

# Designing with the Mind in Mind

Simple Guide to  
Understanding  
User Interface  
Design Rules

**MK**  
MORGAN KAUFMANN

Jeff Johnson

# Designing with the Mind in Mind

Simple Guide to Understanding  
User Interface Design Rules

# Designing with the Mind in Mind

Simple Guide to Understanding  
User Interface Design Rules

**Jeff Johnson**



AMSTERDAM • BOSTON • HEIDELBERG • LONDON  
NEW YORK • OXFORD • PARIS • SAN DIEGO  
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann Publishers is an imprint of Elsevier



Morgan Kaufmann Publishers is an imprint of Elsevier  
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA

This book is printed on acid-free paper

© 2010 Elsevier Inc. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from the publisher. Details on how to seek permission, further information about the Publisher's permissions policies and our arrangements with organizations such as the Copyright Clearance Center and the Copyright Licensing Agency, can be found at our website: [www.elsevier.com/permissions](http://www.elsevier.com/permissions).

This book and the individual contributions contained in it are protected under copyright by the Publisher (other than as may be noted herein).

#### Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

#### Library of Congress Cataloging-in-Publication Data

Johnson, Jeff, Ph. D.

Designing with the mind in mind: simple guide to understanding user interface design rules / Jeff Johnson.

p. cm.

Includes bibliographical references and index.

ISBN 978-0-12-375030-3 (alk. paper)

1. Graphical user interfaces (Computer systems) I. Title.

QA76.9.U83J634 2010

005.4'37—dc22

2010001844

#### British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library.

ISBN: 978-0-12-375030-3

For information on all Morgan Kaufmann publications, visit our Web site at [www.mkp.com](http://www.mkp.com) or [www.elsevierdirect.com](http://www.elsevierdirect.com)

Typeset by MPS Limited, a Macmillan Company, Chennai, India  
[www.macmillansolutions.com](http://www.macmillansolutions.com)

Printed in China

10 11 12 13 5 4 3 2 1

Working together to grow  
libraries in developing countries

[www.elsevier.com](http://www.elsevier.com) | [www.bookaid.org](http://www.bookaid.org) | [www.sabre.org](http://www.sabre.org)

ELSEVIER

BOOK AID  
International

Sabre Foundation



# Acknowledgments

I could not have written this book without a lot of help and support.

First to mention are the students of the Human-Computer Interaction course I taught as an Erskine Fellow at the University of Canterbury in New Zealand in the winter semester of 2006. It was for them that I developed a lecture providing a brief background in perceptual and cognitive psychology—just enough to enable them to understand and apply user interface design guidelines. That lecture expanded into a professional development course, then into this book.

Second are the reviewers of the first draft: Susan Fowler, Robin Jeffries, Tim McCoy, and Jon Meads. They made many helpful comments and suggestions that allowed me to greatly improve the book.

Third are three cognitive science researchers who provided useful content, directed me to valuable readings, or allowed me to bounce ideas off of them: Prof. Edward Adelson (M.I.T. Dept. of Brain and Cognitive Sciences), Prof. Dan Osherson (Princeton University Dept. of Psychology), and Dr. Dan Bullock (Boston University Dept. of Cognitive and Neural Systems).

The book also was helped immeasurably by the care, oversight, logistical support, and nurturing provided by the staff at Elsevier, especially Mary James, David Bevans, and Andre Cuello.

Valuable additional copyediting was provided by Cate de Heer. Most importantly, I thank my wife and friend Karen Ande for her love and support while I was researching and writing this book, all the more remarkable because it coincided with the period when she was completing a book of her own: *Face To Face: Children of the AIDS Crisis in Africa*, a photography book documenting the plight of children orphaned by AIDS in sub-Saharan Africa ([FaceToFaceAfrica.com](http://FaceToFaceAfrica.com)).

# Foreword

The design of interactive computer systems is not only an art, but, at least aspirationally, a science. Well, not a science, actually, but rather a kind of joint computer-cognitive engineering, that is, science-based techniques to create interactive systems satisfying specified requirements.

Like cars, buildings, and clothes, interactive computing artifacts can emotionally delight, exhibit style and fashion, and have social significance. There is much room for art and industrial design in making things that pop, flash, and interact. But the resulting artifacts also have to work correctly and to flow with human activity. A beautiful building whose soaring windows roast its inhabitants in the summer or whose trusses buckle in a storm is a failure. Designers need methods to put latitude, season, fenestration, volume, and circulation together to predict heating loads before building the building. They also need a stockpile of technology component solutions, like thermopane glass, blinds, overhangs, and fans to choose among as part of the standard engineering of a solution. Engineering does not replace art in a design, it makes it possible.

Engineering is hard enough for architecture; it is harder still for interactive artifacts, for the simple reason that it is easier to get a science of buildings than one of people. Providing such a supporting science and engineering has been a founding aspiration of the field of human-computer interaction. How to do it? The most basic method is by “usability testing”—watch users doing tasks, discover their difficulties, and fix these through redesign. Usability testing is useful, necessary, and inefficient. The results don’t cumulate very well into a discipline anything like engineering, and it isn’t very insightful about why things break. It’s the cognitive equivalent of roasting the users to find the effect of the large windows. But usability testing can find many of a system’s flaws. It is a feasible method, because interactive systems are often much easier to change than rebuilding a building.

Better would be to avoid many of the errors in the first place, and one method is through *design rules*. Instead of rediscovering over and over through usability testing that interfaces depending on red and green are bad for color-blind users, just make it a design rule to use color redundantly with other cues. Design rules, however, turn out to have their own problems. In practice, design rules may be ambiguous or require subtle interpretation of context or contradict other guidelines. And that brings us to the current book.

The idea of the present book is to unite design rules with the supporting cognitive and perceptual science that is at their core. This format has several merits: the psychological science is made concrete and easy to absorb by connecting to actual designs, and the design rules are made easier to adjust for context, since they are related to their deeper rationale.

Jeff Johnson is the perfect author to attempt such a book. His whole career has combined work on both the interface design side and the psychological science **ix**

side. I first met him when he was on the user interface team of the Xerox Star series of products—the first commercial graphical user interface. So on the design side, he was essentially in at the beginning of GUIs. On the psychology side, he did degrees at Yale and Stanford. Putting design and psychology together, he worked on commercial interactive systems, taught at the university, and worked as a consultant. His trademark is using concrete design examples to illustrate abstract principles. In fact, he is famous for driving his points home memorably by exhibiting “blooper” examples of bad designs—and so he does in this book.

There is a third method of using science to help engineer a system that goes beyond design rules—*design models*. Jeff’s book shows examples of how to use this method, too. He shows how to model the task context in terms of object and actions and how to understand real-time interaction constraints.

In sum, this is a book that advances the goal of a supporting engineering method for interactive system design. At the same time, it is a primer to understand the *why* of the larger human action principles at work—a sort of cognitive science for designers in a hurry. Above all, this is a book of profound insight into the human mind for practical people who want to get something done.

—Stuart Card

# Introduction

---

## USER-INTERFACE DESIGN RULES: WHERE DO THEY COME FROM AND HOW CAN THEY BE USED EFFECTIVELY?

For as long as people have been designing interactive computer systems, some have attempted to promote *good* design by publishing user-interface design guidelines (also called design rules). Early ones included:

- **Cheriton** (1976) proposed user-interface design guidelines for early interactive (time-shared) computer systems.
- **Norman** (1983a, 1983b) presented design rules for software user interfaces based on human cognition, including cognitive errors.
- **Smith and Mosier** (1986) wrote perhaps the most comprehensive set of user-interface design guidelines.
- **Shneiderman** (1987) included “Eight Golden Rules of Interface Design” in the first edition of his book *Designing the User Interface* and in all later editions.
- **Brown** (1988) wrote a book of design guidelines, appropriately titled *Human-Computer Interface Design Guidelines*.
- **Nielsen and Molich** (1990) offered a set of design rules for use in heuristic evaluation of user interfaces.
- **Marcus** (1991) presented guidelines for graphic design in online documents and user interfaces.

In the twenty-first century, additional user interface design guidelines have been offered by Stone *et al.* (2005), Koyani, Bailey, and Nall (2006), Johnson (2007), and Shneiderman and Plaisant (2009). Microsoft, Apple Computer, and Oracle publish guidelines for designing software for their platforms (Apple Computer, 2009; Microsoft Corporation, 2009; Oracle Corporation/Sun Microsystems, 2001).

How valuable are user-interface design guidelines? That depends on who applies them to design problems.

---

## USER EXPERIENCE DESIGN AND EVALUATION REQUIRES UNDERSTANDING AND EXPERIENCE

Following user-interface design guidelines is not as straightforward as following cooking recipes. Design rules often describe goals rather than actions. They are purposefully

very general to make them broadly applicable, but that means that their exact meaning and their applicability to specific design situations is open to interpretation.

Complicating matters further, more than one rule will often seem applicable to a given design situation. In such cases, the applicable design rules often conflict, i.e., they suggest different designs. This requires designers to determine which competing design rule is more applicable to the given situation and should take precedence.

Design problems—even without competing design guidelines—often have multiple conflicting goals. e.g.:

- bright screen *and* long battery life
- lightweight *and* sturdy
- multifunctional *and* easy to learn
- powerful *and* simple
- WYSIWIG (what you see is what you get) *and* usable by blind people

Satisfying all the design goals for a computer-based product or service usually requires tradeoffs—lots and lots of tradeoffs. Finding the right balance point between competing design rules requires further tradeoffs.

Given all of these complications, user-interface design rules and guidelines must be applied thoughtfully, not mindlessly, by people who are skilled in the art of UI design and/or evaluation. User-interface design rules and guidelines are more like *laws* than like *rote recipes*. Just as a set of laws is best applied and interpreted by lawyers and judges who are well versed in the laws, a set of user-interface design guidelines is best applied and interpreted by people who understand the basis for the guidelines and have learned from experience in applying them.

Unfortunately, with a few exceptions (e.g., Norman, 1983a), user-interface design guidelines are provided as simple lists of design edicts with little or no rationale or background.

Furthermore, although many early members of the user-interface design and usability profession had backgrounds in cognitive psychology, most newcomers to the field do not. That makes it difficult for them to apply user-interface design guidelines sensibly.

Providing that rationale and background education is the focus of this book.

---

## COMPARING USER-INTERFACE DESIGN GUIDELINES

Table I.1 places the two best-known user-interface guideline lists side by side to show the types of rules they contain and how they compare to each other (see the Appendix for additional guidelines lists). For example, both lists start with a rule calling for consistency in design. Both lists include a rule about preventing errors. The Nielsen-Molich rule “Help users recognize, diagnose, and recover from errors” corresponds closely to the Shneiderman-Plaisant rule to “Permit easy reversal of actions.” “User control and freedom” corresponds to “Make users feel they are in control.” There is a reason for this similarity, and it isn’t just that later authors were influenced by earlier ones.

**Table I.1** The Two Best-Known Lists of User Interface Design Guidelines

<b>Shneiderman (1987); Shneiderman and Plaisant (2009)</b>	<b>Nielsen and Molich (1990)</b>
<ul style="list-style-type: none"> <li>• Strive for consistency</li> <li>• Cater to universal usability</li> <li>• Offer informative feedback</li> <li>• Design task flows to yield closure</li> <li>• Prevent errors</li> <li>• Permit easy reversal of actions</li> <li>• Make users feel <i>they</i> are in control</li> <li>• Minimize short-term memory load</li> </ul>	<ul style="list-style-type: none"> <li>• Consistency and standards</li> <li>• Visibility of system status</li> <li>• Match between system and real world</li> <li>• User control and freedom</li> <li>• Error prevention</li> <li>• Recognition rather than recall</li> <li>• Flexibility and efficiency of use</li> <li>• Aesthetic and minimalist design</li> <li>• Help users recognize, diagnose, and recover from errors</li> <li>• Provide online documentation and help</li> </ul>

## WHERE DO DESIGN GUIDELINES COME FROM?

For present purposes, the detailed design rules in each set of guidelines, such as those in [Table I.1](#), are less important than what they have in common: their basis and origin. Where did these design rules come from? Were their authors—like clothing fashion designers—simply trying to impose their own personal design tastes on the computer and software industries?

If that were so, the different sets of design rules would be very different from each other as the various authors sought to differentiate themselves from the others. In fact, all of these sets of user-interface design guidelines are quite similar if we ignore differences in wording, emphasis, and the state of computer technology when each set was written. Why?

The answer is that all of the design rules are based on human psychology: how people perceive, learn, reason, remember, and convert intentions into action. Many authors of design guidelines had at least some background in psychology that they applied to computer system design.

For example, Don Norman was a professor, researcher, and prolific author in the field of cognitive psychology long before he began writing about human-computer interaction. Norman's early human-computer design guidelines were based on research—his own and others'—on human cognition. He was especially interested in cognitive errors that people often make and how computer systems can be designed to lessen or eliminate the impact of those errors.

Similarly, other authors of user-interface design guidelines—e.g., Brown, Shneiderman, Nielsen, and Molich—used knowledge of perceptual and cognitive psychology to try to improve the design of usable and useful interactive systems.

Bottom line: user-interface design guidelines are based on human psychology.

By reading this book, you will learn the most important aspects of the psychology underlying user-interface and usability design guidelines.

---

## **INTENDED AUDIENCE OF THIS BOOK**

This book is intended mainly for software development professionals who have to apply user-interface and interaction design guidelines. This of course includes interaction designers, user-interface designers, and user-experience designers, graphic designers, and hardware product designers. It also includes usability testers and evaluators, who often refer to design heuristics when reviewing software or analyzing observed usage problems.

A second audience for this book is software development managers who want enough of a background in the psychological basis for user-interface design rules to understand and evaluate the work of the people they manage.

# We Perceive What We Expect

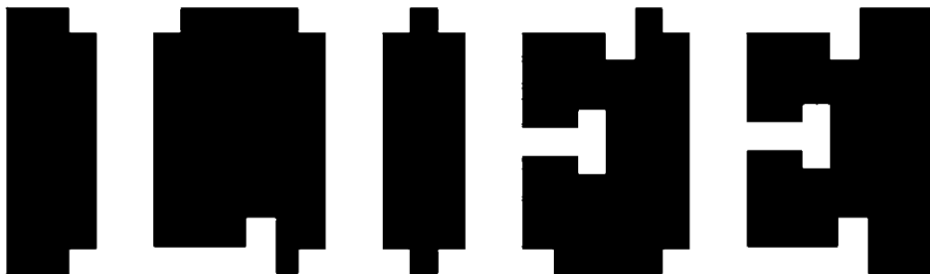
# 1

Our perception of the world around us is not a true depiction of what is actually there. We perceive, to a large extent, what we *expect* to perceive. Our expectations—and therefore our perceptions—are biased by three factors:

- *the past*: our experience
- *the present*: the current context
- *the future*: our goals

## PERCEPTION BIASED BY EXPERIENCE

Imagine that you own a large insurance company. You are meeting with a real estate manager, discussing plans for a new campus of company buildings. The campus consists of a row of five buildings, the last two with T-shaped courtyards providing light for the cafeteria and fitness center. If the real estate manager showed you the map shown in [Figure 1.1](#), you would see five black shapes representing the buildings.



**FIGURE 1.1**

Building map or word? What you see depends on what you were told to see.



## 2 CHAPTER 1 We Perceive What We Expect

Now imagine that instead of a real estate manager, you are meeting with an advertising manager. You are discussing a new billboard ad to be placed in certain markets around the country. The advertising manager shows you the same image, but in this scenario the image is a sketch of the ad, consisting of a single word. In this scenario, you see a word, clearly and unambiguously.

When your perceptual system has been primed to see building shapes, you see building shapes, and the white areas between the buildings barely register in your perception. When your perceptual system has been primed to see text, you see text, and the black areas between the letters barely register.

A relatively famous example of how priming the mind can affect perception is a sketch, supposedly by R. C. James,<sup>1</sup> that initially looks to most people like a random splattering of ink (see Fig. 1.2). Before reading further, look at the sketch.



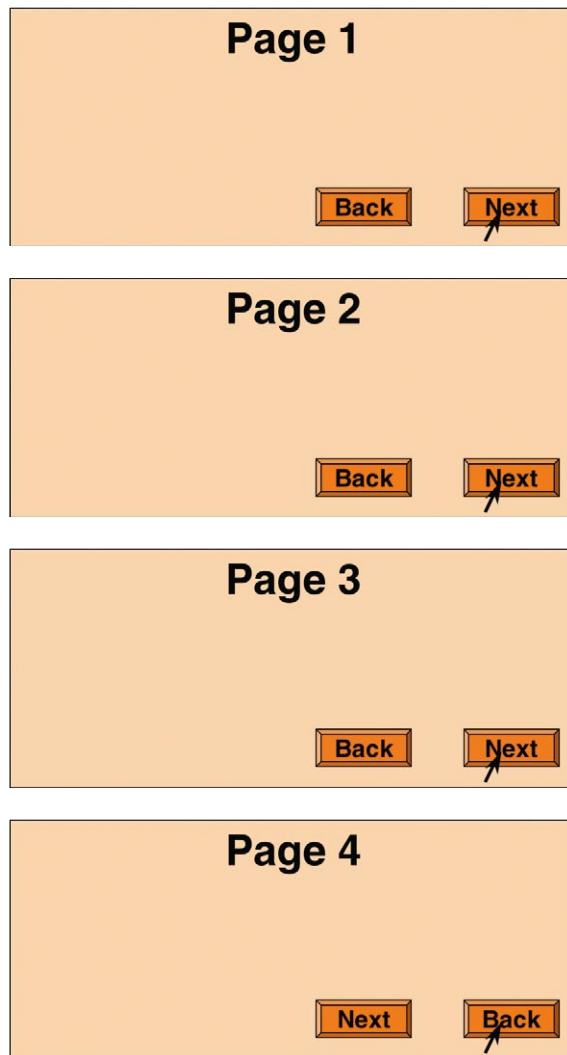
**FIGURE 1.2**

Image showing the effect of mental priming of the visual system. What do you see?

Only after you are told that it is a Dalmatian dog sniffing the ground near a tree can your visual system organize the image into a coherent picture. Moreover, once you've "seen" the dog, it is hard to go back to seeing the image as a random collection of spots.

---

<sup>1</sup>Published in Marr D. (1982) *Vision*. W. H. Freeman, New York, NY, p. 101, Figure 3-1.



**FIGURE 1.3**

The “Next” button is perceived to be in a consistent location, even when it isn’t.

The examples above are visual. Experience can also bias other types of perception, such as sentence comprehension. For example, the headline “New Vaccine Contains Rabies” would probably be understood differently by people who had recently heard stories about contaminated vaccines than by people who had recently heard stories about successful uses of vaccines to fight diseases.

Users of computer software and Web sites often click buttons or links without looking carefully at them. Their perception of the display is based more on what

their past experience leads them to expect than on what is actually on the screen. This sometimes confounds software designers, who expect users to see what is on the screen. But that isn't how perception works.

For example, if the positions of the “Next” and “Back” buttons on the last page of a multipage dialog box<sup>2</sup> switched, many people would not immediately notice the switch (see Fig. 1.3). Their visual system would have been lulled into inattention by the consistent placement of the buttons on the prior several pages. Even after unintentionally going backward a few times, they might continue to perceive the buttons in their standard locations. This is why “place controls consistently” is a common user interface design guideline.

Similarly, if we are trying to find something, but it is in a different place or looks different from usual, we might miss it even though it is in plain view because experience tunes us to look for expected features in expected locations. For example, if the “Submit” button on one form in a Web site is shaped differently or is a different color from those on other forms on the site, users might not find it. This expectation-induced blindness is discussed further later in this chapter, in the section on how our *goals* affect perception.

---

### PERCEPTION BIASED BY CURRENT CONTEXT

When we try to understand how our visual perception works, it is tempting to think of it as a bottom-up process, combining basic features such as edges, lines, angles, curves, and patterns into figures and ultimately into meaningful objects. To take reading as an example, you might assume that our visual system first recognizes shapes as letter and then combines letters into words, words into sentences, and so on.

But visual perception—reading in particular—is not strictly a bottom-up process. It includes top-down influences too. For example, the word in which a character appears may affect how we identify the character (see Fig. 1.4).

Similarly, our overall comprehension of a sentence or of a paragraph can even influence what words we see in it. For example, the same letter sequence can be



THE    CHT

**FIGURE 1.4**

The same character is perceived as H or A depending on the surrounding letters.

---

<sup>2</sup>Multi step dialog boxes are called “wizards” in user interface jargon.

**Fold napkins. Polish silverware. Wash dishes.**

**French napkins. Polish silverware. German dishes.**

**FIGURE 1.5**

The same phrase is perceived differently depending on the list it appears in.

read as different words depending on the meaning of the surrounding paragraph (see Fig. 1.5).

This biasing of perception by the surrounding context works *between* different senses too. Perceptions in any of our five senses may affect simultaneous perceptions in any of our other senses. For example:

- What we see can be biased by what we are hearing, and *vice versa*
- What we feel with our tactile sense can be biased by what we are hearing, seeing, or smelling

Later chapters explain how visual perception, reading, and recognition function in the human brain. For now, I will simply say that the pattern of neural activity that corresponds to recognizing a letter, a word, a face, or any object includes input from neural activity stimulated by the context. This context includes other nearby perceived objects and events, and even reactivated memories of previously perceived objects and events.

Context biases perception not only in people but also in lower animals. A friend of mine often brought her dog with her in her car when running errands. One day, as she drove into her driveway, a cat was in the front yard. The dog saw it and began barking. My friend opened the car door and the dog jumped out and ran after the cat, which turned and jumped through a bush to escape. The dog dove into the bush but missed the cat. The dog remained agitated for some time afterward.

Thereafter, for as long as my friend lived in that house, whenever she arrived at home with her dog in the car, he would get excited, bark, jump out of the car as soon as the door was opened, dash across the yard, and leap into the bush. There was no cat, but that didn't matter. Returning home in the car was enough to make the dog see one—perhaps even smell one. However, walking home, as the dog did after being taken for his daily walk, did not evoke the “cat mirage.”

## PERCEPTION BIASED BY GOALS

In addition to being biased by our *past* experience and the *present* context, our perception is influenced by our goals and plans for the *future*. Specifically, our goals filter our perceptions: things unrelated to our goals tend to be filtered out pre-consciously, never registering in our conscious minds.

For example, when people navigate through software or a Web site, seeking information or a specific function, they don't read carefully. They scan screens quickly

and superficially for items that seem related to their goal. They don't simply *ignore* items unrelated to their goals; they often don't even *notice* them.

To see this, flip briefly to the next page and look in the toolbox (Fig. 1.6) for *scissors*, and then immediately flip back to this page. Try it now.

Did you spot the scissors? Now, without looking back at the toolbox, can you say whether there is a screwdriver in the toolbox too?

Our goals filter our perceptions in other perceptual senses as well as in vision. A familiar example is the “cocktail party” effect. If you are conversing with someone at a crowded party, you can focus your attention to hear mainly what he or she is saying even though many other people are talking near you. The more interested you are in the conversation, the more strongly your brain filters out surrounding chatter. If you are bored by what your conversational partner is saying, you will probably hear much more of the conversations around you.

The effect was first documented in studies of air-traffic controllers, who were able to carry on a conversation with the pilots of their assigned aircraft even though many different conversations were occurring simultaneously on the same radio frequency, coming out of the same speaker in the control room (Arons, 1992). Research suggests that our ability to focus on one conversation among several simultaneous ones depends not only on our interest level in the conversation but also on objective factors such as the similarity of voices in the cacophony, the amount of general “noise” (e.g., clattering dishes or loud music), and the predictability of what your conversational partner is saying (Arons, 1992).

This filtering of perception by our goals is particularly true for adults, who tend to be more focused on goals than children are. Children are more stimulus driven: their perception is less filtered by their goals. This characteristic makes them more distractible than adults, but it also makes them less biased as observers.

