to understand and use. As a result, users become more frustrated, unhappy and less productive. Large portions of functionality in the complicated software products and consumer electronics go unused, and products often fail in the consumer world due to their unnecessarily complicated user interfaces.

Interaction Design exists as the design of behavior, bounded by three core interests:

- 1) Human
- 2) Technical
- 3) Aesthetic.

Interaction design is the branch of user experience design that illuminates the relationship between people and the machines they use. While interaction design has a firm foundation in the traditional user interface design, its focus is on defining the complex dialogues that occur between people and interactive devices of many types-from computers to mobile communication devices to appliances. Best practices in interaction design can be described with an Interaction design pattern

Historically, the term interaction design has its roots in GUI-design. Interaction design has become more and more concerned about other interactions than the ones happening between a single user and a digital device.

"Interaction design used to be primarily about the aesthetics of the interactive experience - how it makes the user feel - whereas now it is increasingly concerned with the social and political implications of new technologies".

One can ask if interaction design is tied only to situations where digital devices are involved. Could, for example, the design of party games and social events be seen as interaction design?

Interaction designers strive to create useful and usable products and services. Following the fundamental tenets of user-centered design, the practice of interaction design is grounded in an understanding of real users-their goals, tasks, experiences, needs, and wants. Approaching design from a user-centered perspective, while endeavoring to balance users' needs with business goals and technological capabilities, interaction designers provide solutions to complex design challenges, and define new and evolving interactive products and services.

The success of products in the marketplace depends on the design of high-quality, engaging interactive experiences. Good interaction design

- effectively communicates a system's interactivity and functionality
- defines behaviors that communicate a system's responses to user interactions
- reveals both simple and complex workflows
- informs users about system state changes
- prevents user error by using techniques such as behavior-shaping constraints

While interaction designers often work closely with specialists in visual design, information architecture, industrial design, user research, or usability, and may even provide some of these services themselves, their primary focus is on defining interactions.

The discipline of interaction design produces products and services that satisfy specific user needs, business goals, and technical constraints. Interaction designers advance their discipline by exploring innovative design paradigms and technological opportunities. As the capabilities of interactive devices evolve and their complexity increases, practitioners of the discipline of interaction design will play an increasingly important role in ensuring that technology serves people's needs. The recent proliferation of technologies used in communication and interaction, often using the internet for connectivity, has inspired interest in social interaction design. Designers increasingly recognize that user interactions often include other users as well as a mediating technology, and that interactions can include interpersonal and social factors as well as the conventional ones described here.

In summary, interaction design defines

- the structure and behaviors of interactive products and services
- user interactions with those products and services

In this paper, we focus on the following styles of interaction.

- 1) Command Entry
- 2) Menus and navigation
- 3) Form fills and spread sheets

3.1.1 Command Line:

A **command line interface** or **CLI** is a method of interacting with a computer. Commands are entered as lines of text (that is, sequences of typed characters) from a keyboard, and output is also received as text. CLIs originated when teletype machines were connected to

computers in the 1950s. In terms of immediate interaction and feedback, they represented an advance over the use of punch cards.

With the use of CRTs as interface devices, CLIs began evolving toward graphical user interfaces (GUIs) like Microsoft Windows, the X Window System and Mac OS, and were largely supplanted by GUIs when Microsoft introduced Windows. A significant minority of computer users prefer to use CLIs, some due to visual disability, and others because they feel that CLIs provide an environment with enhanced productivity. They are most often used in scientific, engineering, and technical environments, by programmers, especially in Unix-based operating systems.

In its simplest form, a CLI displays a prompt, the user types a command on the keyboard and terminates the command (usually with the Enter key), and the computer executes the command, providing textual output.

A program that implements such interface is often called a command line interpreter or **shell**. Examples include the various Unix shells (sh, ksh, csh, tcsh, bash, etc.), the historical CP/M, and DOS's command.com ("Command Prompt"), the latter two based heavily on DEC's RSX and RSTS CLIs. Microsoft's next operating system, Windows Vista, will include support for a new command line interface named MSH (Microsoft Shell, codename *Monad*), which hopes to combine features of traditional Unix shells with their object oriented .NET framework. Windows current CLI programs like DOS and Windows Script Host are commonly considered inadequate or insecure. MinGW is a third-party software for Windows that offers a true Unix CLI.

Some applications provide both a CLI and a GUI. One example is the CAD program AutoCAD. The engineering/scientific numerical computation package Matlab provides no GUI for some calculations, but the CLI can handle any calculation. The three-dimensional-modelling program Rhinoceros 3D (used to design the cases of most cell phones, as well as thousands of other industrial products) provides a CLI (whose language, by the way, is distinct from Rhino's scripting language). In some computing environments, such as the Oberon or Smalltalk user interface, most of the text which appears on the screen may be used for giving commands.

The commands given to a CLI are often of the form

[doSomething] [how] [toFiles]

or

[doSomething] [how] < [inputFile] > [outputFile]

doSomething is in effect to a verb, *how* an adverb (for example, should the command be executed "verbosely" or "quietly") and *toFiles* an object or objects (typically one or more files) on which the command should act. The '>' in the second example is a redirection character, telling the command line interpreter to send the output of the command not to the screen but to the file named on the right of the '>'. Another redirection character is the pipe ('|'), which tells the CLI to use the output of one command as the input to the next command; this "operator-stream" mechanism can be very powerful, as explained under pipeline (Unix).

A comparison: Commands provide a way of expressing instructions to the computer directly. They can take the form of function keys, single characters, short abbreviations, whole words, or a combination of the first two. (For ex, using the <Ctrl > key with the letter 'e' to mean Exit.)

