Design basics

The design principles presented here combine traditional wisdom with extensions to address the evolution of future interfaces. Existing design principles are based on our own experiences in user interface design, on the design experiences of others, and on insights from linguistics and psychology. We have extended these design principles to address evolving interfaces that will provide a more friendly appearance and behavior in the future. The increasing use of 3-D and real-world representations as well as the blossoming popularity of the Internet and the World Wide Web have strongly influenced these progressions.

The most recent influence on these principles has come from our design experience in creating an object-oriented user interface (OOUI). IBM pioneered OOUI architecture and design. Popular operating systems such as Windows 95, IBM OS/2 Warp, and CDE for Unix provide varying degrees of object-orientation for users.

In order to effectively apply these design principles, you need to understand users' tasks and requirements. Understanding and applying principles will be meaningless if users are unhappy with the final product.

Our goal for user interface design is to have the interface positively support users' endeavors and never intrude adversely. The interface should be transparent to the task the user is trying to accomplish and be efficient, satisfying, and fun to use.

Design Principles

Simplicity: Don't compromise usability for function

Keep the interface simple and straightforward. Users benefit from function that is easily accessible and usable. A poorly organized interface cluttered with many advanced functions distracts users from accomplishing their everyday tasks. A well-organized interface that supports the user's tasks fades into the background and allows the user to work efficiently.

Basic functions should be immediately apparent, while advanced functions may be less obvious to new users. Function should be included only if a task analysis shows it is needed. Therefore, keep the number of objects and actions to a

minimum while still allowing users to accomplish their tasks.

Support: Place the user in control and provide proactive assistance

To give users control over the system, enable them to accomplish tasks using any sequence of steps that they would naturally use. Don't limit them by artificially restricting their choices to your notion of the "correct" sequence.

The system should also allow users to establish and maintain a working context, or frame of reference. The current state of the system and the actions that users can perform should be obvious. Users should be able to leave their systems for a moment or a day and find the systems in the same familiar state when they return. This contextual framework contributes to their feeling of stability.

Most users perform a variety of tasks, being expert at some and novice at others. In addition to providing assistance when requested, the system should recognize and anticipate the user's goals, and offer assistance to make the task easier. Ideally, assistance should provide users with knowledge that will allow them to accomplish their tasks quickly. Intelligent assistance is like the training wheels on a bicycle - at some point, most users will want to take them off and go forward on their own. The assistance should allow them to become independent at some point when they choose to be so.

Familiarity: Build on users' prior knowledge

Allow users to build on prior knowledge, especially knowledge they have gained from experience in the real world. A small amount of knowledge, used consistently throughout an interface, can empower the user to accomplish a large number of tasks. Concepts and techniques can be learned once and then applied in a variety of situations. Users should not have to learn new things to perform familiar tasks. The use of concepts and techniques that users already understand from their real world experiences allows them to get started quickly and make progress immediately.

The metaphors used in today's user interfaces tend to be inadequate when compared to the real world. Through the use of visuals and interaction techniques that more closely resemble users' real world experiences, there should be little need to continue reliance on such metaphors.

In the past, designers tended to invoke a principle of consistency


when no single design alternative appeared to be the best answer. By choosing to be consistent with something the user already understands, an interface can be made easier to learn, more productive, and even fun to use.

Avoid the tendency to employ consistency without understanding your users, their tasks, and their shared experiences. When choosing a dimension within which to be consistent, seek to understand what the user expects and be consistent with those expectations. Providing a familiar experience is the ultimate use of consistency in which a truly intuitive interface will result.