A parlor game demonstrates this age difference in perceptual filtering. It is similar to the “look in the toolbox” exercise. Most households have a catch-all drawer for kitchen implements or tools. From your living room, send a visitor to the room where the catch-all drawer is, with instructions to fetch you a specific tool, such as measuring spoons or a pipe wrench. When the person returns with the tool, ask whether another specific tool was in the drawer. Most adults will not know what else was in the drawer. Children—if they can complete the task without being distracted by all the cool stuff in the drawer—will often be able to tell you more about what else was there.

Perceptual filtering can also be seen in how people navigate Web sites. Suppose I put you on the home page of New Zealand's University of Canterbury (see Fig. 1.7) and asked you to print out a map of the campus showing the computer science department. You would scan the page and probably quickly click one of the links that share words with the goal that I gave you: *Departments* (top left), *Departments and Colleges* (middle left), or *Campus Maps* (bottom right). If you're a “search” person, you might instead go right to the Search box (middle right), type words related to the goal, and click “Go.”



FIGURE 1.6

Toolbox: Are there scissors here?

UC Home | Courses | Departments | Library | Teaching | Research | Students | Contacts | Search

UC UNIVERSITY OF CANTERBURY Te Whare Wānanga o Waitaha CHRISTCHURCH NEW ZEALAND

University of Canterbury

UC Home

**For**

- Prospective Students
- Current Students
- International Students
- Visitors and Community
- Business and Industry
- Alumni and Friends
- Staff

**About**

- Courses and Subjects
- Departments and Colleges
- Library
- Teaching and Learning
- Research
- Contacts
- The University

**Welcome to the University of Canterbury**  
Nau Mai, Haere Mai ki te Whare Wānanga o Waitaha

**Mid-Year Enrolment**

- Apply now for mid-year
- Change/add courses at UC Student Web
- Courses starting July

*Stand Up*  
Don't wait until 2007!  
Start your degree this July

**Announcements**

- Mid-Year Examinations 2006
- Proposed Merger between UC and CCE
- UCi3, ICT Innovation Institute
- Information Days 4 - 5 July 2006

**News**

Young UC researchers win prestigious science prizes (8 June 2006)

UC research into replicating the spider web process to produce new fibres, generating power using wind at Scott Base and creating technologies to detect chemical weapon agents have been recognised at the annual MacDiarmid Young Scientists of the Year Awards.

**Search**

**Student Profiles**

Tim Kerr  
More Profiles

**Quick Links**

- UC Student Web
- Intranet
- Key Dates
- Enrolment
- UC Diary
- Campus Maps

**Congratulations!** You have been randomly chosen to receive 100 dollars. Claim at Bursars office.

FIGURE 1.7

University of Canterbury home page: Navigating Web sites includes perceptual filtering.

Whether you browse or search, it is likely that you would leave the home page without noticing that you were randomly chosen to win \$100 (bottom left). Why? Because that was not related to your *goal*.

What is the mechanism by which our current goals bias our perception? There are two:

- ***Influencing where we look.*** Perception is active, not passive. We constantly move our eyes, ears, hands, and so on, so as to sample exactly the things in our environment that are most relevant to what we are doing or about to do (Ware, 2008). If we are looking on a Web site for a campus map, our eyes and pointer-controlling hand are attracted to anything that might lead us to that goal. We more or less ignore anything unrelated to our goal.
- ***Sensitizing our perceptual system to certain features.*** When we are looking for something, our brain can prime our perception to be especially sensitive to features of what we are looking for (Ware, 2008). For example, when we are looking for a red car in a large parking lot, red cars will seem to pop out as we scan the lot, and cars of other colors will barely register in our consciousness, even though we do in some sense “see” them. Similarly, when we are trying to find our spouse in a dark, crowded room, our brain “programs” our auditory system to be especially sensitive to the combination of frequencies that make up his or her voice.

---

## DESIGN IMPLICATIONS

All these sources of perceptual bias of course have implications for user interface design. Here are three.

### Avoid ambiguity

Avoid ambiguous information displays, and test your design to verify that all users interpret the display in the same way. Where ambiguity is unavoidable, either rely on standards or conventions to resolve it, or prime users to resolve the ambiguity in the intended way.

For example, computer displays often shade buttons and text fields to make them look raised in relation to the background surface (see [Fig. 1.8](#)). This appearance



**FIGURE 1.8**

Buttons on computer screens are often shaded to make them look three dimensional, but the convention only works if the simulated light source is assumed to be on the upper left.



relies on a convention, familiar to most experienced computer users, that the light source is at the top left of the screen. If an object were depicted as lit by a light source in a different location, users would not see the object as raised.

### **Be consistent**

Place information and controls in consistent locations. Controls and data displays that serve the same function on different pages should be placed in the same position on each page on which they appear. They should also have the same color, text fonts, shading, and so on. This consistency allows users to spot and recognize them quickly.

### **Understand the goals**

Users come to a system with goals they want to achieve. Designers should understand those goals. Realize that users' goals may vary, and that their goals strongly influence what they perceive. Ensure that at every point in an interaction, the information users need is available, prominent, and maps clearly to a possible user goal, so users will notice and use the information.



# Our Vision is Optimized to See Structure

# 2

Early in the twentieth century, a group of German psychologists sought to explain how human visual perception works. They observed and catalogued many important visual phenomena. One of their basic findings was that human vision is holistic: Our visual system automatically imposes structure on visual input and is wired to perceive whole shapes, figures, and objects rather than disconnected edges, lines, and areas. The German word for “shape” or “figure” is *Gestalt*, so these theories became known as the Gestalt principles of visual perception.

Today’s perceptual and cognitive psychologists regard the Gestalt theory of perception as more of a *descriptive* framework than an *explanatory* and *predictive* theory. Today’s theories of visual perception tend to be based heavily on the neurophysiology of the eyes, optic nerve, and brain (see Chapters 4–7).

Not surprisingly, the findings of neurophysiological researchers support the observations of the Gestalt psychologists. We really are—along with other animals—“wired” to perceive our surroundings in terms of whole objects (Stafford & Webb, 2005; Ware, 2008). Consequently, the Gestalt principles are still valid—if not as a fundamental explanation of visual perception, at least as a framework for describing it. They also provide a useful basis for guidelines for graphic and user interface design (Soegaard, 2007).

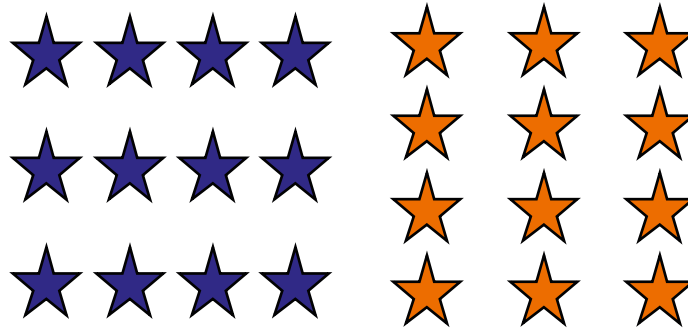
For present purposes, the most important Gestalt principles are: Proximity, Similarity, Continuity, Closure, Symmetry, Figure/Ground, and Common Fate. In the following sections, I describe each principle and provide examples from both static graphic design and user interface design.

## GESTALT PRINCIPLE: PROXIMITY

The principle of *Proximity* is that the relative distance between objects in a display affects our perception of whether and how the objects are organized into subgroups. Objects that are near each other (relative to other objects) appear grouped, while those that are farther apart do not.

In Figure 2.1, the stars on the left are closer together horizontally than they are vertically, so we see three rows of stars, while the stars on the right are closer together vertically than they are horizontally, so we see three columns.

The Proximity principle has obvious relevance to the layout of control panels or data-forms in software, Web sites, and electronic appliances. Designers often separate groups of on-screen controls and data-displays by enclosing them in group boxes or by placing separator lines between groups (see Fig. 2.2).



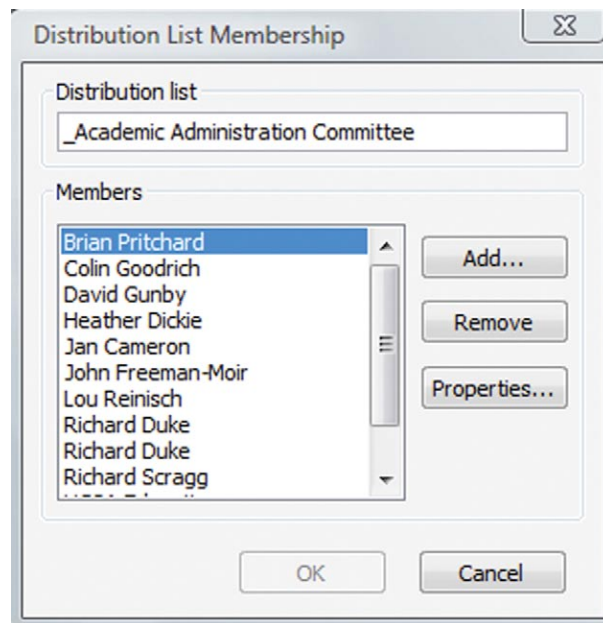
**FIGURE 2.1**

Proximity: Items that are closer appear grouped. Left: rows, Right: columns.



**FIGURE 2.2**

In Outlook's Distribution List Membership dialog box, list buttons are in a group box, separate from the window-control buttons.



However, according to the Proximity principle, items on a display can be visually grouped simply by spacing them closer together to each other than to other controls, without group boxes or visible borders (see Fig. 2.3). Many graphic design experts recommend this approach in order to reduce visual clutter and code size in a user interface (Mullet & Sano, 1994).

Conversely, if controls are *poorly* spaced, e.g., if connected controls are too far apart, people will have trouble perceiving them as related, making the software harder to learn and remember. For example, the Discreet Software Installer displays six horizontal pairs of radiobuttons, each representing a two-way choice, but their spacing, due to the Proximity principle, makes them appear to be two vertical sets of radiobuttons, each representing a six-way choice, at least until users try them and learn how they operate (see Fig. 2.4).

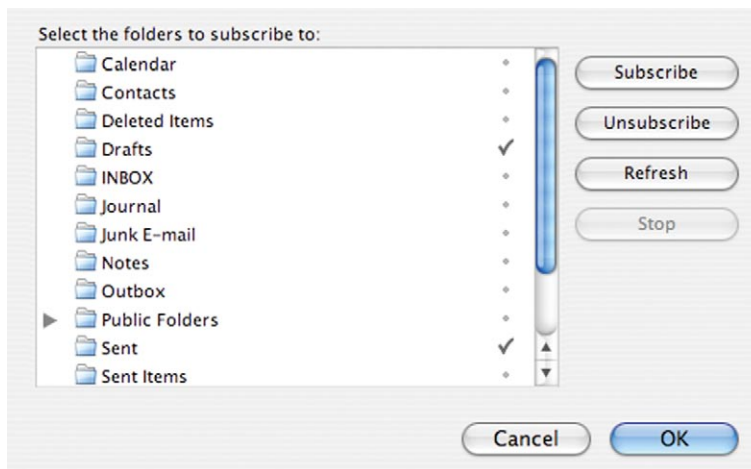


FIGURE 2.3

In Mozilla Thunderbird's Subscribe Folders dialog box, controls are grouped using the Proximity principle.



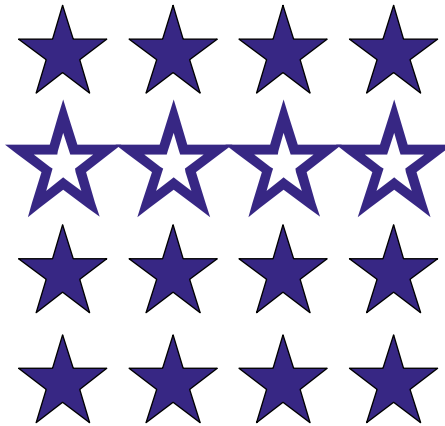
FIGURE 2.4

In Discreet's Software Installer, poorly spaced radiobuttons look grouped in vertical columns.

## GESTALT PRINCIPLE: SIMILARITY

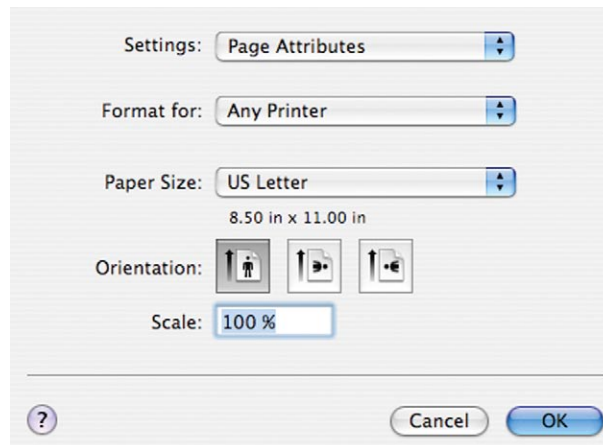
Another factor that affects our perception of grouping is expressed in the Gestalt principle of *Similarity*: Objects that look similar appear grouped, all other things being equal. In [Figure 2.5](#), the slightly larger, “hollow” stars are perceived as a group.

The Page Setup dialog box in Mac OS applications uses the Similarity and Proximity principles to convey groupings (see [Fig. 2.6](#)). The three very similar and tightly spaced



**FIGURE 2.5**

Similarity: Items appear grouped if they look more similar to each other than to other objects.



**FIGURE 2.6**

Mac OS Page Setup dialog box: The Similarity and Proximity principles are used to group the Orientation settings.

Orientation settings are clearly intended to appear grouped. The three menus are not so tightly spaced but look similar enough that they appear related even though that probably wasn't intended.

Similarly, the text fields in a form at the Web site of book publisher Elsevier are organized into an upper group of seven (with three subgroups) for the address, a group of three split fields for phone numbers, and two single text fields. The four menus, in addition to being data fields, help separate the text field groups (see Fig. 2.7). By contrast, the labels are too far from their fields to seem connected to them.

The image shows a screenshot of a web form with the following elements:

- Title (Mr, Ms, Dr etc): A dropdown menu with the text "\*\*Please Select\*\*".
- First name: A single-line text input field.
- Last name: A single-line text input field.
- Job title: A single-line text input field.
- Institution/Organisation: A single-line text input field.
- Number and Street: A single-line text input field.
- City: A single-line text input field.
- State/County: A single-line text input field.
- Zip Code/Postal Code: A single-line text input field.
- Country: A dropdown menu with the text "\*\*Please Select\*\*".
- Work phone: Two adjacent single-line text input fields.
- Home phone: Two adjacent single-line text input fields.
- Fax: Two adjacent single-line text input fields.
- How did you find out about this Web site: A dropdown menu with the text "Please select".
- Other: A single-line text input field.
- Please select the option which most closely describes you as a customer: A dropdown menu with the text "Please select".
- E-mail: A single-line text input field.

**FIGURE 2.7**

Online form at Elsevier.com: Similarity makes the text fields appear grouped.

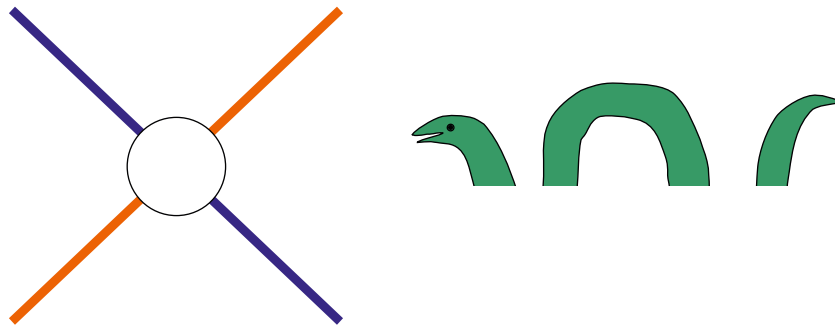
## GESTALT PRINCIPLE: CONTINUITY

In addition to the two Gestalt principles concerning our tendency to organize objects into groups, several Gestalt principles describe our visual system's tendency to resolve ambiguity or fill in missing data in such a way as to perceive whole objects. The first such principle, the principle of *Continuity*, states that our visual perception is biased to perceive continuous forms rather than disconnected segments.

For example, on the left side of [Figure 2.8](#), we automatically see two crossing lines—one blue and one orange. We don't see two separate orange segments and two separate blue ones, and we don't see a blue-and-orange V on top of an upside-down orange-and-blue V. On the right side of [Figure 2.8](#), we see a sea monster in water, not three pieces of one.

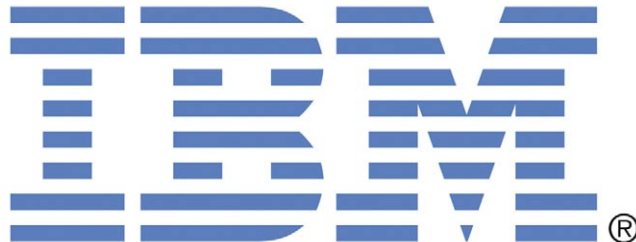
A well-known example of the use of the Continuity principle in graphic design is the IBM® logo. It consists of disconnected blue patches, and yet it is not at all ambiguous; it is easily seen as three bold letters, perhaps viewed through something like venetian blinds (see [Fig. 2.9](#)).

Slider controls are a user-interface example of the Continuity principle. We see a slider as depicting a single range controlled by a handle that appears somewhere on the slider, not as two separate ranges separated by the handle (see [Fig. 2.10A](#)). Even displaying different colors on each side of a slider's handle doesn't completely “break” our perception of a slider as one continuous object, although ComponentOne's choice of strongly contrasting colors (gray vs. red) certainly strains that perception a bit (see [Fig. 2.10B](#)).



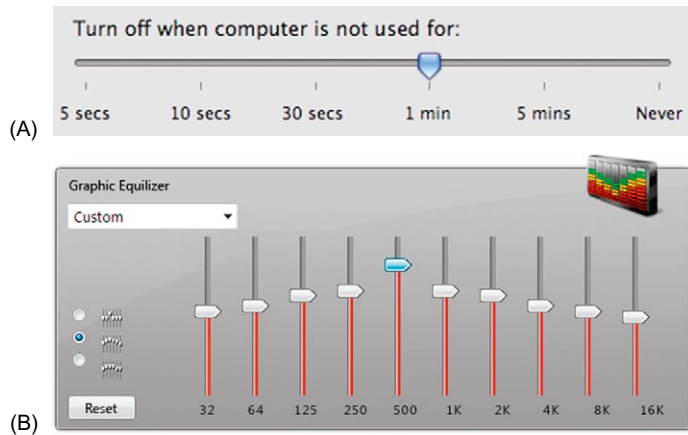
**FIGURE 2.8**

Continuity: Human vision is biased to see continuous forms, even adding missing data if necessary.



**FIGURE 2.9**

The IBM company logo uses the Continuity principle to form letters from disconnected patches.

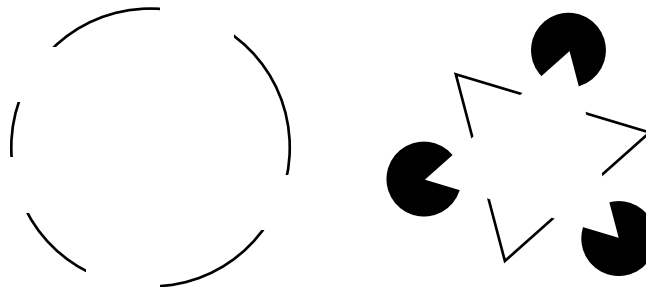


**FIGURE 2.10**

Continuity: We see a slider as a single slot with a handle somewhere on it, not as two slots separated by a handle. (A) Mac OS, (B) ComponentOne.

## GESTALT PRINCIPLE: CLOSURE

Related to Continuity is the Gestalt principle of *Closure*, which states that our visual system automatically tries to close open figures so that they are perceived as whole objects rather than separate pieces. Thus, we perceive the disconnected arcs on the left of Figure 2.11 as a circle.



**FIGURE 2.11**

Closure: Human vision is biased to see whole objects, even when they are incomplete.

Our visual system is so strongly biased to see objects that it can even interpret a totally blank area as an object. We see the combination of shapes on the right of Figure 2.11 as a white triangle overlapping another triangle and three black circles, even though the figure really only contains three V-shapes and three black Pac-men.

The Closure principle is often applied in graphical user interfaces (GUIs). For example, GUIs often represent collections of objects—e.g., documents or messages—as *stacks* (see Fig. 2.12). Just showing one whole object and the edges of others “behind” it is enough to make users perceive a stack of objects, all whole.



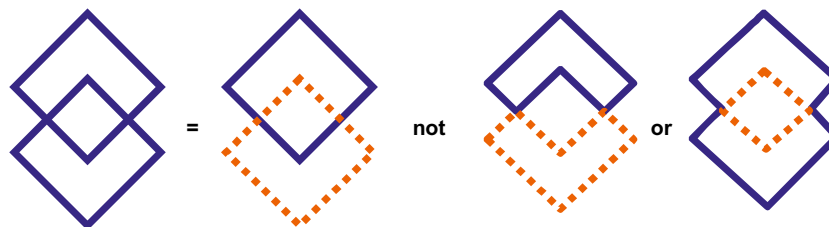
**FIGURE 2.12**

Icons depicting stacks of objects exhibit the Closure principle: partially visible objects are perceived as whole.

## GESTALT PRINCIPLE: SYMMETRY

A third fact about our tendency to see objects is captured in the Gestalt principle of *Symmetry*. It states that we tend to parse complex scenes in a way that reduces the complexity. The data in our visual field usually has more than one possible interpretation, but our vision automatically organizes and interprets the data so as to simplify it and give it symmetry.

For example, we see the complex shape on the left of Figure 2.13 as two overlapping diamonds, not as two touching corner bricks or a pinch-waist octahedron with a square in its center. A pair of overlapping diamonds is simpler than the other two interpretations shown on the right of Figure 2.13: it has fewer sides and more symmetry than the other two interpretations.



**FIGURE 2.13**

Symmetry: The human visual system tries to resolve complex scenes into combinations of simple, symmetrical shapes.

In printed graphics and on computer screens, our visual system’s reliance on the symmetry principle can be exploited to represent three dimensional objects on a two dimensional display. This can be seen in a cover illustration for Paul Thagard’s book *Coherence in Thought and Action* (Thagard, 2002; see Fig. 2.14) and in three-dimensional depiction of a cityscape (see Fig. 2.15).





**FIGURE 2.14**

The cover of the book *Coherence in Thought and Action* (Thagard, 2002) uses the Symmetry, Closure, and Continuity principles to depict a cube.



**FIGURE 2.15**

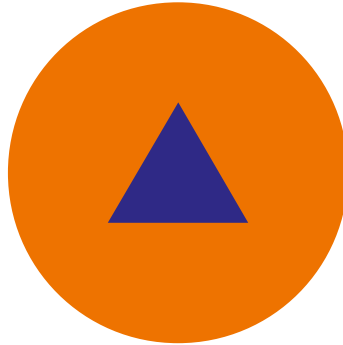
Symmetry: The human visual system parses very complex two dimensional images into three dimensional scenes.

## GESTALT PRINCIPLE: FIGURE/GROUND

The next Gestalt principle that describes how our visual system structures the data it receives is *Figure/Ground*. This principle states that our mind separates the visual field into the figure (the foreground) and ground (the background). The foreground

consists of those elements of a scene that are the object of our primary attention, and the background is everything else.

The Figure/Ground principle also specifies that the visual system's parsing of scenes into figure and ground is influenced by characteristics of the scene. For example, when a small object or color patch overlaps a larger one, we tend to perceive the smaller object as the figure and the larger object as the ground (see Fig. 2.16).



**FIGURE 2.16**

Figure/Ground: When objects overlap, we see the smaller as figure on ground.

However, our perception of figure vs. ground is not completely determined by scene characteristics. It also depends on the viewer's focus of attention. Dutch artist M. C. Escher exploited this phenomenon to produce ambiguous images in which figure and ground switch roles as our attention shifts (see Fig. 2.17).



**FIGURE 2.17**

M. C. Escher exploited figure/ground ambiguity in his art.

In user interface and Web design, the Figure/Ground principle is often used to place an impression-inducing background “behind” the primary displayed content (see Fig. 2.18). The background can convey information—e.g., the user’s current location—or it can suggest a theme, brand, or mood for interpretation of the content.



