Human–Computer Interaction

Fundamentals and Practice

Gerard Jounghyun Kim

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed on acid-free paper
Version Date: 20141208

International Standard Book Number-13: 978-1-4822-3389-6 (Hardback)

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Contents

Preface

Human–computer interaction (HCI) is becoming ever more important in interactive software. Such software has long been evaluated in terms of the availability and breadth of its functions and its algorithmic efficiency. While such a developer's perspective is still somewhat valid, it has become difficult to differentiate among similar software components from such an aspect given the amazing computing performance of today's hardware and the spread of algorithmic knowledge and systems development know-how. Thus software quality is increasingly judged from the users' external point of view in terms of their expectations, satisfaction, and experience. This external view or user experience may be defined in many ways, but it is most obvious that it has quite a lot to do with how the software users interact with it and, hence, its design. HCI will become even more critical as everything around us becomes digital and unknowingly embedded with interactive computing services that make our everyday lives more exciting, efficient, and convenient.

Therefore, software (at least software that is highly interactive and targeted for a high number of users) must now be developed with HCI as one of its higher priorities. However, at the undergraduate level, it is still often the case that HCI is not given the attention it deserves in the education of future software developers. Most entry-level HCI textbooks are structured around high-level concepts and guidelines and are not directly tied to the software development process. Some of these books may offer design patterns, but students at the undergraduate level might still find it puzzling as to how HCI fits in with their basic software development knowledge. In fact, most of the HCI concepts and guidelines are fairly commonsense or very easy to comprehend. (After all, how difficult would it be to make one understand that users are important?) But it is in the practice and within the context of actual development that one has to make the difficult choices to produce highly usable interactive software.

Following this line of thinking, this book was designed around the overall development cycle for an interactive software product. It starts with the required basic HCI knowledge, which is kept as compact as possible by including only the basic essentials (Chapters 1–3). The intention is to convey the spirit of HCI rather than a long list of compiled knowledge. The book then moves into the application of this knowledge by iteratively forming the HCI requirements and modeling the interaction process (Chapter 4), designing the interface (Chapter 4), implementing the resulting design (Chapters 5–7), and finally evaluating the implemented product (Chapter 8). The book is targeted mainly at undergraduate students of computer science and information technology (IT), but it is easy enough to be taken up by readers in other fields. Some knowledge of computers and programming would be desirable, but it is not absolutely necessary. (Those not interested in the detailed aspects of implementation can skip some of Chapters 5–7.)

The core content of the book is based on the introductory undergraduate HCI course (advanced junior or senior level) that I have taught since 2006 at Korea University. The following table shows how one might structure a similar course using this book (or pace oneself for self-teaching).

	Lecture
Weeks 1–2	Chapters 1–2: Introduction, HCI principles, and guidelines
Weeks 3–5	Chapter 3: Cognitive science, GOMS, human factors Homework 1: • Application of HCI principles/guidelines • GOMS exercise
Weeks 6–8	Chapter 4: HCI design

	Homework 2: • Project proposal (Part 1): Functional and UI requirements, user analysis, etc. • Design of the app (Part 2): Interaction model, scenario, storyboards, basic interface design, and wire-framing • Short presentation
Week 9	Midterm exam (Chapters 1–4)
Weeks 10–11	Chapters 5–7: Implementation issues Homework 3: • First implementation of project (using the MVC model) • Presentation (MVC structure) and working demo 1
Weeks 12–13	Chapter 8: Evaluation
Weeks 14–15	Chapter 9: Future of HCI Homework 4: • Self-heuristic evaluation for the project • Carry out and receive peer review for other projects and one's own project • Redesigning/reimplementation of the project app • Presentation of "before" and "after" and working demo 2
Week 15/16	Final exam

The PowerPoint lecture slides and the source code for the example application used in this book ("No Sheets 1.0," also downloadable through Google Play) are available through the publisher's resource website (see http://www.crcpress.com/product/isbn/9781482233896). I sincerely hope that the book will help readers to develop and acquire an HCI mindset as an important step to becoming a capable IT professional in the field.

