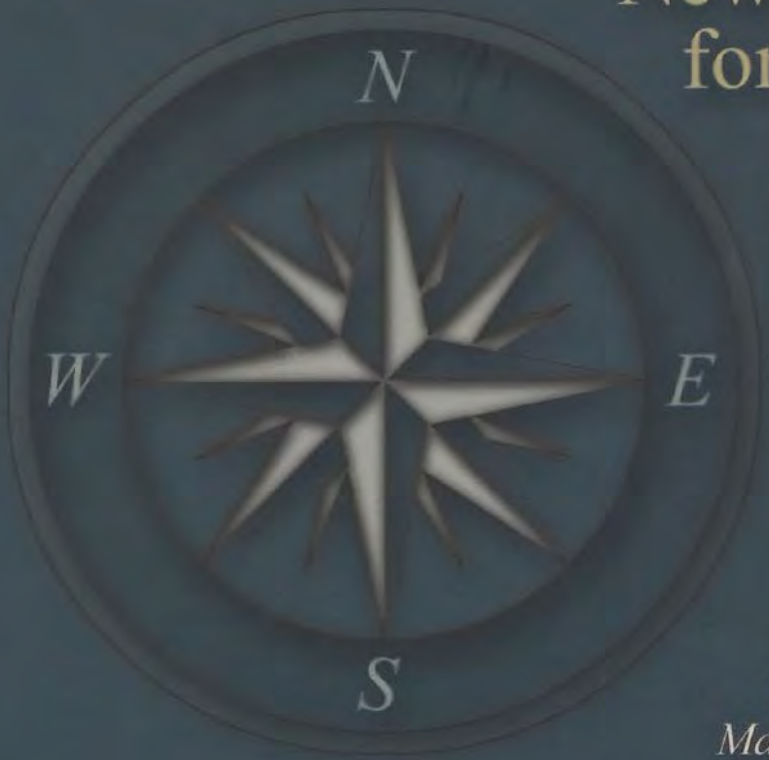
THE HUMANE INTERFACE

New Directions for Designing Interactive Systems



The creator of the Macintosh goes beyond today's graphic user interfaces to show how the Web, computers, and information appliances can be made easier to learn and use.

Jef Raskin

Petitioner has added romanettes to pages (i)-(v) and page numbers to 156(a)-156(p) and 234-236. Otherwise, it leaves the original page numbering.

MICROSOFT EXHIBIT 1021

THE HUMANE INTERFACE

THE HUMANE INTERFACE

New Directions for Designing Interactive Systems

JEF RASKIN



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

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book and we were aware of a trademark claim, the designations have been printed in initial capital letters or all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers discounts on this book when ordered in quantity for special sales. For more information please contact:

Pearson Education Corporate Sales Division
One Lake Street
Upper Saddle River, NJ 07458
(800) 382-3419

corpsales@pearsontechgroup.com

Visit AW on the Web at *www.awl.com/cseng/*

Library of Congress Cataloging-in Publication Data

Raskin, Jef.
The humane interface: new directions for designing interactive systems / Jef Raskin.
p. cm.
Includes bibliographical references and index.
ISBN 0-2-1-37937-6
1. Human-computer interaction. 2. User interfaces (Computer systems) I. Title.
QA76.9.H85 R37 2000
004'.01'9–dc21 00–021300

Copyright © 2000 by ACM Press, A Division of the Association for Computing Machinery, Inc. (ACM).

All rights reserved. No part of the publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, otherwise, without the prior consent of the publisher. Printed in the United States of America. Published simultaneously in Canada.

ISBN 0-201-37937-6
Text printed on recycled paper.
3 4 5 6 7 8 9—CRS—04030201
Third printing, January 2001

We are oppressed by our electronic servants.
This book is dedicated to our liberation.

CONTENTS

PREFACE xi

ACKNOWLEDGMENTS xiii

INTRODUCTION: *The Importance of Fundamentals* xvii

ONE ***Background*** 1

1-1 Interface Definition 2

1-2 Keep the Simple Simple 2

1-3 Human-Centered Design and User-Centered Design 3

1-4 Tools That Do Not Facilitate Design Innovation 4

1-5 Interface Design in the Design Cycle 5

1-6 Definition of a Humane Interface 6

TWO ***Cognetics and the Locus of Attention*** 9

2-1 Ergonomics and Cognetics: What We Can and Cannot Do 9

2-2 Cognitive Conscious and Cognitive Unconscious 11

2-3 Locus of Attention 17

2-3-1 Formation of Habits 18

2-3-2 Execution of Simultaneous Tasks 20

2-3-3 Singularity of the Locus of Attention 24

2-3-4 Origins of the Locus of Attention 27

2-3-5 Exploitation of the Single Locus of Attention 29

2-3-6 Resumption of Interrupted Work 31

THREE *Meanings, Modes, Monotony, and Myths* 33
3-1 Nomenclature and Notations 33
3-2 Modes 37
3-2-1 Definition of Modes 42
3-2-2 Modes, User-Preference Settings, and Temporary Modes 47
3-2-3 Modes and Quasimodes 55
3-3 Noun-Verb versus Verb-Noun Constructions 59
3-4 Visibility and Affordances 62
3-5 Monotony 66
3-6 Myth of the Beginner-Expert Dichotomy 68

FOUR *Quantification* 71
4-1 Quantitative Analyses of Interfaces 71
4-2 GOMS Keystroke-Level Model 72
4-2-1 Interface Timings 73
4-2-2 GOMS Calculations 76
4-2-3 GOMS Calculation Examples 77
4-2-3-1 Hal's Interface: Solution 1, Dialog Box 78
4-2-3-2 Hal's Interface: Solution 2, GUI 80
4-3 Measurement of Interface Efficiency 83
4-3-1 Efficiency of Hal's Interfaces 87
4-3-2 Other Solutions for Hal's Interface 90
4-4 Fitts' Law and Hick's Law 93
4-4-1 Fitts' Law 93
4-4-2 Hick's Law 96

FIVE *Unification* 99
5-1 Uniformity and Elementary Actions 101
5-2 Elementary Actions Cataloged 103
5-2-1 Highlighting, Indication, and Selection 105
5-2-2 Commands 109
5-2-3 Display States of Objects 115
5-3 File Names and Structures 117
5-4 String Searches and Find Mechanisms 124

5-4-1 Search-Pattern Delimiters 127
5-4-2 Units of Interaction 129
5-5 Cursor Design and a Strategy for Making Selections 133
5-6 Cursor Position and LEAP 136
5-7 Applications Abolished 139
5-8 Commands and Transformers 143

SIX *Navigation and Other Aspects of Humane Interfaces* 149
6-1 Intuitive and Natural Interfaces 150
6-2 Better Navigation: ZoomWorld 152
6-3 Icons 168
6-4 Techniques and Help Facilities in Humane Interfaces 174
6-4-1 Cut and Paste 177
6-4-2 Messages to the User 178
6-4-3 Simplified Sign-Ons 183
6-4-4 Time Delays and Keyboard Tricks 184
6-5 Letter from a User 187

SEVEN *Interface Issues Outside the User Interface* 191
7-1 More Humane Programming Language Environments 192
7-1-1 System and Development Environment 192
7-1-2 Importance of Documentation in Program Creation 194
7-2 Modes and Cables 195
7-3 Ethics and Management of Interface Design 198

EIGHT *Conclusion* 205
APPENDIX A: *The One-Button Mouse History* 207
APPENDIX B: *SwyftCard Interface Theory of Operation* 211
REFERENCES 215
INDEX 221

PREFACE

I don't know what percentage of our time on any computer-based project is spent getting the equipment to work right, but if I had a gardener who spent as much of the time fixing her shovel as we spend fooling with our computers, I'd buy her a good shovel. At least you can *buy a good shovel.*

—Erasmus Smums

Creating an interface is much like building a house: If you don't get the foundations right, no amount of decorating can fix the resulting structure. *The Humane Interface* reexamines the cognitive foundations of human-machine interaction to elucidate a crucial aspect of why interface designs succeed or fail. One finding is that present-day graphical user interfaces, such as those of the Windows and Macintosh operating systems, which are based on an architecture of operating system plus application programs, are inherently flawed. A different approach is required if computers are to become more pleasant and if users are to become more productive. This book describes some of the fundamental flaws in user interfaces and describes solutions for overcoming those flaws.

Although the techniques covered in *The Humane Interface* apply to a wide range of products—including web sites, application software, handheld personal data managers and other information appliances, and operating systems—this book does not present a survey of the field of human-machine

interface design. Rather, this book strikes out in new directions while also reviewing those established parts of interface design that are needed in the development of the new material.

If we are to surmount the inherent problems in present human-machine interfaces, it is necessary that we understand the teachings of this volume; it is not, however, *sufficient*. Many important aspects of interaction design are not included here because they are well covered in the literature. This book is intended to complement existing—or to be a prolegomenon to future—treatments of interface design.

The audience for this book includes

- Web designers and managers who want to give their sites a special ease of use that appeals to audiences and helps customers to find the information they need and to buy what they want
- Product designers and product managers who need to be able to create web sites or products that will win and retain customers by offering ease of use and ready learnability and by having a first-rate feature set
- Corporate managers who correctly insist on making products that have low maintenance and that reduce the need for help desks
- Programmers who do interface design—and who doesn't these days?—and who want to understand more of the factors that make their work most useful
- IT (information technology) managers who need to know which interface features will minimize their costs for training and which interface designs are likely to aid productivity
- Consumers who want to learn what to hope for in terms of pleasant interaction with computers and other equipment, and what is wrong with the way today's software is designed
- Computer science and cognitive psychology students who want to understand what lies behind heuristics of interface design

Finally, this book is for human-machine interface researchers, who may find that they will never again be able to view interfaces in quite the same way they did before reading *The Humane Interface*.

ACKNOWLEDGMENTS

Friendly counsel cuts off many foes.

—*William Shakespeare (*King Henry VI, *Act III, Scene 1)*

To list those who have helped is difficult, because they are so numerous, and the debt is so enormous. Many friends, colleagues, relations, reviewers, and some generous strangers I know only via the Internet have contributed ideas, critiques, suggestions, and detailed editorial work. Please forgive (and inform) me if you've helped and I've left you out or gotten your name or title wrong.

Thanks to the groups at Addison Wesley Longman, whether editors, designers, in PR, marketing, or whatever, all of whom seem to have been chosen not only for their competence, but also for their friendliness and forbearance. On the other hand, the anonymous reviewers they chose were merciless, for which I am also grateful.

Among the following list are friends, acquaintances, colleagues, my brother, my son's horn teacher, a fellow model airplane enthusiast—a seemingly unlikely lot. Only a few are experts in human-computer interface design, but all have read my manuscript and made essential contributions to the book or have contributed over the years to its concepts: David Alzofon (who also drew Quasimodo), Bill Atkinson, Thomas Atwood, Paul Baker, Jerry Barenholtz, John Bumgarner, David Caulkins, William Buxton, Ph.D., Renwick Curry, Ph.D., Robert Fowles, Josh Garrett, Ph.D., Jean-François

Groff, Scott Kim, Ph. D., Kathleen Mandis, Pamela Martin, Troy May, Miriam Meisler, Ph.D., Douglas McKenna, Michael S. Miller, David Moshal, M.D., Andrew Nielsen, Jakob Nielsen, Julie Ososke, Ian Patterson, Michael Raskin, Ph.D., Erasmus Smums, Spider Robinson, Minoru Taoyama, Shay Telfer, Yesso Tekerian, Bruce Tognazzini, David Wing, Terry Winograd, Ph.D., the local chapter (BayCHI) of the ACM's Special Interest Group in Computer-Human Interaction, which has let me preach and debate my theses, and the students at the Center for Computer Research in Music and Acoustics at Stanford University and its director, John Chowning.

I am lucky to have a literate as well as a loving wife, Linda Blum, R.N., who has never cared that writing a technical book is no way to support a family, so long as it was a worthy endeavor. Her attention to the ideas, direction, and details of this book have improved many a page. I can take no credit in choosing my parents, but they deserve much credit for teaching me to value people over things and to relish the arts as well as the sciences, choices that lead directly to this work. My son Aza contributed far beyond what you'd expect from someone of his youth, including ideas, editing, and hard work on the illustrations. He and his sisters were amazingly patient with me as I wrote. Especially important in my life is L. Roland Genise, my best teacher, who, during high school, gave me the twin gifts of intellectual self-confidence and a love of mathematics. Among those who have shared warm friendship, philosophy, and music, and who have been devastating editors of my earlier works, I am lucky to be able to name Brian Howard and Douglas Wyatt. I have disagreed with a few details of the writings of Dr. Donald Norman in this book, but these are minor points as I regard his work as essential reading in the field; without his critiques and teachings this book would not have come about. I am grateful to Bill Verplank, a quiet and agreeable sort whose comments are delivered so gently that you don't realize the rug's been pulled out until you hit the floor. His was one of the voices that convinced me to completely change the tone and orientation of the book, for the better. Another who hammered this book into shape was Lyn Dupré, a fierce and nitpicking professional editor. She wrote *BUGS in Writing,* which you should read. Many concepts, a few of which are cited in the text, came from or were polished during discussions and work done with my friend James Winter, M.D., Ph.D. The delightfully acerbic computer scientist Dick Karpinski, who styles himself aptly as the world's largest leprechaun, has been helpful in manifold ways, whether expounding on a technical point, introducing me to a key person or book, or dropping by with dim sum. And, whom I've saved for last, there's Peter Gordon, a man of wisdom, persistence, and (especially) patience, who was my advocate at

Addison-Wesley. Our correspondence must never be made public as it would reveal a penchant for extended word play and awful puns that would forever besmirch both our names, but which lightened the burden of endless details that must be attended to in putting together even so slim a book as this.

