

A Summary of Principles for User-Interface Design.

by Talin

This document represents a compilation of fundamental principles for designing user interfaces, which have been drawn from various books on interface design, as well as my own experience. Most of these principles can be applied to either command-line or graphical environments. I welcome suggestions for changes and additions -- I would like this to be viewed as an "open source" evolving document.

1. The principle of user profiling

-- Know who your user is.

Before we can answer the question "How do we make our user interfaces better", we must first answer the question: Better for *whom*? A design that is better for a technically skilled user might not be better for a non-technical businessman or an artist.

One way around this problem is to create user models. [TOG91] has an excellent chapter on brainstorming towards creating "profiles" of possible users. The result of this process is a detailed description of one or more "average" users, with specific details such as:

- What are the user's goals?
- What are the user's skills and experience?
- What are the user's needs?

Armed with this information, we can then proceed to answer the question: How do we leverage the user's strengths and create an interface that helps them achieve their goals?

In the case of a large general-purpose piece of software such as an operating system, there may be many different kinds of potential users. In this case it may be more useful to come up with a list of *user dichotomies*, such as "skilled vs. unskilled", "young vs. old", etc., or some other means of specifying a continuum or collection of user types.

Another way of answering this question is to talk to some real users. Direct contact between end-users and developers has often radically transformed the development process.

2. The principle of metaphor

-- Borrow behaviors from systems familiar to your users.

Frequently a complex software system can be understood more easily if the user interface is depicted in a way that resembles some commonplace system. The ubiquitous "Desktop metaphor" is an overused and trite example. Another is the *tape deck* metaphor seen on many

audio and video player programs. In addition to the standard transport controls (play, rewind, etc.), the tape deck metaphor can be extended in ways that are quite natural, with functions such as *time-counters* and *cueing buttons*. This concept of "extendibility" is what distinguishes a powerful metaphor from a weak one.

There are several factors to consider when using a metaphor:

- Once a metaphor is chosen, it should be spread widely throughout the interface, rather than used once at a specific point. Even better would be to use the same metaphor spread over several applications (the tape transport controls described above is a good example.) Don't bother thinking up a metaphor which is only going to apply to a single button.
- There's no reason why an application cannot incorporate several different metaphors, as long as they don't clash. Music sequencers, for example, often incorporate both "tape transport" and "sheet music" metaphors.
- Metaphor isn't always necessary. In many cases the natural function of the software itself is easier to comprehend than any real-world analog of it. Don't strain a metaphor in adapting it to the program's real function. Nor should you strain the meaning of a particular program feature in order to adapt it to a metaphor.
- Incorporating a metaphor is not without certain risks. In particular, whenever physical objects are represented in a computer system, we inherit not only the beneficial functions of those objects but also the detrimental aspects.
- Be aware that some metaphors don't cross cultural boundaries well. For example, Americans would instantly recognize the common U.S. Mailbox (with a rounded top, a flat bottom, and a little red flag on the side), but there are no mailboxes of this style in Europe.

3. The principle of feature exposure

-- Let the user see clearly what functions are available

Software developers tend to have little difficulty keeping large, complex mental models in their heads. But not everyone prefers to "live in their heads" -- instead, they prefer to concentrate on analyzing the sensory details of the environment, rather than spending large amounts of time refining and perfecting abstract models. Both type of personality (labeled "Intuitive" and "Sensible" in the Myers-Briggs personality classification) can be equally intelligent, but focus on different aspects of life. It is to be noted that according to some psychological studies "Sensibles" outnumber "Intuitives" in the general population by about three to one.

Intuitives prefer user interfaces that utilize the power of abstract models -- command lines, scripts, plug-ins, macros, etc. Sensibles prefer user interfaces that utilize their perceptual abilities -- in other words, they like interfaces where the features are "up front" and "in their face". Toolbars and dialog boxes are an example of interfaces that are pleasing to this personality type.

This doesn't mean that you have to make everything a GUI. What it does mean, for both GUI and command line programs, is that the features of the program need to be *easily exposed* so that a quick visual scan can determine what the program actually does. In some cases, such as a toolbar, the program features are exposed by default. In other cases, such as a printer configuration dialog, the exposure of the underlying printer state (i.e. the buttons and controls

which depict the *conceptual printing model*) are contained in a dialog box which is brought up by a user action (a feature which is itself exposed in a menu).

Of course, there may be cases where you don't wish to expose a feature right away, because you don't want to overwhelm the beginning user with too much detail. In this case, it is best to structure the application like the layers of an onion, where peeling away each layer of skin reveals a layer beneath. There are various levels of "hiding": Here's a partial list of them in order from most exposed to least exposed:

- Toolbar (completely exposed)
- Menu item (exposed by trivial user gesture)
- Submenu item (exposed by somewhat more involved user gesture)
- Dialog box (exposed by explicit user command)
- Secondary dialog box (invoked by button in first dialog box)
- "Advanced user mode" controls -- exposed when user selects "advanced" option
- Scripted functions

The above notwithstanding, in no case should the primary interface of the application be a reflection of the *true complexity* of the underlying implementation. Instead, both the interface and the implementation should strive to match a simplified conceptual model (in other words, the *design*) of what the application does. For example, when an error occurs, the explanation of the error should be phrased in a way that relates to the current user-centered activity, and not in terms of the low-level fault that caused there error.

4. The principle of coherence

-- The behavior of the program should be internally and externally consistent

There's been some argument over whether interfaces should strive to be "intuitive", or whether an intuitive interface is even possible. However, it is certainly arguable that an interface should be *coherent* -- in other words logical, consistent, and easily followed. ("Coherent" literally means "stick together", and that's exactly what the parts of an interface design should do.)

Internal consistency means that the program's behaviors make "sense" with respect to other parts of the program. For example, if one attribute of an object (e.g. color) is modifiable using a pop-up menu, then it is to be expected that other attributes of the object would also be editable in a similar fashion. One should strive towards the principle of "least surprise".