The completion of this book was possible only with the greatest help and understanding from many people. My first thanks go to my graduate students at the Digital Experience Laboratory at Korea University (Youngsun, Youngwon, Changhyun, Jong-gil, Sang-yong, Jae-dong, Myong-hee, and Euijae). They helped me with proofreading, drawing figures, formatting, and many other tasks in the midst of research, projects, classes, and all the other things that make up the life of a graduate student. My dear colleagues in the HCI community have also given me much valuable feedback regarding the content and structure of the book. In particular, I thank Prof. Jee-in Kim, Dr. Gun Lee, Prof. Woontak Woo, Prof. Jinwoo Kim, Prof. Jongwon Lee, Prof. Jong-il Park, Prof. Seokhee Jeon, Prof. Si-Jung Kim, Dr. Ungyeon Yang, Prof. Junho Kim, Prof. Chang-Guen Song, Prof. Jin-seok Seo, Prof. Sookjin Kim, Prof. Junho Choi, and Prof. Mincheol Hwang. I am very grateful for the support of the

KRF-funded Engineering Center of Kwangwoon University (head: Professor Eunsoo Kim). CRC Press has been very patient and prompt with assistance for all my writing problems, not to mention seeing the value in publishing this book. Finally, I thank my wife Sooah, my parents, and my children (Andrew and Ellen) for their understanding and just for being there!

About the Author

Gerard "Gerry" Jounghyun Kim earned his bachelor's in electrical and computer engineering at Carnegie Mellon University and his master's and PhD at the University of Southern California. He started his academic career at POSTECH in 1996 after a short post at the U.S. National Institute of Standards and Technology as a National Research Council postdoctoral fellow. In 2006, he moved to Korea University. Since 1996, he has conducted research in the field of HCI, including virtual and mixed reality, mobile interaction, and multimodal interaction. Dr. Kim has written more than 100 articles in international and domestic journals and conferences, and he is the author of *Designing Virtual Reality Systems* (Springer, 2005).

1

Introduction

1.1 What HCI Is and Why It Is Important

Human–computer interaction (HCI) is a cross-disciplinary area (e.g., engineering, psychology, ergonomics, design) that deals with the theory, design, implementation, and evaluation of the ways that humans use and interact with computing devices. *Interaction* is a concept to be distinguished from another similar term, *interface*. Roughly speaking, interaction refers to an abstract model by which humans interact with the computing device for a given task, and an interface is a choice of technical realization (hardware or software) of such a given interaction model. Thus, the letter *I* in HCI refers to both interaction and interface, encompassing the abstract model and the technological methodology (Figure 1.1).

HCI has become much more important in recent years as computers (and embedded devices) have become commonplace in almost all facets of our lives. Aside from merely making the necessary computational functionalities available, the early focus of HCI has been in how to design interaction and implement interfaces for high usability. The term *high usability* means that the resulting interfaces are easy to

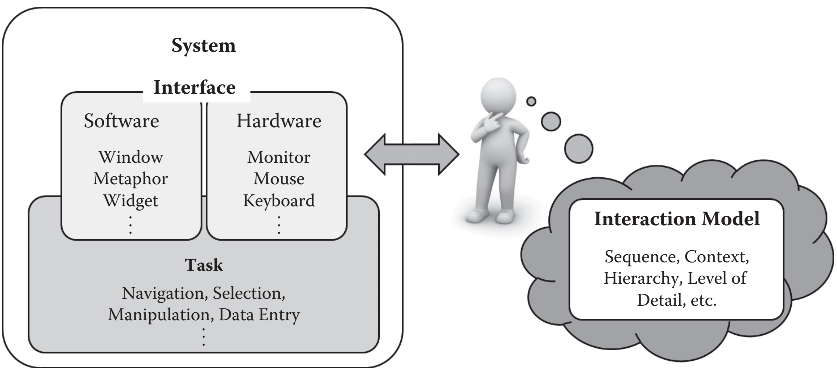


Figure 1.1 The distinguishing concepts of interaction (model) and interface.

use, efficient for the task, ensure safety, and lead to a correct completion of the task. Usable and efficient interaction with the computing device in turn translates to higher productivity.