Thanks to Agfa Corporation for supplying the digital camera used in creating some of the illustrations.

Thanks, also, to the following readers, who first offered me particular corrections, or suggested changes, that have now been incorporated into the book: Eric Blossom, Jon Bondy, Paul Cubbage, Peter Jones, G. A. Michael, Cam Mitchner, Rich Morin, Martin Portman, and Elisabeth Riba. The author is also grateful to Rich Morin for supporting the www.jefraskin.com Web site.

INTRODUCTION

The Importance of Fundamentals

One person, one computer.

—Apple Computer slogan

Imagine that you have just boarded an airliner resplendent in its livery: fitted with a wide choice of video and audio for every glove-soft, leather-covered, oversized seat; its galleys provisioned with fine food and drink. You take your seat and look out the freshly cleaned large window. With a sigh of anticipation for a particularly pleasant flight, you reach into a small compartment in front of you to see what is there. A not-too-small bottle of your favorite beverage comes to hand first, followed by a little booklet about this remarkable airliner.

As the flight attendants swing the doors shut and you settle in, you read the booklet. You learn that the aircraft is the work of some of the finest interior designers from all over the world, that chefs from five-star restaurants have created the menu and personally prepared the dishes, and that because the internationally acclaimed artists who designed the exterior made the craft look so much faster than any other airliner, there had been no

need to include professional aeronautical engineers in the aircraft's development team.

In the small print used for legalese, the booklet warns that the ride tends to be bumpy, even in the absence of turbulence, and that the plane crashes regularly. However, the booklet promises, until any of those events occur, you will be comfortable and well entertained.

Suddenly, the latching of the doors seems menacing instead of promising. Your equanimity is gone; you are trapped. This flight, the only one to your destination, is doomed, and you are on it. At this point, you'd rather be sitting on a hard seat, no drink in your hand and no window by your side but in an aircraft blessed by great engineering.

This absurd situation closely parallels the nature of most human-machine interfaces today. Our computers and cellular telephones have the latest chips and electronics; today's operating systems are a feast for the eyes, with glorious colorful backgrounds and three-dimensional tromp l'oeil effects. You click on a button, and lo! it appears to move most realistically; you hear a digital stereophonic, full-fidelity rendering of a switch clicking, and your ears are enchanted by a resonant harp glissando as a window opens before you.

But when you start to use the system, it begins to poke you with uncomfortable corners of unexpected behavior. You cannot find the command you want among the thousands that the system provides. Simple, routine tasks take forever to do. The program you bought last year does not run under the improved operating system, so you have to buy an upgrade. And, of course, the system crashes regularly.

Some engineering fundamentals that are not widely known underlie good interfaces. And why should those fundamentals be studied? How interfaces should look and work seems well established: They've been incrementally improved for two decades now; we have interface guidelines published by the major software producers to ensure future compliance; development tools allow us to put together interfaces quickly such that they look just like other modern interfaces—just as my mythical airliner *looked just like* a well-designed, safe, and comfortable flying machine.

But consider what these interfaces fail to do for us. When you want to set down an idea, you should be able to go to your computer or information appliance and just start typing: no booting, no opening the word processor, no file names, no operating system. (My definition of an operating system: What you have to hassle with before you get to hassle with the application.) You should not have to learn an entire new application to perform what you know to be only a few simple tasks that you'd like to add to

the repertory of your system. Regrettably, the design of interfaces has taken a wrong turn, leading to a level of difficulty unjustified by technological or logical necessity.

Millions of us have a love-hate relationship with information technology: We can't live without it, but at the same time, we find it difficult to live with. The problem of making technology comfortable does have solutions, but we can't buy them now; they will be available to us only if we drop a lot of the baggage of the past. The customary, desktop-based, applications-oriented interfaces turn out to be part of the problem. This book offers some alternatives. After all, computer problems are not like the weather: We *can* do something about them.

Given the prevalence of the Internet and the obvious importance of products that facilitate group interaction, it may seem odd that *The Humane Interface* concentrates on single-user interface design. One reason is that the design of single-user interfaces is not a solved problem. The primary reason is that the quality of any interface is ultimately determined by the quality of the interaction between one human and one system—between you and it. *If a system's one-on-one interaction with its human user is not pleasant and facile, the resulting deficiency will poison the performance of the entire system, however fine that system might be in its other aspects.*

ONE

Background

Nothing is more impossible than to write a book that wins every reader's approval.

—Miguel de Cervantes

This chapter explains that the nature of interfaces and of interface design is widely misunderstood. There is more to interfaces than windows, icons, pull-down menus, and mice. The need to take interface design into account early in the design cycle is sometimes overlooked. Another factor often overlooked is the commonality in the cognitive equipment handed out to all of us. We must take into account common factors before we can deal with the differences among individual humans. Unfortunately, the tools widely available for interface construction are inadequate to this task.

I reject the idea that computers are difficult to use because what we do with them has become irretrievably complicated. No matter how complex the task a product is trying to accomplish, the simple parts of the task should remain simple. This chapter ends with a definition of a humane interface.

1-1 Interface Definition

Call our USA number above and test your stamina against the incredible frustration provided by our voice mail system.

—Note at the bottom of an advertisement for Simple-brand shoes

In this book, I usually shorten *human-machine interface* or *human-computer interface* to *user interface* or simply *interface*. Many people assume that the term *user interface* refers specifically to today's graphical user interfaces (GUIs), complete with windows and mouse-driven menus. For example, an article in *Mobile Office* magazine said, "Before too long, you may not have to worry about an interface at all: You may find yourself simply speaking to your computer." As I pointed out in response, a voice-controlled system may have no windows, but neither do telephone voice-response systems, and they often have hellaciously bad interfaces. The way that you accomplish tasks with a product—what you do and how it responds—that's the interface. (See also Raskin 1993.)

1-2 Keep the Simple Simple

Technology is a queer thing. It brings you great gifts with one hand, and it stabs you in the back with the other.

—C.P. Snow (quoted in Jarman 1992)

Despite a burgeoning population of interface designers, few consumers claim that new products, such as an electric, four-button wristwatch, are easier to use than they were a few decades ago. If you point out to me that watches, like computers, now have much greater functionality (true) and that, in consequence, the interfaces have had to become more complex (debatable), I respond by pointing out that even the simple tasks that I used to do easily have become mired in complexity. Complex tasks may require complex interfaces, but that is no excuse for complicating simple tasks. Compare the difficulty of setting the time on your electronic, four-button wristwatch to that of completing the *same* task on a mechanical model. No matter how complex the overall system, there is no excuse for not keeping simple tasks simple.

Of the many absurdities foisted on us by inept interface design, perhaps it is the complication of what should be simple that gives comic strips and comedians the most opportunities. In the movie *City Slickers*, three chums are driving a herd of cattle. Billy Crystal's character tries unsuccessfully—apparently for hours—to explain how to use a VCR to record a show

on one channel while watching another. When the friends finally explode in exasperation at the lengthy explanation, Crystal's character cheerfully agrees to drop the subject and offers instead an explanation of how to set the clock on the VCR. This offer enrages his cronies and cracks up the audience. The humor arises from the dissonance between the simplicity of the task and the difficulty of the interface: If the vertical front of a VCR had labeled buttons situated above and below the digits of a clock as shown in Figure 1.1, few people would have any trouble setting the clock.

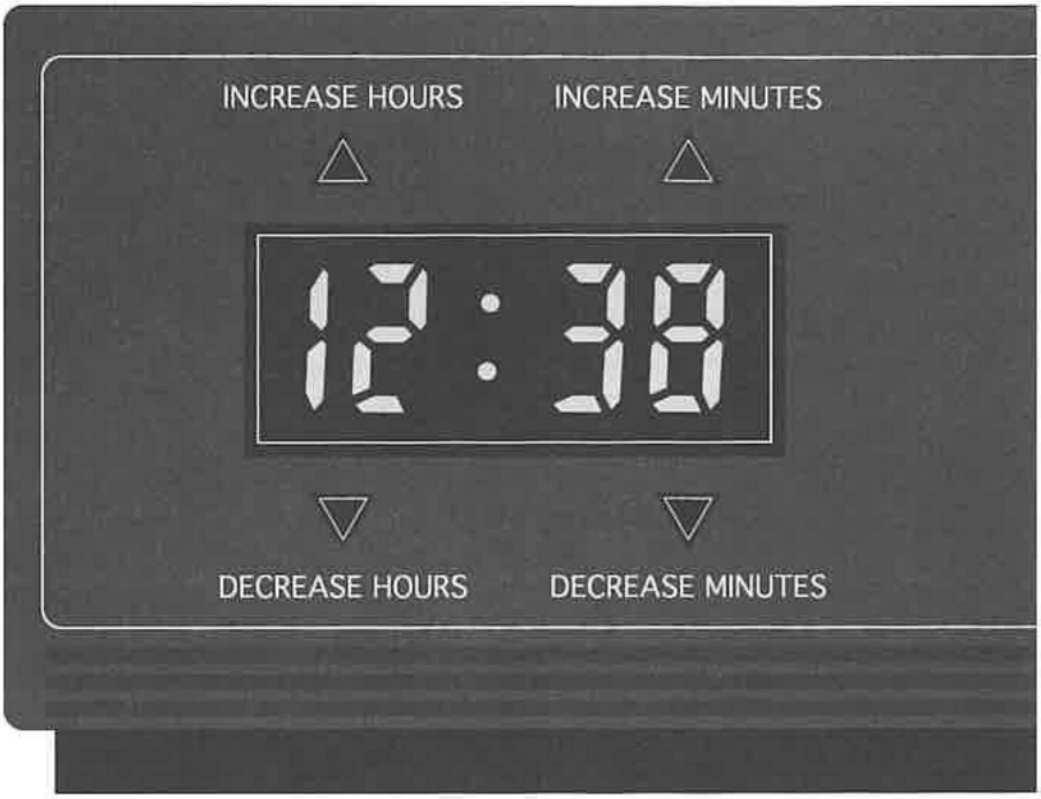


Figure 1.1. An easy-to-set digital clock on a VCR. An even better design would be a clock that set itself based on broadcast time signals.

1-3 Human-Centered Design and User-Centered Design

We have overcomplicated the software and forgotten the primary objective.

—Jim and Sandra Sundfors

Not only interface designers but also managers in the electronics and computer industries understand the need for user- or customer-centered design. The first step in meeting this need is to get to know your users, but

in commercial practice, getting to know the users usually consists of listening to task domain experts. Domain experts often know the parameters and details of the problem to be solved, but their formal expertise does not usually extend to questions of human psychology. Although users' task-related needs differ, your user population shares many common mental attributes. Before exploring the application or even working to accommodate differences among individuals, interface designers can minimize their work by exploiting what is common to all humans with regard to interface-design requirements. After that is accomplished, the interface designers can accommodate the differences across individuals and groups, and, finally, they can satisfy the varying requirements of their tasks. That crucial first step—making sure that the interface design accords with universal psychological facts—is customarily omitted in the design process. For the most part, interface designers have abdicated that responsibility to "industry standards." All current popular interfaces have been built on underpinnings that flout what we know about human thought and behavior. For example, files with file names are a nearly universal feature in computer systems, yet we all have trouble remembering what file name we used to store a document six months ago. (A solution to this problem will be discussed in Section 5-3.) We want comprehensible software that demonstrates, by its impeccable behavior, that its designers were focused more on usability than on glitz.

1-4 Tools That Do Not Facilitate Design Innovation

Creating fine interfaces can require undertaking intensive and expensive work. Interface-building tools, such as Visual Basic or Visual C++, are marketed as lowering development costs and speeding implementation. In spite of their utility, these tools will not be mentioned often in this book; they enshrine current paradigms and thus unduly limit the scope of what you can do. Similarly, the Macintosh or Windows interface guidelines and a portion of the heuristics presented by books on interface design occasionally give advice that is demonstrably incorrect—often due to the company's need to maintain compatibility with earlier versions of the interfaces and to the misperception that users will inevitably revolt if old, familiar interface methods are abandoned. Where real improvement can be achieved by making major changes, the interface designer must balance the legitimate use of familiar paradigms, which ease the learning process, against the enhanced usability that can be attained by abandoning them. In a situation of rapid turnover of personnel or the customer base, familiarity might be the better design choice. *Where most of the users' time will be spent in routine operation of the*

product and where learning is only a small part of the picture, designing for productivity—even if that requires retraining—is often the correct decision.

