(e.g.,

1.6 More on usability: design and usability principles

Another way of conceptualizing usability is in terms of design principles. These are generalizable abstractions intended to orient designers towards thinking about different aspects of their designs. A well-known example is feedback: systems should be designed to provide adequate feedback to the users to ensure they know what to

do next in their tasks. Design principles are derived from a mix of theory-based knowledge, experience, and common sense. They tend to be written in a prescriptive manner, suggesting to designers what to provide and what to avoid at the interface—if you like, the do's and don'ts of interaction design. More specifically, they are intended to help designers explain and improve the design (Thimbleby, 1990). However, they are not intended to specify how to design an actual interface (e.g., telling the designer how to design a particular icon or how to structure a web portal) but act more like a set of reminders to designers, ensuring that they have provided certain things at the interface.

A number of design principles have been promoted. The best known are concerned with how to determine what users should see and do when carrying out their tasks using an interactive product. Here we briefly describe the most common ones: visibility, feedback, constraints, mapping, consistency, and affordances. Each of these has been written about extensively by Don Norman (1988) in his bestseller The Design *of Everyday* Things.

Visibility The importance of visibility is exemplified by our two contrasting examples at the beginning of the chapter. The voice-mail system made the presence and number of waiting messages invisible, while the answer machine made both aspects highly visible. The more visible functions are, the more likely users will be able to know what to do next. In contrast, when functions are "out of sight," it makes them more difficult to find and know how to use. Norman (1988) describes the controls of a car to emphasize this point. The controls for different operations are clearly visible (e.g., indicators, headlights, horn, hazard warning lights), indicating what can be done. The relationship between the way the controls have been positioned in the car and what they do makes it easy for the driver to find the appropriate control for the task at hand.

Feedback Related to the concept of visibility is feedback. This is best illustrated by an analogy to what everyday life would be like without it. Imagine trying to play a guitar, slice bread using a knife, or write using a pen if none of the actions produced any effect for several seconds. There would be an unbearable delay before the music was produced, the bread was cut, or the words appeared on the paper, making it almost impossible for the person to continue with the next strum, saw, or stroke.

Feedback is about sending back information about what action has been done and what has been accomplished, allowing the person to continue with the activity. Various kinds of feedback are available for interaction design—audio, tactile, verbal, visual, and combinations of these. Deciding which combinations are appropriate for different kinds of activities and interactivities is central. Using feedback in the right way can also provide the necessary visibility for user interaction.

Constraints The design concept of constraining refers to determining ways of restricting the kind of user interaction that can take place at a given moment. There are various ways this can be achieved. A common design practice in graphical user interfaces is to deactivate certain menu options by shading them, thereby restrict-