FIGURE 2.18

Figure/Ground is used at AndePhotos.com to display a thematic watermark “behind” content.

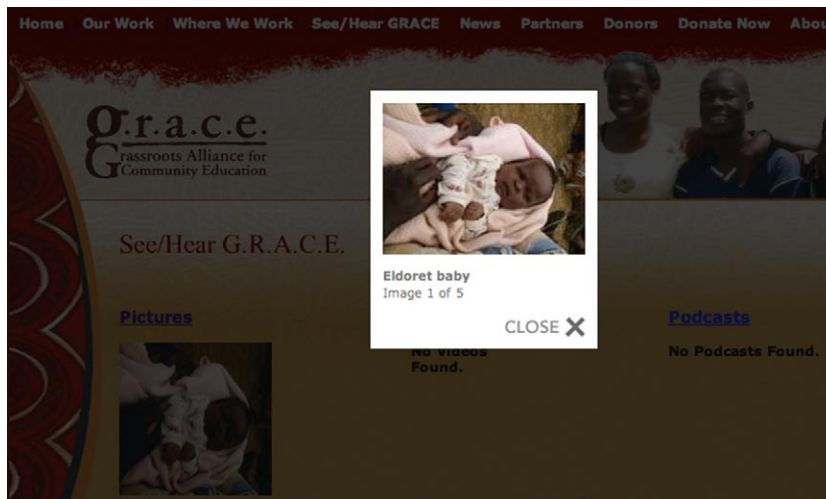


FIGURE 2.19

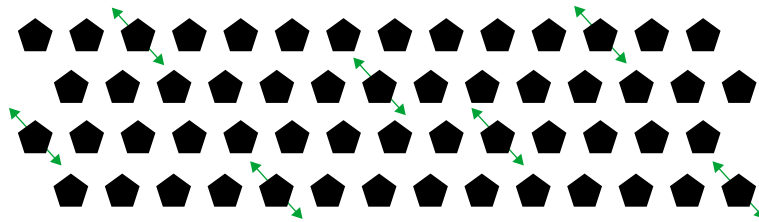
Figure/Ground is used at GRACEUSA.org to pop up a photo “over” the page content.

Figure/Ground is also often used to pop up information over other content. Content that was formerly the figure—the focus of the users’ attention—temporarily becomes the *background* for new information, which appears briefly as the new *figure* (see Fig. 2.19). This approach is usually better than temporarily *replacing* the old information with the new information, because it provides context that helps keep people oriented regarding their place in the interaction.

## GESTALT PRINCIPLES: COMMON FATE

The previous six Gestalt principles concerned perception of static (un-moving) figures and objects. One final Gestalt principle—Common Fate—concerns moving objects. The Common Fate principle is related to the Proximity and Similarity principles: Like them it affects whether we perceive objects as grouped. The Common Fate principle states that objects that move together are perceived as grouped or related.

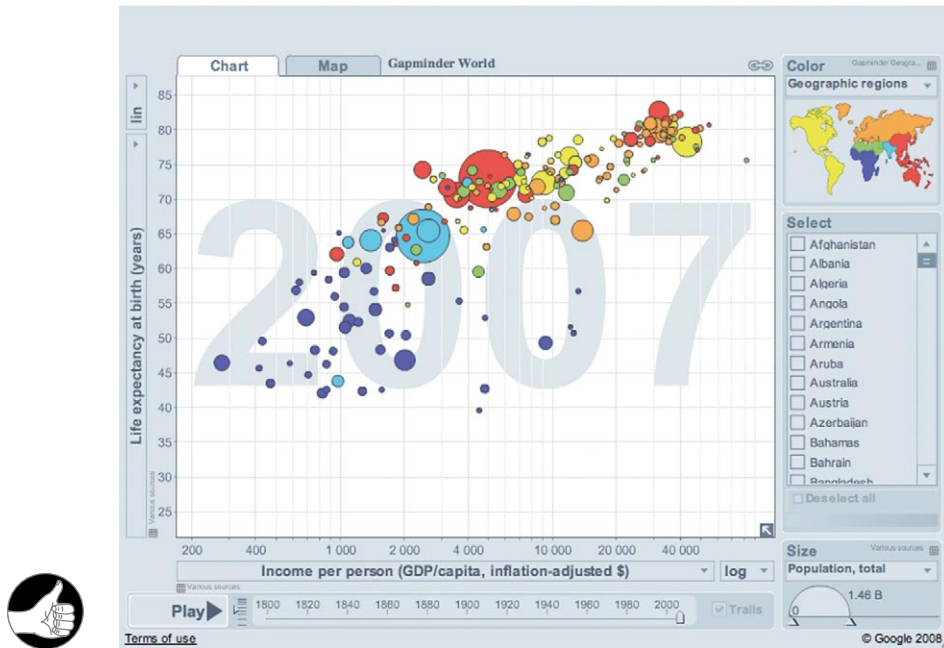
For example, in a display showing dozens of pentagons, if seven of them wiggled back and forth in synchrony, people would see them as a related group, even if the wiggling pentagons were separated from each other and looked no different from all the other pentagons (see Fig. 2.20).



**FIGURE 2.20**

Common Fate: Items appear grouped or related if they move together.

Common motion—implying common fates—is used in some animations to show relationships between entities. For example, GapMinder graphs animate dots representing nations to show changes over time in various factors of economic development. Countries that move together share development histories (see Fig. 2.21).



**FIGURE 2.21**

Common fate: GapMinder animates dots to show which nations have similar development histories.

## GESTALT PRINCIPLES: COMBINED

Of course, in real-world visual scenes, the Gestalt principles work in concert, not in isolation. For example, a typical Mac OS desktop usually exemplifies six of the seven principles described above (excluding Common Fate): Proximity, Similarity, Continuity, Closure, Symmetry, and Figure/Ground (see Fig. 2.22). On a typical desktop, Common Fate is used (along with similarity) when a user selects several files or folders and drags them as a group to a new location (see Fig. 2.23).

With all these Gestalt principles operating at once, *unintended* visual relationships can be implied by a design. A recommended practice, after designing a display, is to view it with each of the Gestalt principles in mind—Proximity, Similarity, Continuity, Closure, Symmetry, Figure/Ground, and Common Fate—to see if the design suggests any relationships between elements that you do *not* intend.

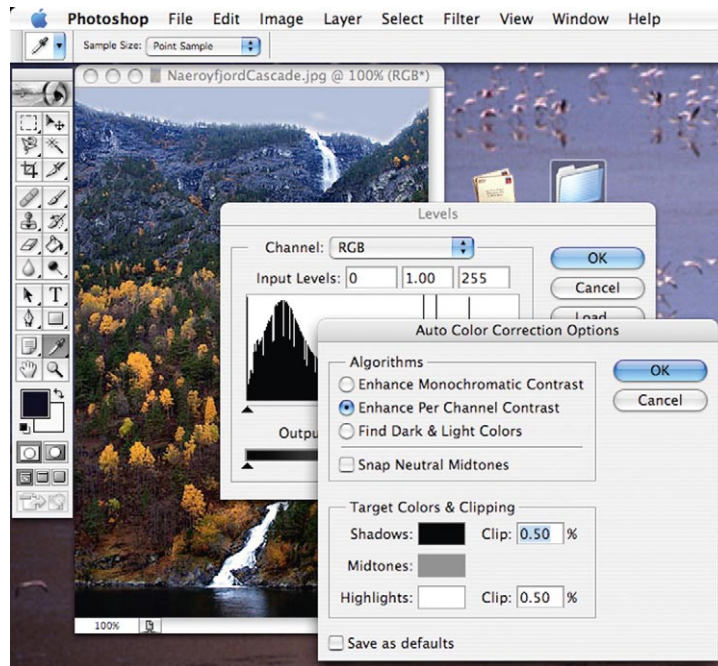


FIGURE 2.22

All of the Gestalt principles except Common Fate play a role in this portion of a Mac OS desktop.

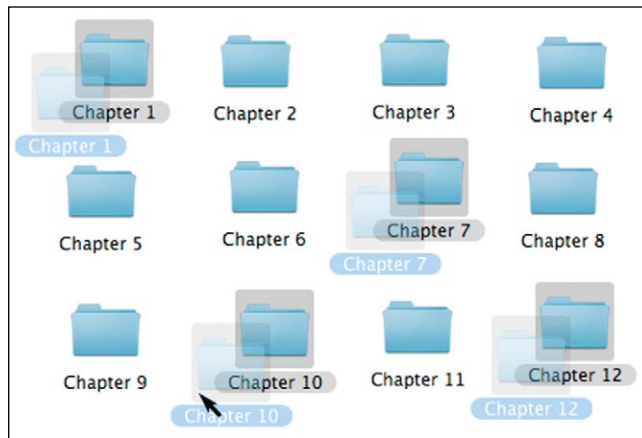


FIGURE 2.23

Similarity and Common Fate: When users drag folders that they have selected, common highlighting and motion make the selected folders appear grouped.



# We Seek and Use Visual Structure

# 3

Chapter 2 used the Gestalt principles of visual perception to show how our visual system is optimized to perceive structure. Perceiving structure in our environment helps us make sense of objects and events quickly. Chapter 2 also mentioned that when people are navigating through software or Web sites, they don't scrutinize screens carefully and read every word. They scan quickly for relevant information. This chapter presents examples to show that when information is presented in a terse, structured way, it is easier for people to scan and understand.

Consider two presentations of the same information about an airline flight reservation. The first presentation is unstructured prose text; the second is structured text in outline form (see Fig. 3.1). The structured presentation of the reservation can be scanned and understood much more quickly than the prose presentation.

The more structured and terse the presentation of information, the more quickly and easily people can scan and comprehend it. Look at the Contents page from the California Department of Motor Vehicles (see Fig. 3.2). The wordy, repetitive links slow users down and "bury" the important words they need to see.



**FIGURE 3.1**

Structured presentation of airline reservation information is easier to scan and understand.

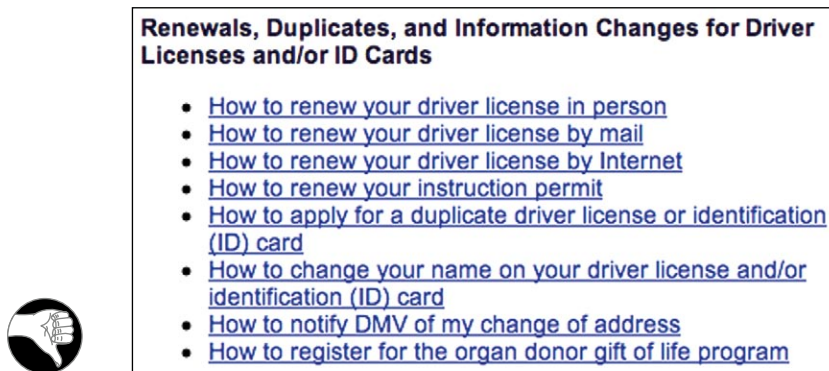


FIGURE 3.2

Contents page at the California Department of Motor Vehicles (DMV) Web site buries the important information in repetitive prose.

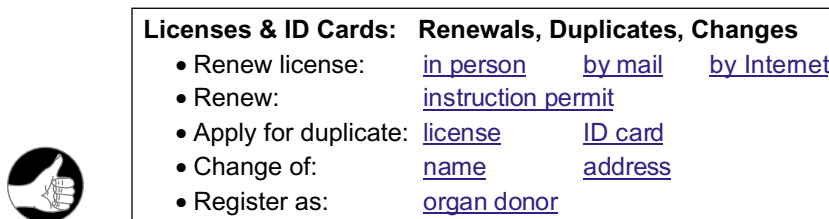


FIGURE 3.3

The California DMV Web site Contents page with repetition eliminated and better visual structure.

Compare that with a terser, more structured hypothetical design that factors out needless repetition and marks as links only the words that represent options (see Fig. 3.3). All options presented in the actual Contents page are available in the revision, yet it consumes less screen space and is easier to scan.

Displaying search results is another situation in which structuring data and avoiding repetitive “noise” can improve people’s ability to scan quickly and find what they seek. In 2006, search results at HP.com included so much repeated navigation data and metadata for each retrieved item that they were useless. By 2009 HP had eliminated the repetition and structured the results, making them easier to scan and more useful (see Fig. 3.4).

Of course, for information displays to be easy to scan, it is not enough merely to make them terse, structured, and nonrepetitious. They must also conform to the rules of graphic design, some of which were presented in Chapter 2.

For example, a prerelease version of a mortgage calculator on a real estate Web site presented its results in a table that violated at least two important rules of graphic design (see Fig. 3.5, left). People usually read (on- or offline) from top to



Search:

**Search results**

- » Buy HP Color LaserJet 1600 Printer(CB373A#ABA), HP color laser printers, HP LaserJets, HP printers, Printers direct from the ...
- » Buy HP Color LaserJet 1600 Printer(CB373A#ABA), HP color laser printers, HP LaserJets, HP printers, Printers direct from the HP Home & Home Office Store. On a budget but want ... 2006-12-05
- » Buy HP Color LaserJet 4700dn Printer(Q7493A#ABA), HP color laser printers, HP LaserJets, HP printers, Printers direct from ...
- » Buy HP Color LaserJet 4700dn Printer(Q7493A#ABA), HP color laser printers, HP LaserJets, HP printers, Printers direct from the HP Home & Home Office Store. Get ready to deliver ... 2006-12-20
- » Buy HP Color LaserJet 4700n Printer(Q7492A#ABA), HP color laser printers, HP LaserJets, HP printers, Printers direct from ...
- » Buy HP Color LaserJet 4700n Printer(Q7492A#ABA), HP color laser printers, HP LaserJets, HP printers, Printers direct from the HP Home & Home Office Store. Classy, brilliantly ... 2006-12-20
- » Buy supplies for Color laser specialty paper, Paper for laser printers, Paper direct from the HP Home & Home Office Store
- » Buy supplies for Color laser specialty paper, Paper for laser printers, Paper direct from the HP Home & Home Office Store. Hewlett Packard computer and printer store. Shop for, ... 2006-12-09
- » Buy supplies for Color laser specialty paper, Paper for laser printers, Paper, Printing supplies direct from the HP Home & Home Office Store
- » Buy supplies for Color laser specialty paper, Paper for laser printers, Paper, Printing supplies direct from the HP Home & Home Office Store. Hewlett Packard computer and printer ... 2006-12-06



Search:

**Search results**

- » HP® Official Store — Buy an HP Tri-fold Color Laser Glossy Brochure Paper (150 sheets, 8.5 x 11-inch) (Q6612A) from HP Why outsource brochure printing when you can do it expertly on your own laser printer, in small batches tailored to your unique projects and clients? Ideal for use with HP Color ...
- » Color Laser Printers At a glance - HP Small & Medium Business products Summary of all HP Color Laser Printers currently available for purchase and recommended for Small & Medium Business. Includes links to compare products, obtain more information about a ...
- » HP Color laserjets - color laser printers for large business - HP Large Enterprise Business HP Large Enterprise Business - Find and compare color laser printers. Review the business features of HP Color LaserJets
- » HP® Official Store — Color laser speciality paper, Paper for laser printers, Paper Use these papers with your HP Color LaserJet printer for outsource-quality marketing materials and photos
- » HP Press Release: Hewlett-Packard Color LaserJet 4550 Family Establishes New Generation of Internet-enabled Color Laser ... Successor to the World's Best-selling Color Laser Printer Features Embedded Web Technologies Designed to Improve Ease of Use and Manageability.
- » Laser Printers, Inkjet Printers, Photo Printers - HP Choose from a wide variety of dependable HP printers, including laser printers and ink printers, all in one/multifunction printers and photo printers.



FIGURE 3.4

In 2006, HP.com's site search produced repetitious, noisy results (left) but by 2009 was improved (right).

Mortgage Summary	
<b>\$1,840.59</b>	<b>\$662,611.22</b>
Monthly Payment	Total of 360 Payments
<b>\$318,861.22</b>	<b>Sep, 2037</b>
Total Interest Paid	Pay-off Date
<b>\$93,750.00</b>	<b>\$0.00</b>
Total Tax Paid	Total PMI Paid



Mortgage Summary	
Monthly Payment	<b>\$ 1,840.59</b>
Number of Payments	<b>360</b>
Total of Payments	<b>\$ 662,611.22</b>
Interest Total	<b>\$ 318,861.22</b>
Tax Total	<b>\$ 93,750.00</b>
PMI Total	<b>\$ 0.00</b>
Pay-off Date	<b>Sep 2037</b>



FIGURE 3.5

Left: Mortgage summary presented by a software mortgage calculator. Right: Improved design.

bottom, but the labels for calculated amounts were *below* their corresponding values. Second, the labels were just as close to the value below as to their own value, so proximity (see Chapter 2) could not be used to perceive that labels were grouped with their values. To understand this mortgage results table, users had to scrutinize it carefully and slowly figure out which labels went with which numbers.

The revised design, in contrast, allows users to perceive the correspondence between labels and values without conscious thought (see Fig. 3.5, right).

## STRUCTURE ENHANCES PEOPLE'S ABILITY TO SCAN LONG NUMBERS

Even small amounts of information can be made easier to scan if they are structured. Two examples are telephone numbers and credit card numbers (see Fig. 3.6 and Fig. 3.7). Traditionally, such numbers were broken into parts to make them easier to scan and remember.

A long number can be broken up in two ways: either the user interface breaks it up explicitly by providing a separate field for each part of the number, or the interface provides a single number field, but lets users break the number into parts with spaces or punctuation (see Fig. 3.8a). However, many of today's computer presentations of phone and credit card numbers do not segment the numbers and do not

Easy:	(415) 123-4567
Hard:	4151234567
Easy:	1234 5678 9012 3456
Hard:	1234567890123456

FIGURE 3.6

Telephone and credit card numbers are easier to scan and understand when segmented.

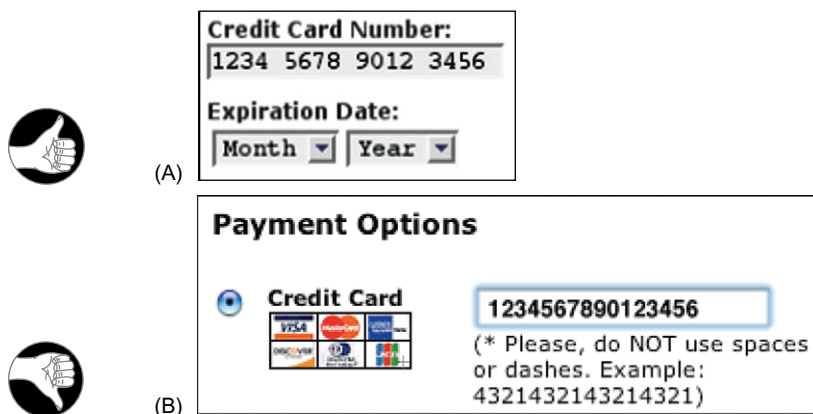


FIGURE 3.7

(A) At Democrats.org, credit card numbers can include spaces. (B) At Stuffit.com, they cannot, making them harder to scan and verify.



#### Date of Birth

You must be at least 18 years of age and either a United States citizen or a permanent resident of the U.S., or at least 21 years of age and a permanent resident of Puerto Rico.

/  /  MM/DD/YYYY

FIGURE 3.8

BankOfAmerica.com: Segmented data fields provide useful structure.

allow users to do it with spaces (see Fig. 3.8b). This limitation makes it harder for people to scan a number or verify that they typed it correctly.

Segmenting data fields can provide useful visual structure even when the data to be entered is not, strictly speaking, a number. Dates are an example of a case in which segmented fields can improve readability and help prevent data entry errors, as shown by a date field at Bank of America's Web site (see Fig. 3.8).

## DATA-SPECIFIC CONTROLS PROVIDE EVEN MORE STRUCTURE

A step up in structure from segmented data fields are data-specific controls. Instead of using simple text fields—whether segmented or not—designers can use controls that are designed specifically to display (and accept as input) a value of a specific type. For example, dates can be presented (and accepted) in the form of menus combined with pop-up calendar controls (see Fig. 3.9).

It is also possible to provide visual structure by mixing segmented text fields with data-specific controls, as demonstrated by an email address field at Southwest Airlines' Web site (see Fig. 3.10).



Depart

Oct 21

Morning

FIGURE 3.9

NWA.com: Dates are displayed and entered using a control that is specifically designed for dates.



E-mail Address: fred @ bedrock . com

FIGURE 3.10

NWA.com: Dates are displayed and entered using a control that is specifically designed for dates.

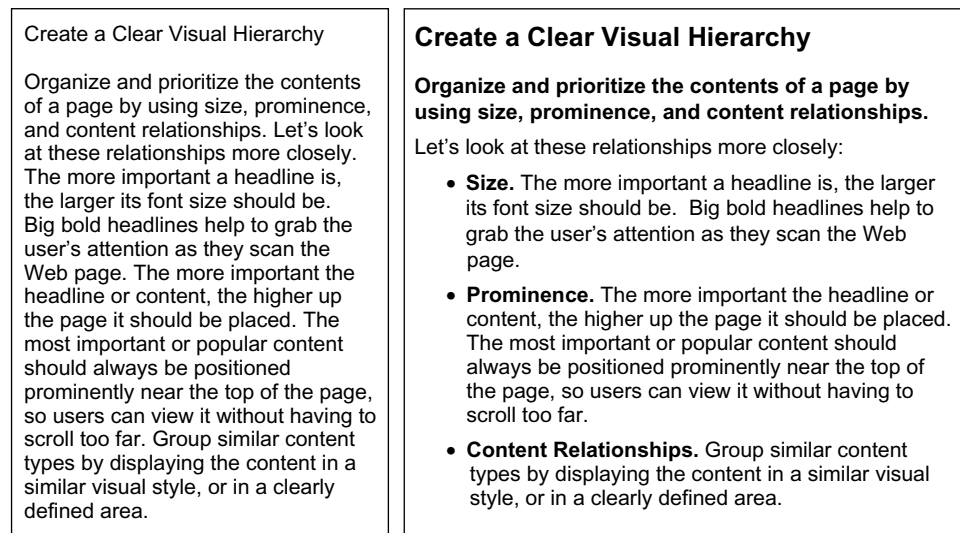
## VISUAL HIERARCHY LETS PEOPLE FOCUS ON THE RELEVANT INFORMATION

One of the most important goals in structuring information presentations is to provide a visual hierarchy—an arrangement of the information that:

- Breaks the information into distinct sections, and breaks large sections into subsections
- Labels each section and subsection prominently and in such a way as to clearly identify its content
- Presents the sections and subsections as a hierarchy, with higher level sections presented more strongly than lower level ones

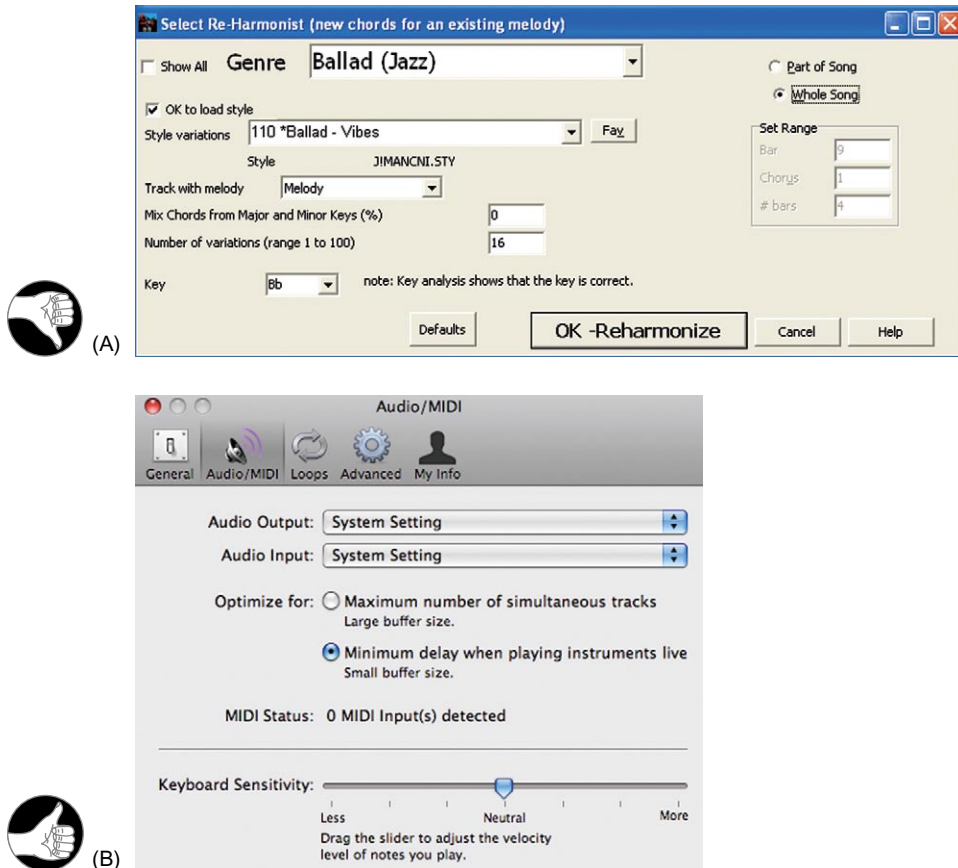
A visual hierarchy allows people, when scanning information, to separate what is relevant to their goals from what is irrelevant instantly, and to focus their attention on the relevant information. They find what they are looking for more quickly because they can easily skip everything else.