The advantage of using the single character or the function keys is that only one or two keystrokes are required to execute the command compared with two or more when using a combination such as a word, or an abbreviation. (for ex, having to type exit)

The disadvantage with using single character is that it is generally more difficult to remember an arbitrary letter than a well chosen command name or abbreviation, especially if there are a large number of them. Giving commands appropriate names is important because it helps users to remember what the command refers to. (for example, in autocad, to draw a line, we need to type 'l', but for more complex process like extruding a solid, we need to type the whole word extrude)

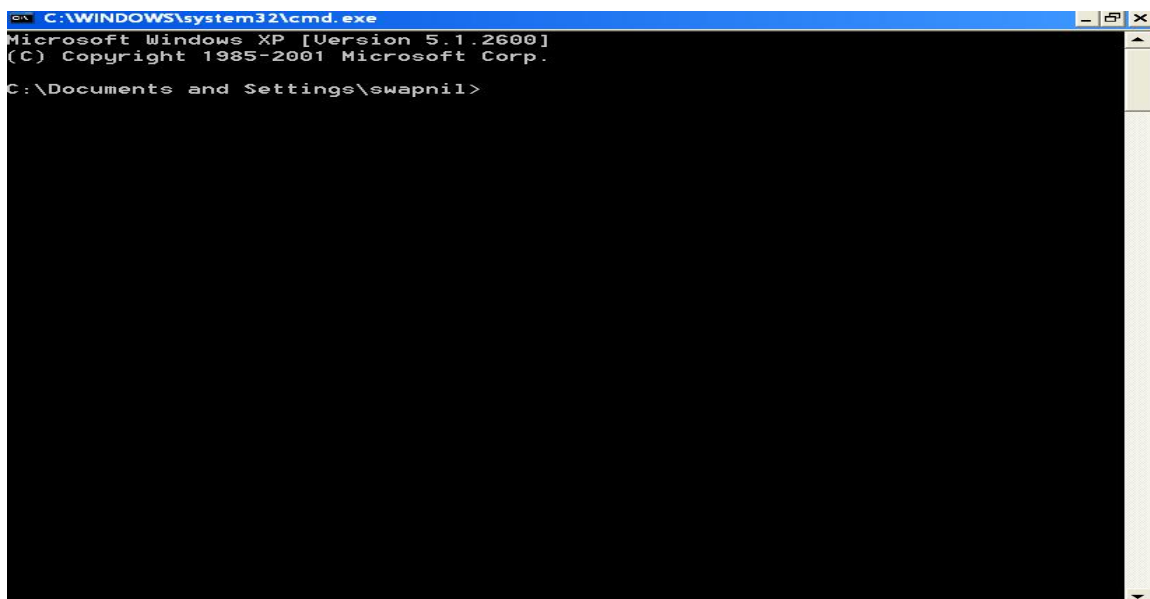


Fig. 3.1 Screen shot of a command line based interaction

Summary of CLI

Earliest commonly used interactive style.

Direct commands issued in a specific language.

Commands may be whole words, abbreviations or function keys.

Typically entered via a keyboard

May support creation of user specific macros

Advantages:

- Flexibility and speed for expert users.
- Provide a sense of control for expert users.
- Low resource usage.

Disadvantages:

- Requires training and recall of exact syntax
- Low visibility of system status
- Poor error handling

Examples:

```
copy *.doc c:\data\
```

```
grep -v ^$ filea fileb
```

Command language: guidelines

- Commands should be short, meaningful with consistent rules for abbreviation -- consider using mnemonics for abbreviations.
- Design for the tasks that the user will be performing.
- Facilitate error correction through allowing recall and editing of previous commands.
- Use a hierarchical structure or permit concatenation to form variations on commands.
- Provide facilities to construct macros – for expert users.

3.1.2 Menus and Navigation:

A menu is set of options displayed on the screen where the selection and execution of one or more of the options results in a change in the state of the interface. Unlike command driven systems, menus have the advantage that user do not have to remember the items they want. The only need to recognize it. This means that for menus to be effective, the names and icons selected have to be self explanatory which is not always the case.

Designers have to decide the best way of displaying the menus so that they are comprehensible and natural to use. In the majority of the case, it is useful to organize the command in a hierarchical way. The problem is deciding which items to include at various levels and which items to group together at different level.

Empirical evidence suggests that there are four alternatives for ordering menu items.

They are :

- 1) Alphabetical
- 2) Categorical
- 3) Conventional
- 4) Frequency forms of organisation