1-5 Interface Design in the Design Cycle

Project methodologies often fail to take full advantage of what is known about interface design. This omission can be the result of bringing in interface designers long after much of the opportunity for improving the quality of interaction between user and product has been lost. Designs are most flexible at their inception. If interface designers are consulted only after the software has been designed and the software tools chosen or when the software is nearly complete, the correct advice—namely, to start over—is unacceptable. The budget and most of the schedule have already been expended, and the option of throwing away much or all of the design and the completed code makes the project managers look bad. Even so, as recent a book on project management as the *UML Toolkit* (Eriksson and Magnus 1998) fails to recognize that the interface has to be part of the requirements analysis, which is Eriksson and Magnus's first phase of project development. Contrary to their suggestion, interface design cannot be postponed until the technical design phase (their third phase). *Once the product's task is known, design the interface first; then implement to the interface design.* This is an iterative process: The task definition will change as the interface is designed, and the implementation will be influenced by the task definition and the interative design as well. Flexibility on all fronts is needed. The place to start the implementation is to list exactly what the user will do to achieve his or her goals and how the system will respond to each user action.

Users do not care about what is inside the box, as long as the box does what they need done. What processor was used, whether the programming language was object oriented or multithreaded, or whether it was the proud possessor of some other popular buzzword does not count. What users want is convenience and results. But all that they see is the interface. *As far as the customer is concerned, the interface* is *the product.*

Your Time Is Sacred; Your Work Is Sacred

I have learned to save my work frequently so that, even in the event of a system crash, I will not lose much time. I use a keyboard-operated save command every time I complete a paragraph or even a few sentences. This command places a copy of my work on the disk, where it is relatively safe from damage in a crash. Every hour or so, I

further back up my work to a nonvolatile memory card that I physically remove from the computer so that the backup is isolated from the computer and is safe, no matter how the computer runs amok; every week, I back up the whole system on an external disk drive. I am not paranoid, just realistic. Yet these elaborate procedures should be unnecessary. *The system should treat all user input as sacred* and —to paraphrase Asimov's first law of robotics, "A robot shall not harm a human, or, through inaction, allow a human to come to harm" (Asimov 1977, p. 44). The first law of interface design should be: *A computer shall not harm your work or, through inaction, allow your work to come to harm.*

While working on this book, and at the suggestion of my editors, I began to use a facility that allows me to accept or to reject suggested changes. After every few decisions, I used the save command. When the system crashed, I was not worried, because of my continual saves. But when I went to find my most recent changes, they were gone, and I had to redo my work. By trying a few experiments, I finally figured out that when the accept-reject feature is in use, the keyboard save command does not operate. No warning is given. I lost more than three hours in work and in experimenting to figure out what had happened, in the hope of preventing a recurrence. Aside from the sheer complexity of today's computer systems, it is vexing details such as this one that prove the need for better interface design.

For a second interface law, you could do worse than to insist on this one: *A computer shall not waste your time or require you to do more work than is strictly necessary.* Section 4-3 introduces a measure of how much work is necessary to complete a task.

1-6 Definition of a Humane Interface

You can have any combination of features the Air Ministry desires, so long as you do not also require that the resulting airplane fly.

—Willy Messerschmidt (preeminent World War II German aircraft designer)

An interface is ***humane*** *if it is responsive to human needs and considerate of human frailties.* If you want to create a humane interface, you must have an understanding of the relevant information on how both humans and

machines operate. In addition, you must cultivate in yourself a sensitivity to the difficulties that people experience. That is not necessarily a simple undertaking. We become accustomed to the ways that products work to the extent that we come to accept their methods as a given, even when their interfaces are unnecessarily complex, confusing, wasteful, and provocative of human error.

Many of us become annoyed, for example, at the amount of time that it takes a computer to start up, or to boot. An advertisement for a computer-based car radio in 1999 assures us that "unlike a home computer, there are no long boot-up times to worry about." A scan through six respected books on the subject of interface design—books that have been written over most of the span of time during which interface design has been a recognized field—finds the topic of booting unmentioned (Shneiderman 1987; Norman 1988; Laurel 1990; Tognazzini 1992; Mayhew 1992; Cooper 1995). I am sure that each of these authors would wholeheartedly agree that reducing or eliminating this delay would improve usability; I have never met a user to whom the delay is not an annoyance. Yet the boot delay is so much a familiar and accepted part of using a computer that it is seldom questioned in the interface-design literature. There has never been any technical necessity for a computer to take more than a few seconds to begin operation when it is turned on. We have slow-booting computers only because many designers and implementers did not assign a high priority to making the interface humane in this regard. In addition, some people believe that the sale of millions of slow-booting computers "proves" that the way these machines now work is just fine.

The annoyance of having to wait for a machine to start up has not been ignored in other product areas. The defunct Apple Newton, the Palm Pilot, and other handheld computers manage to start instantly, and the introduction of a sleep mode—a state in which the computer is using less power than when it is on but from which it can be brought to operational status quickly—on certain computers is a step in the right direction.

Engineering has successfully tackled some more difficult problems. In early television sets, for example, the wait due to the time it took to heat the cathode of the picture tube was nearly a minute. In certain television sets, engineers added a circuit to keep the cathode warm, reducing the wait to reach operating temperature. (Keeping the cathode fully heated would have been wasteful of electricity and would have shortened the life of the tube.) Other engineers designed picture tubes that had cathodes that heated in a few seconds. Either way, the user's needs were satisfied. In the early twentieth century, the failure of the Stanley Steamer, a steam-powered automobile

that apparently was a superior vehicle in other aspects, may have been a result of the 20-minute delay between firing it up and its having sufficient boiler pressure to drive.

That a user should not be kept waiting unnecessarily is an obvious and humane design principle. It is also humane not to hurry a user; the more general principle is: *Users should set the pace of an interaction.*

It does not take much technical knowledge to see, for example, that higher-bandwidth communication channels can hasten downloading of web pages. Other relationships are not so evident. It is useful for designers of a human-machine interface to understand the innards of the technology; otherwise, they have no way to judge the validity of assertions—by, say, programmers or hardware designers—that a given interface detail is not feasible.

TWO

Cognetics and the Locus of Attention

He wept and was nothing content, but it booted not.

— Dominic Mancini, speaking not of a dead computer but of Edward V of England. Occupatione Regni Anglie per Riccardum Tercium *(1483). Quoted in Alison Weir,* The Princes in the Tower *(1992).*

As complicated as computers and other products of our technology may be, it is easier to understand the machine side of the human-machine interface than to come to grips with the far more complex and variable human side. Even so, many—perhaps surprisingly many—human performance factors are independent of a user's age, gender, cultural background, or level of expertise. These properties of human learning and performance are directly applicable to the foundations of any interface design. In particular, that we have one locus of attention affects many aspects of the design of human-machine interfaces.

2-1 Ergonomics and Cognetics: What We Can and Cannot Do

Know thyself.

—Inscription at the Delphic Oracle, from Plutarch, Morals

Use a machine or a tool in accord with its strengths and limitations, and it will do a good job for you. Design a human-machine interface in accord with the abilities and foibles of humankind, and you will help the

user to not only get the job done but also be a happier, more productive person.

Design guidelines for products that interact with us physically are reasonably straightforward. The sizes and capabilities of the human frame and senses have been well cataloged; these studies form the science of **ergonomics**. Chairs, tables, keyboards, and displays can be designed with a high degree of likelihood that they will work reasonably well for their human users, although thorough testing can never be neglected. You would not design a machine that required one person to simultaneously operate two switches 3 meters apart: We all know that humans are not that large. Mayhew (1992, Chapter 12) discusses computer-relevant ergonomics, a topic outside the scope of this book, in her overview of interface design. Ergonomics takes into account the statistical nature of human variability. You might design a car seat to accommodate only 95 percent of the population, even though you know that 5 percent of the potential car purchasers will find the seat uncomfortable. It might be too expensive or mechanically impossible to give the seat the range of adjustment needed to work with the rare 1-meter midget or the rarer still 2.5-meter giant.

For the most part, the machines that our civilizations have built have been mechanical and have interacted principally with our physical selves. Consequently, our physical limitations are relatively well understood. Increasingly, inventions have come to the aid of intellectual rather than physical pursuits. *We must master an ergonomics of the mind if we want to design interfaces that are likely to work well.* As surprising as it may seem, we are often blind to our own mental limits; we must rely on careful experiment and observation to discover the edges of our own mind's abilities.

The study of the applicable, engineering scope of our mental abilities is cognitive engineering, or **cognetics**. Certain cognetic limitations are obvious: You do not expect a typical user to be able to mentally multiply a pair of 30-digit numbers in 5 seconds, and you would not design an interface that requires such an ability. But we are often not aware of other mental limitations that adversely affect our performance when we use human-machine interfaces, although these limitations are inherent in every human. Remarkably, all of the well-known computer interfaces—and many non-computer human-machine interfaces—are designed as though their designers expect us to have cognitive abilities that experiment shows we do not possess. Much of the difficulty that we have with computers and related devices is due to poor interface design rather than to any complexity inherent to the task or to any lack of effort or intelligence on the part of users.

Just as ergonomics takes into account the statistical nature of human variability, so too should cognetics. However, because there has been so little practical use of what is known of the limits of human cognition that are common to all of us, it seems wise to look first at those limits.

Fortunately—if only because our present knowledge of these topics is uncertain—we do not need to examine the structure of the physical brain. We can design successful interfaces based on a pragmatic and empiric view of what the human mind can and cannot do, of how long the mind and body take to do particular tasks, and of the circumstances that increase the likelihood that we will make mistakes.

2-2 Cognitive Conscious and Cognitive Unconscious

Oh Doctor Freud, Oh Doctor Freud, how we wish that you'd been otherwise employed!

—"Doctor Freud," David Lazar, 1951

It can be difficult to deal with the psychologically, philosophically, and historically laden terms—such as *conscious* and *unconscious*—that we use to describe aspects of the way our minds work. In an engineering context, it is useful to work with the more limited concepts of the **cognitive conscious** and the **cognitive unconscious**. More accurate would be the terms *empirical conscious* and *empirical unconscious,* but Kihlstrom's more euphonic coinage has priority (Cohen and Schooler 1997, p. 137). Understanding that we possess these two distinct sets of limited mental abilities and understanding how they work in relationship to human-machine interfaces is as essential to designing interfaces as is knowing the size and the strength of the human hand when we are designing a keyboard.

Here is a first-cut definition: Unconscious mental processes are those of which you are not aware at the time they occur. The cognitive unconscious is not the seething, mythic creature of Freudian psychology but rather a phenomenon that you can demonstrate with a straightforward experiment, which you will be asked to try shortly. Although a growing mountain of books discusses the questions and paradoxes of consciousness, following the approach taken in Bernard J. Baars's *A Cognitive Theory of Consciousness* (1988)—a book that has broadly informed this chapter—helps us to avoid the dilemmas and to deal only with what we can observe directly and produce concretely. As Baars says in his preface, "one time-honored strategy in science is to side-step philosophical issues for a time by focusing on empirically

decidable ones." Cognetics is a practical discipline. Although theoretical studies can be illuminating and eventually may lead to firm and practical results, we avoid them until they do. (Analogously, although a study of human bone growth can inform ergonomics, such a study would fall within the purview of physiology rather than of ergonomics itself.)

Consciousness and Models of the Mind

In following Baars's useful treatment of the cognitive properties of consciousness, it is not necessary also to subscribe to his theory of mind, which uses contemporary digital computer structures as models. I am leery of that approach, especially because throughout the ages, thinkers have borrowed the latest technology as models for understanding humans only to abandon or to limit the models when technology advanced.

Don Norman, a cognitive psychologist who is especially conversant with computers, spoke of the mind in terms of computational units (Norman 1981). J. R. Anderson, beginning in 1976, based his model of mental operations on productions—a tool prominent at the time for describing the syntax of computer languages (Anderson 1993). Although the analogies are usually enlightening, some scholars tend to overapply the metaphor of the moment. I predict a flowering of psychological theories in the next few years with the client/server architecture, the Internet, and the World Wide Web with its hypertext links and browsers appearing as models for how certain aspects of the mind work.