External consistency means that the program is consistent with the environment in which it runs. This includes consistency with both the operating system and the typical suite of applications that run within that operating system. One of the most widely recognized forms of external coherence is compliance with *user-interface standards*. There are many others, however, such as the use of standardized scripting languages, plug-in architectures or configuration methods.

5. The principle of state visualization

-- Changes in behavior should be reflected in the appearance of the program

Each change in the behavior of the program should be accompanied by a corresponding change in the appearance of the interface. One of the big criticisms of "modes" in interfaces is that many of the classic "bad example" programs have modes that are visually indistinguishable from one another.

Similarly, when a program changes its appearance, it should be in response to a behavior change; A program that changes its appearance for no apparent reason will quickly teach the user not to depend on appearances for clues as to the program's state.

One of the most important kinds of state is the *current selection*, in other words the object or set of objects that will be affected by the next command. It is important that this internal state be visualized in a way that is consistent, clear, and unambiguous. For example, one common mistake seen in a number of multi-document applications is to forget to "dim" the selection when the window goes out of focus. The result of this is that a user, looking at several windows at once, each with a similar-looking selection, may be confused as to exactly which selection will be affected when they hit the "delete" key. This is especially true if the user has been focusing on the selection highlight, and not on the window frame, and consequently has failed to notice which window is the active one. (Selection rules are one of those areas that are covered poorly by most UI style guidelines, which tend to concentrate on "widgets", although the Mac and Amiga guidelines each have a chapter on this topic.)

6. The principle of shortcuts

-- Provide both concrete and abstract ways of getting a task done

Once a user has become experienced with an application, she will start to build a mental model of that application. She will be able to predict with high accuracy what the results of any particular user gesture will be in any given context. At this point, the program's attempts to make things "easy" by breaking up complex actions into simple steps may seem cumbersome. Additionally, as this mental model grows, there will be less and less need to look at the "in your face" exposure of the application's feature set. Instead, pre-memorized "shortcuts" should be available to allow rapid access to more powerful functions.

There are various levels of shortcuts, each one more abstract than its predecessor. For example, in the emacs editor commands can be invoked directly by name, by menu bar, by a modified keystroke combination, or by a single keystroke. Each of these is more "accelerated" than its predecessor.

There can also be alternate methods of invoking commands that are designed to increase power rather than to accelerate speed. A "recordable macro" facility is one of these, as is a regular-expression search and replace. The important thing about these more powerful (and more abstract) methods is that they should not be the most exposed methods of accomplishing the task.

This is why emacs has the non-regexp version of search assigned to the easy-to-remember "C-s" key.

7. The principle of focus

-- Some aspects of the UI attract attention more than others do

The human eye is a highly non-linear device. For example, it possesses edge-detection hardware, which is why we see Mach bands whenever two closely matched areas of color come into contact. It also has motion-detection hardware. As a consequence, our eyes are drawn to animated areas of the display more readily than static areas. Changes to these areas will be noticed readily.

The mouse cursor is probably the most intensely observed object on the screen -- it's not only a moving object, but mouse users quickly acquire the habit of tracking it with their eyes in order to navigate. This is why global state changes are often signaled by changes to the appearance of the cursor, such as the well-known "hourglass cursor". It's nearly impossible to miss.

The text cursor is another example of a highly eye-attractive object. Changing its appearance can signal a number of different and useful state changes.

8. The principle of grammar

-- A user interface is a kind of language -- know what the rules are

Many of the operations within a user interface require both a *subject* (an object to be operated upon), and a *verb* (an operation to perform on the object). This naturally suggests that actions in the user interface form a kind of grammar. The grammatical metaphor can be extended quite a bit, and there are elements of some programs that can be clearly identified as adverbs, adjectives and such.

The two most common grammars are known as "Action->Object" and "Object->Action". In Action->Object, the operation (or tool) is selected first. When a subsequent object is chosen, the tool immediately operates upon the object. The selection of the tool persists from one operation to the next, so that many objects can be operated on one by one without having to re-select the tool. Action->Object is also known as "modality", because the tool selection is a "mode" which changes the operation of the program. An example of this style is a paint program -- a tool such as a paintbrush or eraser is selected, which can then make many brush strokes before a new tool is selected.

In the Object->Action case, the object is selected first and persists from one operation to the next. Individual actions are then chosen which operate on the currently selected object or objects. This is the method seen in most word processors -- first a range of text is selected, and then a text style such as bold, italic, or a font change can be selected. Object->Action has been called "non-modal" because all behaviors that can be applied to the object are always available. One

powerful type of Object->Action is called "direct manipulation", where the object itself is a kind of tool -- an example is dragging the object to a new position or resizing it.

Modality has been much criticized in user-interface literature because early programs were highly modal and had hideous interfaces. However, while non-modality is the clear winner in many situations, there are a large number of situations in life that are clearly modal. For example, in carpentry, it's generally more efficient to hammer in a whole bunch of nails at once than to hammer in one nail, put down the hammer, pick up the measuring tape, mark the position of the next nail, pick up the drill, etc.

9. The principle of help

-- Understand the different kinds of help a user needs

In an essay in [LAUR91] it states that there are five basic types of help, corresponding to the five basic questions that users ask:

- **1. Goal-oriented:** "What kinds of things can I do with this program?"
- **2. Descriptive:** "What is this? What does this do?"
- **3. Procedural:** "How do I do this?"
- **4. Interpretive:** "Why did this happen?"
- **5. Navigational:** "Where am I?"

The essay goes on to describe in detail the different strategies for answering these questions, and shows how each of these questions requires a different sort of help interface in order for the user to be able to adequately phrase the question to the application.

For example, "about boxes" are one way of addressing the needs of question of type 1. Questions of type 2 can be answered with a standard "help browser", "tool tips" or other kinds of context-sensitive help. A help browser can also be useful in responding to questions of the third type, but these can sometimes be more efficiently addressed using "cue cards", interactive "guides", or "wizards" which guide the user through the process step-by-step. The fourth type has not been well addressed in current applications, although well-written error messages can help. The fifth type can be answered by proper overall interface design, or by creating an application "roadmap". None of the solutions listed in this paragraph are final or ideal; they are simply the ones in common use by many applications today.