Try it for yourself. Look at the two information displays in [Figure 3.11](#) and find the information about prominence. How much longer does it take you to find it in the nonhierarchical presentation?



**FIGURE 3.11**

Find the advice about prominence in each of these displays. Prose text format (left) makes people read everything. Visual hierarchy (right) lets people ignore information irrelevant to their goals.



**FIGURE 3.12**

Visual hierarchy in interactive control panels and forms lets users find settings quickly. (A) Band in a Box (bad), (B) GarageBand (good).

The examples in Figure 3.11 show the value of visual hierarchy in a textual, read-only information display. Visual hierarchy is equally important in interactive control panels and forms—perhaps even more so. Compare dialog boxes from two different music software products (see Fig. 3.12). The Reharmonize dialog box of Band-in-a-Box has poor visual hierarchy, making it hard for users to find things quickly. In contrast, GarageBand's Audio/MIDI control panel has good visual hierarchy, so users can quickly find the settings they are interested in.

# Reading is Unnatural

# 4

Most people in industrialized nations grew up in households and school districts that promoted education and reading. They learned to read as young children and became good or excellent readers by adolescence. As adults, most of our activities during a normal day involve reading. The process of reading—deciphering words into their meaning—is for most educated adults automatic, leaving our conscious minds free to ponder the meaning and implications of what we are reading. Because of this background, it is common for good readers to consider reading to be as “natural” a human activity as speaking is.

## WE’RE WIRED FOR LANGUAGE, BUT NOT FOR READING

Speaking and understanding spoken language *is* a natural human ability, but reading is *not*. Over hundreds of thousands—perhaps millions—of years, the human brain evolved the neural structures necessary to support spoken language. As a result, normal humans are born with an innate ability to learn as toddlers, with no systematic training, whatever language they are exposed to. After early childhood, our innate ability to learn spoken languages decreases significantly. By adolescence, learning a new language is the same as learning any other skill: it requires instruction and practice, and the learning and processing are handled by different brain areas from those that handled it in early childhood (Sousa, 2005).

In contrast, *writing* and *reading* did not exist until a few thousand years BC and did not become *common* until only four or five centuries ago—*long* after the human brain had evolved into its modern state. At no time during childhood do our brains show any special innate ability to learn to read. Instead, reading is an artificial skill that we learn by systematic instruction and practice, like playing a violin, juggling, or reading music (Sousa, 2005).

Because people are not innately “wired” to learn to read, children who either lack caregivers who read to them, or who receive inadequate reading instruction in school may never learn to read. There are many such people, especially in the developing world. By comparison, very few people never learn a spoken language.

For a variety of reasons, even people who learn to read may never become good at it. Perhaps their parents did not value and promote reading. Perhaps they attended substandard schools or didn’t attend school at all. Perhaps they learned a second language but never learned to read well in that language. Finally, people who have cognitive or perceptual impairments such as dyslexia may never become good readers.

Learning to read involves training our brain—including our visual system—to recognize patterns. The patterns that our brain learns to recognize run a gamut from low level to high level:

- Lines, contours, and shapes are basic visual *features* that our brain recognizes innately. We don’t have to learn to recognize them.
- Basic features combine to form patterns that we learn to identify as characters—letters, numeric digits, and other standard symbols. In ideographic scripts, such as Chinese, symbols represent entire words or concepts.
- In alphabetic scripts, patterns of characters form *morphemes*, which we learn to recognize as packets of meaning, e.g., “farm,” “tax,” “-ed,” and “-ing” are morphemes in English.
- Morphemes combine to form patterns that we recognize as *words*, e.g., “farm,” “tax,” “-ed,” and “-ing” can be combined to form the words “farm,” “farmed,” “farming,” “tax,” “taxed,” and “taxing.” Even ideographic scripts include symbols that serve as morphemes or modifiers of meaning rather than as words or concepts.
- Words combine to form *patterns* that we learn to recognize as phrases, idiomatic expressions, and sentences.
- Sentences combine to form *paragraphs*.

To see what text looks like to someone who has not yet learned to read, just look at a paragraph printed in a language and script that you do not know (see Fig. 4.1A and B).

Alternatively, you can approximate the feeling of illiteracy by taking a page written in a familiar script and language—such as a page of this book—and turning it upside down. Turn this book upside down and try reading the next few paragraphs. This exercise only approximates the feeling of illiteracy. You will discover that the inverted text appears foreign and illegible at first, but after a minute you will be able to read it, albeit slowly and laboriously.

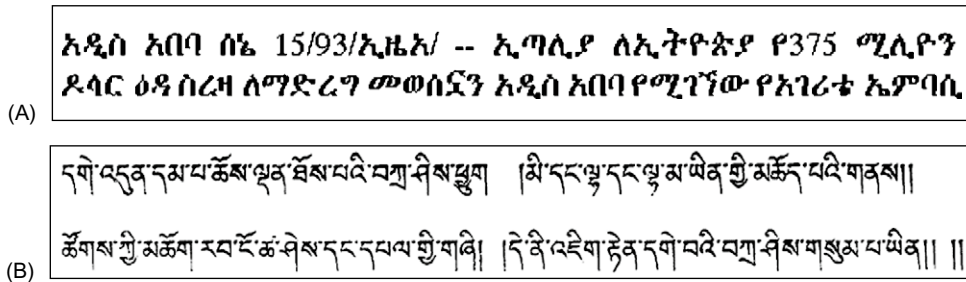


FIGURE 4.1

To see how it feels to be illiterate, look at text printed in a foreign script. (A) Amharic, (B) Tibetan.

## IS READING FEATURE-DRIVEN OR CONTEXT-DRIVEN?

As stated earlier, reading involves recognizing features and patterns. Pattern recognition, and therefore reading, can be either a bottom-up, feature-driven process or a top-down, context-driven process.

In feature-driven reading, the visual system starts by identifying simple features—line segments in a certain orientation or curves of a certain radius—on a page or display and then combines them into more complex features, such as angles, multiple curves, shapes, and patterns. Then the brain recognizes certain shapes as characters or symbols representing letters, numbers, or, for ideographic scripts, words. In alphabetic scripts, groups of letters are perceived as morphemes and words. In all types of scripts, sequences of words are parsed into phrases, sentences, and paragraphs that have meaning.

Feature-driven reading is sometimes referred to as “bottom-up” or “context-free.” The brain’s ability to recognize basic features: lines, edges, angles, etc.—is built in and therefore automatic from birth. In contrast, recognition of morphemes, words, and phrases has to be learned. It starts out as a non-automatic, conscious process requiring conscious analysis of letters, morphemes, and words, but with enough practice it becomes automatic (Sousa, 2005). Obviously, the more common a morpheme, word, or phrase, the more likely it is that recognition of it will become automatic. With ideographic scripts such as Chinese, which have many times more symbols than alphabetic scripts have, people typically take many years longer to become skilled readers.

Context-driven or top-down reading operates in parallel with feature-driven reading but it works the opposite way: from whole sentences or the gist of a paragraph *down* to the words and characters. The visual system starts by recognizing high-level patterns like words, phrases, and sentences, or by knowing the text’s meaning in advance. It then uses that knowledge to figure out—or guess—what the components of the high-level pattern must be (Boulton, 2009). Context-driven reading is less likely to become fully automatic because most phrase-level and sentence-level patterns and contexts don’t occur frequently enough to allow their recognition to become burned into neural firing patterns. But there are exceptions, such as idiomatic expressions.



To experience context-driven reading, glance down quickly at [Figure 4.2](#) (below), then immediately direct your eyes back here and finish reading this paragraph. Try it now. What did the text say?

Now look at the same sentence again more carefully. Do you read it the same way now?

It has been known for decades that reading involves both feature-driven (bottom-up) processing and context-driven (top-down) processing. In addition to being able to figure out the meaning of a sentence by analyzing the letters and words in it, people can determine the words of a sentence by knowing the meaning of the sentence, or the letters in a word by knowing what word it is (see [Fig. 4.3](#)). The question is: is skilled reading primarily bottom-up or top-down, or is neither mode dominant? Which type of reading is preferred?

Educational researchers in the 1970s applied information theory to reading, and assumed that because of redundancies in written language, top-down, context-driven reading would be faster than bottom-up, feature-driven reading. This assumption led them to hypothesize that reading in highly skilled (fast) readers would be dominated by context-driven (top-down) processing. This theory was probably responsible for many speed-reading methods of the 1970s and 1980s, which supposedly trained people to read fast by taking in whole phrases and sentences at a time.

The rain in Spain falls  
manly in the the plain

**FIGURE 4.2**

Top-down “recognition” of this expression may inhibit awareness of its actual content.

(A) Mray had a ltlte lmab, its feclee was withe as sown. And ervey  
wehre taht Mray wnet, the lmab was srue to go.

(B) Twinkle twinkle little star, how I wonder what you are

**FIGURE 4.3**

Top-down reading: Most readers, especially those who know the songs from which these text passages are taken, can read these passages even though the words (A) have all but their first and last letters scrambled and (B) are mostly obscured.

However, empirical studies of readers conducted since then have demonstrated conclusively that the truth is the opposite of what the earlier theory predicted. Reading researcher Keith Stanovich explains:

---

**... Context [is] important, but it's a more important aid for the poorer reader who doesn't have automatic context-free recognition instantiated.**

(in Boulton, 2009)

---

In other words, the most efficient way to read is via context-free, bottom-up, feature-driven processes that are well learned to the point of being automatic. Context-driven reading is today considered mainly a backup method that, although it operates in parallel with feature-based reading, is only relevant when feature-driven reading is difficult or is insufficiently automatic.

Skilled readers may resort to context-based reading when feature-based reading is disrupted by poor presentation of information (see examples later in this chapter). Also, in the race between context-based and feature-based reading to decipher the text we see, contextual cues sometimes win out over features. As an example of context-based reading, Americans visiting England sometimes misread “To let” signs as “Toilet” because in the United States they see the word “toilet” often, but they almost never see the phrase “to let”—Americans use “for rent” instead.

In less skilled readers, feature-based reading is not automatic; it is conscious and laborious. Therefore, much more of their reading is context based. Their involuntary use of context-based reading and nonautomatic feature-based reading consumes short-term cognitive capacity, leaving little for comprehension.<sup>1</sup> They have to focus on deciphering the stream of words, leaving no capacity for constructing the meaning of sentences and paragraphs. That is why poor readers can read a passage aloud but afterward have no idea what they just read.

Why is context-free (bottom-up) reading not automatic in some adults? Some people didn't get enough experience reading as young children for the feature-driven recognition processes to become automatic. As they grow up, they find reading mentally laborious and taxing, so they avoid reading, which perpetuates and compounds their deficit (Boulton, 2009).

---

## SKILLED AND UNSKILLED READING USES DIFFERENT PARTS OF THE BRAIN

Before the 1980s, researchers who wanted to understand which parts of the brain are involved in language and reading were limited mainly to studying people who had suffered brain injuries. For example, in the mid-1800s, doctors found that

---

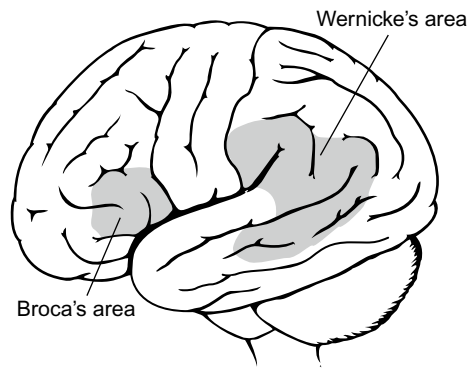
<sup>1</sup>Chapter 10 describes the differences between automatic and controlled cognitive processing. For present purposes, we will simply say that controlled processes burden working memory, while automatic processes do not.

people with brain damage near the left temple—an area now called *Broca's area* after the doctor who discovered it—can understand speech but have trouble speaking, and that people with brain damage near the left ear—now called *Wernicke's area*—cannot understand speech (Sousa, 2005) (see Fig. 4.4).

In recent decades, several new methods of observing the operation of functioning brains in living people, enhancing noninvasive scanning methods with computer-based analysis techniques, have been developed: electroencephalography (EEG), functional magnetic resonance imaging (fMRI), and functional magnetic resonance spectroscopy (fMRS). These methods allow researchers to watch the response in different areas of a person's brain—including the sequence in which they respond—as the person perceives various stimuli or performs specific tasks.

Using these methods, researchers have discovered that the neural pathways involved in reading differ for novice versus skilled readers. Of course, the first area to respond during reading is the occipital (or visual) cortex at the back of the brain. That is the same regardless of a person's reading skill. After that, the pathways diverge (Sousa, 2005):

- **Novice:** First an area of the brain just above and behind Wernicke's area becomes active. Researchers have come to view this as the area where, at least with alphabetic scripts such as English and German, words are “sounded out” and assembled—that is, letters are analyzed and matched with their corresponding sounds. The word-analysis area then communicates with Broca's area and the frontal lobe, where morphemes and words—units of meaning—are recognized and overall meaning is extracted. For ideographic languages, where symbols represent whole words and often have a graphical correspondence to their meaning, sounding out of words is not part of reading.



**FIGURE 4.4**

The human brain, showing Broca's area and Wernicke's area.

- **Advanced:** The word analysis area is skipped. Instead the occipito-temporal area (behind the ear, not far from the visual cortex) becomes active. The prevailing view is that this area recognizes words as a whole without sounding them out, and then that activity activates pathways toward the front of the brain that correspond to the word's meaning and mental image. Broca's area is only slightly involved.

Findings from brain scan methods of course don't indicate exactly what processes are being used, but they do support the theory that advanced readers use different processes from those novice readers use.

---

## POOR INFORMATION DESIGN CAN DISRUPT READING

Careless writing or presentation of text can reduce skilled readers' automatic, context-free reading to conscious, context-based reading, burdening working memory, thereby decreasing speed and comprehension. In unskilled readers, poor text presentation can block reading altogether.

### Uncommon or unfamiliar vocabulary

One way software often disrupts reading is by using unfamiliar vocabulary—words the intended readers don't know very well or at all.

One type of unfamiliar terminology is computer jargon, sometimes known as “geek speak.” For example, an intranet application displayed the following error message if a user tried to use the application after more than 15 minutes of letting it sit idle:

---

**Your session has expired. Please reauthenticate.**

---

The application was for finding resources—rooms, equipment, etc.—within the company. Its users included receptionists, accountants, and managers as well as engineers. Most nontechnical users would not understand the word “reauthenticate,” so they would drop out of automatic reading mode into conscious wondering about the message's meaning. To avoid disrupting reading, the application's developers could have used the more familiar instruction “Login again.” For a discussion of how “geek speak” in computer-based systems affects learning, see Chapter 11.

Reading can also be disrupted by uncommon terms even if they are not computer technology terms. Here are some rare English words, including many that appear mainly in contracts, privacy statements, or other legal documents:

- *Aforementioned:* mentioned previously
- *Bailiwick:* the region in which a sheriff has legal powers; more generally: domain of control

- *Disclaim*: renounce any claim to or connection with; disown; repudiate
- *Heretofore*: up to the present time; before now
- *Jurisprudence*: the principles and theories on which a legal system is based
- *Obfuscate*: make something difficult to perceive or understand
- *Penultimate*: next to the last, as in “the next to the last chapter of a book”

When readers—even skilled ones—encounter such a word, their automatic reading processes probably won’t recognize it. Instead, their brain uses less automatic processes, such as sounding out the word’s parts and using them to figure out its meaning, figuring out the meaning from the context in which the word appears, or looking the word up in a dictionary.

### Difficult scripts and typefaces

Even when the vocabulary is familiar, reading can be disrupted by hard-to-read scripts and typefaces. Bottom-up, context-free, automatic reading is based on recognition of letters and words from their visual features. Therefore, a typeface with difficult-to-recognize feature and shapes will be hard to read. For example, try to read Abraham Lincoln’s Gettysburg Address in an outline typeface in ALL CAPS. (see Fig. 4.5).

ABRAHAM LINCOLN'S GETTYSBURG ADDRESS

FOURSCORE AND SEVEN YEARS AGO, OUR FOREFATHERS BROUGHT FORTH ON THIS CONTINENT A NEW NATION, CONCEIVED IN LIBERTY AND DEDICATED TO THE PROPOSITION THAT ALL MEN ARE CREATED EQUAL.

NOW WE ARE ENGAGED IN A GREAT CIVIL WAR, TESTING WHETHER THAT NATION, OR ANY NATION SO CONCEIVED AND SO DEDICATED, CAN LONG ENDURE. WE ARE MET ON A GREAT BATTLEFIELD OF THAT WAR. WE HAVE COME TO DEDICATE A PORTION OF THAT FIELD, AS A FINAL RESTING PLACE FOR THOSE WHO HERE GAVE THEIR LIVES THAT THAT NATION MIGHT LIVE. IT IS ALTOGETHER FITTING AND PROPER THAT WE SHOULD DO THIS.

BUT, IN A LARGER SENSE, WE CAN NOT DEDICATE - WE CAN NOT CONSECRATE - WE CAN NOT HALLOW - THIS GROUND. THE BRAVE MEN, LIVING AND DEAD, WHO STRUGGLED HERE, HAVE CONSECRATED IT, FAR ABOVE OUR POOR POWER TO ADD OR DETRACT. THE WORLD WILL LITTLE NOTE, NOR LONG REMEMBER WHAT WE SAY HERE, BUT IT CAN NEVER FORGET WHAT THEY DID HERE. IT IS FOR US THE LIVING, RATHER, TO BE DEDICATED HERE TO THE UNFINISHED WORK WHICH THEY WHO FOUGHT HERE HAVE THUS FAR SO NOBLY ADVANCED. IT IS RATHER FOR US TO BE HERE DEDICATED TO THE GREAT TASK REMAINING BEFORE US - THAT FROM THESE HONORED DEAD WE TAKE INCREASED DEVOTION TO THAT CAUSE FOR WHICH THEY GAVE THE LAST FULL MEASURE OF DEVOTION - THAT WE HERE HIGHLY RESOLVE THAT THESE DEAD SHALL NOT HAVE DIED IN VAIN - THAT THIS NATION, UNDER GOD, SHALL HAVE A NEW BIRTH OF FREEDOM - AND THAT GOVERNMENT OF THE PEOPLE, BY THE PEOPLE, FOR THE PEOPLE, SHALL NOT PERISH FROM THE EARTH.

FIGURE 4.5

Text in ALL CAPS is generally hard to read because letters look more similar to each other. Outline typefaces complicate feature recognition. This example demonstrates both.

## Tiny fonts

Another way to make text hard to read in software applications, Websites, and electronic appliances is to use fonts that are too small for their intended readers' visual system to resolve. For example, try to read the first paragraph of the U.S. Constitution in a seven-point font (see Fig. 4.6).

We the people of the United States, in Order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our Posterity, do ordain and establish this Constitution for the United States of America.

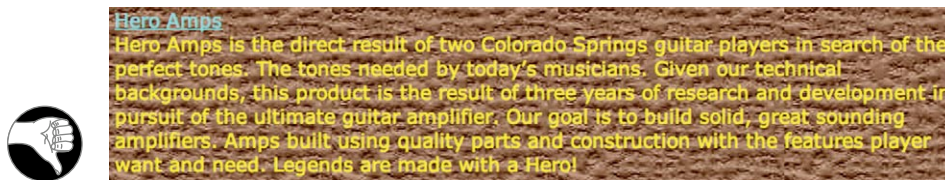
**FIGURE 4.6**

The opening paragraph of the U.S. Constitution, presented in a seven-point font.

Developers sometimes use tiny fonts because they have a lot of text to display in a small amount of space. But if the intended users of the system cannot read the text, or can read it only laboriously, the text might as well not be there.

## Text on noisy background

Visual noise in and around text can disrupt recognition of features, characters, and words and therefore drop reading out of automatic feature-based mode into a more conscious and context-based mode. In software user interfaces and Web sites, visual noise often results from designers' placing text over a patterned background or displaying text in colors that contrast poorly with the background, as an example from RedTele.com shows (see Fig. 4.7).



**FIGURE 4.7**

RedTele.com: Text on noisy background and with poor color contrast compared to the background.

There are situations in which designers *intend* to make text hard to read. For example, a common security measure on the Web is to ask site users to identify distorted words, as proof that they are a live human being and not an Internet "bot." This relies on the fact that most people can read text that Internet 'bots cannot



Type the characters you see in the picture above.

**FIGURE 4.8**

captchas: Text that is intentionally displayed with noise so that Web-crawling software cannot read it.



**FIGURE 4.9**

The Federal Reserve Bank's online mortgage calculator formerly displayed text on a patterned background.

currently read. Text displayed as a challenge to test a registrant's humanity is called a *captcha*<sup>2</sup> (see Fig. 4.8).

Of course, *most* text displayed in a user interface should be *easy* to read. A patterned background need not be especially strong to disrupt people's ability to read text placed over it. For example, the Federal Reserve Bank's collection of Web sites formerly had a mortgage calculator that was decorated with a repeating pastel background with a home and neighborhood theme. Although well-intentioned, the decorated background made the calculator hard to read (see Fig. 4.9). Later, when the Fed redesigned the mortgage calculator to add functionality, it also removed the decorative background (see Fig. 4.10).

<sup>2</sup>The term originally was coined based on the word "capture," but it is also said to be an acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart"—*Wikipedia entry for "Captcha."*



**1. Enter Your Information**

Monthly Income \$  ? Enter your financial profile along with information about the house you would like to purchase in the fields provided. A definition of each input field is accessible by clicking the corresponding Help ("?") button.

Price \$  ?

Down Payment \$  ?

Other Debt Payments \$  ?

---

**2. Mortgage Loan Criteria**

Front Ratio  ? You may either enter the mortgage loan criteria directly or choose default values by selecting the "Conventional Loan" or "FHA Loan" buttons. Qualifying ratios will vary by lender and loan type. Check with your lender to determine the best loan for you.

Back Ratio  ?

Loan/Value  ?

Interest Rate  ?

Term in Months  ?

Taxes and Insurance \$  ?

---

**3. Compute**



**FIGURE 4.10**

A more recent mortgage calculator on the FED Web site displays text on a plain white background.

### Information buried in repetition

Visual noise can also come from the text itself. If successive lines of text contain a lot of repetition, readers receive poor feedback about what line they are focused on, plus it is hard to pick out the important information. For example, recall the example from the California Department of Motor Vehicles Web site in the previous chapter (see Fig. 3.2, page 26).

Another example of repetition that creates noise is the computer store on Apple.com. The pages for ordering a laptop computer list different keyboard options





FIGURE 4.11

Apple.com's "Buy Computer" page lists options in which the important information (keyboard language compatibility) is buried in repetition.

for a computer in a very repetitive way, making it hard to see that the essential difference between the keyboards is the language that they support (see Fig. 4.11).

