Larry L. Constantine & Lucy A.D. Lockwood

SOFTWARE FOR USE

A Practical Guide to the Models and Methods of Usage-Centered Design

SOFTWARE FOR USE

A Practical Guide to the Models and Methods of Usage-Centered Design

LARRY L. CONSTANTINE
LUCY A. D. LOCKWOOD

ACM Press
New York, New York

Addison-Wesley

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

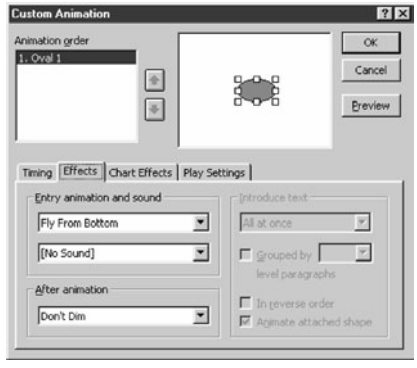


FIGURE 7-6 *Time and motion slowed.*
(MICROSOFT POWERPOINT 97)

use cases. For example, a tabbed dialogue from within a presentation package is illustrated in Figure 7-6. A common task is animating an object, which typically requires setting the motion or style of animation and controlling the timing relative to other events. In this design, the user must always move between two different tabs, **`Effects`** and **`Timing`**, to complete the usual task. While this organization may make sense to users in terms of the categories in which they think and reason about presentation problems, it does not reflect how they carry out tasks.

Aesthetic apprehension. Visual organization based on semantic and task considerations is most important for good usage-centered designs, but visual aesthetics can also be a factor of lesser import. As a rule, the primary goal is to achieve a highly usable design. Aesthetic appeal is a secondary concern best approached through polishing the design once a good basic layout has been obtained.

The primary goal is a highly usable design; aesthetic appeal is a secondary concern best achieved through polishing a good basic layout.

As a holdover from work with the printed page, graphical designers refer to the space between visual features as "white space," even when this space may be gray or black on the user interface. As a rule, graphical user interfaces are somewhat more appealing and may even be marginally easier to understand when white space is approximately balanced and distributed somewhat evenly over the interaction context. For user interface designers with a "tin eye," one guideline is to make the total amount of white space above the midline and below it roughly equal, as well as the white space to the left and to the right of center in the interaction context. The same guideline applies to the number of user interface controls or visual features on either side of the midline and of the centerline. The goal is not perfect symmetry, but something

resembling a rough balance. Radically skewed designs are likely to be the most visually disconcerting.

White space is especially important between visual elements and between visual elements and borders or edges. For readability, space is needed between labels and the edges of command buttons, for instance, and controls should not be crowded up against the edges of dialogues or the frames within them.

User interfaces where components do not quite line up can look amateurish, and some research [see, for example, Comber and Maltby, 1994; 1995] suggests that alignment of visual components is more than just an aesthetic matter; it can have some impact on ease of interpretation and use. For a tidy appearance, aligning both the top edges and left edges of visual elements has the greatest effect.

The component size is another visual dimension that can convey meaning and establish associations or disassociations among visual features. Primary navigation controls and other standard command buttons should all be of the same standard size throughout the user interface, but other functions may employ distinct sizes or even different shapes to help the visual organization. However, any one interaction context should not use too many different sizes of controls if it is to appear tidy and well laid out. We will return to this issue of the size and alignment of visual features when we take up quantitative measures of user interface designs in Chapter 17.

8

PRACTICAL WIDGETRY: Choosing and Designing Visual Components

BUY OR BUILD

As part of the design process, the user interface must be populated with visual components in the form of the actual user interface widgets that will serve as the tools and materials employed by the user. In some few cases, there will be only one obvious choice for the widget, but most of the time, the designer will have to choose among a number of alternatives to find the most appropriate embodiment for some abstract tool or material from the content model. The choices may include not only standard visual components but also the possibility of components that are custom designed for a given application or context.

Many designers avoid heavy thinking either by choosing components on the basis of what they have seen before in other systems or by doing whatever Microsoft does.

How do you pick the right widget to solve a particular user interface design problem? In some cases, the choice can be guided by some kind of formula or set of rules. Many designers avoid heavy thinking through the practice of design by imitation—choosing components on the basis of what they have seen before in other systems. Others bow to de facto standards and just do whatever Microsoft does.

In the usage-centered approach, the design and selection of visual components is model-driven. In every case, we are seeking the most understandable and usable interpretation of the content model and the most efficient realization of the use cases to be supported.

It would be impractical to go into detail about all the assortment of widgets available for modern graphical user interface design, and any attempt to catalogue

them in a book would soon be outdated in any event. Instead, we will focus on several categories of design issues that typify the kinds of decision making involved in implementation modeling.

ICONIC COMMUNICATION

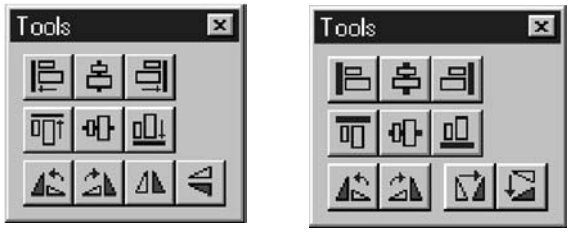
Where would graphical user interfaces be without icons? Icons—the graphical symbols on tools and other visual features—are so ubiquitous in modern software that, to many people, icons are a virtual synonym for a graphical user interface. In the inflated language of software marketing, icons make a user interface intuitive. Although icons do not necessarily make a user interface intuitively obvious, well-conceived and well-designed icons can improve both the learnability and the rememberability of user interfaces by promoting easy recognition of features. They are definitely a part of the look-and-feel that helps mark a user interface as contemporary.