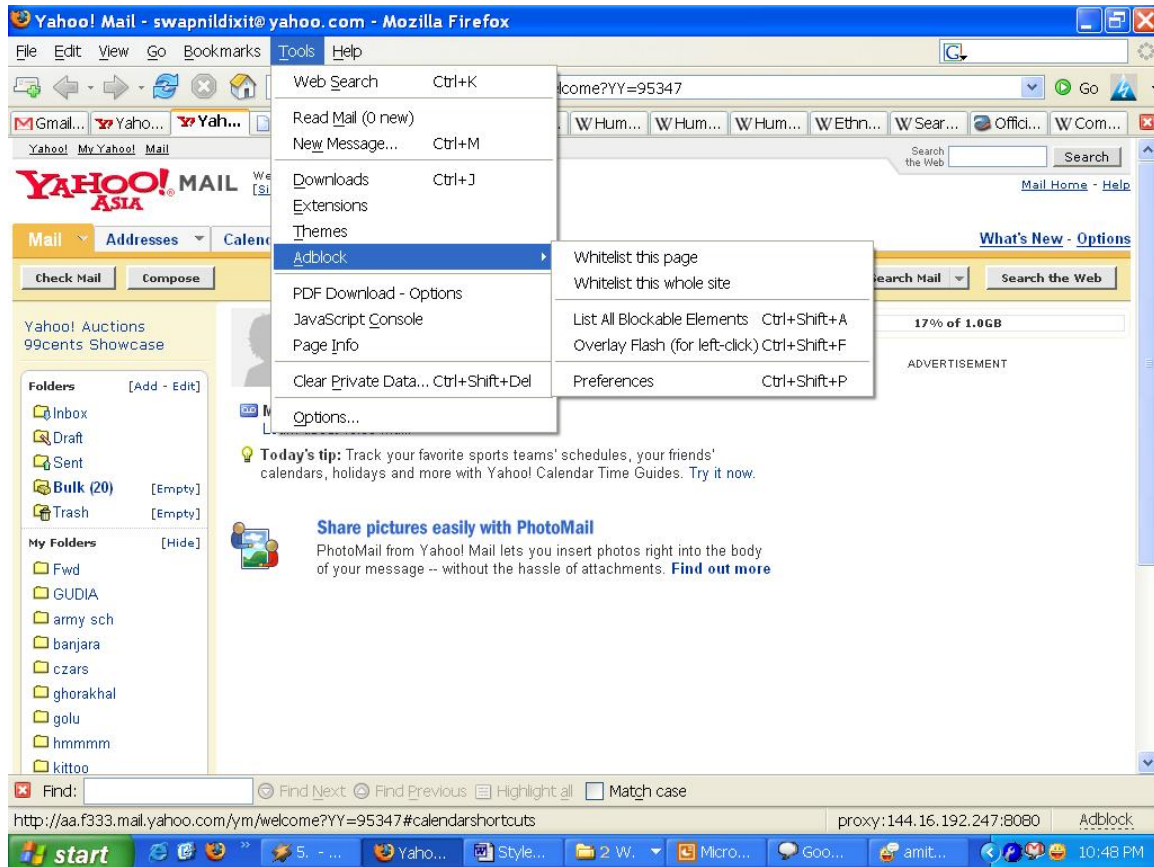


Fig. 3.2 A screenshot of a menu based GUI

Alphabetical is self explanatory.

Categorical organisation relies on the selection of the suitable categories which can be quite difficult.

Conventional menus are neither alphabetical not categorical but can have a temporal order such as the days of the week, or the months of the year. They do not occur very often.

Frequency of use is another way of organizing a menu if the number of options is small.

A more recent innovation is the pie menu. In these menus, the list of choices is presented as a **pie graph**. Compared with linear menus, it has been found that error rates and search items can be lower when searching a pie menu.

However, there are some **basic problems**,

if the menus are arranged in alphabetical forms. It defeats the very purpose of the menu interaction. In order to find the required command to click, a person will have to remember the name of the command in order to find which alphabet to look for in.

If the menus are arranged in categorical way, there can be many commands which lie in more than one category. Including all the commands in all the concerned menus can cause unnecessary computational complexities. For example, in autocad, if you have to adjust the line width, there can be a menu which lets you do it, It however, can belong to the drafting on screen section as well the printing preference section. Moreover, there can be other requirement of deciding when do you want to display the linewidth. Do you want it during drafting or only during printing.

If the menus are arranged in a conventional way, there is a problem of choosing where to put the new innovative command developed. The file edit view menu of Microsoft products always will have cut and paste in edit menu. But what about tarnish the table into two. Will that be in view, options or edit is a problem.

If the menus are arranged in the frequency form, it makes an interesting case. The more frequently used commands will come into a menu that is easy to find. Now, since the user are using these commands very frequently, they know where to find these commands. A good interaction style is where one does not find it difficult to do tasks that he normally does not do and has to find the menu where it is listed.

3.2.3 Menu based interaction style and navigation

a **menu** is a list of commands presented to an operator by a computer or communications system. They may be thought of as *shortcuts* to frequently used commands that avoid the operator having to have a detailed knowledge or recall of syntax. A menu is used in contrast to a command line interface where instructions to the computer are given in the form of commands (or verbs).

Choices given from a menu may be selected by the operator by a number of methods (called interfaces):

- using an electromechanical pointer, such as a light pen
- touching the display screen with a finger
- speaking to a voice-recognition system
- positioning a cursor or reverse-video bar by using a keyboard or mouse
- Depressing one or more keys on the keyboard or mouse.



Fig. 3.3 Desktop metaphor



In the Windows XP graphical user interface, Music tasks and File and Folder tasks listed on the left side are a kind of *menu*

A computer using a graphical user interface presents menus (pronounced mee-nis, rhymes with Venus) with a combination of text and symbols to represent choices. By *clicking* on one of the symbols, the operator is selecting the instruction that the symbol represents. A *context menu* is a menu in which the choices presented to the operator are automatically modified according to the current context in which the operator is working.

A common use of menus is to provide convenient access to various operations such as saving or opening a file, quitting a program, or manipulating data. Most widget toolkits provide some form of pull-down or pop-up menu. Pull-down menus are the type commonly used in menu bars (usually near the top of a window or screen), which are most often used for performing actions, whereas pop-up menus are more likely to be used for setting a value, and might appear anywhere in a window.

According to traditional human interface guidelines, menu names were always supposed to be *verbs*, such as "file" "edit" and so on. This has been largely ignored in subsequent UI developments. A single word verb however is sometimes unclear, and so as to allow for multiple word menu names, the idea of a vertical menu was invented)

Menu selection is a preferable interaction style for many functions.

Advantages:

- Easy to use and learn as it offers memory cues (recognition not recall, but have to see it to recognize it).
- Reduces keystrokes
- Assists users to structure decision making

- Easy to support error handling

Disadvantages:

1. May require very large numbers of menus to cover all choices – hard to find a specific target
2. Expert or frequent users may be slowed down by the need to navigate through many levels.
3. May require extensive amount of screen space

Menu types: There following types of menu which are shown in Fig-9 below.

1. Single menus
2. Linear sequence
3. Tree structure
4. Acyclic networks and cyclic networks

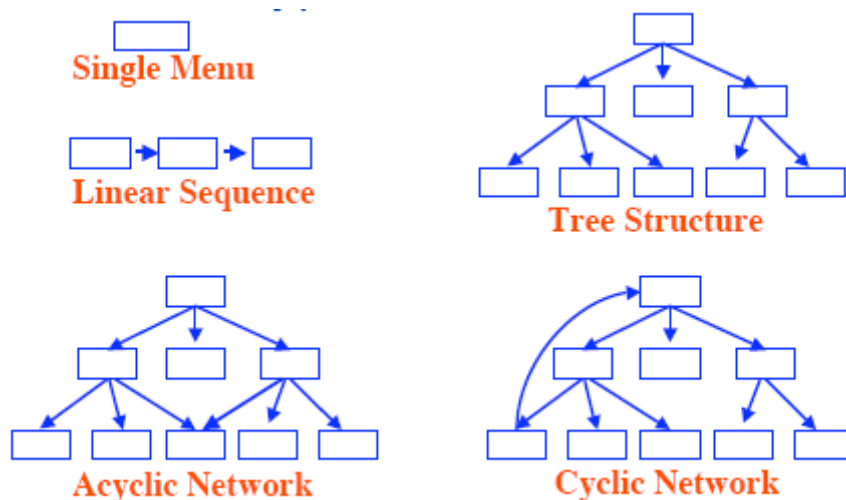


Fig. 3.4 Menu Types

Single menus

- Binary menu (one menu - two items - yes-no, true-false);
- Multiple item menu (egg quiz question];
- Multiple-selection menus or checkboxes (a-b-c-d with many simultaneous choices);
- Pull-down and pop-up menus [from a menu bar or on the desktop];