The simple aesthetic appeal of interfaces (while satisfying the need for usability) is now a critical added requirement for commercial success as well. The family of distinctly designed Apple® products is a good example. Apple products are attractive and have created a multitude of faithful followers even though their functionality may be virtually equal to their competitors. In this context, the concept of *user experience* (UX) has lately become a buzzword, a notion that not only encompasses the functional completeness, high usability, and aesthetic appeal of the interactive artifact, but also its seamless integration into one's lifestyle or even creating a new one around it (Figure 1.2).

A less acknowledged fact is how HCI has had a huge impact in the history of computing and changed our daily lives. It was probably the invention (or rediscovery) of the mouse that was the linchpin in the personal

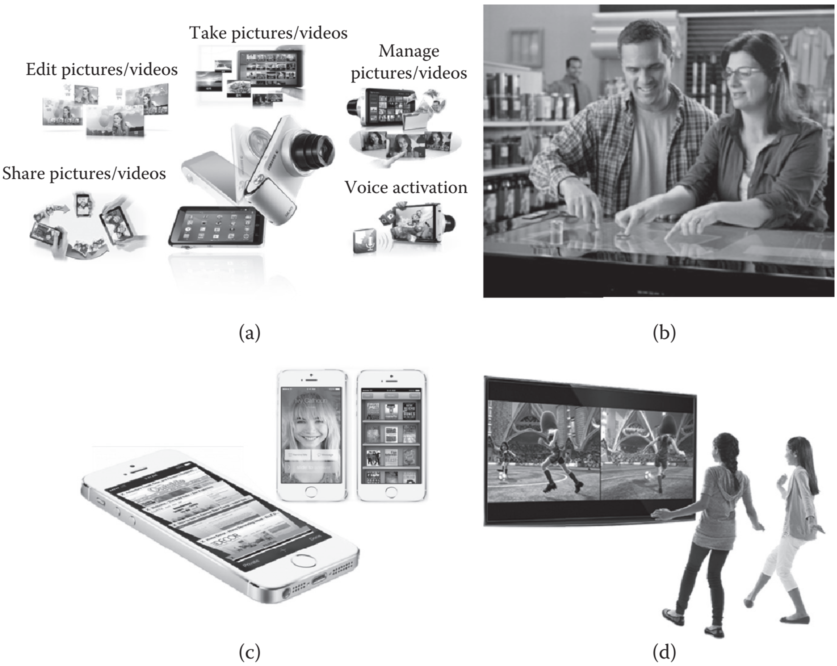


Figure 1.2 Goals of human–computer interaction (HCI): (a) functional completeness (Apple iPhone 5s, http://www.apple.com/iphone-5s), (b) high usability (Microsoft® Pixelsense, http://blogs.msdn.com/b/pixelsense), (c) aesthetic appeal (Apple iPhone 5s), and (d) compelling user experience (UX) (Microsoft Kinect, http://www.xbox.com/ko-KR/Kinect).

Figure 1.3 The evolution of interfaces in the course of the history of computing (i.e., terminal and keyboard, graphic user interface and mouse, and handheld and touch-based interface). (Courtesy of Cox, J., https://www.flickr.com/photos/15587432@N02/3281139507, Melbourne, FL.)

computer revolution, making the operation of a computer intuitive and much easier than the previous system of keyboard commands. The spreadsheet interface made business computing a huge success. The Internet phenomenon could not have happened without the web-browser interface. Smartphones, with their touch-oriented interfaces, have nearly replaced the previous generation of feature phones. Body-based and action-oriented interfaces are now introducing new ways to play and enjoy computer games. HCI still continues to redefine how we view, absorb, exchange, create, and manipulate information to our advantage (Figure 1.3).