### Centered text

One aspect of reading that is highly automatic in most skilled readers is eye movement. In automatic (fast) reading, our eyes are trained to go back to the same horizontal position and down one line. If text is centered or right-aligned, each line of text starts in a different horizontal position. Automatic eye movements therefore take our eyes back to the wrong place, so we must consciously adjust our gaze to the *actual* start of each line. This drops us out of automatic mode and slows us down greatly. With poetry and wedding invitations, that is probably OK, but with any other type of text, it is a disadvantage. An example of centered prose text is provided by the Web site of FargoHomes, a real estate company (see Fig. 4.12). Try reading the text quickly to demonstrate to yourself how your eyes move.

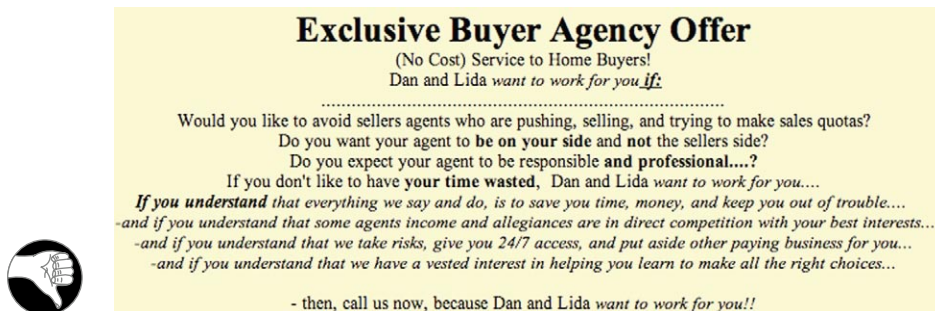


FIGURE 4.12

FargoHomes.com centers text, thwarting automatic eye movement patterns.

The same site also centers numbered lists, *really* messing up readers' automatic eye movement (see Fig. 4.13). Try scanning the list quickly.

**BUYER'S ! MORE Searches HERE**  
 .....if you don't have a Realtor click [Here](#)

**1. Search All The Fargo Moorhead Listings [CLICK HERE](#)** Step One (Very Important).... if you don't have a Realtor click [Here](#)

[Dream Home Finder](#) request form: *All area Best listings from [Top Area Realtors](#)*  
 for Fargo, Moorhead, and FM Area real estate. Moorhead homes. Moorhead Real Estate. West Fargo homes and West Fargo Real Estate

**2. [Todays \\*\\*HOT SHEET\\*\\* Click Here:](#) [New Listings in Fargo, Moorhead area](#)**

**3. Rural Minnesota... [Featured Listings](#)**

**4. Multiple Listing Number search [Click Here](#)**

**5. <http://www.fargoMLS.com> - Blog - MLS "Value of the Day"**

**6. Eid - Co Builders - Access to Models, New Developments in Fargo, West Fargo, Moorhead, and Dilworth, and Floor Plan Options [Click Here](#)**

**bb - New "Heritage Homes" - [Available Properties](#)**

**Minnesota Lake and River Property**  
**[Detroit Lakes Resorts, Lots, and Cabins Search](#)**  
with Tom Ackman, Coldwell Banker At The Lakes  
**[Detroit Lakes Find-A-Listing - Search Here](#)**



**FIGURE 4.13**

FargoHomes.com centers numbered items, really thwarting automatic eye movement patterns.

### Combining flaws that disrupt reading

The website of Keller Williams, another real-estate firm, combines many of the above-described ways of disrupting reading. In some places it has insufficient contrast between foreground and background. In other places it has too much contrast, e.g., it places blue against red, causing an annoying shimmering. It also has centered prose text and text on patterned backgrounds. All of the above combine to make this site very hard to read (see Fig. 4.14).

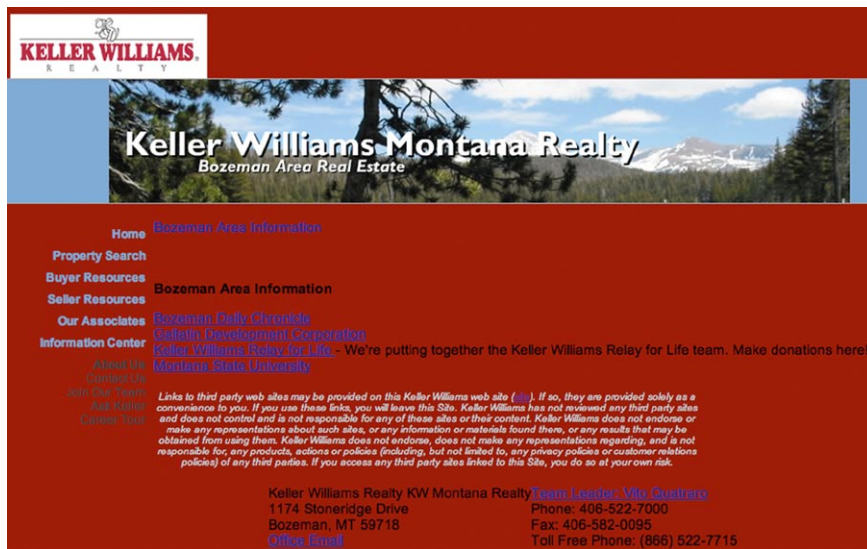


FIGURE 4.14

Keller Williams's Web site makes text hard to read in several different ways.

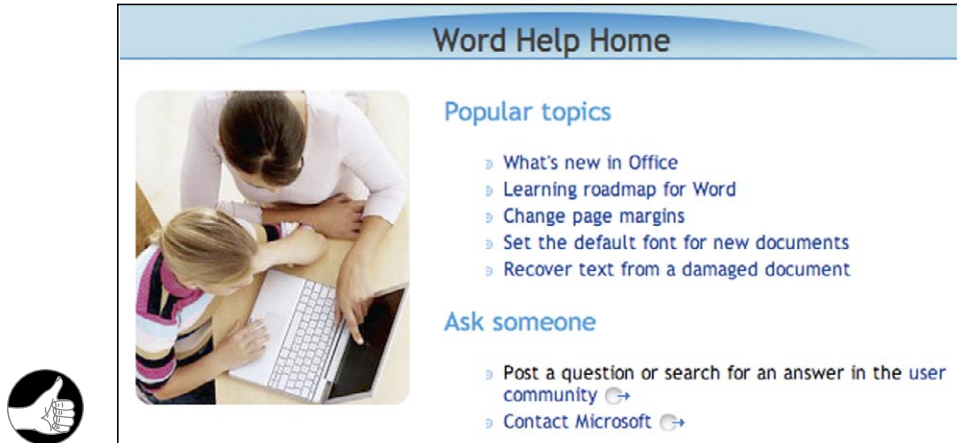
### Design implications: don't disrupt reading; support it!

Obviously, a designer's goal should be to support reading, not disrupt it. Skilled (fast) reading is mostly automatic and mostly based on feature, character, and word recognition. The easier the recognition, the easier and faster the reading. Less skilled reading, by contrast, is greatly assisted by *contextual* cues.

Designers of interactive systems can support both reading methods by following these guidelines:

- Ensure that text in user interfaces allows the feature-based automatic processes to function effectively by avoiding the disruptive flaws described above: difficult or tiny fonts, patterned backgrounds, centering, etc.

- Use restricted, highly consistent vocabularies—sometimes referred to in the industry as *plain language*<sup>3</sup> or *simplified language* (Redish, 2007).
- Format text to create a visual hierarchy (see Chapter 3) to facilitate easy scanning: use headings, bulleted lists, tables, and visually emphasized words (see Fig. 4.15).



**FIGURE 4.15**

Microsoft Word's Help home page is easy to scan and read.

Experienced information architects, content editors, and graphic designers can be very useful in ensuring that text is presented so as to support easy scanning and reading.

## MUCH OF THE READING REQUIRED BY SOFTWARE IS UNNECESSARY

In addition to committing design mistakes that disrupt reading, many software user interfaces simply present *too much* text, requiring users to read more than is necessary. Consider how much unnecessary text there is in a dialog box for setting text entry properties in the SmartDraw application (see Fig. 4.16).

Software designers often justify lengthy instructions by arguing: “We need all that text to explain clearly to users what to do.” However, instructions can often be shortened with no loss of clarity. Let’s examine how the Jeep company, between

<sup>3</sup>For more information on plain language, see the U.S. government Web site: [www.plainlanguage.gov](http://www.plainlanguage.gov).

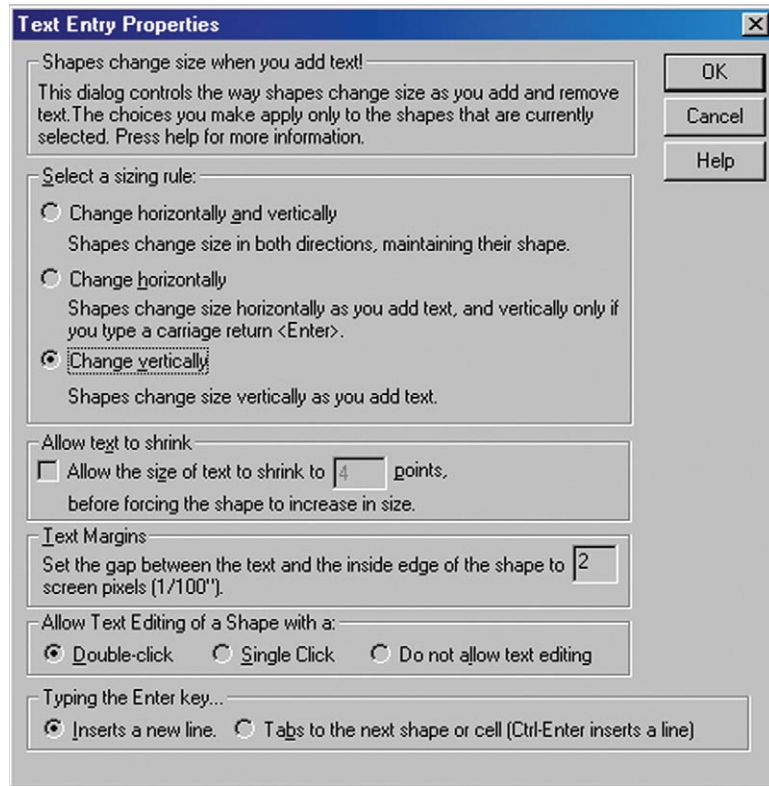


FIGURE 4.16

SmartDraw's Text Entry Properties dialog box displays too much text for its simple functionality.

2002 and 2007, shortened its instructions for finding a local Jeep dealer (see Fig. 4.17):

- **2002:** The “Find a Dealer” page displayed a large paragraph of prose text, with numbered instructions buried in it, and a form asking for more information than needed to find a dealer near the user.
- **2003:** The instructions on the “Find a Dealer” page had been boiled down to three bullet points, and the form required less information.
- **2007:** “Find a Dealer” had been cut to one field (zip code) and a Go button on the Home page.

Even when text describes products rather than explaining instructions, it is counterproductive to put all a vendor wants to say about a product into a lengthy prose description that people have to read from start to end. Most potential customers



2002

**Jeep**

## FIND A DEALER

It's easy to locate a dealer. 1. Click and hold box number 1 to select your search by Zip Code, City, Dealership Name or State. 2. Enter the Zip Code, City, or Dealership Name in the box marked number 2. 3. If searching by State only, select the state from the pull-down menu in box number 3. \*\*If choosing to search by city or state, type the city in box 2 then select a state in the box marked number 3 to make your search complete. 4. Once finished, simply click the "Search" button.

**Search by:**  
 ①

**Enter Zip Code, City, or Dealership name:**  
 ②

**Select a State:**  
 ③

④

If you are a member of the U.S. Military, an executive, or a diplomat living outside the U.S., [click here](#) for special options.



2003

**Jeep**

## FIND A DEALER

It's easy to locate a Jeep Dealer near you.

- Select Zipcode, City or Dealership Name  
(If you choose to search by city, you will be prompted to provide the state.)
- Provide the Zip Code, City or Dealership Name
- Click on Search

**Search by:**  
 ①

**Enter Zip Code, City, or Dealership name:**  
 ②

③



2007

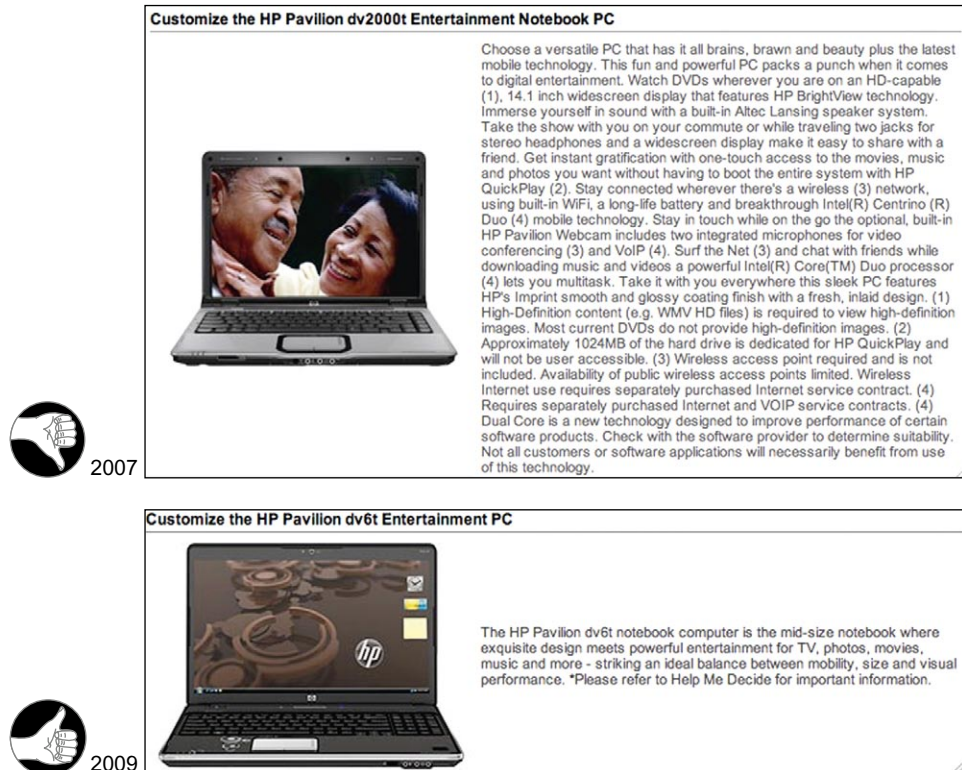
**FIND A DEALER** ▾

FIGURE 4.17

Between 2002 and 2007, Jeep.com drastically reduced the reading required by “Find a Dealer.”



cannot or will not read it. Compare Costco.com's descriptions of laptop computers in 2007 with those in 2009 (see Fig. 4.18).



**FIGURE 4.18**

Between 2007 and 2009, Costco.com drastically reduced the text in product descriptions.

### Design implications: minimize the need for reading

Too much text in a user interface loses poor readers, who unfortunately are a significant percentage of the population. Too much text even alienates *good* readers: it turns using an interactive system into an intimidating amount of *work*.

Minimize the amount of prose text in a user interface; don't present users with long blocks of prose text to read. In instructions, use the *least* amount of text that gets most users to their intended goals. In a product description, provide a brief overview of the product and let users request more detail if they want it. Technical writers and content editors can assist greatly in doing this. For additional advice on how to eliminate unnecessary text, see Krug (2005) and Redish (2007).

---

## TEST ON REAL USERS

Finally, designers should test their designs on the intended user population to be confident that the users can read all essential text quickly and effortlessly. Some testing can be done early, using prototypes and partial implementations, but it should also be done just before release. Fortunately, last-minute changes to text font sizes and formats are usually easy to make.



# Our Color Vision is Limited

# 5

Human color perception has both strengths and limitations. Many of those strengths and limitations are relevant to user interface design:

- Our vision is optimized to detect contrasts (edges), not absolute brightness.
- Our ability to distinguish colors depends on how colors are presented.
- Some people have color-blindness.
- The user's display and the viewing conditions affect color perception.

To understand these qualities of human color vision, let's start with a brief description of how the human visual system processes color information from the environment.

## HOW COLOR VISION WORKS

If you took introductory psychology or neurophysiology in college, you probably learned that the retina at the back of the human eye—the surface onto which the eye focuses images—has two types of light receptor cells: rods and cones. You probably also learned that the rods detect light levels but not colors, while the cones detect colors. Finally, you probably learned that there are three types of cones, sensitive to red, green, and blue light, respectively, suggesting that our color vision is similar to video cameras and computer displays, which detect or project a wide variety of colors through combinations of red, green, and blue pixels.

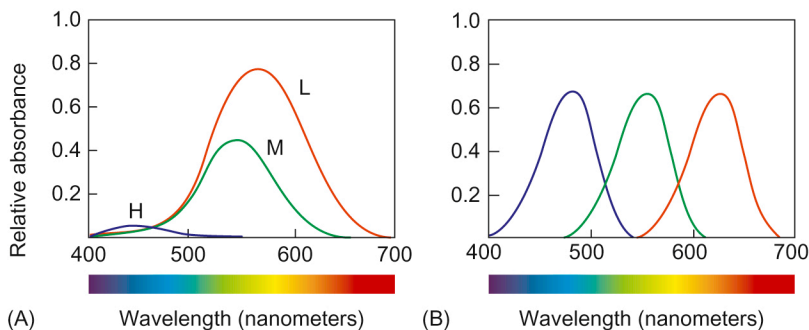
What you learned in college is only partly right. We do in fact have rods and three types of cones in our retinas. The rods are sensitive to overall brightness while the three types of cones are sensitive to different frequencies of light. But that's where the truth departs from what most people learned in college until recently.

First, those of us who live in industrialized societies hardly use our rods at all. They function only at low levels of light. They are for getting around in poorly lighted environments—the environments our ancestors lived in until the nineteenth century. Today, we use our rods only when we are having dinner by candlelight, feeling our way around our dark house at night, camping outside after dark, etc. In bright daylight and modern artificially lighted environments—where we spend most of our time—our rods are completely maxed out, providing no useful information. Most of the time, our vision is based entirely on input from our cones (Ware, 2008).

So how do our cones work? Are the three types of cones sensitive to red, green, and blue light, respectively? In fact, each type of cone is sensitive to a wider range of light frequencies than you might expect, and the sensitivity ranges of the three types overlap considerably. In addition, the overall sensitivity of the three types of cones differs greatly (see Fig. 5.1A):

- **Low frequency:** These cones are sensitive to light over almost the entire range of visible light, but are most sensitive to the middle (yellow) and low (red) frequencies.
- **Medium frequency:** These cones respond to light ranging from the high-frequency blues through the lower middle-frequency yellows and oranges. Overall, they are less sensitive than the low-frequency cones.
- **High frequency:** These cones are most sensitive to light at the upper end of the visible light spectrum—violets and blues—but they also respond weakly to middle frequencies, such as green. These cones are much less sensitive overall than the other two types of cones, and also less numerous. One result is that our eyes are much less sensitive to blues and violets than to other colors.

Compare a graph of the light sensitivity of our retinal cone cells (Fig. 5.1A) to what the graph might look like if electrical engineers had designed our retinas as a mosaic of receptors sensitive to red, green, and blue, like a camera (Fig. 5.1B).



**FIGURE 5.1**

Sensitivity of the three types of retinal cones (A) versus artificial red, green, blue receptors (B).

Given the odd relationships between the sensitivities of our three types of retinal cones cells, one might wonder how our brain combines the signals from the cones to allow us to see a broad range of colors.

The answer is: by *subtraction*. Neurons in the visual cortex at the back of our brain *subtract* the signals coming over the optic nerves from the medium- and low-frequency cones, producing a “red-green” *difference* signal channel. Other neurons in the visual cortex subtract the signals from the high- and low-frequency cones, yielding a “yellow-blue” *difference* signal channel. A third group of neurons in the visual cortex *adds* the signals coming from the low- and medium-frequency cones to produce an overall *luminance* (or “black-white”) signal channel.<sup>1</sup> These three channels are called *color-opponent* channels.

The brain then applies additional subtractive processes to all three color-opponent channels: signals coming from a given area of the retina are effectively subtracted from similar signals coming from nearby areas of the retina.

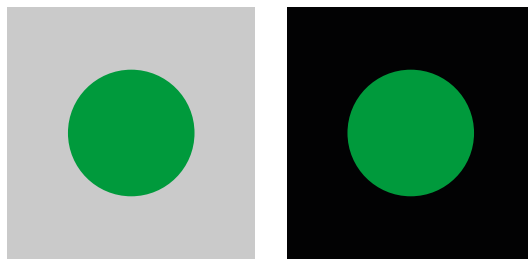
---

## VISION IS OPTIMIZED FOR EDGE CONTRAST, NOT BRIGHTNESS

All this subtraction makes our visual system much more sensitive to *differences* in color and brightness—i.e., to contrasting edges—than to absolute brightness levels.

To see this, compare the two green circles in [Figure 5.2](#). They are the same exact shade of green—the circle on the right was copied from the one on the left—but the different backgrounds make the one on the left appear darker to our contrast-sensitive visual system.

The sensitivity of our visual system to contrast rather than to absolute brightness is an advantage: it helped our distant ancestors recognize a leopard in the nearby bushes as the same dangerous animal whether they saw it in bright noon sunlight or in the early morning hours of a cloudy day. Similarly, being sensitive to color contrasts rather than to absolute colors allows us to see a rose as the same red whether it is in the sun or the shade.

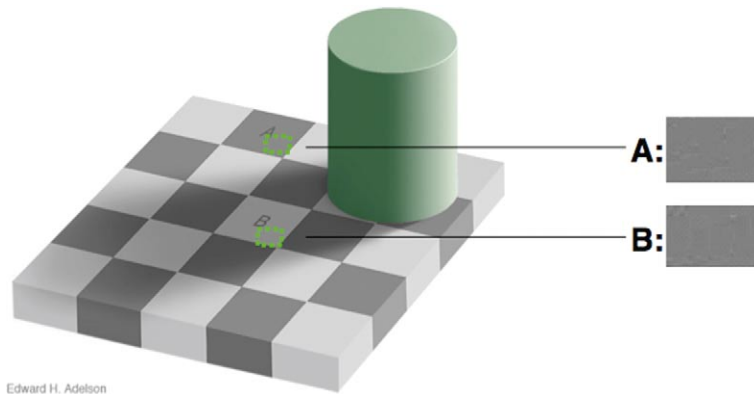


**FIGURE 5.2**

The circles appear as different shades because their backgrounds are different, but they are the same.

---

<sup>1</sup> The overall brightness sum omits the signal from the high-frequency (blue-violet) cones. Those cones are so insensitive that their contribution to the total would be negligible, so omitting them makes little difference.



**FIGURE 5.3**

The squares marked A and B are the same gray. We see B as white because it is “shaded.”

Brain researcher Edward H. Adelson at the Massachusetts Institute of Technology developed an outstanding illustration of our visual system’s insensitivity to absolute brightness and its sensitivity to contrast (see Fig. 5.3). As difficult as it may be to believe, square A on the checkerboard is exactly the same shade as square B. Square B only appears white because it is depicted as being in the cylinder’s shadow.

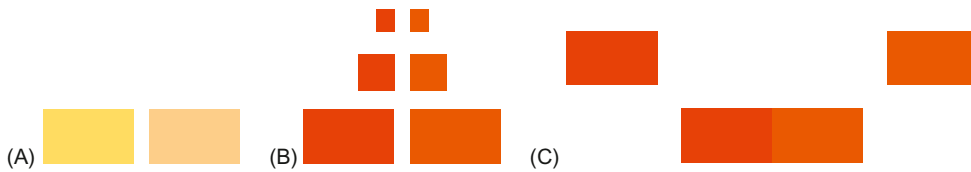
## ABILITY TO DISCRIMINATE COLORS DEPENDS ON HOW COLORS ARE PRESENTED

Even our ability to detect differences between colors is limited. Because of how our visual system works, three presentation factors affect our ability to distinguish colors from each other:

- **Paleness:** The paler (less saturated) two colors are, the harder it is to tell them apart (see Fig. 5.4A).
- **Color patch size:** The smaller or thinner objects are, the harder it is to distinguish their colors (see Fig. 5.4B). Text is often thin, so the exact color of text is often hard to determine.
- **Separation:** The more separated color patches are, the more difficult it is to distinguish their colors, especially if the separation is great enough to require eye motion between patches (see Fig. 5.4C).