10. The principle of safety

-- Let the user develop confidence by providing a safety net

Ted Nelson once said "Using DOS is like juggling with straight razors. Using a Mac is like shaving with a bowling pin."

Each human mind has an "envelope of risk", that is to say a minimum and maximum range of risk-levels which they find comfortable. A person who finds herself in a situation that is too risky

for her comfort will generally take steps to reduce that risk. Conversely, when a person's life becomes too safe -- in other words, when the risk level drops below the *minimum* threshold of the risk envelope -- she will often engage in actions that *increase* their level of risk.

This comfort envelope varies for different people and in different situations. In the case of computer interfaces, a level of risk that is comfortable for a novice user might make a "power-user" feel uncomfortably swaddled in safety.

It's important for new users that they feel safe. They don't trust themselves or their skills to do the right thing. Many novice users think poorly not only of their technical skills, but of their intellectual capabilities in general (witness the popularity of the "...for Dummies" series of tutorial books.) In many cases these fears are groundless, but they need to be addressed. Novice users need to be assured that they will be protected from their own lack of skill. A program with no safety net will make this type of user feel uncomfortable or frustrated to the point that they may cease using the program. The "Are you sure?" dialog box and multi-level undo features are vital for this type of user.

At the same time, an expert user must be able to use the program as a *virtuoso*. She must not be hampered by guard rails or helmet laws. However, expert users are also smart enough to turn off the safety checks -- if the application allows it. This is why "safety level" is one of the more important application configuration options.

Finally, it should be noted that many things in life are not meant to be easy. Physical exercise is one -- "no pain, no gain". A concert performance in Carnegie Hall, a marathon, or the Guinness World Record would be far less impressive if anybody could do it. This is especially pertinent in the design of computer game interfaces, which operate under somewhat different principles than those listed here (although many of the principles in fact do apply).

11. The principle of context

-- Limit user activity to one well-defined context unless there's a good reason not to

Each user action takes place within a given context -- the current document, the current selection, the current dialog box. A set of operations that is valid in one context may not be valid in another. Even within a single document, there may be multiple levels -- for example, in a structured drawing application, selecting a text object (which can be moved or resized) is generally considered a different state from selecting an individual character within that text object.

It's usually a good idea to avoid mixing these levels. For example, imagine an application that allows users to select a range of text characters within a document, and also allows them to select one or more whole documents (the latter being a distinct concept from selecting all of the characters in a document). In such a case, it's probably best if the program disallows selecting both characters and documents in the same selection. One unobtrusive way to do this is to "dim" the selection that is not applicable in the current context. In the example above, if the user had a range of text selected, and then selected a document, the range of selected characters could

become dim, indicating that the selection was not currently pertinent. The exact solution chosen will of course depend on the nature of the application and the relationship between the contexts.

Another thing to keep in mind is the relationship between contexts. For example, it is often the case that the user is working in a particular task-space, when suddenly a dialog box will pop up asking the user for confirmation of an action. This sudden shift of context may leave the user wondering how the new context relates to the old. This confusion is exacerbated by the terseness of writing style that is common amongst application writers. Rather than the "Are you sure?" confirmation mentioned earlier, something like "There are two documents unsaved. Do you want to quit anyway?" would help to keep the user anchored in their current context.

12. The principle of aesthetics

-- Create a program of beauty

It's not necessary that each program be a visual work of art. But it's important that it not be ugly. There are a number of simple principles of graphical design that can easily be learned, the most basic of which was coined by artist and science fiction writer William Rotsler: "Never do anything that looks to someone else like a mistake." The specific example Rotsler used was a painting of a Conan-esque barbarian warrior swinging a mighty broadsword. In this picture, the tip of the broadsword was just off the edge of the picture. "What that looks like", said Rotsler, "is a picture that's been badly cropped. They should have had the tip of the sword either clearly within the frame or clearly out of it."

An interface example can be seen in the placement of buttons -- imagine five buttons, each with five different labels that are almost the same size. Because the buttons are packed using an automated-layout algorithm, each button is almost but not exactly the same size. As a result, though the author has placed much care into his layout, it looks carelessly done. A solution would be to have the packing algorithm know that buttons that are almost the same size look better if they are exactly the same size -- in other words, to encode some of the rules of graphical design into the layout algorithm. Similar arguments hold for manual widget layout.

Another area of aesthetics to consider is the temporal dimension. Users don't like using programs that feel sluggish or slow. There are many tricks that can be used to make a slow program "feel" snappy, such as the use of off-screen bitmaps for rendering, which can then be blitted forward in a single operation. (A pet peeve of this particular author is buttons that *flicker* when the button is being activated or the window is being resized. Multiply redundant refreshing of buttons when changing state is one common cause of this.)

13. The principle of user testing

-- Recruit help in spotting the inevitable defects in your design

In many cases a good software designer can spot fundamental defects in a user interface. However, there are many kinds of defects which are not so easy to spot, and in fact an

experienced software designer is often less capable of spotting them than the average person. In other cases, a bug can only be detected while watching someone else use the program.

User-interface testing, that is, the testing of user-interfaces using actual end-users, has been shown to be an extraordinarily effective technique for discovering design defects. However, there are specific techniques that can be used to maximize the effectiveness of end-user testing. These are outlined in both [TOG91] and [LAUR91] and can be summarized in the following steps:

- Set up the observation. Design realistic tasks for the users, and then recruit end-users that have the same experience level as users of your product (Avoid recruiting users who are familiar with your product however).
- Describe to the user the purpose of the observation. Let them know that you're testing the product, not them, and that they can quit at any time. Make sure that they understand if anything bad happens, it's not their fault, and that it's helping you to find problems.
- Talk about and demonstrate the equipment in the room.
- Explain how to "think aloud". Ask them to verbalize what they are thinking about as they use the product, and let them know you'll remind them to do so if they forget.
- Explain that you will not provide help.
- Describe the tasks and introduce the product.
- Ask if there are any questions before you start; then begin the observation.
- Conclude the observation. Tell them what you found out and answer any of their questions.
- Use the results.