1.2 Principles of HCI

Despite its importance, good HCI design is generally difficult, mainly because it is a multiobjective task that involves simultaneous consideration of many things, such as the types of users, characteristics of the tasks, capabilities and cost of the devices, lack of objective or exact quantitative evaluation measures, and changing technologies, to name just a few. A considerable knowledge in many different fields is required. Over the relatively young history of HCI, researchers and developers in the field have accumulated and established basic principles for good HCI design in hopes of achieving some of the main objectives (as a whole) that were laid out in the previous section. These HCI principles are general, fundamental, and commonsensical, applicable to almost any HCI design situation. Here, we provide a short review of the main HCI principles.

1.2.1 "Know Thy User"

The foremost creed in HCI is to devise interaction and interfaces around the target users. This overall concept was well captured by the phrase, "Know thy user," coined by Hansen [1] in 1971, even though the so-called user-centered design approach has become a buzzword only in recent years. This principle simply states that the interaction and interface should cater to the needs and capabilities of the target user of the system in design. However, as easy as this sounds, it is more often the case that the HCI designers and implementers proceed without a full understanding of the user, for example, by just guessing and pretending to know and be able to predict how the representative user might respond to one's design. Ideally, comprehensive information (e.g., age, gender, education level, social status, computing experience, cultural background) about the representative target user should be collected and analyzed to determine their probable preferences, tendencies, capabilities (physical and mental), and skill levels. Such information can be used to properly model interaction and pick the right interface solution for the target users.

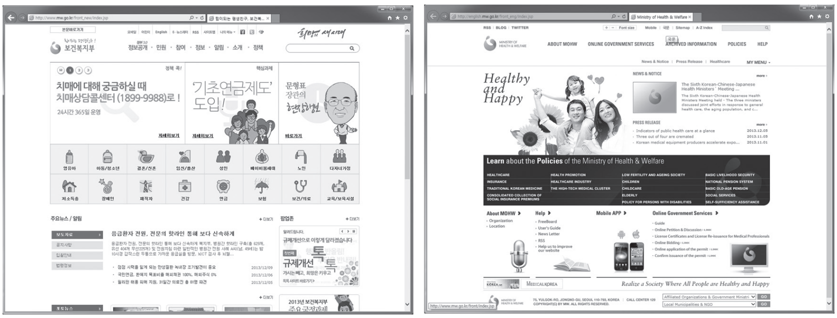
Consider a situation where a developer is working to change an interface, supposedly to achieve higher usability. However, we might need to remember that while young adults are extremely adept at and open to adopting new interfaces, older generations are much less so. Here is another example. Males are generally known to be better than females in terms of spatial ability and, as such, one might consider such a fact in employing three-dimensional (3-D) user interfaces. However, other studies point to females majoring in engineering and science to possess an equivalent level of spatial ability as their male counterparts [2]. So sometimes, conventional wisdom alone may not be sufficient to warrant proper interface design. These examples illustrate that there are a great many aspects that need to be considered in this regard. If a direct field study is not feasible, an experienced and humble HCI designer will at least try to leverage the vast knowledge available from cognitive psychology, ergonomics, and anthropomorphic data to assess the capabilities and characteristics of the target user group. Figure 1.4 shows examples of user-centered designs of web pages for kids and the elderly.

A related (or perhaps even opposing) notion to the user-centered design is the concept of "universal usability," which roughly promotes "humane" interfaces that cater to a wide (rather than a specific) range of users, i.e., across age groups, skill levels, cultural backgrounds, and disability levels. Such a notion has become almost required in our advanced multicultural societies. However, as wonderful as it sounds, it is generally very difficult to achieve this with a single interface.

(a) (b)

Figure 1.4 Examples of user-centered designs of web pages for (a) kids (courtesy of Junior Naver, http://jr.naver.com), and (b) the elderly (courtesy of SilverNet News, http://www.silvernews.or.kr).

Usually, universal usability is achieved by justifying the investment required to build separate interfaces for distinct user groups. For example, in advanced countries, many government web pages are now legally required to provide interfaces in different languages and for color-blind and visually challenged users (Figure 1.5). Many interactive systems provide both menu-driven commands for novices and keyboard-based hot keys for experts (Figure 1.6).