We must monitor this tendency carefully lest we reify the analogies without neurophysiological support. In the seventeenth century, the universe and its denizens were often spoken of in terms of clockwork (Dijksterhuis 1961, p. 495); with our present understanding of both clockwork and organisms, however, this metaphor has become pallid. In the nineteenth century, the metaphor of the steam engine permeated many philosophical musings on human function; now we know that the analogy's value is limited primarily to explaining metabolism, insofar as the metabolic system is a heat engine.

Two popular books that consider the subject of consciousness are Roger Penrose's *The Emperor's New Mind* (Penrose 1989) and Daniel

Dennett's *Consciousness Explained* (Dennett 1991). These treatments, however fascinating to read, unfortunately turn out to be of no use in the design of human-machine interfaces. Without direct evidence of functional parallelism in hand, it is wise to avoid taking computer-brain metaphors too literally.

Because concerns about what is conscious and unconscious usually seem remote from our workaday world, let us tangibly demonstrate their reality in your life by means of a question: What is the final character in your first name? Until you read the previous sentence, you were probably not thinking about this alphabetic character and its relationship to your name. You know—and have long known—what that character is and where in your first name it lies, but you were not paying attention to that knowledge. You were not thinking of it; you were not considering it. Or, to use our preferred terminology, you were not conscious of it. The information was not being accessed, yet you could recover it on demand. We will call that place from which the character was fetched the cognitive unconscious. The cognitive unconscious may not be a physical place, although it must be represented by physical phenomena in the brain. To help dispel the notion that the unconscious or conscious *must* be places in the brain—which is different from the notion that they *might* be—we can think of possible alternative mechanisms. For example, when you became aware, or conscious, of the alphabetic character, this becoming might have been a change of state. Alternatively, perhaps our brains use a pointer mechanism, and a pointer was changed while the memory or thought was left where it was in your brain. It is possible that thoughts and memories could be distributed, like a holographic record of an image.

We could consider other mechanisms or descriptions, but we have no need to do so. For our purposes, all we need is to acknowledge that you were not conscious of the character at one moment and were conscious of it at another moment. I use a positional metaphor from time to time, speaking of a thought as moving from the conscious to the unconscious or back, but you should not think of this verbal convenience as denoting a model of how the brain works. (Research into the physical brain may show such a model to be valid, but that is not our concern here.)

When I use the terms *conscious* and *unconscious* in this book, I will generally intend them as abbreviations for the cognitive conscious and the cognitive unconscious. When you did think about the final character in your first name—a thought triggered by a sentence in this book—that thought

became part of your conscious awareness. This change of state of your thought, from unconscious to conscious, demonstrates that you have at least two forms of knowing. To build a science of cognetics, we must banish solipsism and assume that other humans would make the same observations about their mental processes as you just did about yours.

A stimulus, such as reading a particular portion of this book, can trigger the migration of not only an item of information but also a sensation, a feeling, or another aspect of your memory or knowledge from the unconscious, where it is stored, to the conscious, where you are aware of it. Notice how your clothes feel: where they are tight and where they are loose. Until you came across the sentence that directed your attention to your clothes, you probably were not consciously aware of the various pressures that your clothes were putting on portions of your body. You can also call up memories—say, of a recent happy event—and perhaps can even retrieve a bit of the accompanying emotion from your cognitive unconscious and can bring the memories into your cognitive conscious. Now pay attention to the feel of this book in your hand—if you are reading this text in book form—or to the input device that you are using to control your computer—if you are reading from a display. I trust that these experiments on your part have convinced you that you have a cognitive unconscious and a cognitive conscious and that a stimulus can bring a mental construct from the former to the latter.

By definition, you cannot experience, or be conscious of, any unconscious process. An unconscious process, such as the one that monitors your bladder pressure, can be a stimulus, whereupon the need to relieve yourself becomes conscious.

A bothersome philosophical question in connection with the previous sentence is: Who is the "you" about whom I am talking? Can I distinguish between you and your conscious? Taking an engineering perspective, I sidestep this question, saying simply that the "you" in this case is the union of your physical self and all of the physical and mental phenomena that your physical self manifests. We do not have to address the question of the possible distinctions among you, your conscious, and your unconscious to understand interface-design principles.

It seems likely that memories or keys to memories are stored in physical places in the brain, because in some cases, direct electrical stimulation of parts of the brain, as is sometimes done during brain surgery, can evoke a memory—that is, can make that memory conscious. Stimulating one place may reproducibly call up a particular memory, feeling, or sensation; stimulating another place may reproducibly evoke a different experience. Studies of

the brain performed with such techniques as magnetic-resonance imaging (MRI) and positron-emission tomography (PET) are helping researchers to elucidate the physical correlates of various mental activities. These technologies are mentioned because they may, at a future time, be directly helpful in the design—and, especially, in the testing—of interfaces. For example, there is an inverse correlation between a person's localized glucose uptake—an indicator of how much energy the brain is using in a particular physical structure—and the ease with which that person uses a tested interface feature. Interface testing in the future may well make increasing use of direct measures of brain activity, but a further exploration of these methods lies outside the scope of this book.

Our examples so far have followed mental constructs from unconscious to conscious. Here is an example moving the other way: A sudden noise or other unexpected event can pull your attention away from what you are doing—for example, reading this book—and to the question of what caused the sound; the sound of a lamp falling from a shelf (what was your cat doing up there?) is an example. After you return to reading, your knowledge of the event will move from the cognitive conscious to the cognitive unconscious.

There are borderline cases, and there is much that we do not understand about the conscious and the unconscious. For example, you sometimes feel that someone's name is on "the tip of your tongue"; you can almost think of the name, but you cannot quite do so. Sometimes, the memory becomes fully conscious and you recall the name. Other times, the memory remains elusive. Is there a state between the conscious and the unconscious? Have we caught the mind in the middle of "moving" the information about the name from one region of the brain to another? Is an almost remembered name evidence for a link partially formed or intermittently connected, sputtering like a loose electrical contact? Such open questions are interesting, but we do not need to answer them, just as in cosmology, we can understand that the universe is expanding, without having a clue about what happened before the expansion started, that is, before the big-bang event that most cosmologists believe was the raising of the curtain on our universe.

As I have said, I do not want to reify any particular metaphor of how the brain works. However, we cannot avoid building mental models of our brain: We build these models of our brain in our brain, a mind-boggling thought. For the time being, then, picture the conscious and the unconscious as separate compartments. These compartments are more than just different places or states for storing thoughts or memories: They have different ways of interacting with the world and with concepts. As has been elucidated by cognitive psychologists over the past century, the cognitive

conscious and the cognitive unconscious have properties beyond our awareness and unawareness of them.

Table 2.1 summarizes the differences between the cognitive conscious and the cognitive unconscious. The table tells us that the cognitive conscious is brought into play whenever you encounter a situation that seems new or threatening and whenever you have to make a nonroutine decision—that is, one based on what is happening in the here and now. Only when you are conscious of a proposition can you determine whether it is logically consistent. The cognitive conscious operates sequentially and can consider only one question or control only one action at a time. You can be conscious of only between four and eight distinct thoughts or things at once. Your conscious memory fades in, at most, a few seconds.

The conscious is invoked in branching tasks. It is sometimes difficult to distinguish a branching task from a nonbranching task. For example, braking for a traffic light may be either. It is nonbranching—and is handled by the cognitive unconscious—if you are simply reacting to a red light by pressing the brake pedal. However, if a light that you are approaching turns yellow, such that you have to decide whether to continue through the intersection without pause or to stop, your cognitive conscious comes into play. While you are learning a task, you may see and react to it as a branching event requiring conscious attention. With repetition, your execution of the task may become nonbranching and automatic. In Section 2-3, we start looking into these properties and their implications for interface design.

TABLE 2.1. PROPERTIES OF THE COGNITIVE CONSCIOUS AND THE COGNITIVE UNCONSCIOUS

PROPERTY	CONSCIOUS	UNCONSCIOUS
Engaged by	Novelty	Repetition
	Emergencies	Expected Events
	Danger	Safety
Used in	New circumstances	Routine situations
Can handle	Decisions	Nonbranching tasks
Accepts	Logical propositions	Logic or inconsistencies
Operates	Sequentially	Simultaneously
Controls	Volition	Habits
Capacity	Tiny	Huge
Persists for	Tenths of seconds	Decades (lifelong)

2-3 Locus of Attention

You have a degree of control over making unconscious thoughts conscious, as you demonstrated when you brought the final character of your first name "into mind." You cannot deliberately make conscious thoughts unconscious, however. "Don't think about an elephant," a girl whispers to a boy, knowing that the boy cannot comply. But in a few moments, unless the conversation stays on elephants, the animal will fade into the boy's unconscious. When that happens, the boy is no longer paying attention to the thought of an elephant: The elephant is not his locus of attention.

I use the term *locus* because it means *place,* or *site.* The term *focus,* which is sometimes used in a similar connection, can be read as a verb; thus, it conveys a misimpression of how attention works. When you are awake and conscious, your **locus of attention** is a feature or an object in the physical world or an idea about which you are intently and actively thinking. You can see the distinction when you contemplate this phrase: "We can deliberately focus our attention on a particular locus." Whereas *to focus* implies volition, we cannot completely control what our locus of attention will be. If you hear a firecracker unexpectedly exploding behind you, your attention will be drawn to the source of the sound. *Focus* is also used to denote, among the objects on a computer display, the one that is currently selected. Your attention may or may not be on this kind of focus when you are using an interface. Of all the world that you perceive through either your senses or your imagination, you are concentrating on at most one entity. Whatever that one object, feature, memory, thought, or concept might be, it is your locus of attention. Attention, as used here, includes not only the case of actively paying attention but also the passive case of going with the flow, or just experiencing what is taking place.

You see and hear much more than whatever is the locus of your attention. If you go into a room to look for a misplaced item, what you seek may be plainly in view but remain unnoticed. We can demonstrate through optical considerations that the image of the sought object was on your retina; it might even have been within the 5-degree cone of your foveal vision. We know through experiments in neurophysiology that a signal representing the object was being generated and transmitted over the optic nerve, yet you do not notice it, because it never became your locus of attention. If I listen for them, I notice that the fluorescent lights in the hall near my office buzz annoyingly, but otherwise I do not hear them. The sound is there, as a tape recording can demonstrate, even when I am unaware of it. I

most often notice the sound when I turn the lights on or off. The sudden start of the buzzing calls my attention to it; the sudden stop makes me realize—amazingly, because it is after the fact—that I had been hearing it. Indeed, what seems to be a full-fidelity recollection of the sound I had just been hearing suddenly becomes my locus of attention. Experiments show that direct perceptions—the contents of what psychologists call perceptual memory—seem to persist for a brief period: The well-known phenomenon of the persistence of vision is what makes the discrete frames of a movie appear to flow in continuous motion. In particular, visual perceptions decay in typically 200 milliseconds (200 msec), with a range of 90 msec to 1,000 msec; auditory perceptions decay in typically 1,500 msec, with a range of 900 msec to 3,500 msec (Card, Moran, and Newell 1983, pp. 29–31). I cannot now, sitting at my desk, recreate the buzz in that same vivid, immediate way as I did right after it had stopped and my attention had been directed by the sudden onset of silence to the previous presence of the sound. Now, hours later, the perception is long gone, and only a relatively pale memory—one having the character more of a description than of a sensation—remains of the annoying fluorescent buzz.

Perceptions do not automatically become memories. Most perceptions are lost after they decay. One implication for interface design of the rapid decay of sense perceptions is that you cannot assume that, because someone has seen or heard a particular message 5 seconds earlier, that person will remember its wording. If that particular wording is important or if there is an important detail—for example, if the message is, "Report error type 39-152," with the critical detail being the particular number—either you must keep the message displayed until it is no longer needed (the best strategy), or the user must be able to apply the information immediately—that is, before memory of it decays. As the information becomes the locus of attention, it moves into short-term memory, which we define in Section 2-3-4; it will persist there for as long as 10 seconds.

2-3-1 Formation of Habits

Anything worth doing is worth doing badly—at first.

—Dick Karpinski

When you perform a task repeatedly, it tends to become easier to do. Juggling, table tennis, and playing piano are everyday examples in my life; they all seemed impossible when I first attempted them. Walking is a more

widely practiced example. With repetition, or *practice*, your competence becomes *habitual*, and you can do the task without having to think about it. Lewis Thomas (1974), whose writings on biology are always a joy to read, expanded lyrically on the subject.

> Working a typewriter by touch, like riding a bicycle or strolling on a path, is best done by not giving it a glancing thought. Once you do, your fingers fumble and hit the wrong keys. To do things involving practiced skills, you need to turn loose the systems of muscles and nerves responsible for each maneuver, place them on their own, and stay out of it. There is no real loss of authority in this, because you get to decide whether to do the thing or not, and you can intervene and embellish the technique any time you like; if you want to ride a bicycle backward, or walk with an eccentric loping gait giving a little skip every fourth step, whistling at the same time, you can do that. But if you concentrate your attention on the details, keeping in touch with each muscle, thrusting yourself into free fall with each step and catching yourself at the last moment by sticking out the other foot in time to break the fall, you will end up immobilized, vibrating with fatigue. (p. 64)

When an observer suggested that a baseball player should think about his technique as he was hitting, baseball star Yogi Berra echoed Lewis's theme but with characteristic brevity: "How can you think and hit at the same time?" (Kaplan 1992, p. 754).