Ex: Primary window menu which is shown in Fig. 3.5(a).

Actions on the object itself (eg the document) - always there Eg: Workspace menu which is shown in Fig. 3.5(b). Management of possible child windows– always there.

Eg. Active object menu shown in Fig. 3.5(c).



Primary window menu

(a) Primary window menu



Workspace menu

(b) Workspace menu



Active objects menu

(c) Active object menu



(c) Iconic menu

Fig. 3.5 Different types of menu

Actions you can apply to the activated object (e.g. current text window) should go on these menus. Often associated with the left-click menu.

- Scrolling and two dimensional menus - fast and vast- needed if there is a will of having only one conceptual category - can also be a multiple column display instead of a scrolling list;
- Alphasliders if number of items in the menu is too big, then a slider can be used to alphabetically locate an item. ALSO consider for example the index in the tabbed panel usually found in the help context of most windows applications;
- embedded links: all others previously mentioned being explicit menus, embedded links are used in the well-know hypertext design - HTML for example - where items in a context other than an explicit menu are selectable (here one can use the example of SEA? for browsing through a telecom system;
- iconic menus shown in Fig. 3.5(c), which contains only icons, toolbars or palettes.

Menus offering many actions that a user can select with a click and apply to a displayed object. User should be able to control the positioning of the toolbar and even get rid of it.

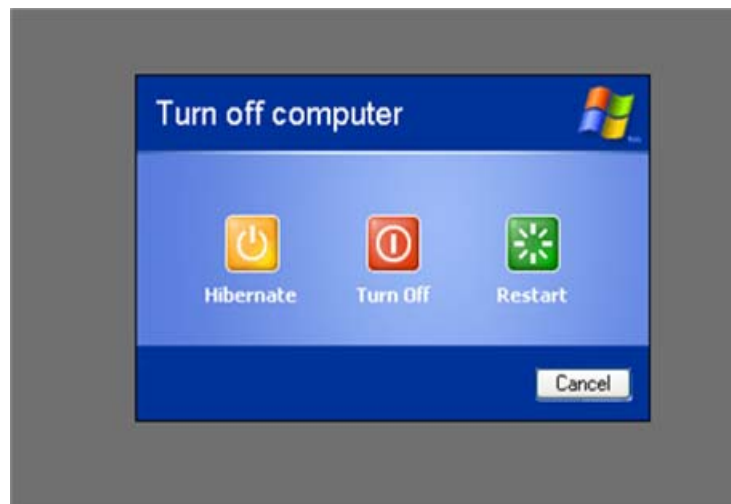


Fig. 3.6 - Single menu

Pie menus - A circular pop-up menu:

- No bounds on selection area
- Basically only angle counts
- A “dead area” at center

An example of Pie Menu is “The Sims” as seen in the game from Maxis which is shown in the Fig. 3.7 below.

Menu navigation vs. CLI

Menu navigation was introduced in reaction to the steep learning curve of *Command Line Interfaces* (CLI), text-based user interfaces requiring commands to be typed on the keyboard. Since the command words in CLIs are usually numerous and composable, very complicated operations can be invoked using a relatively short sequence of words and symbols. This leads to high levels of efficiency once the many commands are learned, but reaching this level can take some time, because the command words are not easily discoverable. WIMPs (“window, icon, menu, pointing device”), on the other hand, present the user with numerous widgets that represent and can trigger some of the system's available commands.

WIMPs extensively use modes as the meaning of all keys and clicks on specific positions on the screen are redefined all the time. CLIs use modes only in the form of a current directory.

Most modern operating systems provide both a GUI and some level of a CLI, although the GUIs usually receive more attention. The GUI is usually WIMP-based, although occasionally other metaphors surface, such as Microsoft Bob, 3dwm or (partially) FSV.

Applications may also provide both interfaces, and when they do the GUI is usually a WIMP wrapper around the CLI version. The latter used to be implemented first because it allowed the developers to focus exclusively on their product's functionality without bothering about interface details such as designing icons and placing buttons. Nowadays, the GUI is no longer an optional part of a successful application, because users have grown accustomed to the ease of use provided by their familiar GUIs.

An example of Pie Menus is “The Sims” as seen in the game from Maxis which is shown in the Fig. 3.7(a).



(a) Pie Menus as seen in the game “The Sims” from Maxis

Pop-up Linear Menu

Today
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday

(b) Pop-up Linear menu

Pop-up Pie Menu



(c) Pop-up Pie menu

Fig. 3.7 Different pie menus

Which will be faster on average?

Pie menu (bigger targets & less distance)

Pie menus - Why don't we see Pie menus much?

- Possibly just lack of awareness
- However, harder to implement - particularly drawing labels.
- They don't scale past a few items - no hierarchy.

Tree-structured menus

- Large numbers of items creates the need for task related logical groupings.
- Hierarchical decomposition familiar and comprehensible - however, some items can be hard to classify.
- Try to avoid overlapping items.
- Always use same titles throughout levels.