(a) (b)

Figure 1.5 Two different interfaces to achieve universal usability (one in Korean and the other in English). (From the Korean Ministry of Health and Welfare, http://english.mw.go.kr/front_eng/index.jsp.)

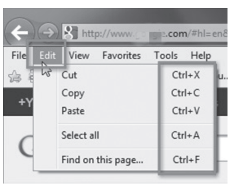


Figure 1.6 An interface providing both menus (for novice users) and hot keys (for expert users).

1.2.2 Understand the Task

Another almost-commonsensical principle is to base HCI design on the understanding of the task. The term *task* refers to the job to be accomplished by the user through the use of the interactive system. In fact, understanding the task at hand is closely related to the interaction modeling and user analysis. It really boils down to identifying the sequence and structure of subtasks at an abstraction level appropriate for the typical user within the larger application context. Take the subtask (for a larger application) for "changing the Wi-Fi connection access point" for a smartphone. For an expert user experienced in computer networks, the task might be modeled with detailed steps, asking the user to select from a pool of available nearby access points based on their characteristics such as the signal strength, bandwidth, security level, and so forth. On the other hand, for a casual user, the subtask might only involve entering a password for the automatically selected access point (Figure 1.7).

Note again that the task (or, equivalently, the interaction) model must ideally come from the user. Different users will have different mental models of the task at hand, and this must be reflected in the structure of the interface to simplify implementation for all users. We will study the process of task/interaction modeling in Chapter 2 in more detail. However, it is not always the case that modeling interaction after the user is the most efficient approach. One must remember that humans are very adaptive and, as such, a nonuser-based

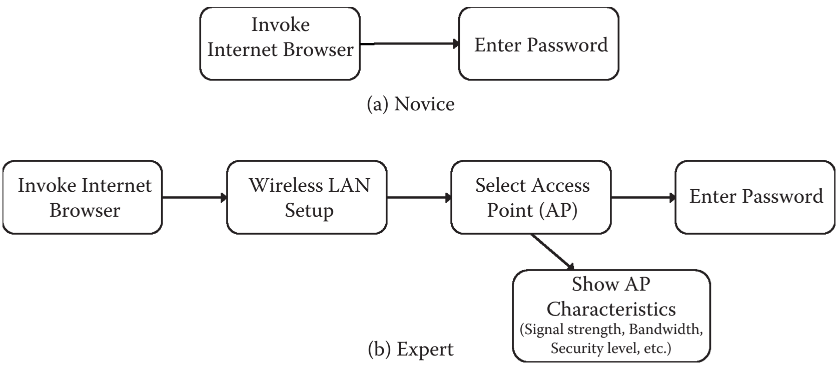


Figure 1.7 Two interaction models at different levels of detail for the task of "connecting to the Internet from a smartphone," depending on the user type.

task/interaction model may sometimes be developed based solely on the general human capacity.

1.2.3 Reduce Memory Load

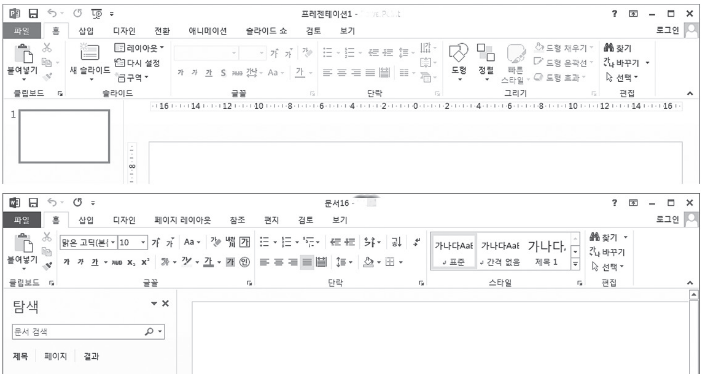
Designing interaction with as little memory load as possible is a principle that also has a theoretical basis. Humans are certainly more efficient in carrying out tasks that require less memory burden, long or short term. Keeping the user's short-term memory load light is of particular importance with regard to the interface's role as a quick and easy guidance to the completion of the task. The capacity of the human's short-term memory (STM) is about 5–9 chunks of information (or items meaningful with respect to the task), famously known as the "magic number" [3]. Light memory burden also leads to less erroneous behavior. This fact is well applied to interface design, for instance, in keeping the number of menu items or depth to less than this amount to maintain good user awareness of the ongoing task or in providing reminders and status information continuously throughout the interaction (Figure 1.8).