Obviousness: Make objects and their controls visible and intuitive


Where you can, use real-world representations in the interface. Real-world representations and natural interactions (direct action) give the interface a familiar look and feel and can make it more intuitive to learn and use. Icons and windows were early attempts to draw on user experiences outside the computing domain. As we move toward real-world representations, reliance on such computer artifacts should decline. In an object-oriented interface the objects and concepts presented to users parallel familiar things from the real world; for example:

- Trash can - when we throw things away we usually use some type of trash receptacle or "trash can". An object on the desktop displayed as a trash can communicates to users that it is a place for discarding things. It should look like the real object rather than like an abstract container, and the user should be able to show its contents in a meaningful way.
- Telephone - the actions we take with telephones are so familiar to most of us that they require little thought. A telephone object on the desktop indicates to users that it will allow them to perform phone-related tasks, and users will expect it to behave like the real thing.


The controls of the system should be clearly visible and their functions identifiable. Visual representations provide cues and reminders that help users understand roles, remember relationships, and recognize what the computer is doing. For example, the numbered buttons on the telephone object indicate that they can be used to key in a telephone number.

Allow users to interact directly with objects and minimize the use of indirect techniques. Identifying an object and doing something with it (like picking up the handset of a phone to answer it) usually are not separate actions in the real world. Likewise, with direct action techniques, explicit selection is not necessary because selection is implicit in the actions users take

with objects. Real-world 3D interfaces are especially conducive to direct interaction.

Encouragement: Make actions predictable and reversible

A user's actions should cause the results the user expects. In order to meet those expectations, the designer must understand the user's tasks, goals, and mental model. Use terms and images that match users' task experience, and that help users understand the objects and their roles and relationships in accomplishing tasks.

Users should feel confident in exploring, knowing they can try an action, view the result, and undo the action if the result is unacceptable. Users feel more comfortable with interfaces in which their actions do not cause irreversible consequences.

Even seemingly trivial user actions, such as deselection or moving objects, should be reversible. For example, a user who spends several minutes deliberating and selecting individual files to be archived from a group will be very upset if all the files are accidentally deselected and the deselection cannot be undone.

Avoid bundling actions together, because the user may not anticipate the side effect. For example, if a user chooses to cancel a request to send a note, only the send request should be cancelled. Do not bundle another action, such as deletion of the note, with the cancel request. Rather than implementing composite actions, make actions independent and provide ways to allow users to combine them when they wish.

Satisfaction: Create a feeling of progress and achievement

Allow the user to make uninterrupted progress and enjoy a sense of accomplishment. Reflect the results of actions immediately; any delay intrudes on users' tasks and erodes confidence in the system. Immediate feedback allows users to assess whether the results were what they expected and to take alternative action immediately. For example, when a user chooses a new font, the font of all applicable text, or of sample text, should change immediately. The user can then decide if the effect is what was desired and, if not, can change it before switching attention to something else.

Offer a preview of the results of an action when it would be inconvenient for a user to apply the action and then reverse it. For example, if a user wants to bold, underscore, and use helvetica font in certain places throughout a document, provide a sample part of that document with those changes applied, allowing the

user to decide if that is the right action to take. This saves the user a lot of time by not having to reverse the action that's been applied to an entire document and enhances the user's confidence in the system.

Avoid situations where users may be working with information that is not up-to-date. Information should be updated immediately or refreshed as soon as possible so that users are not making incorrect decisions or assumptions. If, for some reason, the results of a refresh cannot be displayed immediately, the situation should be communicated to users. This becomes especially important in networked environments where it is more difficult to maintain state between networked systems dynamically. For example, most Web browsers display a completion percentage in the information area so that users know the progress of the graphics loading process.

Availability: Make all objects available at all times

Users should be able to use all of their objects in any sequence and at any time. Avoid the use of modes, those states of the interface in which normally available actions are no longer available, or in which an action causes different results than it normally does.

Modes restrict the user's ability to interact with the system. For example, one of the most common uses of modes in menu-driven systems is the modal dialog box (such as "Print" and "Save as") used to request command parameters. Modal dialogs tend to lock users out of their system; to continue, users must complete - or cancel - the modal dialog. If users need to refer to something in an underlying window to complete the dialog, they must cancel the dialog, access the information they need and re-invoke the dialog.