"EYE-CON" DESIGN

Like other aspects of good user interface design, creative icon designs can take time. However, it is important to keep in mind that, if it takes you as the designer more than a few minutes to invent an icon, it is almost certain not to be intuitively obvious to the casual user. Anything that takes the user more than a second or two to figure out can never be said to be intuitive or "guessable." Obscure icons not only do not help the user guess their meaning but also can be hard to memorize and may actually slow down the process of learning how to use a system.

Icons are often claimed to make a user interface intuitive, but obscure icons can be hard to memorize and may actually slow down the process of learning how to use a system.

Good icon and tool design calls for a simple and consistent style, yet with icons that are readily distinguishable from one another. The selective and restrained use of color can help communicate meaning and enhance apparent differences. A fairly good example is shown in Figure 8-1a, a collection of tools from a presentation package for arranging graphical objects. These icons make use of "minimalist" graphics clarified by selective use of color. Knowing the context in which they are used, it is not hard to figure out what they mean. However, they can be improved somewhat. The tiny arrows in the icons of the first six icons are hard to see and add no information. Thickening the bars representing the points of alignment and highlighting them by making them darker would increase the visual differences



(a) (b)

FIGURE 8-1 *Tool icons for object arrangement operations.*
(MICROSOFT POWERPOINT)

FIGURE 8-2 *Iconic conventions for abstract and concrete actions.*
(MICROSOFT POWERPOINT)

among these icons, as can be seen in Figure 8-1b. The last two icons—for flipping horizontally and vertically—are clarified by adding small arrows like those on the rotation tools.

Wherever possible, the icon designer should draw on established conventions, prior associations, and accepted standards. Care should be taken not to violate or contradict highly familiar signs. For example, the widely accepted slash-and-circle sign for "no" or "prohibited" should not be pressed into service for "create a wheel" in a vehicle design package.

Objects or concrete nouns are usually the easiest to represent as icons; they are merely depicted directly, often in simplified or abstract form. Actions are somewhat more difficult, but nouns can sometimes stand in for verbs. For example, an image of a printer can stand for "print," or an opened folder can represent "open file." Abstract verbs are the hardest of all to represent as icons. Conventions can help. Representing the concept of "undo" as an icon has long challenged graphics designers, but the hegemony of Microsoft has reinforced the convention of the little counterclockwise loop, shown at the left in Figure 8-2, to the point that it is all but instantly recognizable. Conventions can also collide with other needs and lead to confusion. The differences among the undo, redo, do again, and rotate icons, shown left to right in Figure 8-2, are minimal and somewhat arbitrary.

SEMIOTICS

An icon means one thing to a software designer and something else to a specialist in the study of signs and symbols. Some knowledge of semiotics, the science of signs, can be useful for user interface designers [Callahan, 1994; Constantine and Henderson-Sellers, 1996]. We refer to any graphical symbol on a tool or other visual control as an icon, but, in semiotics, icons are one of three basic categories of signs: indexical, symbolic, and iconic.

In semiotics, the science of signs, icons are one of three basic categories of signs: indexical, symbolic, and iconic.

Indexical signs are associated with what they represent by a necessary physical or logical connection. Smoke is an indexical sign for fire, for example. *Symbolic* signs are arbitrary. The letter *h* in the Latin alphabet has no necessary association whatsoever with the soft puff of air we use to pronounce it in English, nor does the similarity in shape between an *h* and a *b* give any clue to their pronunciation. The sign composed of the letters *h-a-t* does not in any meaningful way resemble the thing we cover our heads with that it represents.

Iconic signs bear some form of similarity or likeness to the things for which they stand. For example, a symbol on a user interface might represent software "tools" with pictures of a hammer and screwdriver. There are many variations on iconic representation that can be used to create software icons, including direct depiction of objects, analogy and metaphor, synecdoche, and homonyms.

Analogies and metaphors can be a rich source of icon ideas; a magnifying glass can be used to signify "search," for example. However, analogies and metaphors that are too clever or creative can create problems for users, especially when they are based on inside knowledge or a programming perspective. One system used an image of a globe to represent user preferences or options. A programmer from the software vendor justified this design by pointing out that such preferences were "global" parameters. This is a graphical example of inside-out design, a form of software malpractice in which geek-speak becomes the visual language at the user interface.

In synecdoche, a part is used to represent a whole, the specific represents the general, or vice versa. A telephone or a telephone connection might be represented by an image of a handset, for example.

Homonyms or visual puns for icons often work only in one particular language and culture.

Many programmers are such avid punsters that homonyms or visual puns can be particularly appealing to them. However, homonyms and puns are particularly risky for user interface design because they "work" only in one language and often only within a particular culture. For example, one system used a picture of a dog, a golden retriever, to represent "retrieve