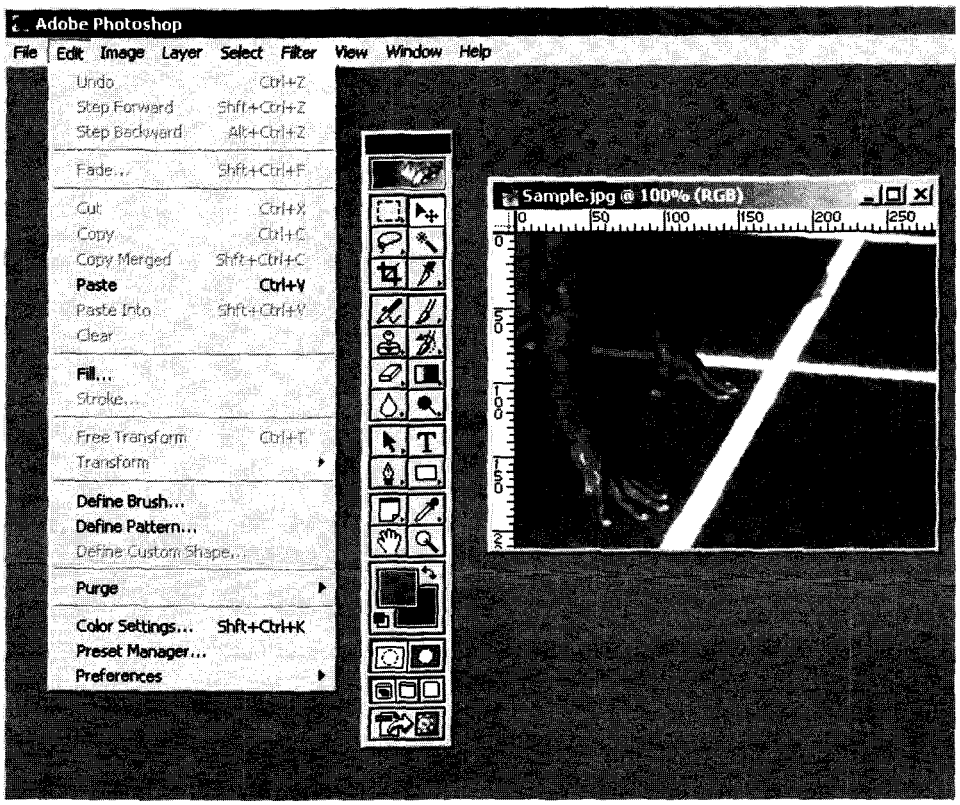

Figure 1.8 A menu illustrating restricted availability of options as an example of logical constraining. Shaded areas indicate deactivated options.

ing the user to only actions permissible at that stage of the activity (see Figure 1.8). One of the advantages of this form of constraining is it prevents the user from selecting incorrect options and thereby reduces the chance of making a mistake. The use of different kinds of graphical representations can also constrain a person's interpretation of a problem or information space. For example, flow chart diagrams show which objects are related to which, thereby constraining the way the information can be perceived.

Norman (1999) classifies constraints into three categories: physical, logical, and cultural. Physical constraints refer to the way physical objects restrict the movement of things. For example, the way an external disk can be placed into a disk drive is physically constrained by its shape and size, so that it can be inserted in only one way. Likewise, keys on a pad can usually be pressed in only one way.

Logical constraints rely on people's understanding of the way the world works (cf. the marbles answering machine design). They rely on people's common-sense reasoning about actions and their consequences. Picking up a physical marble and placing it in another location on the phone would be expected by most people to

Figure 1.9 (a) Natural mapping between rewind, play, and fast forward on a tape recorder device. (b) An alternative arbitrary mapping.

trigger something else to happen. Making actions and their effects obvious enables people to logically deduce what further actions are required. Disabling menu options when not appropriate for the task in hand provides logical constraining. It allows users to reason why (or why not) they have been designed this way and what options are available.

Cultural constraints rely on learned conventions, like the use of red for warning, the use of certain kinds of audio signals for danger, and the use of the smiley face to represent happy emotions. Most cultural constraints are arbitrary in the sense that their relationship with what is being represented is abstract, and could have equally evolved to be represented in another form (e.g., the use of yellow instead of red for warning). Accordingly, they have to be learned. Once learned and accepted by a cultural group, they become universally accepted conventions. Two universally accepted interface conventions are the use of windowing for displaying information and the use of icons on the desktop to represent operations and documents.

Mapping This refers to the relationship between controls and their effects in the world. Nearly all artifacts need some kind of mapping between controls and effects, whether it is a flashlight, car, power plant, or cockpit. An example of a good mapping between control and effect is the up and down arrows used to represent the up and down movement of the cursor, respectively, on a computer keyboard. The mapping of the relative position of controls and their effects is also important. Consider the various musical playing devices (e.g., **MP3,** CD player, tape recorder). How are the controls of playing, rewinding, and fast forward mapped onto the desired effects? They usually follow a common convention of providing a sequence of buttons, with the play button in the middle, the rewind button on the left and the fast-forward on the right. This configuration maps directly onto the directionality of the actions (see Figure 1.9a). Imagine how difficult it would be if the mappings in Figure 1.9b were used. Look at Figure 1.10 and determine from the various mappings which is good and which would cause problems to the person using it.

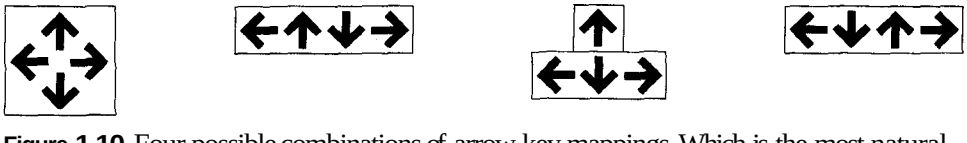

Figure 1.10 Four possible combinations of arrow-key mappings. Which is the most natural mapping?

Consistency This refers to designing interfaces to have similar operations and use similar elements for achieving similar tasks. In particular, a consistent interface is one that follows rules, such as using the same operation to select all objects. For example, a consistent operation is using the same input action to highlight any graphical object at the interface, such as always clicking the left mouse button. Inconsistent interfaces, on the other hand, allow exceptions to a rule. An example of this is where certain graphical objects (e.g., email messages presented in a table) can be highlighted only by using the right mouse button, while all other operations are highlighted using the left button. A problem with this kind of inconsistency is that it is quite arbitrary, making it difficult for users to remember and making the users more prone to mistakes.