User testing can occur at any time during the project, however, it's often more efficient to build a mock-up or prototype of the application and test that before building the real program. It's much easier to deal with a design defect before it's implemented than after. Tognazzini suggests that you need no more than three people per design iteration -- any more than that and you are just confirming problems already found.

14. The principle of humility

-- Listen to what ordinary people have to say

Some of the most valuable insights can be gained by simply watching other people attempt to use your program. Others can come from listening to their opinions about the product. Of course, you don't have to do exactly everything they say. It's important to realize that each of you, user and developer, has only part of the picture. The ideal is to take a lot of user opinions, plus your insights as a developer and reduce them into an elegant and seamless whole -- a design which, though it may not satisfy everyone, will satisfy the greatest needs of the greatest number of people.

One must be true to one's vision. A product built *entirely* from customer feedback is doomed to mediocrity, because what users want most are the features that they cannot anticipate.

But a single designer's intuition about what is good and bad in an application is insufficient. Program creators are a small, and not terribly representative, subset of the general computing population.

Some things designers should keep in mind about their users:

- Most people have a biased idea as to the what the "average" person is like. This is because most of our interpersonal relationships are in some way self-selected. It's a rare person whose daily life brings them into contact with other people from a full range of personality types and backgrounds. As a result, we tend to think that others think "mostly like we do." Designers are no exception.
- Most people have some sort of core competency, and can be expected to perform well within that domain.
- The skill of using a computer (also known as "computer literacy") is actually much harder than it appears.
- The lack of "computer literacy" is not an indication of a lack of basic intelligence. While native intelligence does contribute to one's ability to use a computer effectively, there are other factors which seem to be just as significant, such as a love of exploring complex systems, and an attitude of playful experimentation. Much of the fluency with computer interfaces derives from play -- and those who have dedicated themselves to "serious" tasks such as running a business, curing disease, or helping victims of tragedy may lack the time or patience to be able to devote effort to it.
- A high proportion of programmers are introverts, compared to the general population. This doesn't mean that they don't like people, but rather that there are specific social situations that make them uncomfortable. Many of them lack social skills, and retreat into the world of logic and programming as an escape; As a result, they are not experienced people-watchers.

The best way to avoid misconceptions about users is to spend some time with them, especially while they are actually using a computer. Do this long enough, and eventually you will get a "feel" for how the average non-technical person thinks. This will increase your ability to spot defects, although it will never make it 100%, and will never be a substitute for user-testing.

Bibliography

[TOG91] *Tog On Interface*, Bruce Tognazzini, Addison-Wesley, 1991, ISBN 0-201 60842-1

[LAUR91] *The Art of Human Computer Interface Design*, Brenda Laurel, Addison-Wesley, 1991, ISBN 0-201 51797-3

The Psychology of Everyday Things, Don Norman, Harper-Collins 1988, ISBN 0-465 06709-3

The Macintosh Human Interface Guidelines, Apple Computer Staff, Addison-Wesley 1993, ISBN 0-201 62216-5

The Amiga User Interface Style Guide, Commodore-Amiga, Addison-Wesley 1991, ISBN 0-201 57757-7

Credits and Change History

August 14, 1998: Initial public release.

August 19, 1998: Applied grammatical fixes sent in by Kuraiken and Mark Ellis. Made various additions attempting to fix deficiencies pointed out by James Cook (risks of metaphors), Richard Kail (cross-cultural metaphors) and Stephan Bethke (terse and uninformative dialog boxes).

Last updated: Friday, August 14, 1998

Talin@ACM.org

Five Main Principles of Good Web Site Design

1. Have a clear purpose in mind.

Web sites should have a clear purpose for existing. They should concentrate on a particular theme or topic. The purpose of this web site is to summarize the principles of good web site and page design and to offer links to other sites which cover these principles in greater detail.

2. Know your intended audience.

Often the perceived correctness of a particular interface design is largely dependent on the personal preferences of the individual user. For this reason, an important first step in the design of a user interface is to know who will be using the interface (Lewis and Rieman, 1994). This site is primarily intended for those people who are new to web site design principles.

3. Have a well structured site.

Many authors such as Marcus (1990), McFarland (1995), Andleigh and Thakrar (1996), and Levi and Conrad (1996), mention that a user interface needs to communicate clearly with the intended user. In order to communicate clearly, a user interface should be well organized and structured (Vaughan, 1996). Web pages within the site should have a consistent and predictable appearance, so that it will be easier for users make sense out of the web site. The best approach is to consistently apply a few basic design principles to each web page in the site (Lynch, 1995). Specific structuring principles will be further discussed in the "[WWW Page Design](#)" section of this site.

4. Make your site easy to navigate.

If your web site is going to be useful, it must be easy to navigate. Navigation occurs at two levels; within a particular web site and between web sites. Menus are often used to aid navigation within a web site. Menus should be neither too shallow nor too deep and may lose their effectiveness if they don't have at least four or five links (Lynch, 1997). Zimmerman (1997) points out that one way to enhance the navigability within a site is to include navigation aids such as "return to home page", "previous page", and "next page" links on each web page. This not only increases the navigability within your site, but allows users who enter your site on a page other than your home page, to easily find their way around. While you don't have control over other sites, you do have control over the links from your sites to other sites. It is up to you to ensure that these links are pertinent to your site's topic and current.

5. Keep your site current.

The World Wide Web is constantly changing, as new sites are developed and old ones are changed or go out of existence. It is up to you to keep your site current and accurate. Check your links from time to time to be sure that the external sites still exist, and add new links that are pertinent to your site's topic as you discover them.

Six Basic Principles of Web Page Design

1. Include essential elements on each page.

Any Web page may be accessed directly from another Web site, therefore each Web page needs to contain essential information which allows it to act as an independent document. This essential information is usually placed into one of three main parts of the Web page; the header, the body, or the footer (Lynch, 1997).

The header is used to bring continuity to the various pages of the Web site, as well as indicate the main topic of the particular Web page. Therefore, the header usually contains a banner graphic which ties the various Web pages of the site together, the title of the document, and navigational aids which link to other pages within the Web site.