Any habit is a surrender of detail control, but habits are essential to the earth's higher life forms. At the other extreme, life is entirely possible—for example, in microbes—in the absence of any consciousness whatever, at least as far as we know or have any reason to believe. We also use the term *habit* in a pejorative sense. Despite Lewis's claim that there is no real loss of authority, bad habits do develop. Habits can be so strong as to approach addiction, sometimes reaching the point of a total loss of conscious control. (I am speaking not of physiological addictions here, such as to nicotine or opiates, but rather of undesired learned habits, such as nail biting.) Insofar as our conscious selves are who we are, I am reminded of Unamuno's observation: "To fall into a habit is to begin to cease to be" (Unamuno 1913). Unamuno was, perhaps, warning us against the pernicious aspects of habit formation; when it comes to the routine aspects of everyday life, however, you want your conscious attention to "cease to be."

You can readily imagine how difficult it would be to drive your car if you had to think, "Uhh, I want to stop. Let me see now: The engine needs

to slow down, so I have to take my foot off the accelerator. Now I have to dissipate my car's kinetic energy into heat by pressing on the brake pedal. . . ." Fortunately, as an experienced driver, you perform the operation habitually. Similarly, you have developed many small habits that help you to use your computer, watch, alarm clock, telephone, and every other device that has an interface.

Persistent use of any interface will cause you to develop habits that you will find yourself unable to avoid. Our mandate as designers is to create interfaces that do not allow habits to cause problems for the user. We must design interfaces that (1) deliberately take advantage of the human trait of habit development and (2) allow users to develop habits that smooth the flow of their work. *The ideal humane interface would reduce the interface component of a user's work to benign habituation. Many of the problems that make products difficult and unpleasant to use are caused by human-machine design that fails to take into account the helpful and injurious properties of habit formation.* One notable example is the tendency to provide many ways of accomplishing the same task. Having multiple options can shift your locus of attention from the task to the choice of method (a topic explored in Section 3-7).

You cannot often break a habit by an act of volition. As often or as fiercely as you tell yourself that you will not perform the habitual action, you may not always be able to stop yourself. Say that, for example, next Sunday your car will interchange the functions of the brake and accelerator pedals. A red light will illuminate on your dashboard to warn you of this change. You might manage to drive a few blocks successfully with the pedals reversed, but most of us would not get out of the driveway without making an error. As soon as your locus of attention is pulled away from the novel arrangement—for example, if a child runs into the street—your habitual reaction will make you stomp on the wrong pedal. The red light will be of no help at all. I emphasize that you cannot undo a habit by any single act of willpower; only a time-consuming training process can undo a habit. A designer can lay—or create inadvertently—a nasty trap by permitting to run, on one computer, two or more heavily used applications that differ in only a handful of often-used details. In such a circumstance, the user is guaranteed to develop habits that will cause him errors when he attempts to use in one application a method appropriate to only the other.

2-3-2 Execution of Simultaneous Tasks

In the language of cognitive psychologists, any task that you have learned to do without conscious thought has become *automatic.* Automatic-

ity enables you to do more than one activity at a time: All but at most one of the tasks that you perform simultaneously are automatic. The one that is not automatic is, of course, the one that most directly involves your locus of attention. When you do two tasks simultaneously, neither of which is automatic, your performance on each task degrades—a phenomenon that psychologists call *interference*—compared to your performance on each task alone, because the two tasks compete for your attention. The more predictable, automatic, and unconscious a task becomes, the less it will degrade or compete with other tasks (Baars 1988, p. 33).

We humans apparently simulate the simultaneous accomplishment of tasks that require conscious control by alternating our attention between tasks, attending now to one, then to the others (Card, Moran, and Newell 1983, p. 42). You achieve true simultaneity when all but at most one of your tasks become automatic. For example, you can, at the same time, eat a snack without choking, walk without tripping, and think through a mathematics problem to a satisfactory conclusion. (You may also be working on another math problem unconsciously, but by the definition of the cognitive unconscious, you wouldn't notice that you were. I am claiming only that you cannot simultaneously work *consciously* on two different math problems.) For most people, all of the tasks, except for finding the solution to the mathematics problem, are so well learned that they undertake these tasks on autopilot. However, if you were practicing these simultaneous activities and suddenly discovered a nasty-tasting morsel in the snack, you would become conscious only of what you were eating. You would no longer be conscious of the mathematics problem.

Equally important as the fact that you cannot be conscious of more than one task at any moment is the realization that humans cannot avoid developing automatic responses. This idea is important enough to bear repetition: No amount of training can teach a user *not* to develop habits when she uses an interface repeatedly. That we form habits is part of our fixed mental wiring; habit formation cannot be prevented by any act of volition. If you have ever unintentionally driven toward your normal workplace on a Saturday morning when you intended to go somewhere else, you've been had by a habit that formed through repetition of a fixed sequence of actions. When you learned to read, at first you sounded out and paid attention to each letter and syllable; now (I hope) you read without conscious attention to the process of translating marks into words.

Any sequence of actions that you perform repeatedly will, eventually, become automatic. A set of actions that form a sequence also becomes clumped into a single action; once you start a sequence that takes less than 1

or 2 seconds to complete, you will not be able to stop the sequence but will continue executing the actions until you complete that chunk. You also cannot interrupt sequences that take longer than a few seconds to execute unless the sequence becomes your locus of attention. Thus, after you take the wrong turn on Saturday, you may suddenly realize that you intended to drive in the opposite direction; this realization makes your navigation your locus of attention, and you can interrupt the automatic sequence of actions that would have led you to your workplace.

When you repeat a sequence of operations, making *and keeping* what you are doing your locus of attention is the only way to keep a habit from forming. This is very difficult to do. As expressed in a common phrase, our attention wanders.

The inevitability of habit formation has implications for interface design. For example, many of us have used computer systems that, before they will perform an irreversible act, such as deleting a file, ask, "Are you sure?" You then must type, say, a *Y* for yes or an *N* for no in response to the question. The idea is that, by making you confirm your decision, the system will give you a chance to correct an otherwise irrecoverable error. This idea is widely accepted. For example, Smith and Duell (1992), addressing a nursing environment, say, "If you inadvertently delete part of the permanent record (which is hard to do because the computer always asks if you're sure) . . ." (p. 86). Unfortunately, Smith and Duell are unrealistic in their assessment: You can readily make an accidental deletion even when this kind of confirmation is required. Because errors are relatively rare, you will usually type *Y* after giving any command that requires confirmation. Due to the continual repetition of the action, typing *Y* after deleting soon becomes habitual. Instead of being a separate mental operation, typing the *Y* becomes part of the delete-file action; that is, you do not pause, check your intentions, and then type the *Y*. The computer system's query, intended to serve as a safety measure, is rendered useless by habituation; it serves only to complicate the normal file-deletion process. The key idea is that *any confirmation step that elicits a fixed response soon becomes useless.* Designers who use such confirmations and administrators who think that the confirmations confer protection are unaware of the powerful habit-forming property of the cognitive unconscious (see Section 6-4-2).

A more effective strategy is to allow users to undo an erroneous command, even if they have performed intervening actions since issuing it. You cannot protect against a user developing a habit of confirming without reestablishing the decision as the locus of attention, even by making the required confirmation action unpredictable. For example, have the computer specify that the user must type either twice or backward—that choice

being presented at random—a word displayed, also chosen randomly, in the dialog box.

> **The action that you have requested cannot be undone. It will cause permanent loss of the information in the file. If you are sure you wish to delete the information forever, type backward the tenth word in this box.**

Requiring this kind of confirmation is as draconian as it is futile. Any attempt at an effective confirmation process is necessarily obnoxious because it prevents the user from forming a habitual response and from ever becoming comfortable with it. If, for legal or other reasons, a file should never be deleted by the user, it should be made impossible for such a deletion to be performed. Such measures also create a new locus of attention; the user is not attending to the correctness of their prior response, thus frustrating the purposes of both the confirmation and the user.

No method of confirming intent is perfect. Even having the user type in the reason for a deletion—a technique especially useful in a situation that carries legal implications—will soon lead to the user's supplying one of a few stock answers. If the rationale for performing an irreversible act was flawed from the outset, no warning or confirmation method can prevent the user from making a mistake.

Trapped in the Pitfall of Automaticity

I was trapped in the pitfall of automaticity while I was writing this chapter: I italicized a word, then tried to unitalicize it. In most Macintosh word processors, pressing and holding the key marked with a drawing of an apple (called the Command key) and then pressing and releasing the key marked with the letter T (Command-T) returns the text to normal status. In Microsoft Word, however, Command-T alters the paragraph format. If you had asked me—if you had thus made it my locus of attention—I would have told you that I was using Word, yet I reached (automatically!) for Command-T, typed it, and mangled the paragraph formatting. The only way to prevent such errors is through interface designs that take into consideration the inevitability of habit formation.

2-3-3 Singularity of the Locus of Attention

I can't think about X *if I'm thinking about* Y.

—*Chris, a character on the television show* Northern Exposure, *31 October 1994*

For our purposes, an essential fact about your locus of attention is that there is but one of them. This observation underlies the solution of numerous interface problems. Many people do not believe that they or others have only one locus of attention, but experiments, described in the cited literature, strongly support the hypothesis that we are unable to attend to multiple simultaneous stimuli. This notion, which parallels our discussion on the limitations of the cognitive conscious, is sufficiently surprising to justify examining the support for it.

As Roger Penrose (1989) noted, "A characteristic feature of conscious thought . . . is its 'oneness'—as opposed to a great many independent activities going on at once" (p. 398). Bernard Baars (1988), a widely recognized leader in the study of the cognitive conscious, explains that when people are "asked to monitor a demanding stream of information [they are] largely unconscious of alternative streams of information presented at the same time, even to the same sensory organ. Similarly, in absorbed states of mind, when one is deeply involved with a single train of information, alternative events are excluded from consciousness" (p. 33). The alternative events are not loci of your attention.

Common parlance recognizes this observation. For example, we may have a thought, and we may have many thoughts, but we speak of our attention only in the singular. We never speak of a person's *attentions* except in unrelated usages, such as *unwanted attentions.* Although you are unconscious of all but the one line of thought—the single conceptual thread that you are following—an unexpected or surprising event can pull your attention from that thread. I have described how surprising events trigger conscious attention. What is salient here is that you have acquired a new locus of attention and lost the old; it is not the case that a second locus has been brought into play.

An interrupting event does not need to be external: A sudden pain or the realization that it is time for an appointment may break into your cognitive conscious, derailing your current train of thought and putting it on a new track.[1] However, if outside or internal events are routine and unpress-

1. Considerable attention has been paid to the biological mechanisms that allow animals to synchronize with external time cycles, but I am unaware of any work regarding how we set and respond to internal alarm clocks.

ing, your unconscious recognizes that status, and you ignore those events—without being conscious that you are ignoring them. In other words, in the presence of the ordinary, your attention is not pulled away. You can train yourself to scan the environment consciously from time to time to notice events to which your attention would not be otherwise called. To illustrate, pilots are taught to scan their instruments regularly, without outside stimuli to initiate scanning. A scan allows pilots to detect, for example, an instrument that subtly shows an abnormal condition. (Not every instrument in an aircraft has an alarm associated with it.) Nonetheless, pilots regularly fail to perform their scan when events force their attention to a particular locus.

Absorption Kills 101 People

An extreme example is an accident that killed 101 people in December 1972. Normally a green indicator in the airliner's cockpit lights to signal that the landing gear is down and ready for landing. When the indicator failed to light, the pilot decided to circle at an altitude of 2,000 feet, and the copilot put the aircraft on autopilot to maintain the altitude. All three crew members then tried to change the bulb, but it stuck and they could not get it out. Perhaps due to their moving around and working on the bulb, they accidentally turned off the autopilot; in any case, it became disengaged. Soon, as the cockpit recording later showed, an automatic warning sounded; a 0.5-second chord warned them that they had gone 250 feet below their assigned altitude. A yellow warning indicator also lit up. The crew, absorbed in the problem with the green bulb, failed to notice either warning. A little later, while still struggling with the bulb, the copilot noticed that the altimeter indicated 150 feet, alarmingly low. He then asked the pilot, "We're still at two thousand, right?" The pilot replied, "Hey, what's happening here?"

As the pilot spoke, a low-altitude warning horn went off. "And even with the altimeter approaching zero, an amber light on the altimeter indicating they were off their assigned altitude, the radio altimeter approaching zero, and a radio altimeter warning horn beeping, everyone in the crew was so sure they were at 2,000 feet that no one could bring himself to act and eight seconds after the first officer noticed the altimeter, the aircraft crashed into the Everglades." (Quoted material from Garrison 1995.)