1.2.4 Strive for Consistency

In the longer term, one way to unburden the memory load is to keep consistency [4]. This applies to (a) both within an application and across

Figure 1.8 Interfaces designed for minimal short-term memory: (a) a menu system with fewer than 10 items (left) and (b) categorization by colors, areas, icons, and labels. Badges are used to display status information such as the current weather (see circled portions) and number of unread mails as a constant reminder. (From Microsoft®, Microsoft Metro interface, http://www.microsoft.com.)

(a)



(b)

Figure 1.9 (a) A consistent look of the interface within an application (a game called Subway Surfers, https://play.google.com/store/apps/details?id=com.kiloo.subwaysurf) and (b) a consistent interface between Microsoft PowerPoint and Word.

different applications and (b) both the interaction model and interface implementation. For instance, the user is likely to get confused and exhibit erroneous responses if the same subtask is involved, at different times, for different interaction steps or interface methods. Note that the exact same subtasks may appear across different applications as well. Aside from being able to remember what to do, consistency and familiarity also lead to higher acceptability and preference. One way the Microsoft Windows®–based applications maintain their competitiveness is by promoting consistent and familiar interfaces (Figure 1.9).

1.2.5 Remind Users and Refresh Their Memory

Any significant task will involve the use of memory, so another good strategy is to employ interfaces that give continuous reminders of important information and thereby refresh the user's memory. The human memory dissipates information quite quickly, and this is especially true when switching tasks in multitasking situations (which is a very prevalent form of interaction these days). In fact, research shows that our brain internally rehearses information encoding during multitasking [5]. Even a single task may proceed in different contextual spans. For instance, in an online shopping application, one might cycle through the entry of different types of information: item selection, delivery options, address, credit card number, number of items, etc. To maintain the user's awareness of the situation and further elicit correct responses, informative, momentary, or continuous feedback will refresh the user's memory and help the user complete the task easily.

One particular type of informative feedback (aside from the current status) is the reaffirmation of the user action to signal the closure of a larger process [6]. An example might be not only explicitly confirming the safe receipt of a credit card number, but also signaling that the book order is complete (and "closed"). Such a closure will bring satisfaction by matching the user's mental picture of the ongoing interactive process (Figure 1.10).

1.2.6 Prevent Errors/Reversal of Action

While supporting a quick completion of the task is important, error-free operation is equally important [6]. As such, the interaction and interface should be designed to avoid confusion and mental overload. Naturally, all of the aforementioned principles apply here. In addition, one effective technique is to present or solicit only the relevant information/action as required at a given time. Inactive menu items are good examples of such a technique. Also, having the system require the user to choose from possibilities (e.g., menu system) is

Figure 1.10 Reaffirming the user's action (i.e., credit card number correctly and securely entered) and a larger interactive process (i.e., the book purchase is complete).

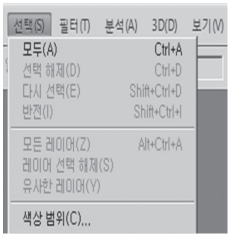


Figure 1.11 Preventing errors by presenting only the relevant information at a given time (inactive menu items) and making selections rather than enforcing recall or full manual input specification.

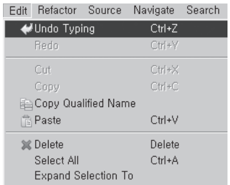


Figure 1.12 Making the user comfortable by always allowing an easy reversal of action.

generally a safer approach than to rely on recall (e.g., direct text input) (Figure 1.11).