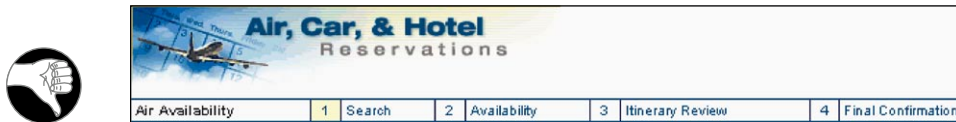
Several years ago, the online travel Web site ITN.net used two pale colors—white and pale yellow—to indicate which step of the reservation process the user was on (see Fig. 5.5). Some site visitors couldn’t see which step they were on.

Small color patches are often seen in data charts and plots. Many business graphics packages produce legends on charts and plots, but make the color patches in the legend very small (see Fig. 5.6). Color patches in chart legends should be large to help people distinguish the colors (see Fig. 5.7).



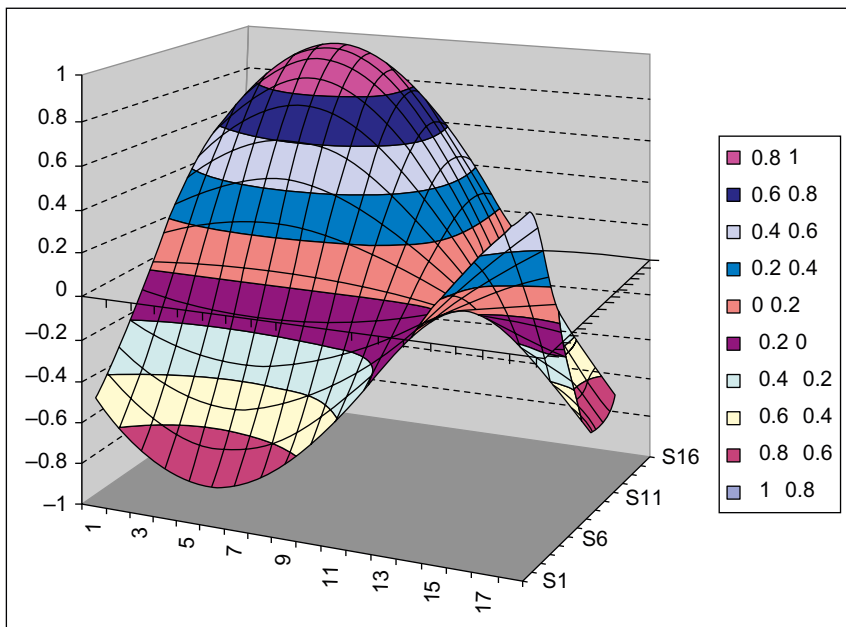
**FIGURE 5.4**

Factors affecting the ability to distinguish colors: (A) paleness, (B) size, (C) separation.



**FIGURE 5.5**

ITN.net (2003): Pale color marking current step makes it hard for users to see which step in the airline reservation process they are on.



**FIGURE 5.6**

Tiny color patches in this chart legend are hard to distinguish.

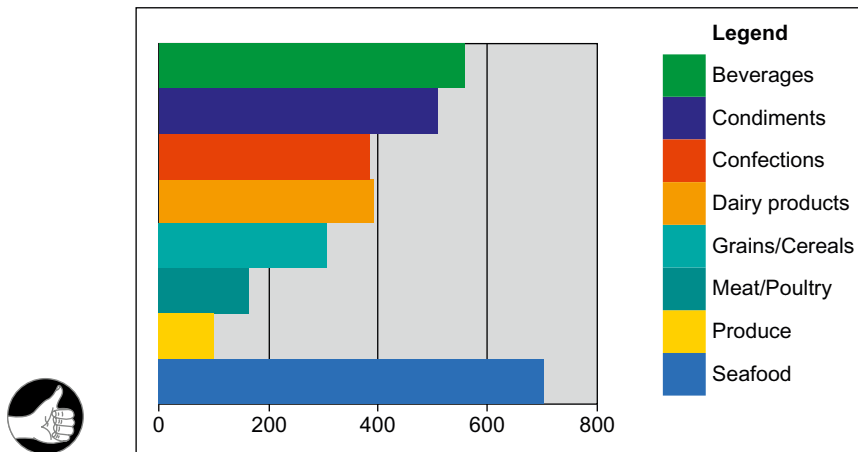


FIGURE 5.7

Large color patches make it easier to distinguish the colors.



FIGURE 5.8

MinneapolisFed.org: The difference in color between visited and unvisited links is too subtle.<sup>2</sup>

On Web sites, a common use of color is to distinguish unfollowed links from already followed ones. On some sites, the “followed” and “unfollowed” colors are too similar. The Web site of the Federal Reserve Bank of Minneapolis (see Fig. 5.8) has this problem. Furthermore, the two colors are shades of blue, the color range in which our eyes are least sensitive. Can you spot the two followed links? (The answer is below.)

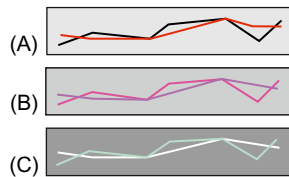
## COLOR-BLINDNESS

A fourth factor of color presentation that affects design principles for interactive systems is whether the colors can be distinguished by people who have common types of

<sup>2</sup>Already followed links in Figure 5.8: Housing Units Authorized and House Price Index.

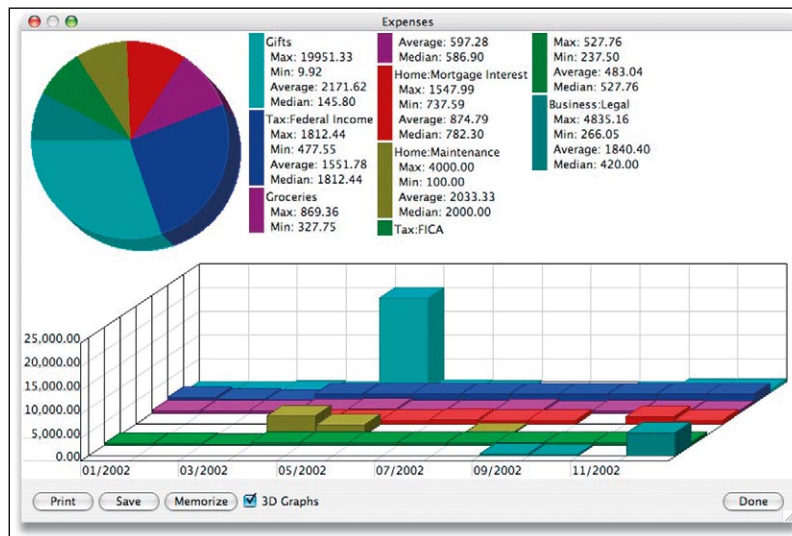
color-blindness. Having color-blindness doesn't mean an inability to see colors. It just means that one or more of the color subtraction channels (see above) don't function normally, making it difficult to distinguish certain pairs of colors. Approximately 8% of men and slightly under 0.5% of women have a color perception deficit:<sup>3</sup> difficulty discriminating certain pairs of colors (Wolfmaier, 1999). The most common type of color-blindness is red/green; other types are much rarer. Figure 5.9 shows color pairs that people with red/green color blindness have trouble distinguishing.

The home finance application MoneyDance provides a graphical breakdown of household expenses, using color to indicate the various expense categories (see Fig. 5.10). Unfortunately, many of the colors are hues that color-blind people cannot



**FIGURE 5.9**

Red/green color-blind people can't distinguish: (A) dark red from black, (B) blue from purple, (C) light green from white.



**FIGURE 5.10**

MoneyDance: Graph uses colors some users can't distinguish.

<sup>3</sup>The common term is color "blindness," but color "vision deficit," "vision deficiency," "vision defect," "confusion," and "weakness" are more accurate. Color "challenged" is also used. A total inability to see color is extremely rare.

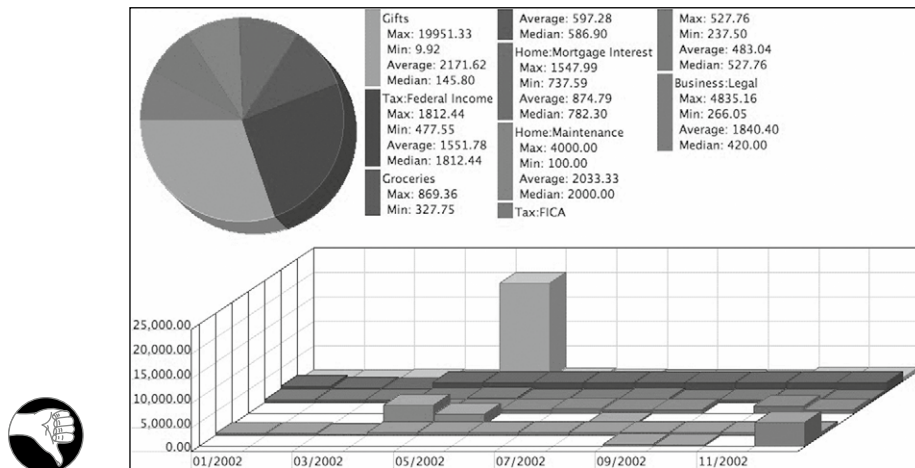


FIGURE 5.11

MoneyDance graph rendered in grayscale.

tell apart. For example, people with red/green color-blindness cannot distinguish the blue from the purple or the green from the khaki. If you are not color-blind, you can get an idea of which colors in an image will be hard to distinguish by converting the image to grayscale (see Fig. 5.11).

## EXTERNAL FACTORS THAT INFLUENCE THE ABILITY TO DISTINGUISH COLORS

Factors concerning the external environment also impact people's ability to distinguish colors. For example:

- **Variation among color displays:** Computer displays vary in how they display colors, depending on their technologies, driver software, or color settings. Even monitors of the same model with the same settings may display colors slightly differently. Something that looks yellow on one display may look beige on another. Colors that are clearly different on one may look the same on another.
- **Grayscale displays:** Although most displays these days are color, there are devices, especially small hand-held ones, with grayscale displays. Figure 5.11 (above) shows that a grayscale display can make areas of different colors look the same.
- **Display angle:** Some computer displays, particularly LCD ones, work much better when viewed straight on than when viewed from an angle. When LCD



displays are viewed at an angle, colors—and color differences—often are altered.

- **Ambient illumination:** Strong light on a display washes out colors before it washes out light and dark areas, reducing color displays to grayscale ones, as anyone who has tried to use a bank ATM in direct sunlight knows. In offices, glare and venetian blind shadows can mask color differences.

These four external factors are usually out of the software designer’s control. Designers should therefore keep in mind that they don’t have full control of the color viewing experience of users. Colors that seem highly distinguishable in the development facility on the development team’s computer displays and under normal office lighting conditions may not be as distinguishable in some of the environments where the software is used.

## GUIDELINES FOR USING COLOR

In interactive software systems that rely on color to convey information, follow these five guidelines to assure that the users of the software receive the information:

1. *Distinguish colors by saturation and brightness as well as hue.* Avoid subtle color differences. Make sure the contrast between colors is high (but see guideline 5). One way to test whether colors are different enough is to view them in grayscale. If you can’t distinguish the colors when they are rendered in grays, they aren’t different enough.
2. *Use distinctive colors.* Recall that our visual system combines the signals from retinal cone cells to produce three “color opponent” channels: red-green, yellow-blue, and black-white (luminance). The colors that people can distinguish most easily are those that cause a strong signal (positive or negative) on one of the three color-perception channels, and neutral signals on the other two channels. Not surprisingly, those colors are red, green, yellow, blue, black, and white (see Fig. 5.12). All other colors cause signals on more than one color channel, and so our visual system cannot distinguish them from other colors as quickly and easily as it can distinguish those six colors (Ware, 2008).



**FIGURE 5.12**

The most distinctive colors. Each color causes a strong signal on only one color-opponent channel.

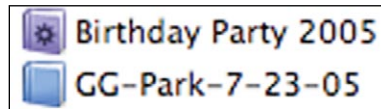


FIGURE 5.13

Apple's iPhoto uses uses color plus a symbol to distinguish two types of albums.



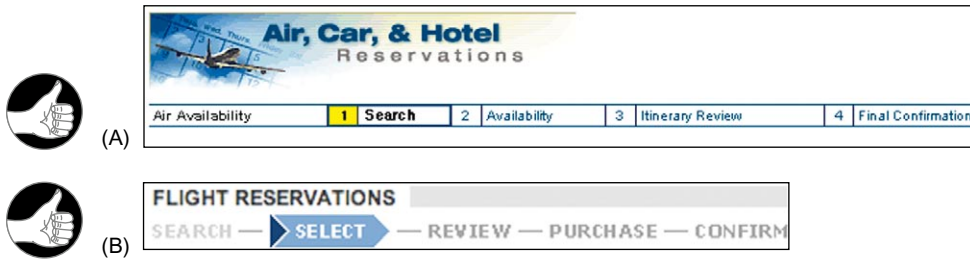
FIGURE 5.14

Opponent colors, placed on or directly next to each other, clash.

3. *Avoid color pairs that color-blind people cannot distinguish.* Such pairs include dark red versus black, dark red versus dark green, blue versus purple, light green versus white. Don't use dark reds, blues, or violets against any dark colors. Instead, use dark reds, blues, and violets against light yellows and greens. Use [Vischeck.com](http://Vischeck.com) to check Web pages and images to see how people with various color vision deficiencies would see them.
4. *Use color redundantly with other cues.* Don't rely on color alone. If you use color to mark something, mark it another way as well. Apple's iPhoto uses both color and a symbol to distinguish "smart" photo albums from regular albums (see [Fig. 5.13](#)).
5. *Separate strong opponent colors.* Placing opponent colors right next to or on top of each other causes a disturbing shimmering sensation, and so should be avoided (see [Fig. 5.14](#)).

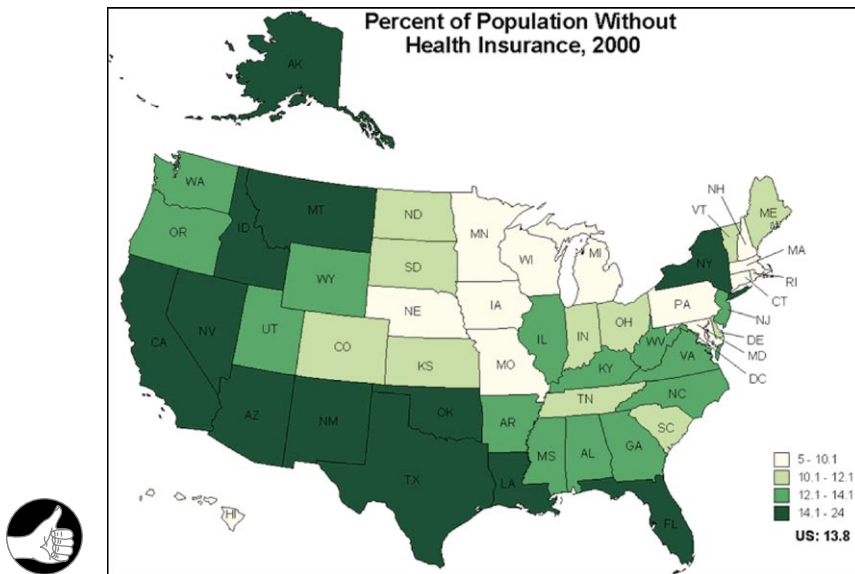
As mentioned above, ITN.net used only pale yellow to mark customers' current step in making a reservation (see [Fig. 5.5](#), above), which is too subtle. A simple way to strengthen the marking would be to make the current step bold and increase the saturation of the yellow (see [Fig. 5.15A](#)). But ITN.net opted for a totally new design, which also uses color redundantly with shape (see [Fig. 5.15B](#)).

A graph from the Federal Reserve Bank uses white and shades of green ([Fig. 5.16](#)). This is a well-designed graph. Any sighted person could read it, even on a grayscale display.



**FIGURE 5.15**

ITN.net: The current step is highlighted in two ways: with color and shape.



**FIGURE 5.16**

MinneapolisFed.org: Graph uses color differences visible to all sighted people, on any display.

# Our Peripheral Vision is Poor

# 6

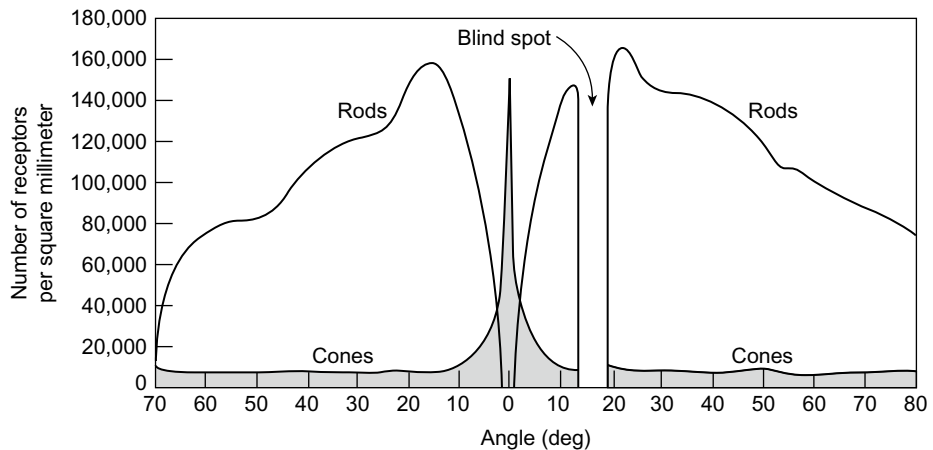
The previous chapter explained that the human visual system differs from a digital camera in the way it detects and processes color. Our visual system also differs from a camera in its resolution. On a digital camera's photo sensor, photoreceptive elements are spread uniformly in a tight matrix, so the spatial resolution is constant across the entire image frame. The human visual system is not like that.

This chapter explains why:

- Stationary items in muted colors presented in the periphery of people's visual field often will not be noticed.
- Motion in the periphery is usually noticed.

## RESOLUTION OF THE FOVEA COMPARED TO THAT OF THE PERIPHERY

The spatial resolution of the human visual field drops greatly from the center to the edges. Each eye has approximately six million retinal cone cells. They are packed much more tightly in the center of our visual field—a small region called the *fovea*—than they are at the edges of the retina (see Fig. 6.1). The fovea is only about 1% of the retina, but the brain's visual cortex devotes about 50% of its area to input from the fovea. Furthermore, foveal cone cells connect 1:1 to the ganglial neuron cells that begin the processing and transmission of visual data, while elsewhere on the retina, *multiple* photoreceptor cells (cones and rods) connect to each ganglion cell. In technical terms, information from the visual periphery is compressed (with data loss) before transmission to the brain, while information from the fovea is not. All of this causes our vision to have much, *much* greater resolution in the center of our visual field than elsewhere (Lindsay and Norman, 1972; Waloszek, 2005).



**FIGURE 6.1**

Distribution of photoreceptor cells (cones and rods) across the retina.

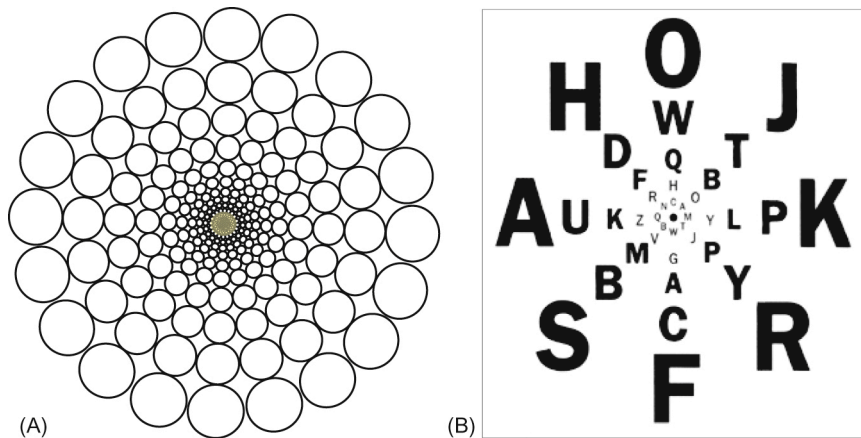
*Lindsay and Norman, 1972.*

To visualize how small the fovea is compared to your entire visual field, hold your arm straight out and look at your thumb. Your thumbnail, viewed at arm's length, corresponds approximately to the fovea (Ware, 2008). While you have your eyes focused on the thumbnail, everything else in your visual field falls outside of your fovea on your retina.

In the fovea, people with normal vision have very high resolution: they can resolve several thousand dots within that region—better resolution than many of today's pocket digital cameras. Just outside of the fovea, the resolution is already down to a few dozen dots per inch viewed at arm's length. At the edges of our vision, the “pixels” of our visual system are as large as a melon (or human head) at arm's length (see Fig. 6.2A and B).

If our peripheral vision has such low resolution, one might wonder why we don't see the world in a kind of tunnel vision where everything is out of focus except what we are directly looking at now. Instead, we seem to see our surroundings sharply and clearly all around us. We experience this illusion because our eyes move rapidly and constantly about three times per second even when we don't realize it, focusing our fovea on selected pieces of our environment. Our brain fills in the rest in a gross, impressionistic way based upon what we know and expect.<sup>1</sup> Our brain does not have to maintain a high-resolution mental model of our environment because it can order the eyes to sample and resample details in the environment as needed (Clark, 1998).

<sup>1</sup>Our brains also fill in the perceptual gaps that occur during saccadic eye movements, when vision is suppressed.



**FIGURE 6.2**

The resolution of our visual field is high in the center but much lower at the edges.

(B) Image from *Vision Research*, Vol. 14 (1974), Elsevier.

For example, as you read this page, your eyes dart around, scanning and reading. No matter where on the page your eyes are focused, you have the impression of viewing a complete page of text, because, of course, you are. But now, imagine that you are viewing this page on a computer screen, and the computer is tracking your eye movements and knows where your fovea is on the page. Imagine that wherever you look, the right text for that spot on the page is shown clearly in the small area corresponding to your fovea, but everywhere else on the page, the computer shows random, meaningless text. As your fovea flits around the page, the computer quickly updates each area where your fovea stops to show the correct text there, while the last position of your fovea returns to textual noise. Amazingly, experiments have shown that people *do not notice* this: not only can they read normally, they still believe that they are viewing a full page of meaningful text (Clark, 1998).

Related to this is the fact that the center of our visual field—the fovea and a small area immediately surrounding it—is the only part of our visual field that can read. The rest of our visual field cannot read. What this really means is that the neural networks starting in the fovea, running through the optic nerve to the visual cortex, and then spreading into various parts of our brain, have been trained to read, but the neural networks starting elsewhere in our retinas cannot read. All text that we read comes into our visual system after being scanned by the central area, which means that reading requires a lot of eye movement. Of course, based on what we have already read and our knowledge of the world, our brains can sometimes predict text that the fovea has not yet read (or its meaning), allowing to us skip reading it, but that is different from actually reading.

The fact that retinal cone cells are distributed tightly in and near the fovea, and sparsely in the periphery of the retina affects not only spatial resolution but color

resolution as well. We can discriminate colors better in the center of our visual field than at the edges.

Another interesting fact about our visual field is that it has a gap—a small area in which we see nothing. The gap corresponds to the spot on our retina where the optic nerve and blood vessels exit the back of the eye (see Fig. 6.1, above). There are no retinal rod or cone cells at that spot, so when the image of an object in our visual field happens to fall on that part of the retina, we don't see it. We usually don't notice this hole in our vision because our brain fills it in with the surrounding content, like a graphic artist using Photoshop to fill in a blemish on a photograph by copying nearby background pixels.