The body contains the main textual content of the document, as well as hypertext links to other related Web sites.

The footer is used to verify the origin and authorship of the Web page. Therefore, the footer should contain the author or contact person of the site, as well as the institution with which the author is affiliated (if any), navigational links to other pages within the Web site, the date of last revision, and a statement of copyright. Other useful information might also include the author's e-mail and mailing addresses, as well as the URL of the document.

2. Use appropriate navigational aids.

Good navigational aids are essential to good Web page design. Zimmerman (1997) points out that one way to increase the navigability of Web pages is to include "return to home page", "previous page", and "next page" links on each page. Lynch (1995), and others, suggest that local navigational links be located both at the beginning and end of the page layout so they are easily accessible to the viewer of the Web page. By including the navigational links at the end of the page, the user is not forced to return to the beginning of the Web page after browsing it in order to access another page. Another method used to increase navigability is to provide a menu or table of contents of the Web site on each Web page.

3. Keep page lengths short.

It is usually recommended that Web page lengths not exceed two or three screens worth of information (Lynch, 1995). A major disadvantage of long Web pages is that the user needs to depend on the vertical scroll bar to navigate within the page, a process which can be disorienting to the viewer. In order to keep Web pages short, longer topics can be subdivided into logical chunks of information on separate Web pages. Individual Web pages should include only relevant, yet complete, information on a single topic.

4. Use appropriate text fonts and styles.

Different web page browsers may display special non-standard text fonts in various ways. For this reason, Web page designers should use standard text fonts in designing Web pages (Vaughan, 1996). Lynch (1995) suggests using major HTML heading styles very sparingly and reducing header sizes in general so that more information can be displayed on the screen at one time without losing legibility or causing over crowding to the screen layout.

5. Use color appropriately.

Marcus (1990), McFarland (1995), and others point out that color should be used sparingly and only to highlight key elements of the page or to indicate specific functions. Just because you can view a particular color with a particular browser does not mean that others will be able to view it on their monitor with their browser. Therefore it is important to use browser-safe colors, which are also known as web-safe colors.

Black is traditionally used on Web pages for the main body of text because of its legibility on a light background. Some colors are traditionally used to convey a particular meaning. Red, for example, is often used to signify danger or warning, and thus should be used sparingly to convey such meaning. Blue is traditionally used to indicate hypertext links to other Web pages, and a shade of purple to indicate links which have already been accessed by the current user. Altering these traditional color schemes will most likely confuse the new viewer to the Web site.

6. Keep graphics small.

Graphics can effectively be used to add interest to a web page, but the amount and size of graphics should be kept to a minimum (Zimmerman, 1997). Too many graphics, or a single large graphic, can take a long time to download. Usually, using several smaller graphics, as opposed to one large one, can create a better impression for visitors to your site.

Interface Design Principles

Why should you need to follow user interface principles? In the past, computer software was designed with little regard for the user, so the user had to adapt to the system. This approach to system design is not at all appropriate today.

The key to design principles is knowing which ones are most important when making **design tradeoffs**. For certain products and specific design situations, these design principles may be in conflict with each other or at odds with product design goals and objectives. "Principles are not meant to be followed blindly, rather they are meant as guiding lights for sensible interface design" (Mandell, 1997, p48).

The three areas of user interface design principles are:

1. Place users in control of the interface.
2. Reduce users' memory load.
3. Make the user interface consistent.

Note: this is the simplest version found of design principles.

To learn more about HCI and effective Web design principles, select a topic below.

Golden Rules of Interface Design

1. Place users in control of the interface (User-centered design): Graphic user interfaces were designed to give people control over their personal computers. Users now expect a level of design sophistication from all graphic interfaces. They no longer expect or want to see the older text based interfaces like lynx. The goal is to provide for the needs of all of your potential users, adapting Web technology to their expectations, and never requiring the reader to simply conform to an interface that puts unnecessary obstacles in their paths.

2. Reduce users' memory load: Research has shown that people can remember up to seven variables plus or minus two (Miller, 1956). This is true of colors, buttons, and many other computer interface variables. This amount of information has become known as chunks of information and has led to chunking theory in information processing theories.

3. Make the user interface consistent: Consistency is a key aspect of usable interfaces. It's also a major area of debate (Mandell, 1996). However, just like all principles, consistency might be a lower priority than other factors, do not follow consistency principles and guidelines if they do not make sense in your environment. The major benefit of consistency is that users can transfer their knowledge and learning to a new program.;

For more information contact:
John Sullivan, Professor -- RVCC
E-mail comments to: jsulliva@raritanval.edu
Copyright @ 1997 John Sullivan
URL: <http://rvcc2.raritanval.edu/~jsulliva/john.html>

Users of Web documents do not just look at information, they interact with it in ways that have no precedents in paper document design. The graphic user interface (GUI) of a computer system includes the interaction metaphors and multimedia building blocks (images, sound, video, and animation) concepts that convey function and meaning on the computer screen (Lynch & Norton, 1995).

Principle 1: Use a Print Style Manual

Web pages versus conventional document design: Most of the current interface styles stem from the organization of printed books and periodicals, and the library indexing and catalog systems that developed around printed information. The interface standards of books are well established and widely agreed upon in publishing. Detailed instructions for creating books may be found in any style manual of your choice, e.g., The Chicago Manual of Style, APA style manual, MLA style manual or Xerox Publishing Standards: A Manual of Style and Design. **Whatever the choice the important point is to stay consistent with one manual.**

Design precedents in print: Although networked interactive hypermedia documents do pose novel challenges to information designers, most of the guidance you need to design, create, assemble, edit, and organize multiple forms or media is not radically different from current practice in print media. Most Web documents can be made to conform to The Chicago Manual of Style conventions for editorial style and text organization.

Make Your Web pages Freestanding: WWW pages are different from books and other documents in one crucial respect: hypertext links allow users to access a single Web page with no preamble. Thus, Web pages need to be more independent than a conventional book.