Safety: Keep the user out of trouble

Users should be protected from making errors. The burden of keeping the user out of trouble rests on the designer. The interface should provide visual cues, reminders, lists of choices, and other aids, either automatically or on request. Humans are much better at recognition than recall. Contextual and hover help, as well as agents, can provide supplemental assistance. Simply stated, eliminate the opportunity for user error and confusion.

Users should never have to rely on their own memory for something the system already knows, such as previous settings, file names, and other interface details. If the information is in the system in any form, the system should provide it.

Two-way communication may be necessary at times to allow users to clarify or confirm requests, or to remedy a problem. In the past, many interfaces have treated communication with users as primarily one-way, computer-to-user. The communication should be interactive - as rich in presentation and interaction capabilities as the rest of the interface. It should present relevant information, provide access to related information and help, and allow users to make task-specific decisions to continue. For instance, spell check, as designed in some systems, highlights potentially misspelled words as users work, allowing them to either select a new word or continue to work until they reach a point where they can go back and validate the potentially misspelled words.

Adopt the following design perspective: users know what they want to accomplish, but sometimes they find it difficult to express their desires using the objects and actions provided, and the system is unable to recognize their request. Two-way communication may be used to help users reach their goals.

Versatility: Support alternate interaction techniques

Allow users to choose the method of interaction that is most appropriate to their situation. Interfaces that are flexible in this way are able to accommodate a wide range of user skills, physical abilities, interactions, and usage environments.

Each interaction device is optimized for certain uses or users and may be more convenient in one situation than another. For example, a microphone used with voice-recognition software can be helpful for fast entry of text or in a hands-free environment. Pen input is helpful for people who sketch, and mouse input works well for precisely indicating a selection. Alternative output formats, such as computer-generated voice output for foreign language instruction, are useful for some purposes. No single method is best for every situation.

Users should be allowed to switch between methods to accomplish a single interaction. For example, allow the user to swipe-select using the mouse, then to adjust the selection using the keyboard. At the same time, users should not be *required* to alternate between input devices to accomplish what they perceive as a single step or a series of related steps in a task. For example, it would be tedious to require the use of a mouse for scrolling while editing text from the keyboard. Users should be able to complete an entire useful sequence through the same input device.

Providing a range of interaction techniques recognizes that users are individuals with different abilities and situations. The differences include disabilities,

preferences, and work environments.

Personalization: Allow users to customize

The interface should be tailorable to individual users' needs and desires. No two users are exactly alike. Users have varying backgrounds, interests, motivations, levels of experience, and physical abilities. Customization can help make an interface feel comfortable and familiar.

Personalizing a computer interface can also lead to higher productivity and user satisfaction. For example, allowing users to change default values can save them time and hassle when accessing frequently used functions.

In an environment where multiple users are using a shared machine, allow the users to create their own system personality and make it easy to reset the system. In an environment where one user may be using many computers, make personalization information portable so the user can carry that "personality" from one system to another.

Affinity: Bring objects to life through good visual design

The goal of visual design in the user interface is to surface to the user in a cohesive manner all aspects of the design principles. Visual design should support the user model and communicate the function of that model without ambiguities. Visual design should not be the "icing on the cake" but an integral part of the design process. The final result should be an intuitive and familiar representation that is second nature to users.

The following are visual design principles that promote clarity and visual simplicity in the interface:

- **Subtractive design** - reduce clutter by eliminating any visual element that doesn't contribute directly to visual communication.
- **Visual hierarchy** - by understanding the importance of users' tasks, establish a hierarchy of these tasks visually. An important object can be given extra visual prominence. Relative position and contrast in color and size can be used.
- **Affordance** - when users can easily determine the action that should be taken with an object, that object displays good affordance. Objects with good affordance usually mimic real world objects.
- **Visual scheme** - design a visual scheme that maps to the user model and lets the user customize the interface. Do not eliminate extra space in your image just to save space. Use white space to provide visual "breathing room."

Printable section ↑ Back to top ← Previous Next →