You can be more or less absorbed in the task that involves your locus of attention. The more intensely you are focused, the more difficult to transit to a different locus of attention, and the greater the stimulus needed to effect such a change. In the extreme case, when we are completely absorbed by a task, we cease to monitor our environment. You have probably experienced the absorbed state when you are reading a book, are thinking deeply about a problem, or are in the midst of a crisis that, as the expression goes, demands your attention. The use of a computer is often so stressful and difficult that a user will become absorbed in working on the computer system, and therefore distracted from the completion of tasks. Our goal is to leave the task as the locus of the user's attention.

Absorption in a task or a problem decreases the ease with which a person can change her locus of attention. On the other hand, such absorption—if it is confined to the task and if the system does not pull attention to itself—is essential to productivity. Systems should be designed to allow users to concentrate on their jobs. Interfaces should be designed as though the user will be so absorbed in her task that she may not respond to your attempts to communicate with her. An interface must work, whatever the user's state of absorption. For example, interface designers sometimes assume that the user's locus of attention is the cursor and that changing the cursor's shape will inevitably command the user's attention. The cursor location is a good place to put indicators, but even there, the indicator can go unnoticed; the shape of the cursor is not the locus of attention; rather, the place or object to which it is pointing may well be. An example is given in Section 3-2.

Many examples of absorption seem unbelievable until you experience a similar incident or until you have seen so many reports that you become convinced of the strength of absorption's grip. Because aviation accidents are often well researched and carefully documented, they are a good source for case studies. Here is another (Garrison 1994). A well-known pilot was flying an aircraft unfamiliar to him, one that required him to lower the retractable landing gear as he made his descent. As a reminder, a buzzer sounds when this particular model of aircraft is a certain distance from the ground and the gear has not been lowered. "I landed gear-up at one point, having persuaded myself that the insistent buzzer I kept hearing as I approached the gravel runway had something to do with the airbrakes. (This was one of my early lessons in the bizarre mental mix-ups that can lead to accidents)" (Garrison 1994). But there was no bizarre mental mix-up: Garrison was concentrating on making a good landing, one of the most difficult

tasks that a pilot must accomplish and one that requires a great deal of concentration.[2]

The human ability to tune out disturbances is not necessarily an all-or-nothing response, as in the previous examples; it can be proportional to the level of absorption and the degree of disturbance. As stress increases, "people concentrate more and more on but a few features of their environment, paying less and less attention to others" (Loftus 1979, p. 35). Thus, *if the computer behaves unexpectedly while you are using an interface, you become less likely to see hints, help messages, or other user aids as you become increasingly agitated about the problem.*

The more critical the task, the less likely it is for users to notice warnings that alert them to potentially dangerous actions. A computer warning message is most likely to be missed when it is most important for it not to be missed; this sounds like a humorous corollary of Murphy's law,[3] but it is not. One way we can help is to *make sure that users cannot make interface operation errors, or that the effects of any actions are readily reversible rather than simply notifying users about the potential consequences of their actions.* Most interface situations can be designed such that error messages are unnecessary. A forceful diatribe against using error messages appears in *About Face* (Cooper 1995, pp. 421–440).

2-3-4 Origins of the Locus of Attention

That we have only one locus of attention may seem odd. Let us explore how we may have come to have this trait. Baars (1988) speaks eloquently to the question; he seeks a biological rationale for our having evolved in this limited fashion, asserting that

> consciousness and related mechanisms pose a great challenge to functional explanations because of the paradoxical limits of conscious capacity. Why can't we experience two different "things"

2. An interface designer might wonder why, if the aircraft could sound the buzzer, could it not also automatically lower the landing gear? This book is not the forum for a discussion of the details, but at times, automatically lowering the landing gear could be dangerous to the occupants. Therefore, it is always left up to the pilot to choose whether to lower the gear.

3. If anything can go wrong, it will. The first corollary is, If nothing can go wrong, it will anyway.

> at one time? Why is Short Term Memory[4] limited to half a dozen unrelated items? How could such narrow limits be adaptive? Reasoning naively, it would seem wonderful to be able to consciously read one book, write another one, talk to a friend, and appreciate a fine meal, all at the same time. Certainly the nervous system seems big enough to do all these things simultaneously. The usual answers, that the limitations are "physiological" or that we only have two hands and one mouth to work with, are quite unsatisfactory because they simply move the issue one step backwards: Why have organisms blessed with the most formidable brain in the animal kingdom not developed hands and mouths able to handle true parallel processing? And why does our ability to process information in parallel increase with automaticity, and decrease with conscious involvement? (p. 348)

Baars suspects that the answer lies in there being only one "whole system": There is but one "I" in each of us. But to say that there is one personhood per human being begs the question. That is, why are there not multiple personhoods per mind-body ensemble? I am speaking not of changes that occur serially,[5] but rather of true simultaneous and independent minds in a single, connected physical entity. It may simply be that having a single personhood is a biological accommodation to the linearity of time or results from an accident of evolution rather than from a functional adaptation. Nevertheless, it seems more likely that our single personhood is adaptive: an accommodation to the purely physical impediments to having multiple simultaneous persons in one body. Given our evolved body plan, both personalities would not be able to speak at once or to turn the head in different directions simultaneously. Even if our eyes could have evolved to operate as independently as a gecko's, would they be able to serve two independent curiosities? I can imagine—to mention just one possible catas-

---

4. Short-term memory, often abbreviated STM in the literature, describes the behavior of memory with respect to stimuli just seen, heard, or otherwise sensed. If we do not make use of the memory or make it our locus of attention, STM fades in 10 to 20 seconds, or far less if we pay attention to new events. As Baars notes, STM is not only short but also of very limited capacity, and new events will drive out the old irretrievably. For a nontechnical and eminently readable account of the structure of human memory, see Loftus (1980).

5. Continuing change of personality is a human constant: We grow and change all the time. These changes, as well as the changes called multiple-personality disorder, are not what is being discussed.

trophe—that any multiple-mind mutation was eaten when it tried to escape a predator by deciding to run in two different directions at once.[6]

Siamese twins and two-headed animals do occur from time to time, and they have two independent minds, but they are developmental accidents, a mismatching or a misreading of the genetic code. They are not successful from an evolutionary standpoint and are not products of natural selection. In the wild, such sports of nature rarely survive and reproduce.

2-3-5 Exploitation of the Single Locus of Attention

We have examined the effects and the possible origins of having a single locus of attention. The next step is to make use of that singularity. We can redesign neither our own internal wiring nor that of other users, but we create products that have interfaces that accommodate these cognitive capabilities.

That people have a single locus of attention is not always a drawback. Magicians exploit this characteristic shamelessly. A good magician can fix the attention of an entire audience on one hand so that not a single spectator will see what the other hand is doing, although that hand is in no way concealed. If we know where the user's attention is fixed, we can make changes in the system elsewhere, knowing that the changes will not distract the user. This phenomenon was exploited in designing the Canon Cat (Figure 2.1). When a user stopped working, the Cat stored a bit-for-bit image of the screen—exactly as the display appeared when she stopped—on the first track of the disk. When the user again loaded her disk, the Cat placed the most recently viewed image on the screen, in only a fraction of a second. It takes about 10 seconds for a person to switch contexts or to prepare mentally for an upcoming task (Card, Moran, and Newell 1983, p. 390), but it took only 7 seconds for the Cat to read into memory the rest of the disk. So while the user was staring at the static screen image, recalling what she had been doing and deciding what she was going to do next—her locus of attention being the preparations for the coming task—the system finished loading. Only

---

6. I have often seen my cat poised between curiosity and fear, its senses intently trained on an unfamiliar object, its body tensed for immediate flight. I have acted the same way myself. Sometimes, what is learned by not fleeing is of value, and sometimes the delay is fatal: whence the expression "curiosity killed the cat." This occasional "being of two minds," as the idiom goes, is an internal sequential dialogue rather than two independent simultaneous processes.

then did the screen become active, although it did not change appearance, except that the cursor began to blink. Only a handful of users ever noticed this trick. Most Cat owners thought that the product magically managed to read in the whole diskette in the fraction of a second that it took to display the first screen. Presto!

Many people do not believe that it takes a person approximately 10 seconds to switch contexts; the time is measured between the final command executed in the previous context and the first command issued in the new context. The hiatus is not noticed because the minds of the users are occupied; they are not aware of the passage of time. However, this phenomenon should be used carefully when designing an interface. If the work flow is such that a user makes a particular context switch repeatedly, so that it becomes habitual, the user will make the switch in far less time.

Time delays can be masked; for example, a card game that takes time to create a new deal will feel faster if a card-shuffling noise is played during the delay. The value of masking was vividly demonstrated when such a game had its sound turned off inadvertently and the players suddenly found the delay annoying (Dick Karpinski, personal communication 1999).



Figure 2.1. The Canon Cat. Note the two LEAP keys below the space bar.

2-3-6 Resumption of Interrupted Work

Usually, after dealing with an interruption to a task, you then return to the interrupted task. If the interruption lasts only a few seconds—within the decay time of short-term memory—no further stimulus is required to signal you to return to the prior task. After a longer break, however, your return to the interrupted task must be triggered, often by seeing your incomplete work lying before you. Such cueing is as common in daily life as it is when using a computer: A banana peel left on the kitchen table by your 4-year-old child becomes a cue to dispose of the peel.

A metaphor that permeates personal computers and derivative technologies is that of a central, neutral dispatch area, or desktop, from which you can launch a variety of applications. When computers are turned on, most of them present the desktop, although some of them can be set to launch a fixed set of applications. When you quit an application, you are usually returned to the desktop. This interface strategy is inefficient and nonhumane. The reason is straightforward: When you quit an application, you (1) want to either return to the previous task on which you were working or (2) start a new task.

In current desktop-based systems, you must *always* navigate to the task. *This is the same as the* worst *case for an interface that always returns you to what you were last doing, because in the case of wishing to return to the task from which you left off, you have to do* no *work at all.*

Similarly, when you return to an online site, such as a web page, it would generally be better to return you to where you last were than to a home page that, if the site is well designed, is always available with a single click. The same reasoning suggests that *when you open a document in an application, such as a word processor, you should be returned to the place where you were working when you last closed or saved it.*

The Canon Cat had the property of always returning you to the previous task when you started to use it; moreover, it presented exactly the same screen appearance, including cursor placement, as when it was last used. Many users reported that seeing the same display helped them to remember what they had been doing when they had last used the machine, which made returning to the Cat a more pleasant experience than returning to a computer that boots into a desktop. More recently, the Apple iBook has taken a similar approach, saving the current state to disk and whisking it back in when you turn the machine on.

Designers of digitally tuned radios and televisions make sure that their products retain the most recently tuned station and volume settings, a

requirement that adds to the complexity and cost of those products in the form of nonvolatile memory that otherwise would be unnecessary. For computer designers, who work with products that already have substantial nonvolatile memory, such as a hard disk, the hardware needed to do this is already in place, and there is no excuse.

THREE

Meanings, Modes, Monotony, and Myths

There is no progress without struggle.

—Frederick Douglass

So that we can discuss interface with precision, a few definitions and conventions of notations are introduced here. We use these tools, building on the notion of locus of attention from Chapter 2, to understand modes and their harmful effects on interface design. We introduce a beneficial property of interfaces, called *monotony*, which leads to a critique of interfaces that have beginner and expert modes.

3-1 Nomenclature and Notations

The world is divided into people who think that they're right.

—Diedre McGrath

Content is the information that resides in a computer or other information-processing device and that has meaning and utility for you. Creation

or alteration of content is the task that you intend to accomplish with the device. If you are a writer, the content is those of your writings that are stored in the system. If you are an artist, your graphics are the content of the system. Where the system is a computer, the menus, the icons, and other paraphernalia of the computer are not your content—unless you are an interface designer or a programmer. We can now paraphrase Asimov's first law of robotics from Chapter 1 in terms of content: Any system shall not harm your content or, through inaction, allow your content to come to harm.

A **graphical input device** (GID) is a mechanism for communicating information, such as a particular location or choice of object on a display, to a system. Typical GID examples are mice, trackballs, lightpens, tablet pens, joysticks, or touchpads. The **GID button** is the principal button on any GID—for example, the left button on a two-button mouse. In general, you use the graphical input device to control the position of the cursor, which is an arrow or other graphical emblem on the display to indicate the system's interpretation of where you are pointing. Because we can pay attention to only one cursor at a time, systems should not display more than one cursor for each graphical input device on the system. The rationale for the one-button mouse is discussed in Appendix A.

A **tap** is the act of pressing and releasing a key or switch without any intervening action. *Tap* applies only to keys on a keyboard or to other momentary contact switches that, when released, automatically return to their original position and electrical state.