People sometimes experience the blind spot when they gaze at stars. As you look at one star, a nearby star may disappear briefly into the blind spot until you shift your gaze. You can also observe the gap by trying the exercise in Figure 6.3. Some people have other gaps resulting from imperfections on the retina, retinal damage, or brain strokes that affect the visual cortex, but the optic nerve gap is an imperfection everyone shares.



**FIGURE 6.3**

To “see” the retinal gap, cover your left eye, hold this book near your face, and focus your right eye on the +. Move the book slowly away from you, staying focused on the +. The @ will disappear at some point.

## IS THE VISUAL PERIPHERY GOOD FOR ANYTHING?

It seems that the fovea is better than the periphery at just about everything. One might wonder why we even have peripheral vision. What is it good for?

The answer is that our peripheral vision exists mainly to provide low-resolution cues to guide our eye movements so that our fovea visits all the interesting and crucial parts of our visual field. Our eyes don't scan our environment randomly. They move so as to focus our fovea on important things, the most important ones (usually) first. The fuzzy cues on the outskirts of our visual field provide the data that helps our brain plan where to move our eyes, in what order.

For example, when we scan a medicine label for a “use by” date, a fuzzy blob in the periphery with the vague form of a date is enough to cause an eye movement that lands the fovea there to allow us to check it. If we are browsing a produce market looking for strawberries, a blurry reddish patch at the edge of our visual field draws our eyes and our attention, even though sometimes it may turn out to be radishes instead of strawberries. If we hear an animal growl nearby, a fuzzy animal-like shape in the corner of our eye will be enough to zip our eyes in that direction, especially if the shape is moving toward us (see Fig. 6.4).



**FIGURE 6.4**

A moving shape at the edge of our vision draws our eye: it could be food, or it might consider us food.

That brings us to another advantage of peripheral vision: it is good at detecting motion. Anything that moves in our visual periphery, even slightly, is likely to draw our attention—and hence our fovea—toward it. The reason for this phenomenon is that our ancestors—including pre-human ones—were selected for their ability to spot food and avoid predators. As a result, even though we can move our eyes under conscious, intentional control, some of the mechanisms that control where they look are preconscious, involuntary, and very fast.

What if we have no reason to expect that there might be anything interesting in a certain spot in the periphery,<sup>2</sup> and nothing in that spot attracts our attention? Our eyes may never move our fovea to that spot, so we may never see what is there.

---

## EXAMPLES FROM COMPUTER USER INTERFACES

The low acuity of our peripheral vision explains why software and Web site users fail to notice error messages in some applications and Web sites. When someone clicks a button or a link, that is usually where his or her fovea is positioned. Everything on the screen that is not within 1–2 centimeters of the click location (assuming normal computer viewing distance) is in peripheral vision, where resolution is low. If, after the click, an error message appears in the periphery, it should not be surprising that the person might not notice it.

---

<sup>2</sup>See Chapter 1 on how expectations bias our perceptions.



For example, at InformaWorld.com, the online publications Web site of Informa Healthcare, if a user enters an incorrect username or password and clicks “Sign In”, an error message appears in a “message bar” far away from where the user’s eyes are most likely focused (see Fig. 6.5). The red word “Error” might appear in the user’s peripheral vision as a small reddish blob, which would help draw the eyes in that direction. However, the red blob could fall into a gap in the viewer’s visual field, and so not be noticed at all.

Consider the sequence of events from a user’s point of view. The user enters a username and password and then clicks “Sign In”. The page redisplayed with blank fields. The user thinks “Huh? I gave it my login information and hit Sign In, didn’t I? Did I hit the wrong button?” The user reenters the username and password, and clicks “Sign In” again. The page redisplayed with empty fields again. Now the user is really confused. The user sighs (or curses), sits back in his chair and lets his eyes scan the screen. Suddenly noticing the error message, the user says “Aha! Has that error message been there all along?”

Even when an error message is placed nearer to center of the viewer’s visual field than in the above example, other factors can diminish its visibility. For example, until recently the Web site of Airborne.com signaled a login failure by displaying an error message in red just above the Login ID field (see Fig. 6.6). This error message is entirely in red and fairly near the “Login” button where the user’s eyes are probably focused. Nonetheless, some users would not notice this error message

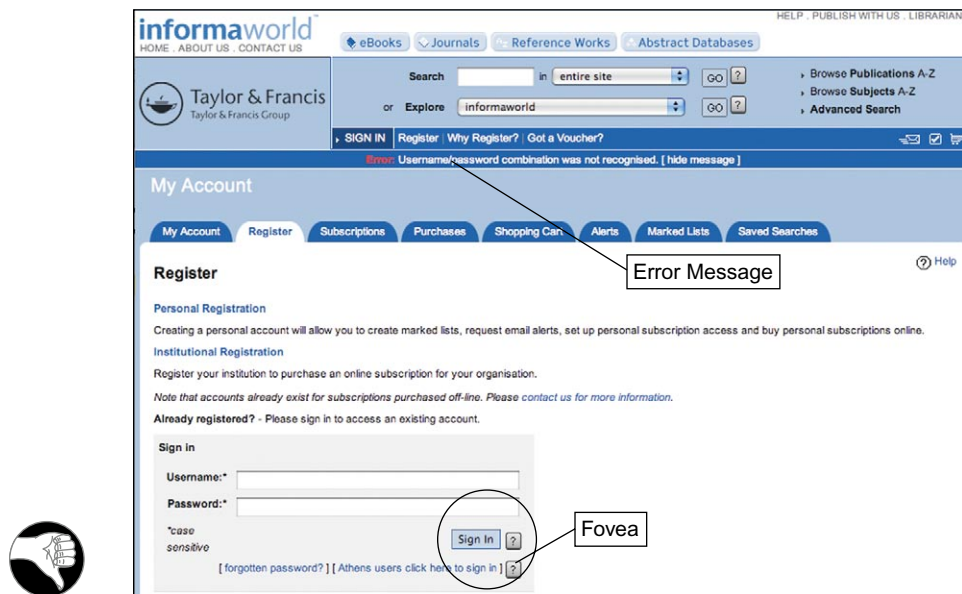


FIGURE 6.5

This error message for faulty sign-in appears in peripheral vision, where it will probably be missed.



**FIGURE 6.6**

This error message for faulty login is missed by some users even though it is not far from “Login” button.

RETURNING CUSTOMER LOGIN

Login ID not found.

Login ID: f000

Password:

Remember my Login ID for faster logins.

LOGIN



**FIGURE 6.7**

A simulation of a user’s visual field while the fovea is fixed on the “Login” button.

when it first appeared. Why? Can you think of any reasons why people might not initially see this error message?





One reason is that even though the error message is much closer to where users will be looking when they click the “Login” button, it is still in the periphery, not in the fovea. The fovea is small: just a centimeter or two on a computer screen, assuming the user is the usual distance from the screen.

A second reason is that the error message is not the only thing near the top of the page that is red. The page title is also red. Resolution in the periphery is low, so when the error message appears, the user’s visual system may not register any change: there was something red up there before, and there still is (see Fig. 6.7).

If the page title were black or any other color besides red, the red error message would be more likely to be noticed, even though it appears in the periphery of the users' visual field.

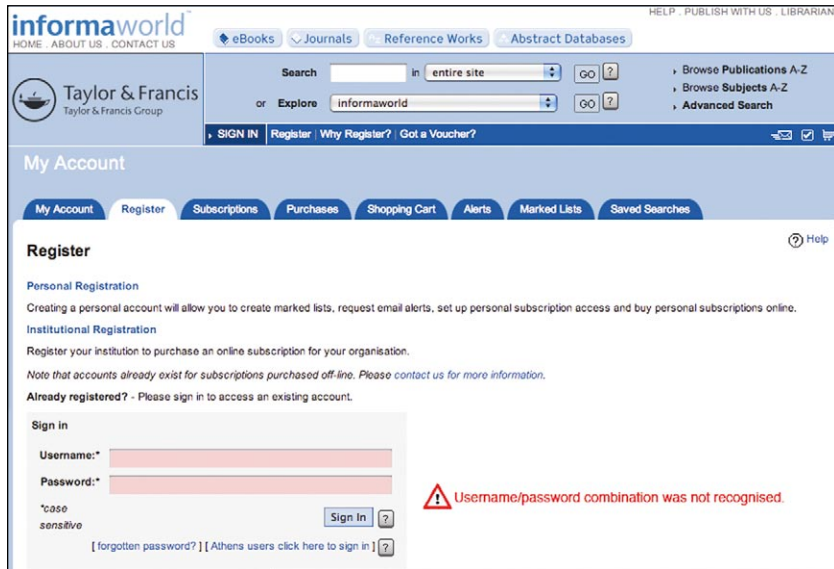
## COMMON METHODS OF MAKING MESSAGES VISIBLE

There are several common and well-known methods of ensuring that an error message will be seen:

- **Put it where users are looking:** People focus in predictable places when interacting with graphical user interfaces (GUIs). In Western societies, people tend to traverse forms and control panels from upper left to lower right. While moving the screen pointer, people usually look either at where it is or where they are moving it to. When people click a button or link, they can usually be assumed to be looking directly at it, at least for a few moments afterward. Designers can use this predictability to position error messages near where they expect users to be looking.
- **Mark the error:** Somehow mark the error prominently to indicate clearly that something is wrong. Often this can be done by simply placing the error message near what it refers to, unless that would place the message too far from where users are likely to be looking.
- **Use an error symbol:** Make errors or error messages more visible by marking them with an error symbol, such as , , , or .
- **Reserve red for errors:** By convention, in interactive computer systems the color red connotes *alert*, *danger*, *problem*, *error*, etc. Using red for any other information on a computer display invites misinterpretation. But suppose you are designing a Web site for Stanford University, which has red as its school color. Or suppose you are designing for a Chinese market, where red is considered an auspicious, positive color. What do you do? Use another color for errors, mark them with error symbols, or use stronger methods (see below).

An improved version of the InformaWorld sign-in error screen uses several of these techniques (see [Fig. 6.8](#)).

At America Online's Web site, the form for registering for a new email account follows the guidelines pretty well (see [Fig. 6.9](#)). Data fields with errors are marked with red error symbols. Error messages are displayed in red and are near the error. Furthermore, most of the error messages appear as soon as an erroneous entry is made, when the user is still focused on that part of the form, rather than only after the user submits the form. It is unlikely that AOL users will miss seeing these error messages.

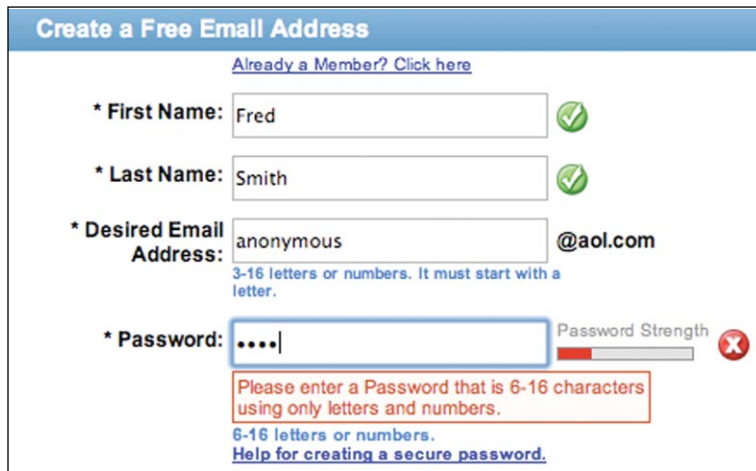


The screenshot shows the Taylor & Francis website's registration page. At the top, there are navigation links for 'eBooks', 'Journals', 'Reference Works', and 'Abstract Databases'. A search bar is present with 'informaworld' entered. Below the search bar, there are links for 'SIGN IN', 'Register', 'Why Register?', and 'Got a Voucher?'. The main heading is 'My Account' with sub-links for 'My Account', 'Register', 'Subscriptions', 'Purchases', 'Shopping Cart', 'Alerts', 'Marked Lists', and 'Saved Searches'. The 'Register' section is active, showing options for 'Personal Registration' and 'Institutional Registration'. A 'Sign in' form is visible with fields for 'Username:\*' and 'Password:\*'. A red error message with a warning icon is displayed: 'Username/password combination was not recognised.' The error message is positioned to the right of the password field, making it highly visible to the user.



FIGURE 6.8

This error message for faulty sign-in is displayed more prominently, near where users will be looking.



The screenshot shows the AOL.com registration page titled 'Create a Free Email Address'. It includes a link for 'Already a Member? Click here'. The form has four main sections:
 

- \* First Name:** 'Fred' with a green checkmark.
- \* Last Name:** 'Smith' with a green checkmark.
- \* Desired Email Address:** 'anonymous@aol.com' with a note: '3-16 letters or numbers. It must start with a letter.'
- \* Password:** '....|' with a 'Password Strength' indicator showing a red bar and a red 'X' icon. A red error message box is overlaid on the password field: 'Please enter a Password that is 6-16 characters using only letters and numbers. 6-16 letters or numbers. Help for creating a secure password.'



FIGURE 6.9

New member registration at AOL.com displays error messages prominently, near each error.

## HEAVY ARTILLERY FOR MAKING USERS NOTICE MESSAGES: USE SPARINGLY

If the common, conventional methods of making users notice messages are not enough, three stronger methods are available to user interface designers. However, these methods, while very effective, have significant negative effects so they should be used sparingly and with great care.

### Pop-up message in error dialog box

Displaying an error message in a dialog box sticks it right in the user's face, making it hard to miss. Error dialog boxes interrupt the user's work and demand immediate attention. That is good if the error message signals a critical condition, but it can annoy people if such an approach is used for a minor message, such as confirming the execution of a user-requested action.

The annoyance of pop-up messages rises with the degree of modality. *Nonmodal* pop-ups allow users to ignore them and continue working. *Application-modal* pop-ups block any further work in the application that displayed the error, but allow users to interact with other software on their computer. *System-modal* pop-ups block any user action until the dialog has been dismissed.

Application-modal error pop-ups should be used sparingly, e.g., only when application data may be lost if the user doesn't attend to the error. System-modal pop-ups should be used extremely rarely—basically only when the system is about to crash and take hours of work with it or if people will die if the user misses the error message.

On the Web, an additional reason to avoid pop-up error dialog boxes is that some people set their browsers to block *all* pop-up windows. If your Web site relies on pop-up error messages, some users may never see them.

### Use sound (e.g., beep)

When a computer beeps, that tells its user something has happened that requires attention. The person's eyes reflexively begin scanning the screen for whatever caused the beep. This can allow the user to notice an error message that is someplace other than where the user was just looking, such as in a standard error message box on the display. That is the value of beeping.

However, imagine many people in a cubicle work environment or a classroom, all using an application that signals all errors and warnings by beeping. Such a workplace would be very annoying, to say the least. Worse, people wouldn't be able to tell whether their own computer or someone else's was beeping.

In noisy work environments, e.g., factories or computer server rooms, beeps might be masked by ambient noise.

Finally, sound is muted or turned way down on some people's computers. Therefore, signaling errors and other conditions with sound are remedies that can be used only in very special, controlled situations.

## Flash or wiggle briefly

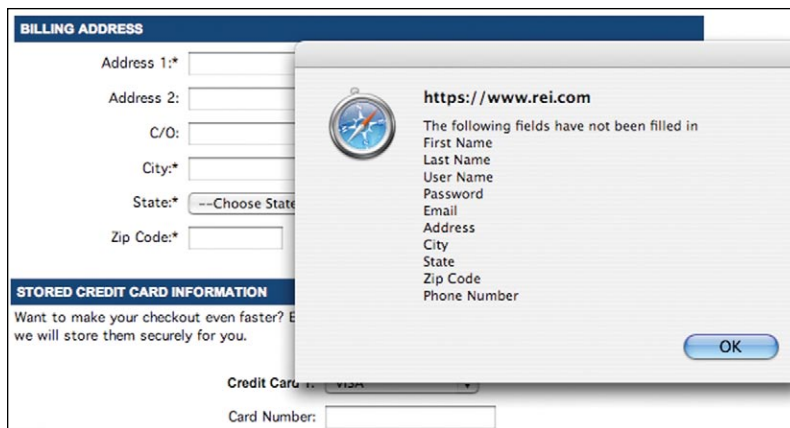
As described earlier, our peripheral vision is good at detecting motion, and motion in the periphery causes reflexive eye movements that bring the motion into the fovea. User interface designers can make use of this by wiggling or flashing messages briefly when they want to ensure that users see them. It doesn't take much motion to trigger eye movement toward the motion. Just a tiny bit of motion is enough to make a viewer's eyes zip over in that direction. Millions of years of evolution have had quite an effect.

However, motion, like pop-up dialog boxes and beeping, must be used sparingly. Most experienced computer users consider wiggling, blinking objects on screen to be annoying. Most of us have learned to ignore displays that blink because many such displays are advertisements. Conversely, a few computer users have attentional impairments that make it difficult for them to ignore something that is blinking or wiggling.

Therefore, if motion or blinking is used, it should be brief: it should last about a quarter- to a half-second—no longer. Otherwise, it quickly goes from an unconscious attention-grabber to a conscious annoyance.

## Use them sparingly

Use all of these “heavy artillery” methods sparingly—only for critical messages. When pop-ups, sound, motion, and blinking are used frequently to attract users' attention, a psychological phenomenon called *habituation* sets in. Our brain pays less and less attention to any stimulus that occurs frequently. It is like the old fable about the boy who cried “Wolf” too often: eventually, the villagers learned to ignore



**FIGURE 6.10**

REI's pop-up dialog box signals required data that was omitted. It is hard to miss, but perhaps overkill.

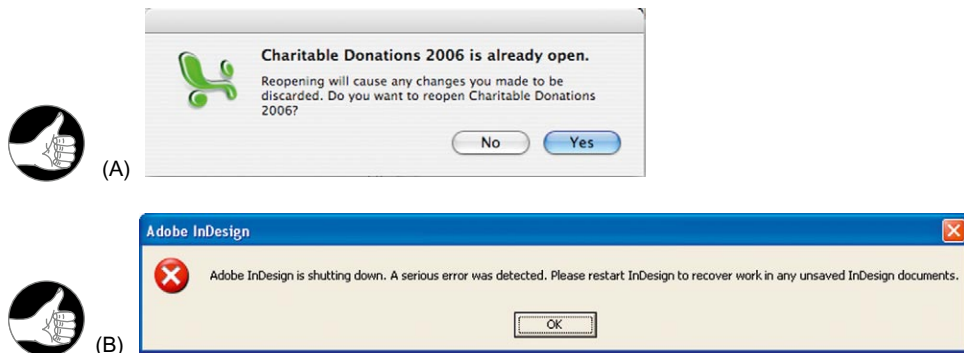
his cries, so when a wolf actually did come, his cries went unheeded. Overuse of strong attention-getting methods can cause important messages to be blocked by habituation.

REL.com has an example of a pop-up dialog being used to display an error message. The message is displayed when someone who is registering as a new customer omits required fields in the form (see Fig. 6.10). Is this an appropriate use of a pop-up dialog? AOL.com (see Fig. 6.9 above) shows that missing-data errors can be signaled quite well without pop-up dialogs, so REL.com's use of them seems a bit heavy-handed.

Examples of more appropriate use of error dialog boxes come from Microsoft Excel (see Fig. 6.11A) and Adobe InDesign (see Fig. 6.11B). In both cases, loss of data is at stake.

Computer games use sound a lot to signal events and conditions. In games, sound isn't annoying; it is expected. Its use in games is widespread, even in game arcades, where dozens of machines are all banging, roaring, buzzing, clanging, beeping, and playing music at once. (Well, it is annoying to parents who have to go into the arcades and endure all the screeching and booming to retrieve their kids, but the games aren't designed for parents.)

The most common use of blinking in computer user interfaces (other than advertisements) is in menu bar menus. When an action—e.g., Edit > Copy—is selected from a menu, it usually blinks once before the menu closes to confirm that the system “got” the command, i.e., that the user didn't miss the menu item. This use of blinking is very common. It is so quick that most computer users aren't even aware of it, but if menu items didn't blink once, we would have less confidence that we actually selected them.



**FIGURE 6.11**

Appropriate pop-up error dialogs: (A) Microsoft Excel, (B) Adobe InDesign.



**FIGURE 6.12**

An error message can wiggle briefly after it appears, to attract a user's fovea toward it.

As an example of using motion to attract users' eye attention, imagine that the error message on the improved InformaWorld sign-in error screen (Fig. 6.8, above) appeared, wiggled one pixel up, down, left, and right within 0.25 second (see Fig. 6.12), and then stopped to become a simple static image. Without being annoying, that would attract the users' eyeballs, guaranteed. Because, after all, the motion in the corner of your eye might be a leopard.



# Our Attention is Limited; Our Memory is Imperfect

# 7

Just as the human visual system has strengths and weaknesses, so does human memory. This chapter describes some of those strengths and weaknesses as background for understanding how we can design interactive systems to support and augment human memory rather than burdening or confusing it. We will start with an overview of how memory works.

## SHORT VS. LONG-TERM MEMORY

Psychologists historically have distinguished *short-term* memory from *long-term* memory. Short-term memory covers situations in which information is retained for very short intervals ranging from a fraction of a second up to several seconds—perhaps as long as a minute. Long-term memory covers situations in which information is retained over longer periods, e.g., minutes, hours, days, years, even lifetimes.

It is tempting to think of short-term and long-term memory as separate memory stores. Indeed, some theories of memory have considered them separate. After all, in a digital computer, the short-term memory stores (central processing unit data registers) are separate from the long-term memory stores (random access memory or RAM, hard disk, flash memory, CD-ROM, etc.). More direct evidence comes from findings that damage to certain parts of the human brain results in short-term memory deficits but not long-term ones, or vice versa. Finally, the speed with which information or plans can disappear from our immediate awareness contrasts sharply with the seeming permanence of our memory of important events in our lives, faces of significant people, activities we have practiced, and information we have studied. These phenomena led many researchers to theorize that short-term memory is a separate store in the brain where information is held temporarily after entering through our perceptual senses (e.g., visual or auditory) or after being retrieved from long-term memory (see [Fig. 7.1](#)).

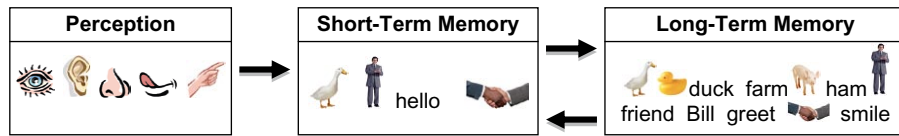


FIGURE 7.1

Traditional (antiquated) view of short-term versus long-term memory.

## A MODERN VIEW OF MEMORY

Recent research on memory and brain function indicates that short- and long-term memory are functions of a single memory system—one that is more closely linked with perception than previously thought (Jonides et al., 2008).

### Long-term memory

Perceptions enter through the visual, auditory, olfactory, gustatory, or tactile sensory systems and trigger responses starting in areas of the brain dedicated to each sense (e.g., visual cortex, auditory cortex), then spreading into other areas of the brain that are *not* specific to any particular sensory modality. The sensory-modality-specific areas of the brain detect only simple features of the data, such as a *dark-light edge*, *diagonal line*, *high-pitched tone*, *sour taste*, *red color*, or *rightward motion*. Downstream areas of the brain combine low-level features to detect higher-level features of the input, such as *animal*, *Uncle Kevin*, *minor key*, *threat*, or *the word “duck.”*

The set of neurons affected by a perception depends largely on its features and context. The context is just as important as the features of the perception. For example, a dog barking near you when you are walking in your neighborhood stimulates a different pattern of neural activity in your brain than the same sound heard when you are safely inside your car. The more similar two perceptual stimuli are—i.e., the more features and contextual elements they share—the more overlap there is between the sets of neurons that fire in response to them.

The initial strength of a perception depends on how much it is amplified or dampened by other brain activity. All perceptions create some kind of trace, but some are so weak that they can be considered as not registered: the pattern was activated once but never again.