One of the benefits of consistent interfaces, therefore, is that they are easier to learn and use. Users have to learn only a single mode of operation that is applicable to all objects. This principle works well for simple interfaces with limited operations, like a mini CD player with a small number of operations mapped onto separate buttons. Here, all the user has to do is learn what each button represents and select accordingly. However, it can be more problematic to apply the concept of consistency to more complex interfaces, especially when many different operations need to be designed for. For example, consider how to **design** an interface for an application that offers hundreds of operations (e.g. a word-processing application). There is simply not enough space for a thousand buttons, each of which maps onto an individual operation. Even if there were, it would be extremely difficult and time-consuming for the user to search through them all to find the desired operation.

A much more effective design solution is to create categories of commands that can be mapped into subsets of operations. For the word-processing application, the hundreds of operations available are categorized into subsets of different menus. All commands that are concerned with file operations (e.g., save, open, close) are placed together in the same file menu. Likewise, all commands concerned with formatting text are placed in a format menu. Selecting an operation then becomes a matter of homing in on the right category (menu) of options and scanning it for the desired one, rather than scrolling through one long list. However, the consistency rule of having a visible one-to-one mapping between command and operation is broken. Operations are not immediately visible at the interface, but are instead hidden under different categories of menus. Furthermore, some menu items are immediately visible, when a top-level menu is first pulled down, while others remain hidden until the visible items are scrolled over. Thus, users need to learn what items are visible in each menu category and which are hidden in submenus.

The way the items are divided between the categories of menu items can also appear inconsistent to users. Various operations appear in menus where they do not belong. For example, the sorting operation (very useful for listing references or names in alphabetical order) in Microsoft Word 2001 is in the Table menu (the Mac Version). In the previous Word 98 version, it was in both the Tools and Table menus. I always thought of it as a Tool operation (like Word Count), and became very frustrated to discover that as a default for Word 2001 it is only in the Table menu. This makes it inconsistent for me in two ways: (i) with the previous version and (ii) in the category it has been placed. Of course, I can customize the new ver-

sion so that the menus are structured in the way I think they should be, but this all takes considerable time (especially when I use different machines at work, home, and when travelling).

Another problem with consistency is determining what aspect of an interface to make consistent with what else. There are often many choices, some of which can be inconsistent with other aspects of the interface or ways of carrying out actions. Consider the design problem of developing a mechanism to let users lock their files on a shared server. Should the designer try to design it to be consistent with the way people lock things in the outside world (called external consistency) or with the way they lock objects in the existing system (called internal consistency)? However, there are many different ways of locking objects in the physical world (e.g., placing in a safe, using a padlock, using a key, using a child safety lock), just as there are different ways of locking electronically (e.g., using PIN numbers, passwords, permissions, moving the physical switches on floppy disks). The problem facing designers is knowing which one to be consistent with.

Affordance is a term used to refer to an attribute of an object that allows people to know how to use it. For example, a mouse button invites pushing (in so doing activating clicking) by the way it is physically constrained in its plastic shell. At a very simple level, to afford means "to give a clue" (Norman, 1988). When the affordances of **a** physical object are perceptually obvious it is **easy** to know how to interact with it. For example, a door handle affords pulling, a cup handle affords grasping, and a mouse button affords pushing. Norman introduced this concept in the late '80s in his discussion of the design of everyday objects. Since then, it has been much popularized, being used to describe how interface objects should be designed so that they make obvious what can be done to them. For example, graphical elements like buttons, icons, links, and scroll bars are talked about with respect to how to make it appear obvious how they should be used: icons should be designed to afford clicking, scroll bars to afford moving up and down, buttons to afford pushing.

Unfortunately, the term affordance has become rather a catch-all phrase, losing much of its potency as a design principle. Norman (1999), who was largely responsible for originally promoting the concept in his book ***The Design*** of ***Everyday Things*** (1988), now despairs at the way it has come to be used in common parlance:

> ***"Zput an affordance there," a participant would say, "I wonder if the object affords clicking. . . " affordances this, affordances that. And no data, just opinion. Yikes! What had I unleashed upon the world?*** Norman's ***(1999)*** reaction to a recent CHI-Web discussion.

He has since tried to clarify his argument about the utility of the concept by saying there are two kinds of affordance: perceived and real. Physical objects are said to have real affordances, like grasping, that are perceptually obvious and do not have to be learned. In contrast, user interfaces that are screen-based are virtual and do not have these kinds of real affordances. Using this distinction, he argues that it does not make sense to try to design for real affordances at the interface--except when designing physical devices, like control consoles, where affordances like pulling and pressing are helpful in guiding the user to know what to do. Alternatively, screen-based

BOX 1.4 Can You Afford a Screen?