To **click** is to position the GID and then to tap the GID button. Thus, "click the word *alligator*" indicates that you are to move the cursor such that it points to *alligator* and then, while the cursor remains stationary, tap the GID button. This operation constitutes "clicking on *alligator.*" To **drag** is to press the GID button at one location and then to move the GID before releasing the GID button at a new location; this action is sometimes called *click and drag*. To **double click** is to position the GID and then to tap the GID button twice quickly, without any intervening motion of the GID or other user action. (In practice, a small amount of GID motion is allowed, because it will usually move slightly when the button is pressed.) Triple- and higher-order clicks have been used in interface design.

The usual notations for the overlapping keystrokes that are used to control most software are sometimes ambiguous. For example, the act of pressing and holding the Control key and—while holding it—pressing and holding Shift and—while continuing to hold them both down—tapping the letter *t*, is often written *Control-Shift-t* or *Control+Shift+t* in manuals. However, these encodings of keyboard operations are indistinguishable from the

conventional notations for the sequences of keystrokes *Control,' hyphen, Shift, hyphen, t*; and *Control, plus, Shift, plus, t*, respectively.

This ambiguity can cause errors. For example, while I was writing this manuscript, I looked up a key combination for a command I needed; the manual listed it (at the end of a sentence) as

Control +.

So I pressed Control and, while holding it, typed a plus sign. Due to the manual's ambiguous notation, my interpretation was wrong: The manual was trying to direct me to press and hold Control and, while holding Control, to tap the period key.

Furthermore, the usual notations do not allow you to notate pressing and holding down, say, both the Shift and Command keys, and—while holding them both down—typing multiple alphabetic characters, releasing Shift but not Command after the first two alphabetic characters have been typed. The alternative that I have just used—expressing the operation in natural language—is clumsy albeit unambiguous.

To denote keyboard operations with precision and brevity, I use a downward-pointing arrow immediately after the name of a key to indicate that the key has been pressed and is to be held; for example, *Shift*↓ signals that the Shift key is to be pressed and held. An upward-pointing arrow after the name of a key indicates the release of that key: *Shift*↑. A tap of a single key—for example, *t*—is rigorously notated *t*↓ *t*↑. The tap of a single key can be abbreviated, where the shorthand is unambiguous, as *t*↓↑ or, where it will not cause confusion, simply *t*.

A space separates the notation of consecutive actions. A tap of the space bar is represented by the notation

Space

This convention does not introduce any ambiguity with typing the word *space*, because typing the letters of the word *space* is represented by letters separated by spaces, namely,

s p a c e

When there is a risk of misinterpretation, or when exactly what is being typed requires emphasis, I use the full notation:

s↓ *s*↑ *p*↓ *p*↑ *a*↓ *a*↑ *c*↓ *c*↑ *e*↓ *e*↑

Arbitrary combinations of keys can be represented linearly in this notation. For example, pressing and holding Shift and then tapping *n*, pressing and

holding Control while Shift is still held and then tapping *k*, releasing Shift while still holding Control and then tapping *w* before releasing Control is written as follows:

Shift↓ *n Control*↓ *k Shift*↑ *w Control*↑

The release of any of the keys currently being held—if the order in which they are released is immaterial to the functioning of the interface—is denoted by an up arrow for each key being held. This convention allows us to denote typing the word *space* this way:

s↓↑ *p*↓↑ *a*↓↑ *c*↓↑ *e*↓↑

Here is a more useful example: To reset parameter RAM on a Macintosh computer, you use the following sequence of commands:

Command↓ *Control*↓ *Power*↓ ↑↑↑ *Command*↓ *Option*↓ *p*↓ *r*↓

Hold these keys and wait until a chime sounds

↑↑↑↑

(In English: Press and hold the Command key—the lack of an up arrow means that you should not release the key immediately. Rather, while you are holding it, press and hold the Control key. There is no up arrow here either, so while you are holding both of those keys, you press the Power key. There have been no up arrows, so now you are holding down three keys. The three up arrows that follow mean that you should release all three keys in any order or all at once. Next, you are instructed to press and hold the Command key, press and hold the Option key, press and hold *p*, and then press and hold *r*, such that, eventually, you are holding down four keys at once. You hold these keys until after the chime and then let all four keys go.)

If a typed sequence has special time dependencies, they should be noted in accompanying text. For example, for the user to type a single letter *t*, we use this sequence:

t↓ *t*↑

With most keyboards, however, if a delay, usually of at least 500 msec, occurs between *t*↓ and *t*↑, extra instances of the letter *t* will be produced at a rate of approximately one every 100 msec. This feature is often called autorepeat. Delays, such as the one that triggers autorepeat, cause problems in interfaces. The use of delays in interfaces and an improved way to implement autorepeat are described in Section 6-4-4.

3-2 Modes

Since humans are more pliable than computers, it can be easier to make a human fit the computer's limitations than to design the computer to fit the human's needs. When that happens, the human becomes a prisoner trapped by the computer rather than liberated by it.

—Karla Jennings

Modes are a significant source of errors, confusion, unnecessary restrictions, and complexity in interfaces. Many of the problems modes cause have been widely recognized; nonetheless, making systems truly modeless is an underused tactic in interface design. Before we can discuss methods for eliminating modes, we must understand them in detail, especially because even interface professionals have disagreed about what constitutes a mode (Johnson and Englebeck 1989).

To understand modes, we must first define a gesture. A **gesture** is a sequence of human actions completed automatically once set in motion. For example, typing a common word, such as *the*, is a single gesture for an experienced typist, whereas the typing of each letter would be a separate gesture for a beginning typist. Combining a sequence of actions into a gesture related to the psychological process is called **chunking**: the combining of separate items of cognition into a single mental unit, a process that allows us to deal with many items as though they were one (Buxton 1986, pp. 475–480; Miller 1956).

Most interfaces have multiple interpretations of a given gesture. For example, at one moment, tapping Return inserts a return character into the text, whereas at another time, tapping Return causes the text typed immediately prior to that tap to be executed as a command.

Modes are manifested by how an interface responds to gestures. *For any given gesture, the interface is in a particular mode if the interpretation of that gesture is constant.* When that gesture has a different interpretation, the interface is in a different mode. This definition gives us a useful initial view of what constitutes a mode; we will refine the definition later.

A flashlight that is operated via a push-button toggle can be either on or off, assuming that it is in good operating condition. Pressing the button turns the light on if the present state is off, and turns the light off if the present state is on. The two states of the flashlight correspond to two modes in the interface: In one mode, tapping the button turns the light on; in the other mode, tapping the button turns the light off. If you do not know the present state of the flashlight, you cannot predict what a press of

the flashlight's button will do. If your flashlight is deep in a duffel bag where you cannot see it, you cannot check by feel whether the light is off or on, assuming that it does not get detectably warm; if your intent is to make sure that it is off, you will have to take it out. Your inability to determine the state of the flashlight is a classic problem caused by an interface that has modes; you cannot tell by inspecting the controlling mechanism what operation you need to perform to accomplish your goal. If you operate the controlling mechanism without separately verifying the state of the system, you cannot predict the effect that the operation will have.

Toggles are difficult to label. For example, in an interface I was shown, an on-screen button was labeled *Lock*. When users first encountered the button, they understood that they had to click on it to lock the data in that window. When they did so, the button's label changed to *Unlock*, to indicate that operating that button would unlock the data. Later, most of the users wondered why the data was unlocked, a conclusion they came to because the button said *Unlock*. As often happens when toggles are used on buttons or in menus, users read the label as an indicator of state. And they are justifiably confused: The button said *Lock* when the data was unlocked and *Unlock* when the data was locked. Obviously, you cannot solve the problem by interchanging the labels and having the button say *Unlock* when the data is unlocked and *Lock* when it is locked.

What can help is to use a check box rather than a button and to use the word *Locked* rather than *Lock* on it. This is usually understood correctly: When the box is checked, the data is locked. When the box is not checked, the data is not locked. The labels do not change. You can make more complete labels, such as "Click this button to unlock the data" or even "The data is currently locked; click this button to unlock the data." But it is difficult to put full explanations on a button, alongside a check box, or in a menu, unless a zooming interface is available, as discussed in Section 6-2.

Check boxes can leave the user guessing what the alternative is. For example, if a check box labeled "Save to archive on closing" is checked, the data will be saved to an archive when the window is closed, but the label gives little clue as to what will happen if the box is not checked. Will the data be saved somewhere else, not saved at all, or will another option appear when you close the window? Often, the best solution is to use a set of radio buttons (Figure 3.1); they are not modal, and the user can clearly see not only the current state but also the alternative(s). Whether check boxes or radio buttons are used, it is important to label them with adjectives, which describe the state of the object affected, rather than verbs, which describe an action, in which case the user does not know whether the action has taken place or is yet to take place.

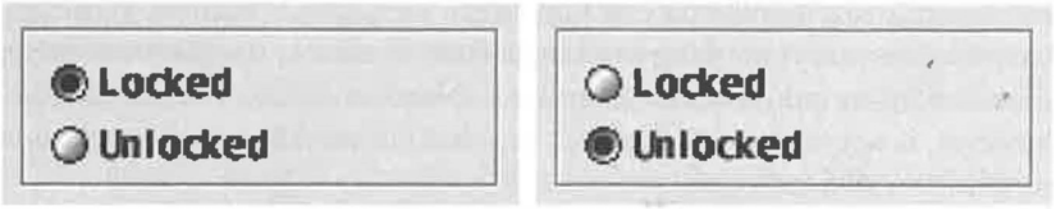


Figure 3.1. A pair of radio buttons labeled with adjectives. The option chosen has the dot in the circle; the left image reflects the locked state and the right image the unlocked state. Confusion is unlikely, and the user is aware of the available options.

For one-of-many choices, radio buttons are already the standard, and there is rarely reason to use other mechanisms. *Use radio buttons rather than toggles; toggles work reliably only when the value of the state controlled by the toggle is your locus of attention and is visible, or is in short-term memory.*

Except when the state is the locus of attention—and it is usually not—toggles will at times cause users errors. The kinds of errors caused by toggles are usually transient, and recovery is easy, but this is no reason to overlook them in interface design. To work in interface design without attending to details at this level is like trying to play a violin concerto, only occasionally forgetting to play a sharp or a flat given in the key signature. The errors annoy by pulling the listener's attention away from the continuity of the music. Similarly, occasional little "gotchas" in an interface impede the user's work flow.

Another troublesome aspect of modes—one that causes computer users considerable exasperation—is exemplified by the way that the Caps Lock key, found on most keyboards, functions. Often, your first indication that this key has become engaged accidentally is that you notice that the sentence you have typed is set in all uppercase letters, and only then do you also notice that the Caps Lock key's light is on, if it has such a light. "It is no accident that swearing is denoted by #&%!#$&," writes my colleague, Dr. James Winter; it is "what a typewriter used to do when you typed numbers when the Caps Lock was engaged" (personal communication 1998).

Decades ago, Larry Clark observed that modes cause problems because they make habitual actions have unexpected effects (Clark 1979).[1] The most commonly prescribed remedy for mode errors is to indicate

1. I am not sure for how long the term *mode* has been used in describing interfaces. An internal Xerox report on the Gypsy Typescript System boasted, "There are no 'modes' in Gypsy" (Tesler and Mott 1975). There were, in fact, modes in Gypsy, but some of the more problematical ones of contemporary word processors had been eliminated. The term *mode* was new enough in 1975 that the authors felt it necessary to put it in quotes.

prominently to a user the state of the system. Dr. Donald Norman characterized mode errors as resulting from inadequate feedback, the feedback being provided by an indicator of system state (Norman 1983). The real culprit, however, is not inadequate feedback but that the user's locus of attention is not the provided indicator.

An especially clear example of the failure of using an indicator to provide feedback of system state to the user occurs in the generally excellent interface to the computer-aided design (CAD) package Vellum (Ashlar 1995). To anybody who is designing a drawing package, I strongly recommend a study of Ashlar's Drafting Assistant; it represents an extraordinarily good interface, one that is more productive and pleasant than that of the better-known AutoCAD.[2] One of Vellum's time-saving functions is a tracer that can follow the inside or the outside of a geometrical figure, selecting the border as it travels. To enable the tracer, you click on a palette item that turns the standard cursor ↖ into a distinct shape, the tracer cursor ▸. After you finish tracing, you often forget to switch back to the standard cursor. As I click to select an object after I have executed a trace operation—selecting by clicking being an action that I do so often that it long ago became automatic—instead of succeeding in making the selection, I sit surprised as the tracer goes off on its rounds. I have been making this mistake for years, as have other users with whom I have spoken. Although I sometimes catch myself in time, I never will learn—I never *can* learn to always catch that I am in tracer mode—because clicking to make a selection is, for me, automatic. My locus of attention is the object I am selecting, not the cursor shape. Vellum's changing cursor shape is one example of many that confirm that feedback and indicators of system state are not sufficient to guarantee the elimination of mode errors, even if the indicators are physically colocated with your locus of attention. They may be in your high-resolution foveal field, and, in this case, you are even using the indicator as your cursor, yet because it is not your locus of attention, you are not aware of the message it is trying to convey. Essential topics for future research include quantifying the frequency of mode errors and understanding the conditions that affect the frequency with which these errors occur.