This is the view of the Tree menu:

Facilitate the decision-making process

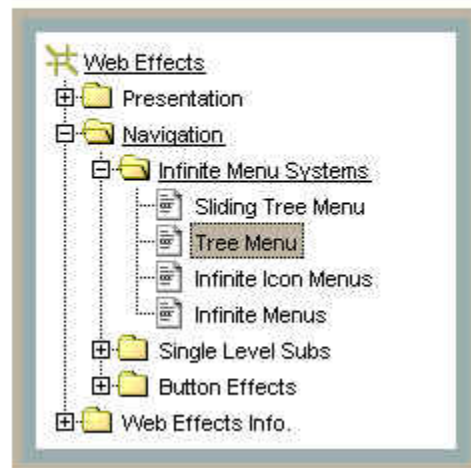
Group the menu items by task relationship

- Create groups of logically related items;
- Form groups that covers all possibilities;
- Make sure that items are no overlapping;
- Use familiar terminology but make sure that items are distinct from each other.p250

Consistency across menus is the most important

For example it is better to gray out an item that is not available than to remove it completely

Open Cube Tree Menu



This tree menu is used for web site navigation

Considerations:

- Depth (no of levels) vs. breadth (number of items per level)
- Many studies indicate more breadth better than more depth – I.e. keep menus shallow.
- May be trade off with cluttered screen versus shallow depth....

Other types of menus

- **Linear sequence** (eg like an online exam that has multiple choice questions items or the printer menu) One decision at the time;
- **Acyclic and Cyclic Networks** - menu items reachable by multiple paths, Natural & useful for some applications e.g. transportation routing, scientific-journal citations but can cause confusion and disorientation. (a way to see it is like browsing through directory that have soft links...) Why would we want this?
 - may be useful to allow users to access the same item from 2 different paths;
 - switch between sections without restarting the search;
 - the www is a large cyclic network;
 - (-) the mental model for this is more difficult to assimilate for users;
- **Menu Maps** - can be effective for large menu sets in providing an overview to minimize user disorientation. Overview of Menu Maps is shown in Fig-18 below.

Menu map example

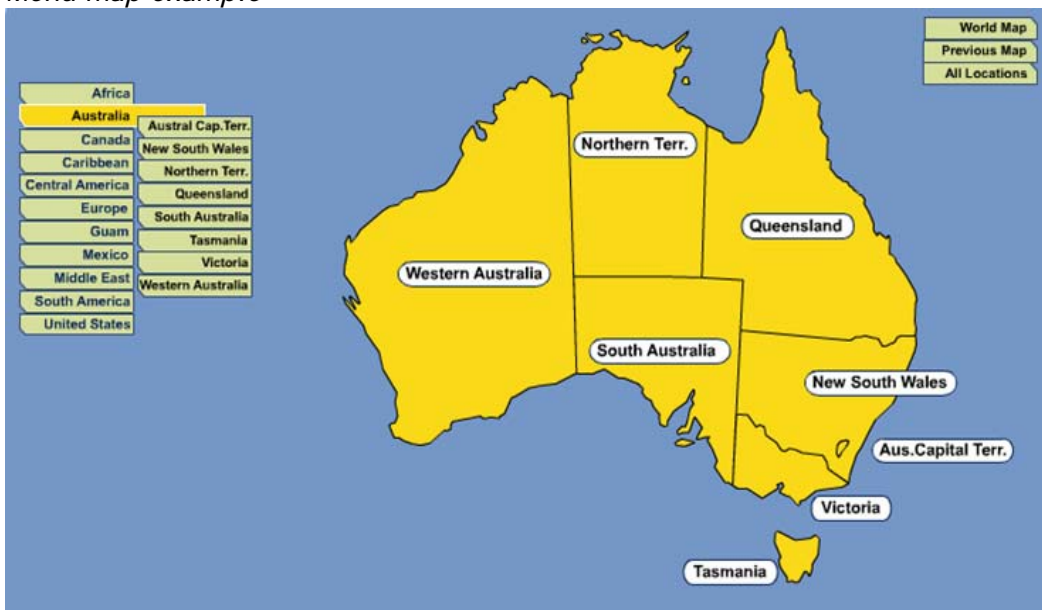


Fig. 3.8 Example of a menu Map

Design issues to consider

- Use titles which clearly identify action:
- Use familiar terminology
- Ensure items are distinct from one another
- Be concise and consistent
- Place keyword to left
- Fonts and layout - size, spacing, consistency
- Scrolling vs. sections vs. multicolumn

3.2.4 Form fill and spreadsheet

When several different categories are fed into a system, using a keyboard, It is often helpful to design the screen as though it is a form (form fill) , particularly if the same type of data has to entered repeatedly as occurs in retailing. (type, number, price, stock , delivery) and personal records (date of birth, name, roll no, marks, whether married or not, employed etc)

The advantage of having fill in forms is that they can help users to position data in correct place, thereby reducing the need to watch the screen too carefully and interpreting the various icons and trying to find out in which menu could the needed command be possibly located.

As one part of the form is completed, the cursor usually moves directly to the position where the next item of data should be entered. This means that users do not have to bother to position the cursor themselves, They only have to press <return> key to make a mouse click. (For Ex . the various login screens,)

As with any form, it is possible to make life difficult for users by designing it poorly. Forms need to be designed so that they enable users to know which kinds of data are permissible in each field. It should also be obvious to users how to make corrections. One way of making forms easy to use is to design them so that they are similar to well designed paper forms in the way they look and filled in.

Spreadsheet program have also been designed using the principal of electronically mimicking a familiar paper predecessor. In the case of spreadsheet, however, the electronic version provides much more functionality. (and are now typically combined with elements of direct manipulation style) For example, they often enable various calculations, such as sums, percentages, ratios, to be performed automatically on data that have been entered on the screen. The advantage of this is that users can try out alternative plans and see the results instantaneously.