Navigation

And

Menus

Menu Design Guidelines

A system will often contain large amounts of data and perform a variety of functions. Regardless of its purpose, the system must provide some means to tell users about the information it possesses or the things it can do. Whether to display a menu continually, or on demand, is determined by the menu's frequency of use. Always permanently display menus that are frequently referenced, while occasionally needed menus may be presented on request via pop-ups or pull-downs. Critical options should always be continuously displayed (Galitz, 1994).

1. Provide a main menu
2. Provide only relevant alternatives
3. Minimize the number of levels within limits of clarity
4. Provide hierarchical groupings of elements into categories
5. Provide consistency between menus
6. Provide control
7. Provide one selection per menu
8. Provide familiar, fully spelled-out descriptions of alternatives.

Menu Guidelines

1. Provide a main menu.

The top level menu in a hierarchical menu scheme should be a general or main menu consisting of basic system options. This will provide a consistent starting point for all system activities and a "home-base" to which the user may always return.

2. Provide only relevant alternatives.

Including non-relevant choices on a menu screen increases learning requirements and has been found to interfere with performance (Baker and Goldstien, 1966).

3. Minimize the number of levels within limits of clarity.

According to Lynch & Horton (1997) menu's lose their value if they don't carry at least 4 or 5 links.

4. Provide hierarchical groupings of elements into categories.

Items displayed on menus should be logically grouped to aid learning and speed up the visual search process (Card, 1982). They should be in natural order (month, year), alphabetical order, or when there is a small number of options in group of use or importance.

5. Provide consistency between menu.

Options found on more than one menu should be consistently positioned on all menus. If menus are of variable length, maintain relative positioning of all item options (for example, place EXIT at the bottom or end of the list). If menus are of fixed length, place options in the same physical position within the list.

6. Provide control.

Navigation through menu levels should be accomplished through simple key actions. It should always be very easy to return to the next higher level and the main menu.

7. Provide navigation maps.

It is often difficult to maintain a sense of position or orientation as one wanders deep into a multipath menu system. The result is getting lost in the menu maze. In Web pages map where the user has been.

8. Provide familiar, fully spelled out descriptions of alternatives.

Nothing detracts more from an interface than spell mistakes for lack of clarity.

Text and Interface Design

One of the key contributors to legibility and readability is the use of typography in the user interface. Typography includes the characteristics of individual elements (typefaces and timesteps) or as computer users know them as fonts and styles and their groupings (typesetting techniques). The third and final typographic recommendation concerns typesetting: Within menus, dialogue boxes, control panels, forms, and other window components.

Text Guidelines:

1. You should use a maximum of two fonts. Galitz p.252
2. Generally, sans serif typefaces are recommended.
3. Include no more than 40-60 characters per line
4. Text should be flush left Marus p.428
5. Numbers should be flush right
6. Avoid centered text in lists
7. Avoid short justified lines
8. Never center numbers.

Text Guidelines:

1. Never use more than two fonts at one time.

A regular type and its Italics is a good combination. Also, restrict a type to weights, regular and bold for example.

2. Generally, sans serif typefaces are recommended.

Similar typefaces are grouped into what are called roman which contains the Times typeface illustrated. A second race is sans serif where the Serif Versus Sans Serif. Sans is French for without serif is the little flag or decoration at the end of a letter stroke. For example, serif: Times, New Century whereas sans serif: Helvetica, Arial, Avant Garde (Villimil-Cassanova).

3. Include no more than 40-60 characters per line.

Lines of text should have from 40 to 60 characters maximum, and words should be spaced correctly, usually the width of a lower case "r" or "i" for a variable-width text. (Marcus)

4 - 8. Text and Number justification.

Text should be set in appropriate formats, that is, set text flush left, set number flush right, avoided centered text in lists, and avoid short justified lines of text. Last, never center numbers. According to Marcus p. 428. Fixed-width fonts, justified lines of text can slow reading speed by 12 percent.

**For more information contact:
John Sullivan, Professor -- RVCC
E-mail comments to: jsulliva@raritanval.edu
Copyright @ 1997 John Sullivan**

Button Guidelines

A button when activated causes the action or command inscribed upon them to be immediately performed. The following are button guidelines that should be followed.

1. Maintain consistency of style
2. Maintain consistency of order
3. Button Labels should:
 - Use meaningful text and symbols
 - Use standard names and uses
4. Button size: should fit the longest label
5. Number of buttons: Keep number of buttons to six or fewer
6. Location:
 - Horizontally on the lower part of the window
 - Vertically on the right side
7. Organization
 - Keep related buttons grouped together
 - Keep destructive buttons separate from frequently used

Buttons

Keep buttons consistently positioned across screens
The order should never change

con Design Guidelines:

Objects and actions are depicted on screens through pictograms or symbols called icons. Objects and actions are depicted on screens through pictograms or symbols called icons.

1. Use familiar objects and actions.
2. Create and simply reflect objects represented.
3. Create consistent shapes.
4. Consider animating the icons.
5. Attach a caption to assure intended meaning.
6. Make sure icon stands out from background.

For additional information concerning buttons and icons, please check out some of the topics listed below:

Buttons

1. Maintain consistency.

Although buttons come in varying shapes and sizes, they should not change from one screen to the next. Buttons should be the same type and in the same location across pages. A quit button should not be in one location on one screen and another location on another screen. Keep buttons consistently positioned across screens.

2. Button labels should use meaningful text and symbols as well as standard names and uses.

The user should not have to guess what the text or symbols mean. Standard names should be names like help, exit, etc. In addition, they should be big enough to fit the longest label.

3. Number of buttons: keep to six or fewer.

4. Organization:

The first rule is when in doubt keep buttons horizontal in the lower part of the window or position the buttons vertically along the right side. Keep related buttons grouped together, but keep destructive buttons separate from frequently used buttons.

Icon Design Guidelines

1. Use familiar objects and actions.

Shneiderman (1987) suggests that simple metaphors, analogies, or models with a minimal set of concepts are the best places to start in developing icons. He also suggests that mixing metaphors from two or more sources should be avoided.