Expertise is no protection against modes; an expert has developed firm habits. Naiveté is also no protection. In the example of Vellum's cursors, the beginner has not yet established in his mind that it is necessary to change the cursor back after tracing. By the time a beginner has learned to do this step without having to think about it explicitly, he is an expert with respect

2. In fact, AutoCAD licensed this technology while this book was being written.

to this feature and is subject to the habituation problem. If the current state of the interface is not the user's locus of attention and if an interface has modes, the user will sometimes make errors because his locus of attention is not the current mode.

Here is another example: On some popular computers, many programs give you a new, clean form when you type

Command↓ *n*↓↑ ↑

The *n* stands for *new*. In the America Online electronic mail package, in contrast, you obtain a new, blank e-mail form by typing

Command↓ *m*↓↑ ↑

I presume that *m* stands for *mail*. The error that is repeatedly made when a user wants to start a new electronic mail message is to use

Command↓ *n*↓↑ ↑

The interface state that gives rise to a mode in this example consists of having a particular application active. The problem occurs when users employ the *Command*↓ *n*↓↑ ↑ command habitually. A beginner would probably make this same error for a different reason; he would assume that *Command*↓ *n*↓↑ ↑ operates identically across applications and thus would make the error from excusable ignorance.

Norman (1983) lists three ways to minimize mode errors.

1. Do not have modes.
2. Make sure that the modes are distinctively marked.
3. Make sure that the commands required by different modes are not the same, so that a command issued in the wrong mode will not lead to difficulty.

Of these three, only the first always prevents mode errors. As we have seen, the second works sporadically. The third does not decrease the number of errors, but it does reduce the penalty for erring.

Extreme techniques will usually draw a user's attention to a mode indicator, but by their intrusive strength, they redirect the user's locus of attention to the current state of the system and not on what he is trying to accomplish—a result that is as undesirable as the mode error that the indicators might prevent. Norman defined mode errors as mistakes that occur when a user misclassifies, or when the user makes an erroneous analysis of a situation (Norman 1981). The terms *misclassifies* and *analysis* hint at active, conscious participation on the part of the user and therefore apply while she

is unfamiliar with a command but do not apply after her use of it has become automatic.

3-2-1 Definition of Modes

If the definition of a mode is based exclusively on the design of the interface, as our definition has been up to now, all users would make the same errors, albeit with different frequencies. They do not. A given interface feature can be modal for one user and not modal for another. A more complete definition of modes must incorporate how the user views the interface: *A human-machine interface is* modal *with respect to a given gesture when (1) the current state of the interface is not the user's locus of attention and (2) the interface will execute one among several different possible responses to the gesture, depending on the system's current state.*

An interface can be modal with respect to one gesture and not modal with respect to a second gesture. *For an interface as a whole to be classified as not modal, it must not be modal for any gesture.*

A measure, Q, of how modal a particular interface is can be given by classifying each of the gestures in an interface as modal or not modal. Then, given the probability, $p(N_i)$, that a particular nonmodal gesture, N_i, is used, measured for a given user or averaged over a population of users, $Q = \sum_i p(N_i)$. Q ranges from 0, completely modal, to 1, completely nonmodal.

Both parts of the definition of a modal gesture are necessary to decide whether, for example, the gesture of pressing the Backspace key is modal. In most computer interfaces, assuming you are in the process of entering text, the Backspace command erases the most recently typed character. If that character was an *e*, the key erases an *e*. If that character was an *x*, the key erases an *x*. That is, sometimes Backspace is an *e* eraser, and sometimes it is an *x* eraser. Considering only the second part of the definition, the use of Backspace is modal because what it erases depends on the character most recently typed; content is part of system state. However, when you realize that your locus of attention is the object that you are erasing, it is the first part of the definition that explains why the operation is not modal and why you do not make mode errors when you use the Backspace key to erase characters in text or any other kind of object that you can select and delete in the same way.[3]

A command that sets whether the Backspace key operates as a forward or a reverse erase, on the other hand, does make the interface modal.

3. On the Macintosh, the key is, more accurately, labeled *Delete*.

The setting of erase direction, made earlier, is usually not your locus of attention when, subsequently, you use Backspace, so sometimes you will erase in the unexpected direction.

Preventing (Aircraft) Crashes by Eliminating a Mode

Due to simple interface design error, both remote-piloted vehicles (RPVs), as used by the military, and civilian radio-controlled model aircraft are occasionally destroyed. To understand the error and a method of solving it, you must first know something about how these devices work.

To fly an RPV, the operator, who stays safely on the ground, moves a small joystick embedded in a control box (Figure 3.2). Typically, you push the stick to pitch down the nose of the RPV, pull on the stick to pitch the nose temporarily upward, and move the stick to the left or right to cause the RPV to roll to the left or right, respectively. Moving the stick causes a proportional motion of a device called a servo in the RPV. The servo is linked mechanically to a control surface, such as the elevator in the rear of the RPV, that controls pitch.



Figure 3.2. A commercial RPV controller. Note the many controls scattered across the face and the top of the device.

A servo has no standard direction of motion with respect to a stick motion, so it is convenient and conventional to have a switch, located on the controller, that allows the user to choose which direction of stick motion corresponds to which direction of servo motion. This clearly modal convention runs into conflict, as modes usually do, with a habituated reaction on the part of the user. Often, one controller is used with multiple different RPVs. Each may have different servo directions corresponding to the actions of the aircraft.

When changing between different RPVs, the user must operate the switches that choose servo direction so that a given stick motion corresponds to the appropriate RPV action.

It is nearly impossible to train a pilot to operate the control sticks in reverse fashion when a servo reversal is unexpected. For example, consider that the pilot about to perform a takeoff with the RPV is unaware that the switch that controls the direction of the roll is in the wrong position. Taking off is a critical maneuver that requires rapt attention. It is common for the RPV to spontaneously roll slightly to the left or the right just after leaving the ground, and the expert pilot will automatically apply a small opposite control input to compensate for the roll; the reaction is so fast that, with a good pilot, a nonpilot observer will not detect the incipient roll. With a reversed-direction control switch, the roll, of course, will get worse, because the servo is moving in the wrong direction. On seeing this, a well-trained pilot will apply a stronger compensating stick motion that, because of the servo reversal, exacerbates the roll, and the RPV will, at this point in the flight, usually be demolished as soon as the wingtip contacts the ground or the RPV rolls entirely on its back. (This entire sequence can take a fraction of a second.) I have never seen a pilot successfully figure out the problem and reverse reactions in time to complete the takeoff. This kind of crash has happened to me on two occasions, in spite of a well-trained ritual of checking that control surface motion is in the correct direction before each takeoff. Checking control direction, after its having been done dozens of times, also becomes a habit, to the point that you don't always verify that the directions in which the control surfaces move are correct. (If they don't move at all, that is sufficiently surprising to alert you to the presence of a problem.) During these checks, I have more often than not caught reversed servos on RPVs I was going to fly. However, when under pressure or with other things on my mind, I, like hundreds of other pilots, have missed the reversal.

Interestingly, this error does not usually occur on a new aircraft's first flight, during which you are especially alert for potential problems. It is only after flying the particular craft has become "second nature," or habitual, that the problem strikes.

This is an example of a mode that cannot be removed from the world. Modes are inherent in the nature of servos and control sticks.

Yet the problem can be greatly ameliorated. The solution: Put the servo reversal switch in the aircraft, perhaps as part of the servos, and have no servo reversal switches on the controller. The interiors of RPVs are not normally accessible: The switches would be set during assembly of the RPV. Correct direction of operation would be checked by the assemblers, rechecked by inspectors, and checked again (usually) by the pilot prior to the first flight. This method does not eliminate the possibility of a crash, but once the first flight is completed, servo reversal on that RPV will not be a problem, no matter how many different RPVs the controller is used with in the interim. To the operator of the RPV, the mode is gone.

Nearly all model aircraft controllers have servo reversal switches. Crashes due to having the controller in the wrong mode are not uncommon. Worse still is the "feature" of many controllers to have other modes set by long-handled paddle switches, easily knocked to the wrong position by accident. On the controller I designed for my own use, all such switches are of a design such that the switch levers have to be pulled away from the controller against spring tension before they can be moved (Figure 3.3) I have never had an accident due to a switch being unknowingly pushed to the wrong position. Sometimes, the solution to a mode problem is mechanical.



Figure 3.3. These switches on an RPV controller cannot be operated accidentally; the handle must be pulled out before the switch can be operated.

Modes also restrict your scope of activity. If a gesture *g* invokes action *a* in mode *A* and action *b* in mode *B*, you must, if you are in mode *B* and want to invoke action *a*, first leave mode *B* and reset the interface into mode *A*. Only then can you use the gesture *g* to invoke *a*. The division of an interface into bounded areas is a necessary consequence of modes; the set of states in which the gesture *g* has a particular interpretation can be called the

range of *g*. Software sold as an application—for example, a spreadsheet—usually consists of one or more overlapping ranges. Certain ranges are relatively large. For example, the following sequence performs a cut in nearly all applications on both the Macintosh and Windows platforms:

Command↓ *x*↓

Other ranges are tiny. The following sequence in a certain computer game opens a treasure chest only when the chest is visible:

Command↓ *h*↓

Grouping commands into separate ranges, or what we usually call *applications*, can be an aid to understanding and using a complex interface, but there are ways to organize an interface that are less restrictive than are modes. *A fully humane interface would consist of exactly one range.*

When an interface is controlled by another machine, you might think that modelessness is of no importance, because machines have no problem remembering state; they just model it internally with a state of their own. However, if an interface is modal and the program operating it does not initially know the current state of the interface—for example, if it connects to the system after the system is already running—the program, prior to operating any modal control, must be provided with a means for testing that state. Interface toggles are particularly troublesome in this regard, where consecutive invocations of the toggling mechanism cycle the control through a number of states before returning to the initial state (the cycle then repeats).

This discussion of programs that operate interfaces is relevant to human-machine interface design because you might want to construct a set of stored commands that can be played back with a single gesture, a *macro*, which is a rudimentary form of computer program. A macro cannot set a toggle into a specific state unless the macro first interrogates the system to determine the system's current state. We saw this problem in the example of the flashlight in the duffel bag. One solution is to have any multiple-option switch always reset to a specified initial state immediately after it is operated. Then, counting the number of invocations of the switch always informs the operator of the switch's current state. More than about five states is excessive if the switching is to be done by a person. As mentioned earlier, another solution is to use radio buttons.

We have not yet exhausted what harm modes do. Modes can put the computer, instead of the user, in charge of an interaction. This aspect is especially noticeable when you are forced to stop what you are doing to

reply to a message box. Certain designers consider forcing the user to stop and to work in lockstep with a planned sequence to be an interface advantage, in that the system is "guiding" the user. There may be circumstances under which it is imperative that the user make a particular decision—if there is no user decision, there is no need for a dialogue at all—by a certain time or before doing another step in the sequence. In the first case, put up a count-down clock, but do not restrict the user from doing other operations with the system. In the second case, have a message stating that the decision must be made before the next step will be presented, but the system should not prevent the user from performing other operations that are not part of the programmed sequence. For example, what if the user needs to look up something in a file or do a calculation in order to answer the question? Guidance should be offered modelessly, so that the user remains in control of as much as possible of the system.

3-2-2 Modes, User-Preference Settings, and Temporary Modes

Facilities for setting user preferences constitute an example of modes and are a major source of user frustration. Ironically, these features are usually touted as a user benefit. Present interfaces are often so difficult to use that a user may feel an urge to rearrange them. Microsoft (1995, p. 4) specifically recommends that such facilities be provided: "Users, because of their widely varying skills and preferences, must be able to personalize aspects of the interface . . . such as color, fonts, or other options." On the other hand, a user of Microsoft Word described the effect of setting a preference as being like dropping a time bomb into the word processor. She needed to build a list in a format different from the one she usually used, so she looked up how to make the changes and chose new settings. The next time she wanted to build a list, she used the familiar List command and, of course, unexpectedly obtained the one she had reset Word to give her rather than the one she was used to. It took her more than an hour to figure out what had happened and to fix it. (Her first impulse was to believe that there was something wrong with the program or in her use of the command, and she repeated the command many times before she remembered that she had reset the preferences.)

Bob Fowles, of the Pennsylvania State University Computer Center, observed:

> People not aware of the complexity of Word can really get into trouble when they accidentally press Command, Option, or Control plus