A problem in applying the concept of affordance to interface objects is that virtual objects have quite different properties from physical objects. A physical door handle affords pulling because its physical properties constrain what can be done with it in relation to the person and environment. It results in it opening (if it is closed) and it closing (if it is open). It is obvious to the person how to interact with it. However, a virtual object like an icon button invites clicking on only because a user has learned initially that the graphical element on the screen is a representation that, when clicked on, makes something else happen (such as moving to another page). It could equally trigger other system responses, like a window closing down. Hence, the mapping between a virtual representation and its behavior is arbitrary, relying on the user learning the accepted conventions.

A problem in using the concept of affordance in this context is that it can be misleading. Designers can be misled into thinking that virtual objects should be designed to look and behave like physical objects because people know intuitively how to interact with these. This may lead them into thinking that interfaces that exhibit this kind of realism will be easier to learn and use. These assumptions, however, are incorrect for the reason stated above. To illustrate this point further, consider the design of screen buttons. A number of them have been designed to have a 3D look, giving the appearance of protruding. An assumption is that this kind of illusion will give the buttons the *affordance* of pushing, inviting the user to click on them, as they would do with actual physical buttons. While users may readily learn this association, it is equally the case that they will be able to learn how to interact with a simple, 2D representation of a button on the screen. The effort to learn the association is similar. However, the effort to design 3D buttons is likely to be greater than simple 2D buttons.

A danger with trying to design graphical interfaces to afford in the way physical objects do is that it can inadvertently lead to poor design. The use of shadowing and other perceptual illusions to give the effect of 3D can have the undesirable effect of cluttering up an interface, often making it more difficult to find particular objects. Simple, plain 2D abstract shapes (e.g., a square or a circle) used to represent objects like buttons, on the other hand, can be easier to perceive and recognize at the interface (See Figure 1.11 on Color Plate 1).

interfaces are better conceptualized as **perceived** affordances, which are essentially learned conventions. In conclusion, Norman argues that other design concepts--conventions, feedback and cultural and logical constraints—are far more useful for helping designers develop graphical user interfaces.

1.6.1 Heuristics and usability principles

When design principles are used in practice they are commonly referred to as heuristics. This term emphasizes that something has to be done with them when they are applied to a given problem. In particular, they need to be interpreted in the design context, drawing on past experience of, for example, how to design feedback and what it means for something to be consistent.

Another form of guidance is usability principles. An example is "speak the user's language." These are quite similar to design principles, except that they tend to be more prescriptive. In addition, whereas design principles tend to be used mainly for informing a design, usability principles are used mostly as the basis for evaluating prototypes and existing systems. In particular, they provide the framework for heuristic evaluation (see Chapter **13).** They, too, are called heuristics when used as part of

an evaluation. Below are the ten main usability principles, developed by Nielsen (2001) and his colleagues. Note how some of them overlap with the design principles.

1. Visibility of system status—always keep users informed about what is going on, through providing appropriate feedback within reasonable time
2. Match between system and the real world—speak the users' language, using words, phrases and concepts familiar to the user, rather than system-oriented terms
3. User control and freedom—provide ways of allowing users to easily escape from places they unexpectedly find themselves, by using clearly marked 'emergency exits'
4. Consistency and standards—avoid making users wonder whether different words, situations, or actions mean the same thing
5. Help users recognize, diagnose, and recover from errors—use plain language to describe the nature of the problem and suggest a way of solving it
6. error prevention-where possible prevent errors occurring in the first place
7. Recognition rather than recall—make objects, actions, and options visible
8. Flexibility and efficiency of use—provide accelerators that are invisible to novice users, but allow more experienced users to carry out tasks more quickly
9. Aesthetic and minimalist design—avoid using information that is irrelevant or rarely needed
10. Help and documentation—provide information that can be easily searched and provides help in a set of concrete steps that can easily be followed

ACTIVITY 1.5

Comment

Simplicity is certainly an important design principle. Many designers try to cram too much into a screenful of space, making it unwieldy for people to find what they are interested in. Removing design elements to see what can be discarded without affecting the overall function of the website can be a salutary lesson. Unnecessary icons, buttons, boxes, lines, graphics, shading, and text can be stripped, leaving a cleaner, crisper, and easier-to-navigate website. However, a certain amount of graphics, shading, coloring, and formatting can make a site aesthetically pleasing and enjoyable to use. Plain vanilla sites with just lists of text and a few hyperlinks may not be as appealing and may put certain visitors off returning. The key is getting the right balance between aesthetic appeal and the right amount and kind of information per page.