2. Create and simply reflect objects represented.

The characteristics of the display itself should permit drawings of adequate quality. Poorly formed or fuzzy shapes will inhibit recognition.

3. Create consistent shapes.

Create consistency in shapes of families of icons and in identical icons of differing sizing.

4. Consider animating the icons.

Recent research (Baecker et al., 1991) has explored the use of bringing to life on screens the icons representing the objects and actions.

5. Attach a caption to assure intended meaning.

The ability to comprehend, and learn, icons can be greatly improved by attaching textual captions or labels to the symbols.

6. Make sure icon stands out from background.

For more information contact:
John Sullivan, Professor -- RVCC
E-mail comments to: jsulliva@raritanval.edu
Copyright © 1997 John Sullivan

Interface Color and Graphics

Color is probably the most often misused element in user interface design. As information-processing creatures, we try to attach meaning to what we see and perceive. Therefore, color in the user interface must be very carefully researched and designed. Color has often been misused as a decorative element. This limits its ability to portray meaningful information in the interface (Mandell, 1997)

Color Guidelines:

1. Use a maximum of five, plus or minus two colors.
2. Use center and peripheral colors appropriately.
3. Use same color for grouping related elements.
4. Use redundant coding of shape as well as color.
5. Avoid blue for text, thin lines, and small shapes.
6. Avoid adjacent colors differing only in the amount of blue.
7. Avoid red and green in the periphery of large-scale displays.
8. Opponent colors go well together.

Read below for more detailed information on the above topics.

Color Guidelines:

1. Use a maximum of five, plus or minus two colors.

Four distinct colors are appropriate. Allows extra room in short-term memory that can store: five words or shapes, six letters, seven colors, eight digits. Computers can provide 16 million colors; humans can discriminate about 7.5 million colors.

2. Use center and peripheral colors appropriately.

Use blue for large areas, not for text type. Blue is good for slide and screen backgrounds. Use red and green to capture attention. The visual field adapts easily to this.

3. Use same color for grouping related elements.

Cognitive science has advanced the notion of set and preattentive processing. In this context, you can prepare or set the user for related events by using a common color code (Murch, p 443) A successive set of images can be shown to be related by using the same color. Similar colors can have similar meanings. Therefore, on the same level in a Web site of Multimedia presentation use the same background color. Do not switch backgrounds unless there is a good reason.

4. Use redundant coding of shape as well as color.

Makes the display more resilient to color distortions. Ambient light changes can cause changes in perceived hue, value (lightness), and chroma (saturation).

5. Avoid blue for text, thin lines, and small shapes.

Our visual system is just not set up for detailed, sharp, short wavelength stimuli. However, blue does make a good background color and is perceived clearly out into the periphery of our visual field.

6. Avoid adjacent colors differing only in the amount of blue.

Edges that differ only in the amount of blue will appear indistinct.

7. Avoid red and green in the periphery of large-scale displays.

Due to the insensitivity of the retinal periphery to red and green, these colors in saturated form should be avoided. Yellow and blue are good peripheral colors.

8. Opponent colors go well together.

Red and green or yellow and blue are good combinations for simple displays. The opposite combinations - red with yellow or green with blue - produce poorer images.

Color Terms and Concepts: (Just think of the controls on the TV)

Color discussions can be confusing because scientists, artists, designers, programmers, and marketing professionals describe color phenomena in different ways. The color terms discussed above are hue, value, and chroma. Color terms are derived from the Munsell system of color used by artists, designers, and manufacturers (Marcus, 1996, p.429). The dimensions differ from the red, green, and blue basis of the "RGB" color system familiar to users of CRT display devices.

Hue is the spectral wavelength composition of a color that produces perceptions of being blue, orange, green, etc.

Value is the relative amount of lightness or darkness of the color in a range of black to white (also called intensity).

Chroma is the purity of the color in a scale from gray to the most vivid variant of the perceived color (also called saturation)

Brightness refers to the amount of light energy creating the color.

Reference:

The July/August 1996 issue of ACM *interactions* features an article by Shubin, Falck, and Johansen (1996) entitled, "Exploring Color in Interface Design." It even has cutout color swatches to play with as you follow the examples in the article.

**For more information contact:
John Sullivan, Professor -- RVCC
E-mail comments to: jsulliva@raritanval.edu
Copyright @ 1997 John Sullivan**

URL: <http://www.raritanval.edu/departments/cis/~jsulliva/john.html>

[Jakob Nielsen's Alertbox:](#)

Summary:

The ten most egregious offenses against users. Web design disasters and HTML horrors are legion, though many usability atrocities are less common than they used to be.

Since my first attempt in 1996, I have compiled many top-10 lists of the biggest mistakes in Web design. See links to **all these lists** at the bottom of this article. This article presents the highlights: the very worst mistakes of Web design. (Updated 2007.)

1. Bad Search

Overly literal search engines reduce usability in that they're unable to handle typos, plurals, hyphens, and other variants of the query terms. Such search engines are [particularly difficult for elderly users](#), but they hurt everybody.

A related problem is when search engines prioritize results purely on the basis of how many query terms they contain, rather than on each document's importance. Much better if your search engine calls out "best bets" at the top of the list -- especially for important queries, such as the names of your products.

Search is the user's lifeline when navigation fails. Even though advanced search can sometimes help, [simple search usually works best](#), and search should be presented as a simple box, since that's what users are looking for.

2. PDF Files for Online Reading

Users hate coming across a PDF file while browsing, because it breaks their flow. Even simple things like printing or saving documents are difficult because standard browser commands don't work. Layouts are often optimized for a sheet of paper, which rarely matches the size of the user's browser window. Bye-bye smooth scrolling. Hello tiny fonts.

Worst of all, PDF is an undifferentiated blob of content that's hard to navigate.

PDF is great for printing and for distributing manuals and other big documents that need to be printed. Reserve it for this purpose and convert any information that needs to be browsed or read on the screen into real web pages.