EDUCATION

Graduation

University/Institute *

Degree *

CGPA / Percentage *

Year *

Branch / Stream / Specialization

Post Graduation

University/Institute *

Degree *

CGPA / Percentage *

Year *

Branch / Stream / Specialization *

WORK EXPERIENCE

Years of Total Experience *

Current Employment Details

Current Employer

Current CTC (in Lacs)

Relevant Experience

Industry 1 Years

Industry 2 Years

MONASH University Textbook List

Lecturer Name

Department

Email Address

Phone

Subject Code

Subject Name

If this subject ran previously under another code, please enter this code below
Old Subject Code

Please complete a separate form for each subject for each semester

Course duration for Monash University
Please tick only one box
Semester One ☐ Semester Two ☐ Summer Semester ☐ Winter Semester ☐ Full Year ☐

Short Course Commencement date Duration

Estimated enrolments numbers for Monash University

Clayton	Caulfield	Gipsland	Peninsula	Berwick
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Fig. 3.5 Screen shot of a form fill in

Advantages:

- Resembles familiar paper form so transfer of model reduces training needs
- Error-handling and help can be context sensitive.
- Simplifies data entry
- Convenient assistance (recognition vs. recall)

Disadvantage:

- Can require large amounts of screen space
- Spelling errors

Design issues to consider

- Use meaningful title and field names.
- Logical grouping and sequencing of fields
- Error prevention, correction and messages
- Fields given explanations, optional fields indicated.
- Field space and boundaries indicated.
- Clear indication of form completion

3.2.5. Natural language based Interface

Commands given in “natural” sentences and phrases rather than set commands Range from simple voice recognition menu-driven systems to more complex ‘natural language’ dialogues.

Examples include search engines, advice-giving systems, help systems.

Most recently, virtual agents at the interface, who converse with you.

A good example which clarifies this concept is Microsoft’s Clippie shown in Fig-7 where the user get any type of help from system by just typing his/her doubt in the dialog box provided.

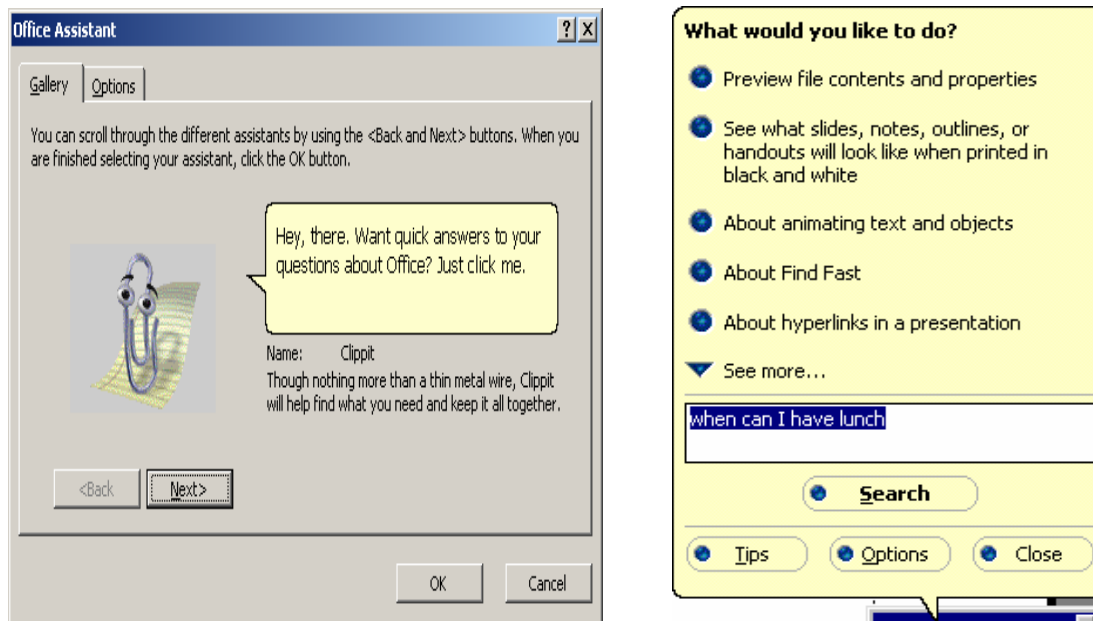




Fig. 3.6 Microsoft's Clippie

You asked: How many legs does a centipede have?

Jeeves knows these answers:


Where can I find a definition for the math term
leg? 

Where can I find a concise encyclopedia article on ?
centipedes? 

Where can I see an image of the human
appendix? 

Why does my leg or other limb fall asleep?



Where can I find advice on controlling the garden pest ?
millipedes and centipedes? 

Where can I find resources from Britannica.com on
leg ? 

Fig. 3.7 Natural language search engine

Fig. 3.7 explains the conversation between users and system. Suppose user asked the question "How many legs does a centipede have?" . Then above will be the response of the system.

3.2.5 Direct Manipulation

Direct manipulation (DM) interfaces allow users to interact directly with the interface objects. Close mapping of task domain and interface domain helps user to focus on tasks rather than interface. An example of DM is Ms Paint which interface is shown in Fig-20.

- Object on the screen and the actions they allow represent real-world objects and actions.
- User actions involve dragging, selecting, opening, closing and zooming actions on virtual objects
- Used for a wide range of applications: Examples, Desktop O/S interface, games, CAD/CAM.
- Also extends to areas such as virtual reality, information visualization.

Characteristics:

- Visible and continuous representation of the task objects and their actions
- Task objects manipulated by physical actions
- Users have focus on the task rather than the technology.
- Extensive use of metaphors and icons