Despite employing some of the principles and techniques described here, there is always a chance that the user will make mistakes. Thus, a very obvious but easy-to-forget feature is to allow an easy reversal of action. This puts the user into a comfortable state and increases user satisfaction as well (Figure 1.12).

1.2.7 Naturalness

The final major HCI principle is to favor "natural" interaction and interfaces. *Naturalness* refers to a trait that is reflective of various operations in our everyday life. For instance, a perfect HCI may one day be realized when a natural language–based conversational interface is possible, because this is the prevalent way that humans communicate. However, it can be tricky to directly translate real-life styles and modes of interaction to and for interaction with a computer. Perhaps a better approach is to model interaction "metaphorically" to the real-life counterpart, extracting the conceptual and abstract essence of the task. For instance, Figure 1.13 shows an interface called the ARCBall [7] for rotating an object in 3-D space using a mouse (2-D device). In

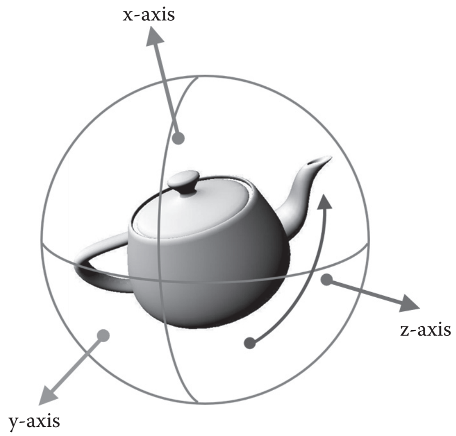


Figure 1.13 ARCBall: 3-D object rotation by using the sphere metaphor. It is also very intuitive with a high level of affordance. (From Shoemake, K., *Graphics Interface,* 92, 151–156, 1992 [7].)

order to rotate, the selected object is overlaid with and enclosed by a transparent sphere, and the user drags on the surface of the sphere to rotate the object inside. One might consider this rotation technique to be metaphoric because it abstracts the interaction object into the shape of a sphere, the most rotational object we know.

A natural or metaphoric interface (assuming that the metaphor is not contrived) will also have *affordance*, a property (or additional cues) that appeals to our innate perception and cognition, thus making it so intuitive that the interface would require almost no learning [4]. In the example of the ARCBall, the spherical shape of the rotator GUI may be regarded to exhibit a high level of affordance, requiring no explanation as to how to rotate the object.

1.3 Summary

In this chapter, I have introduced the field of HCI, namely its objective and importance. We also have reviewed some of the main high-level principles of HCI and presented some relevant examples. These principles are often based on or are just manifestations of deeper theories in cognitive science and ergonomics. However, they are transformed into more detailed and directly usable guidelines when put into actual practice for the specific purpose of designing an effective interface. In Chapters 2 and 3, we take a look at these guidelines and theories, respectively, as they are essential knowledge required for the HCI design process, which we will begin to address in Chapter 4.

References

1. Hansen, Wilfred J. 1971. User engineering principles for interactive systems. In *AFIPS '71 (Fall) Proceedings of the November 16–18, 1971, Fall Joint Computer Conference*, 523–32. New York: ACM.
2. Eisenberg, Theodore A., and Robert L. McGinty. 1977. On spatial visualization in college students. *Journal of Psychology* 95 (1): 99–104.
3. Miller, George A. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review* 63 (2): 81–97.
4. Norman, Donald A. 2002. *The design of everyday things*. New York: First Basic Paperback. (Orig. pub. 1988 as *Psychology of everyday things*. New York: Basic Books.)
5. Salvucci, Dario D., and Neils A. Taatgen. 2010. *The multitasking mind*. Oxford, UK: Oxford University Press.
6. Shneiderman, Ben, and Catherine Plaisant. 2004. *Designing the user interface: Strategies for effective human–computer interaction*. 4th ed. Boston: Addison Wesley.
7. Shoemake, Ken. 1992. ARCBALL: A user interface for specifying three-dimensional orientation using a mouse. *Graphics Interface* 92:151–56.