> Detailed discussion of why [PDF is bad for online reading](#)

3. Not Changing the Color of Visited Links

A good grasp of past navigation helps you understand your current location, since it's the culmination of your journey. Knowing your past and present locations in turn makes it easier to decide where to go next. Links are a key factor in this navigation process. Users can exclude links that proved fruitless in their earlier visits. Conversely, they might revisit links they found helpful in the past.

Most important, knowing which pages they've already visited frees users from unintentionally revisiting the same pages over and over again.

These benefits only accrue under one important assumption: that users can tell the difference between visited and unvisited links because the site shows them in different colors. When visited links **do**'t change color, users exhibit more navigational disorientation in usability testing and unintentionally revisit the same pages repeatedly.

4. Non-Scannable Text

A wall of text is deadly for an interactive experience. Intimidating. Boring. Painful to read.

[Write for online](#), not print. To draw users into the text and support scannability, use well documented tricks:

- subheads
- bulleted lists
- **highlighted keywords**
- short paragraphs
- the inverted pyramid
- a simple writing style, and
- de-fluffed language devoid of marketese.

> [Eyetracking of reading patterns](#)

5. Fixed Font Size

CSS style sheets unfortunately give websites the power to disable a Web browser's "change font size" button and specify a fixed font size. About 95% of the time, this fixed size is *tiny*, reducing readability significantly for most people over the age of 40.

Respect the user's preferences and let them [resize text](#) as needed. Also, specify font sizes in relative terms -- not as an absolute number of pixels.

6. Page Titles With Low Search Engine Visibility

Search is the most important way users discover websites. Search is also one of the most important ways users find their way around individual websites. The humble page title is your main tool to attract new visitors from search listings and to help your existing users to locate the specific pages that they need.

The page title is contained within the HTML <title> tag and is almost always used as the clickable headline for listings on search engine result pages (SERP). Search engines typically show the first 66 characters or so of the title, so it's truly [microcontent](#).

Page titles are also used as the default entry in the Favorites when users bookmark a site. For your homepage, begin the with the company name, followed by a brief description of the site.

Don't start with words like "The" or "Welcome to" unless you want to be alphabetized under "T" or "W."

For other pages than the homepage, start the title with a few of the most salient information-carrying words that describe the specifics of what users will find on that page. Since the page title is used as the window title in the browser, it's also used as the label for that window in the taskbar under Windows, meaning that advanced users will move between multiple windows under the guidance of the first one or two words of each page title. If all your page titles start with the same words, you have severely reduced usability for your multi-windowing users.

[Taglines on homepages](#) are a related subject: they also need to be short and quickly communicate the purpose of the site.

7. Anything That Looks Like an Advertisement

Selective attention is very powerful, and Web users have learned to stop paying attention to any ads that get in the way of their goal-driven navigation. (The main exception being [text-only search-engine ads](#).)

Unfortunately, users also ignore legitimate design elements that look like prevalent forms of advertising. After all, when you *ignore* something, you don't study it in detail to find out what it is.

Therefore, it is best to avoid any designs that look like advertisements. The exact implications of this guideline will vary with new forms of ads; currently follow these rules:

- **banner blindness** means that users never fixate their eyes on anything that looks like a banner ad due to shape or position on the page
- **animation avoidance** makes users ignore areas with blinking or flashing text or other aggressive animations
- **pop-up purges** mean that users close pop-up windows before they have even fully rendered; sometimes with great viciousness (a sort of getting-back-at-GeoCities triumph).

8. Violating Design Conventions

Consistency is one of the most powerful usability principles: when things always behave the same, users don't have to worry about what will happen. Instead, they *know* what will happen based on earlier experience. Every time you release an apple over Sir Isaac Newton, it will drop on his head. That's *good*.

The more users' expectations prove right, the more they will feel in control of the system and the more they will like it. And the more the system breaks users' expectations, the more they will feel insecure. Oops, maybe if I let go of this apple, it will turn into a tomato and jump a mile into the sky.

Jakob's Law of the Web User Experience states that "users spend most of their time on *other* websites."

This means that they form their expectations for your site based on what's commonly done on most other sites. If you deviate, your site will be harder to use and users will leave.

9. Opening New Browser Windows

Opening up new browser windows is like a vacuum cleaner sales person who starts a visit by emptying an ash tray on the customer's carpet. Don't pollute my screen with any more windows, thanks (particularly since current operating systems have miserable window management).

Designers open new browser windows on the theory that it keeps users on their site. But even disregarding the **user-hostile message implied in taking over the user's machine**, the strategy is self-defeating since it disables the *Back* button which is the normal way users return to previous sites. Users often don't notice that a new window has opened, especially if they are using a small monitor where the windows are maximized to fill up the screen. So a user who tries to return to the origin will be confused by a grayed out *Back* button.

Links that don't behave as expected undermine users' understanding of their own system. A link should be a simple hypertext reference that replaces the current page with new content. Users hate unwarranted pop-up windows. When they want the destination to appear in a new page, they can use their browser's "open in new window" command -- assuming, of course, that the link is not a piece of code that interferes with the browser's standard behavior.

10. Not Answering Users' Questions

Users are highly goal-driven on the Web. They visit sites because there's something they want to accomplish -- maybe even buy your product. The ultimate failure of a website is to fail to provide the information users are looking for.

Sometimes the answer is simply not there and you lose the sale because users have to assume that your product or service doesn't meet their needs if you don't tell them the specifics. Other times the specifics are buried under a thick layer of marketese and bland slogans. Since users don't have time to read everything, such hidden info might almost as well not be there.

The worst example of not answering users' questions is to **avoid listing the price** of products and services. No B2C ecommerce site would make this mistake, but it's rife in [B2B](#), where most "enterprise solutions" are presented so that you can't tell whether they are suited for 100 people or 100,000 people. Price is the most specific piece of info customers use to understand the nature of an offering, and not providing it makes people feel lost and reduces their understanding of a product line. We have miles of videotape of users asking "*Where's the price?*" while tearing their hair out.

Even B2C sites often make the associated mistake of forgetting prices in product lists, such as [category pages](#) or [search results](#). Knowing the price is key in both situations; it lets users differentiate among products and click through to the most relevant ones.