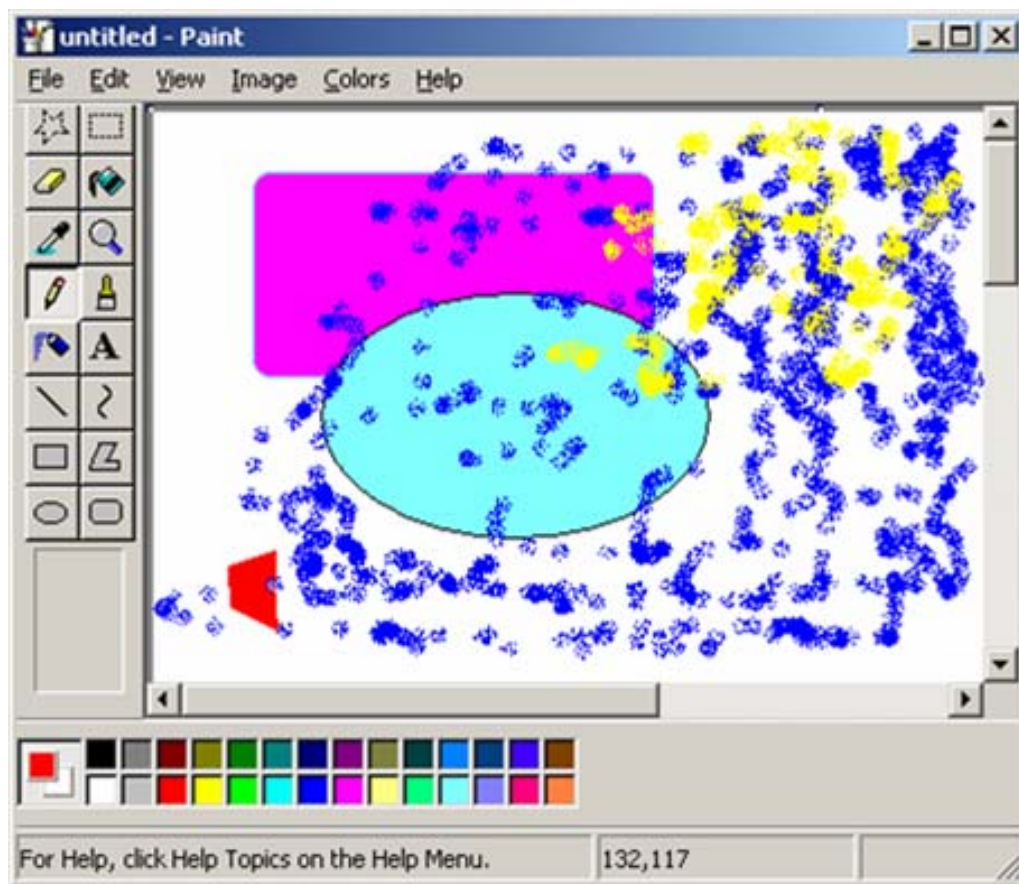


Fig. 3.8 Ms Paint an example of Direct manipulation

Advantages:

- Intuitive, easy to learn and remember
- Reduces errors as minimal syntax required.
- Allows rapid actions, and reversals
- Enjoyable and encourages exploration by immediate feedback and evaluation.

- Users experience less anxiety, sense of confidence and control.
- More difficult to program (especially error handling)
- High resource usage – e.g. memory and CPU
- Requirement for lots of screen space may be cumbersome. e.g. need to scroll.
- Pointing may be slower than typing.

Disadvantages:

- Some data more difficult to manipulate – e.g. lists of file icons versus names.
- Visual representation may mislead:
- Not all objects and tasks can be described visually; and not all actions can be done by direct object manipulation.
- May increase difficulties for visually impaired.

Cognitive styles and DM

Some evidence that "right brain", intuitive personalities prefer visual direct manipulation environment. Whereas "left brain" logical, linear thinkers may prefer command line. However, physical spatial & visual representations generally result in faster performance and less errors.

Design issues to consider

- Choose metaphors carefully.
- Create visual representations of the users' tasks.
- Provide direct, rapid, incremental and reversal actions on objects.
- Provide immediate feedback on actions.
- Indicate which objects may be manipulated at any time.
- Indicate the state of an object.

Application of Direct Manipulation

1. Browsing, for example: Cds, web interface
2. Virtual and augmented reality
3. Haptic and Gesture
4. Direct thought control.

Virtual and augmented reality

Virtual reality - users placed in an immersive environment where their normal surrounding is blocked out.

Augmented reality – users in their normal surroundings with a transparent overlay

Interactions are sensory: visual, aural or haptic.

Natural and realistic interactions

Choosing an interaction style

- Need to determine requirements and user needs.
- Level of expertise of the users
- Take the budget and other constraints into account.
- Also will depend on suitability of technology for activity being supported

Combining interaction styles

The interaction styles may be mixed, with several styles used in the same application. An example of multiple interaction style is shown in Fig-21 below.

Microsoft Windows supports

- _ direct manipulation of the iconic representation of files and directories
- _ menu-based command selection, and
- _ fill-in forms, for commands such as configuration ones

Linux-like systems support

- _ a command language interface (ksh, csh, bsh, zsh), and
- _ a graphical interface (Gnome, KDE)

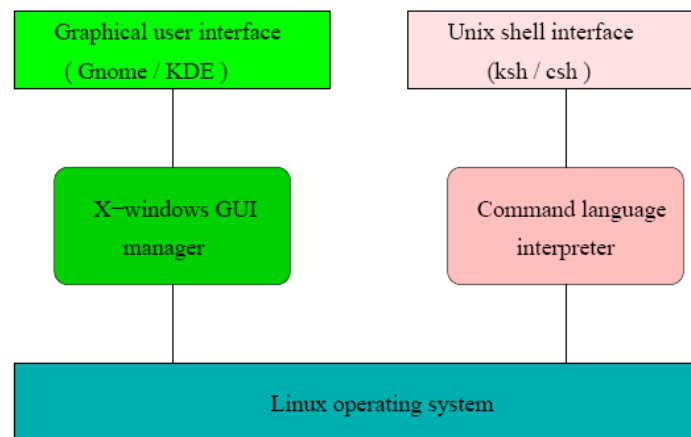


Fig. 3.9 Multiple Interaction Style

Relationship of interaction style to usability

- Who has control?
- Ease of use for novices.
- Learning time to become proficient
- Speed of use (efficiency) once the user becomes proficient.
- Consistency – predictable results of actions.
- Ability to show defaults, current values, etc.
- Skills required (e.g., typing)

Comparisons between different Interactions style has been shown in Table-3 below:

	Advantages	Disadvantages
Direct Manipulation	<ul style="list-style-type: none"> -visually presents task concepts -allows easy learning -allows easy retention -allows errors to be avoided -encourages exploration -affords high subjective satisfaction 	<ul style="list-style-type: none"> -may be hard to program -may require graphics display and pointing device
Menu Selection	<ul style="list-style-type: none"> - shortens learning - reduces keystrokes - structures decision making - permits use of dialogue management tools - allows easy support of error handling 	<ul style="list-style-type: none"> presents danger of many menus may slow frequent users consumes screen space requires rapid display rate
Form Fill-in	<ul style="list-style-type: none"> - simplifies data entry - requires modest training - gives convenient assistance - permits use of form management tools 	<ul style="list-style-type: none"> - consumes screen space
Command Language	<ul style="list-style-type: none"> - is flexible - appeals to 'power' users - supports user initiative - allows convenient creation of user-defined macros 	<ul style="list-style-type: none"> - has poor error handling - requires substantial training and memorization
Natural Language	<ul style="list-style-type: none"> - relieves burden of learning syntax 	<ul style="list-style-type: none"> -requires clarification dialogue - may require more keystrokes - may not show context - is unpredictable

Table 3.1 Comparison between Interaction styles

References

Introduction & UID Strategies

- <http://www.usernomics.com/user-interface-design.html>
- Lecture notes of Prof. D.Samanta

Conceptual, semantic, syntactic, and lexical model

- <http://www.cs.umd.edu/class/fall2002/cmsc838s/tichi/CSSL.html>
- Foley, J.D., van Dam, A., Feiner, S.K., and. Hughes, J.F. (1990) Computer Graphics: Principles and Practice, Reading, MA: Addison- Wesley.
- Schneiderman, Ben, Designing the User Interface. Reading, MA: Addison-Wesley: 1998.
- R. Jacob, "Using Formal Specifications in the Design of a Human-Computer Interface," Communications of the ACM 26(4) pp. 259-264 (1983).

GOMS

- <http://www.cs.umd.edu/class/fall2002/cmsc838s/tichi/printer/goms.html>
- Card, Stuart, Morn, Thomas P., and Newell, Allen, The keystroke-level model for user performance with interactive systems, Communications of the ACM, 23 (1980), 396-210 (This is the paper that presents KLM, the predecessor of GOMS)
- Card, Stuart, Moran, Thomas P., and Newell, Allen, The Psychology of Human-Computer Interaction, Lawrence Erlbaum Associates, Hillsdale, NJ (1983). (This book contains the original presentation of the GOMS model)
- John, Bonnie and Kieras, David E., Using GOMS for user interface design and evaluation: Which technique? ACM Transactions on Computer-Human Interaction 3,4 (December 1996a), 287-319. (This paper explains which GOMS variant to use depending on the application)
- John, Bonnie and Kieras, David E., The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast, ACM Transactions on Computer-Human Interaction 3,4 (December 1996b), 320-351. (In this companion paper to above, the different variants of GOMS are compared)
- Kieras, David, Towards a practical GOMS model methodology for user interface design. In Helander, Martin (Editor), Handbook of Human-Computer Interaction, Elsevier Science Publishers, Amsterdam, The Netherlands (1988),135-137. (This paper presents the NGOMSL variant)
- Gray, W. D., John, B. E., & Atwood, M. E. Project Ernestine: A validation of GOMS for prediction and explanation of real-world task performance. Human-Computer Interaction (1993), 8, 3, 237-309.
John, B. E. (1990) Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. In proceedings of CHI, 1990 (Seattle, Washington, April 30-May 4, 1990) ACM, New York, 107-115. (These two papers present the CPM-GOMS variant)

Stages of action models

- <http://www.it.bton.ac.uk/staff/rng/teaching/notes/NormanGulfs.html>
- Norman, D.A. 1988 "The Design of Everyday Things." MIT Press

Consistency through grammars

- <http://web.cs.wpi.edu/~lab/cs546/Ch1and2.html>
- Norman, D.A. 1988 "The Design of Everyday Things." MIT Press

Widget-level theories

- <http://myweb.lmu.edu/dondi/summer2005-1/cmsi698/theories.pdf>
- <http://web.cs.wpi.edu/~lab/cs546/Ch1and2.html>

- Norman, D.A. 1988 "The Design of Everyday Things." MIT Press

Object/Action Interface Model

- <http://www.cs.umd.edu/class/fall2002/cmsc838s/tichi/printer/oai.html>
- Human-computer interaction in a computer supported collaborative writing environment, <http://infolab.kub.nl/pub/theses/w3thesis/toc.html>
- Ben Shneiderman, Designing the user interface
- Amir Khella, Knowledge and mental models

Principle 1: Recognize the Diversity

Usage profiles & Task profiles

- Designing the User Interface 3rd Ed. by Ben Schneiderman,
- The Psychology of Human-Computer Interaction by Stuart Card and friends,
- Human-Computer Interaction 2nd Ed by Alan Dix and friends

Interaction styles

- <http://www.evl.uic.edu/aej/578/week3.html>
- <http://www.cdt.luth.se/~pelle/smd045/lectures/lecture5/syllabus.shtml>
- <http://w5.cs.uni-sb.de/~butz/teaching/uid-ws03/pdf/uid061103.pdf>
- Designing the User Interface 3rd Ed. by Ben Schneiderman,

Principle 2: Use the Eight Golden Rules of Interface Design

- Designing the User Interface 3rd Ed. by Ben Schneiderman,

Principle 3: Prevent Errors

- www.csc.liv.ac.uk/~weberb/Teaching/106/lec07.pdf
- www.medien.ifi.lmu.de/fileadmin/mimuc/mmi_ws0304/slides/2003-12-03_001.pdf
- http://www.developer.com/design/article.php/10925_1564681_1

Guidelines for Data Entry

- http://www.geekgirls.com/databasics_03.htm

Balance of Automation and Human Control

- http://users.cs.dal.ca/~jamie/course/CS/3160/Lecture/2-Is/stds+guides,v2_1_BW.ppt
- Shneiderman & Plaisant's Designing the User Interface (4e)
- Roger J. Chapman's slides for Shneiderman & Plaisant's Designing the User Interface (4e)

Summary

- Fundamental of Software Engineering- Prof. Rajib Mall