Memory formation consists of long-lasting and even permanent changes in the neurons involved in a neural activity pattern, which make the pattern easier to reactivate in the future.<sup>1</sup> Some such changes involve the release of chemicals into

<sup>1</sup>There is evidence that the long-term neural changes associated with learning occur mainly during sleep, suggesting that separating learning sessions by periods of sleep may facilitate learning (Stafford & Webb, 2005).

the areas around neurons that change their sensitivity to stimulation for fairly long periods of time, until the chemicals dissipate or are neutralized by other chemicals. More permanent changes consist of neurons growing and forming new connections with other neurons.

Activating a memory consists of reactivating the same pattern of neural activity that occurred when the memory was formed. Somehow the brain distinguishes initial activations of neural patterns from *reactivations*—perhaps by measuring the relative ease with which the pattern was reactivated. New perceptions very similar to the original ones reactivate the same patterns of neurons, resulting in *recognition* if the reactivated perception reaches awareness. In the absence of a similar perception, stimulation from activity in other parts of the brain can also reactivate a pattern of neural activity, which if it reaches awareness results in *recall*.

The more often a neural memory pattern is reactivated, the “stronger” it becomes—that is, the easier it is to reactivate—which in turn means that the perception it corresponds to is easier to recognize and recall. Neural memory patterns can also be strengthened or weakened by excitatory or inhibitory signals from other parts of the brain.

A particular memory is not located in any specific spot in the brain. The neural activity pattern comprising a memory involves a network of neurons extending over a wide area. Activity patterns for different memories overlap, depending on which features they share. Removing, damaging, or inhibiting neurons in a particular part of the brain typically does not completely wipe out memories that involve those neurons, but rather just reduces their level of detail or accuracy by deleting features.<sup>2</sup> However, some areas in a neural activity pattern may be critical pathways, so that removing, damaging, or inhibiting them may prevent most of the pattern from activating, thereby effectively eliminating the corresponding memory.

### Short-term memory

The processes discussed above are about long-term memory. Where is short-term memory in all of this? The answer is suggested by the word “awareness.”

Short-term memory is not a *store*—it is not a *place* where memories and perceptions *go* to be worked on. More precisely, it is not a temporary repository for information just brought in from the sensory system or retrieved from long-term memory. Instead, short-term memory is a combination of phenomena arising from perception and attention.

Each of our perceptual senses has its own very brief short-term “memory” that is the result of residual neural activity after a perceptual stimulus ceases, like a bell that rings briefly after it is struck. Until they fade away, these residual perceptions are available as possible input to our brain’s attention mechanisms, which integrate

---

<sup>2</sup>This is similar to the effect of cutting pieces out of a holographic image: it reduces the overall resolution of the image, rather than removing pieces of it as with an ordinary photograph.

input from our various perceptual systems and focus our awareness on some of that input. These sensory-specific residual perceptions together comprise a minor component of our short-term memory.

Also available as potential input to our attention mechanisms are long-term memories reactivated through recognition or recall. As explained above, each memory corresponds to a specific pattern of neural activity distributed across our brain. While activated, a memory pattern is a candidate for our attention.

The human brain has multiple attention mechanisms, some voluntary and some involuntary. They focus our awareness on a very small subset of the perceptions and activated long-term memories while ignoring everything else. That tiny subset of all of the available information from our perceptual systems and our long-term memories that we are conscious of *right now* is the main component of our *short-term memory*, the part that cognitive scientists often call *working memory*.

Don't think of working memory as a temporary buffer where perceptions and memories are *brought* to allow our brains to work on them. Instead, think of it as the combined *focus of attention*—the currently activated neural patterns of which we are aware. The number of items in short-term memory at any given moment is extremely limited and volatile.

What about the earlier finding that damage to some parts of the brain causes short-term memory deficits, while other types of brain damage cause long-term memory deficits? The current interpretation of that finding is that some types of damage decrease or eliminate the brain's ability to focus attention on specific objects and events, while other types of damage harm the brain's ability to store or recall long-term memories.

---

## CHARACTERISTICS OF SHORT-TERM MEMORY

Short-term memory, as described above, is equal to the focus of our attention. Whatever is in that focus is what we are conscious of at any moment.

Right now you are conscious of the last few words and ideas you've read, but probably not the color of the wall in front of you. But now that I've shifted your attention, you *are* conscious of the wall's color, and may have forgotten some of the ideas you read on the previous page.

The primary characteristics of short-term memory are its low capacity and its volatility.

The low capacity of short-term memory is fairly well known. Many college-educated people have read about “the magical number seven, plus or minus two,” proposed by cognitive psychologist George Miller in 1956 as the limit on the number of simultaneous unrelated items in human short-term memory (Miller, 1956).

Miller's characterization of the short-term memory limit naturally raises several questions.

- ***What are the items in short-term memory?*** They are current perceptions and retrieved memories. They are goals, numbers, words, names, sounds, images, odors—anything one can be aware of.
- ***Why must items be unrelated?*** Because if two items are related, they correspond to one big neural activity pattern—one set of features—and hence one item, not two.
- ***Why the fudge-factor of plus or minus two?*** Because researchers cannot measure with perfect accuracy how much people can recall, and because of differences between individuals in how much they can remember.

Later research in the 1960s and 1970s found Miller's estimate to be too high. In the experiments Miller considered, some of the items presented to people to remember could be "chunked" (i.e., considered related), making it appear that people's short-term memory was holding more items than it actually was. When the experiments were revised to disallow unintended chunking, the capacity of short-term memory was shown to be more like four plus or minus one, that is, three to five items (Broadbent, 1975).

More recent research has cast doubt on the idea that the capacity of short-term memory should be measured in whole items or "chunks." It turns out that in early short-term memory experiments, people were asked to briefly remember items (e.g., words or images) that were quite different from each other, i.e., that had very few features in common. In such a situation, people don't have to remember every feature of an item in order to recall it a few seconds later; remembering some of its features is enough. So people appeared to recall items as a whole, and therefore short-term memory capacity seemed measurable in whole items.

Recent experiments have given people items to remember that were similar, i.e., they shared features. In that situation, to recall an item and not confuse it with other items, people need to remember more of its features. In these experiments, the finding was that people remember more details (i.e., features) of some items than of others, and the items they remember in greater detail are the ones they paid more attention to (Bays and Husain, 2008). This finding suggests that the unit of attention—and therefore the capacity limit on short-term memory—is best measured in item-features rather than whole items or "chunks" (Cowan, Chen, & Rouder, 2004).

The second important characteristic of short-term memory is how volatile it is. Cognitive psychologists used to say that new items arriving in short-term memory often bump old ones out, but that way of describing the volatility is based on the view of short-term memory as a temporary storage *place* for information. The

modern view of short-term memory as the current focus of attention makes it even clearer: focusing attention on new information turns it away from some of what it was focusing on.

However we describe it, information can easily be lost from short-term memory. If items in short-term memory don't get combined or rehearsed, they are at risk of having the focus shifted away from them. This volatility applies to goals as well as to the details of objects. Losing items from short-term memory corresponds to forgetting or losing track of something you were doing. We have all had such experiences, for example:

- Going to another room for something, but once there we can't remember why we came.
- Taking a phone call, and afterward not remembering what we were doing before the call.
- Something yanks our attention away from a conversation, and then we can't remember what we were talking about.
- In the middle of adding a long list of numbers, something distracts us, so we have to start over.

One way researchers have shown that short-term memory is limited in capacity and duration is to show people a picture, then show them a second version of the same picture and ask them if the second picture is the same or different from the first. Surprisingly, the second picture can differ from the first in many ways without people noticing any difference. To explore further, researchers gave people questions to answer about the first picture, affecting their goals in looking at it, and therefore what features of the picture they pay attention to. Result: people don't notice differences in features other than those their goals made them pay attention to. This is called "change blindness" (Angier, 2008).

A particularly striking example of the volatility of short-term memory comes from experiments in which experimenters holding city maps posed as lost tourists and asked local people walking by for directions. When the local person focused on the tourist's map in order to figure out the best route, two workmen—actually more experimenters—walked between the "tourist" and the advice-giver carrying a large door, and in that moment the "tourist" was replaced by another experimenter—"tourist." Astoundingly, after the door passed, over half of the local people continued helping the "tourist" without noticing any change, even when the two "tourists" differed in hair color or in whether they had a beard (Simons & Levin, 1998). Some people even failed to notice changes in gender. Conclusion: people focus on the "tourist" only long enough to determine if they are a threat or worth helping, "record" only that the person is a tourist who needs help, and then focus on the map and the task of giving directions.

## A SHORT-TERM MEMORY TEST

To test your short-term memory, get a pen or pencil and two blank sheets of paper and follow these instructions:

1. Place one blank sheet of paper after this page in the book and use it to cover the next page.
2. Flip to the next page for 3 seconds, pull the paper cover down and read the **black numbers** at the top, and flip back to this page. Don't peek at other numbers on that page unless you want to ruin the test.
3. Say your phone number backward, out loud.
4. Now write down the black numbers from memory. ... Did you get all of them?
5. Flip back to the next page for 3 seconds, read the **red numbers** (under the black ones), and flip back.
6. Write down the numbers from memory. These would be easier to recall than the first ones if you noticed that they are the first seven digits of  $\pi$  (3.141592), because then they would be only one number, not seven.
7. Flip back to the next page for 3 seconds, read the **green numbers**, and flip back.
8. Write down the numbers from memory. If you noticed that they are odd numbers from 1 to 13, they would be easier to recall, because they would be three chunks ("odd, 1, 13" or "odd, seven, from 1"), not seven.
9. Flip back to the next page for 3 seconds, read the **orange words**, and flip back.
10. Write down the words from memory. ... Could you recall them all?
11. Flip back to the next page for 3 seconds, read the **blue words**, and flip back.
12. Write down the words from memory. ... It was certainly a lot easier recall them all because they form a sentence, so they could be memorized as one sentence rather than seven words.

3 8 4 7 5 3 9

3 1 4 1 5 9 2

1 3 5 7 9 11 13

town river corn string car shovel

what is the meaning of life

## IMPLICATIONS OF SHORT-TERM MEMORY CHARACTERISTICS FOR USER INTERFACE DESIGN

The capacity and volatility of short-term memory have many implications for the design of interactive computer systems. The basic implication is that user interfaces should help people remember essential information from one moment to the next. Don't require people to remember system status or what they have done, because their attention is focused on their primary goal and progress toward it. Specific examples follow.

### Modes

The limited capacity and volatility of short-term memory is one reason why user-interface design guidelines often say to either avoid designs that have *modes* or provide adequate mode-feedback. In a moded user interface, some user actions have different effects depending on what mode the system is in. For example:

- In a car, pressing the accelerator pedal can move the car either forwards, backwards or not at all, depending on whether the transmission is in drive, reverse, or neutral. The transmission sets a mode in the car's user interface.
- In many digital cameras, pressing the shutter button can either snap a photo or start a video recording, depending on which mode is selected.



- In a drawing program, clicking and dragging normally selects one or more graphic objects on the drawing, but when the software is in “draw rectangle” mode, clicking and dragging adds a rectangle to the drawing and stretches it to the desired size.

Moded user interfaces have advantages; that is why many interactive systems have them. Modes allow a device to have more functions than controls: the same control provides different functions in different modes. Modes allow an interactive system to assign different meanings to the same gestures in order to reduce the number of gestures users must learn.

However, one well-known *disadvantage* of modes is that people often make *mode-errors*: they forget what mode the system is in and do the wrong thing by mistake (Johnson, 1990). This is especially true in systems that give poor feedback about what the current mode is. Because of the problem of mode-errors, many user interface design guidelines say to either avoid modes or provide strong feedback about which mode the system is in. Human short-term memory is too unreliable for designers to assume that users can, without clear, continuous feedback, keep track of what mode the system is in, even when the users are the ones changing the system from one mode to another.

## Search results

When people use a search function on a computer to find information, they enter the search terms, start the search, and then review the results. Evaluating the results often requires knowing what the search terms were. If short-term memory were less limited, people would always remember, when browsing the results, what they had entered as search terms just a few seconds earlier. But as we have seen, short-term memory is very limited. When the results appear, the person’s attention naturally turns away from what they entered and toward the results. Therefore, it should be no surprise that people viewing search results often do not remember the search terms they just typed.

Unfortunately, some designers of online search functions don’t understand that. Search-results sometimes don’t show the search terms that generated the results. For example, in 2006, the search-results page at Slate.com provided search fields so users could search again, but didn’t show what a user had searched for (see Fig. 7.2A). A more recent version of the site does show the user’s search terms (see Fig. 7.2B), reducing the burden on users’ short-term memory.

## Instructions

If you asked a friend for a recipe or for directions to her home, and she gave you a long sequence of steps, you probably would not try to remember it all. You would know that you could not reliably keep all of the instructions in your short-term memory, so you would write them down or ask your friend to send them to you by email. Later, while following the instructions, you would put them where you could refer to them until you reached the goal.



(A)

Search for:

Advanced Search Options

Topics:

Departments:

Authors:

Publication Date from:  to:

Found 968 matches. << 1 - 25 of 968 >>

Ranks	Headline	Author	Published	Department
****	<b>Defendant DeLay? Part 2</b> Who blurted out, "\$100,000"? A hypothesis.	<a href="#">Timothy Noah</a>	Oct 06, 2004	<a href="#">Chatterbox</a>
****	<b>The Tom DeLay Scandals</b> A scorecard.	<a href="#">Nicholas Thompson</a>	Apr 07, 2005	<a href="#">Gist, The</a>
****	<b>The Wall Street Journal vs. Tom DeLay</b> Has the editorial page gotten ... nice?	<a href="#">Timothy Noah</a>	Dec 12, 2001	<a href="#">Chatterbox</a>
****	<b>Defendant DeLay?</b> Nick Smith's bribery accusations.	<a href="#">Timothy Noah</a>	Oct 01, 2004	<a href="#">Chatterbox</a>



(B)

SEARCH FOR:

Date(s):  Past 7 Days  Past 30 Days  Past 90 Days  Past Year  Since 1996

barack obama

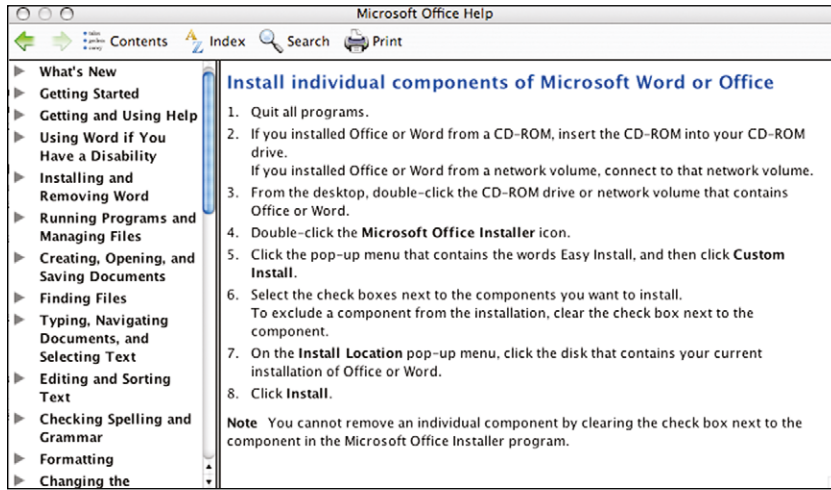
► Advanced

SORT BY « 1 - 25 OF 2131 »

HEADLINE	AUTHOR	DATE	DEPT.
<b>The Truth About Barack Obama</b> Rumors the Obama campaign shouldn't try to correct.	<a href="#">Christopher Beam</a>	Jun 17, 2008	<a href="#">Low Concept</a>
<b>Slate Votes</b> Obama wins this magazine in a rout.		Oct 28, 2008	<a href="#">Politics</a>
<b>Barack Obama's Facebook Feed</b> Every update from Slate's ongoing coverage of the president's secret social networking.		Jun 12, 2009	<a href="#">Politics</a>
<b>Dropping in on Obama's Kenyan Grandmother</b> What it means to be an Obama in Africa.	<a href="#">Andy Isaacson</a>	Oct 28, 2008	<a href="#">Dispatches</a>
<b>The Obama Marriage</b> How does it work for Michelle Obama?	<a href="#">Melinda Henneberger</a>	Oct 26, 2007	<a href="#">First Mates</a>

FIGURE 7.2

Slate.com search results: (A) in 2007, users' search terms not shown, (B) in 2009 search terms shown.


**FIGURE 7.3**

Instructions in Windows Help files remain displayed while users follow them.


**FIGURE 7.4**

Instructions for Windows XP wireless setup start by telling users to close the instructions.

Similarly, interactive systems that display instructions for multistep operations should allow people to refer to the instructions while executing them until completing all the steps. Most interactive systems do this (see Fig. 7.3), but some do not (see Fig. 7.4).

## CHARACTERISTICS OF LONG-TERM MEMORY

Long-term memory differs from short-term memory in many respects. Unlike short-term memory, it actually *is* a memory store.

However, specific memories are not stored in any one neuron or location in the brain. As described above, memories, like perceptions, consist of patterns of activation of large sets of neurons. Related memories correspond to overlapping patterns

of activated neurons. This means that every memory is stored in a distributed fashion, spread among many parts of the brain. In this way, long-term memory in the brain is similar to holographic light images.

Long-term memory evolved to serve our ancestors and us very well in getting around in our world. However, it has many weaknesses: it is error-prone, impressionist, free-associative, idiosyncratic, retroactively alterable, and easily biased by a variety of factors at the time of recording or of retrieval. Let's examine some of these weaknesses.

### Error prone

Nearly everything we've ever experienced is stored in our long-term memory. Unlike short-term memory, the capacity of human long-term memory seems almost unlimited. Although Landauer (1986) used the average human learning rate to calculate the amount of information a person can learn in a lifetime,<sup>3</sup> no one has yet measured or even estimated the maximum information capacity of the human brain.

However, what is in long-term memory is not an accurate, high-resolution recording of our experiences. In terms familiar to computer engineers, one could characterize long-term memory as using heavy compression methods that drop a great deal of information. Images, concepts, events, sensations, actions—all are reduced to combinations of abstract features. Different memories are stored at different levels of detail, that is, with more or fewer features.

For example, the face of a man you met briefly who is not important to you might be stored simply as an average Caucasian male face with a beard, with no other details—a whole face reduced to three features. If you were asked later to describe the man in his absence, the most you could honestly say was that he was a “white guy with a beard.” You would not be able to pick him out of a police lineup of other Caucasian men with beards. In contrast, your memory of your best friend's face includes many more features, allowing you to give a more detailed description and pick your friend out of any police lineup. Nonetheless, it is still a set of features, not anything like a bitmap image.

As another example, I have a vivid childhood memory of being run over by a plow and badly cut, but my father says it happened to my brother. One of us is wrong.

In the realm of human-computer interaction, Microsoft Word users may remember that there is a command to insert a page number, but they may not remember which menu the command is in. That specific feature may not have been recorded when the user learned how to insert page numbers. Alternatively, perhaps the menu-location feature *was* recorded, but did not reactivate with the rest of the memory when the user tried to recall how to insert a page number.

---

<sup>3</sup> 10<sup>9</sup> bits, or a few hundred megabytes.

### Weighted by emotions

Chapter 1 described a dog that remembered seeing a cat in his front yard every time he returned home in the family car. The dog was excited when he first saw the cat, so his memory of it was strong and vivid.

Another example: an adult could easily have strong memories of her first day at nursery school, but probably not of her tenth. On the first day, she was probably upset about being left at the school by her parents, whereas by the tenth day, being left there was nothing unusual.

### Retroactively alterable

Suppose that while you are on an ocean cruise with your family, you see a whale-shark. Years later, when you and your family are discussing the trip, you might remember seeing a whale, and one of your relatives might recall seeing a shark. For both of you, some details in long-term memory were dropped because they did not fit a common concept.

A true example comes from 1983, when the late President Ronald Reagan was speaking with Jewish leaders during his first term as president. He spoke about being in Europe during World War II and helping to liberate Jews from the Nazi concentration camps. The trouble was, he was never in Europe during World War II. When he was an actor, he was in a *movie* about World War II, made entirely in Hollywood.

## A LONG-TERM MEMORY TEST

Test your long-term memory by answering the following questions:

1. Was there a roll of tape in the toolbox in Chapter 1?
2. What was your *previous* phone number?
3. Which of these words were *not* in the list presented in the short-term memory test earlier in this chapter?  
**city stream corn auto twine spade**
4. What was your first grade teacher's name? Second grade? Third grade? ...
5. What Web site was presented earlier that does not show search terms when it displays search results?

Regarding question 3: When words are memorized, often what is retained is the *concept*, rather than the exact word that was presented. For example, one could hear the word "town" and later recall it as "city."

## IMPLICATIONS OF LONG-TERM MEMORY CHARACTERISTICS FOR USER INTERFACE DESIGN

The main thing that the characteristics of long-term memory imply is that people need tools to augment it. Since prehistoric times, people have invented technologies to help them remember things over long periods: notched sticks, knotted ropes, mnemonics, verbal stories and histories retold around campfires, writing, scrolls, books, number systems, shopping lists, checklists, phone directories, datebooks, accounting ledgers, oven timers, computers, portable digital assistants (PDAs), online shared calendars, etc.

Given that humankind has a need for technologies that *augment* memory, it seems clear that software designers should try to provide software that fulfills that need. At the very least, designers should avoid developing systems that *burden* long-term memory. Yet that is exactly what many interactive systems do.

Authentication is one functional area in which many software systems place burdensome demands on users' long-term memory. For example, a Web application developed a few years ago told users to change their personal identification number (PIN) "to a number that is easy ... to remember," but then imposed restrictions that made it impossible to do so (Fig. 7.5). Whoever wrote those instructions seems to have realized that the PIN requirements were unreasonable, because the instructions end by advising users to write down their PIN! Never mind that writing a PIN down creates a security risk and adds yet *another* memory task: users must remember where they hid their written-down PIN.

A contrasting example of burdening people's long-term memory for the sake of security comes from Intuit.com. To purchase software, visitors must register. The site requires users to select a security question from a menu (see Fig. 7.6). What if you can't answer *any* of the questions? What if you don't recall your first pet's name, your high school mascot, or any of the answers to the other questions?

But that isn't where the memory burden ends. Some questions could have several possible answers. Many people had several elementary schools, childhood friends, or heroes. In order to register, they must choose a question and then *remember* which answer they gave to Intuit. How? Probably by writing it down somewhere. Then, when Intuit.com asks them the security question, they have to *remember* where they put the answer. Why burden people's memory, when it would be easy to let users make up a security question for which they can easily recall the one possible answer?

Such unreasonable demands on people's long-term memory counteract the security and productivity that computer-based applications supposedly provide (Schrage, 2005), as users:

- place sticky notes on or near computers or "hide" them in desk drawers
- contact customer support to recover passwords they cannot recall
- use passwords that are easy for others to guess
- setup systems with no login requirements at all, or with one shared login and password



Instruction:

Change your PIN to a number that is easy for you to remember. A PIN can be 6-10 digits and cannot start with 0. Your PIN must be numeric.

New PIN:

Confirm New PIN:

Remember: Please write down your PIN.

**FIGURE 7.5**

Instructions tell users to create an easy-to-remember PIN, but the restrictions make that impossible.



Back to Sign In Cancel ×

**Create an Intuit online account.**

Email Address

Intuit User ID

Password

Confirm Password

Screen Name

Remember me

Security Question

Security Answer

**What is a Security Question?**

The Security Question and Answer are used in the event that you forget your Intuit User ID and Password.

The Security Answer is not case-sensitive.

✓ Select a security question...

- What was the name of your first pet?
- What was the name of your elementary school?
- What was the name of your childhood best friend?
- What was your high school mascot?
- Who was your childhood hero?

**FIGURE 7.6**

Intuit.com registration burdens long-term memory: users may have no unique, memorable answer for any of the questions.

The registration form at NetworkSolutions.com represents a small step toward more usable security. Like Intuit.com, it offers a choice of security questions, but it also allows users to create their own security question—one for which they can more easily remember the answer (see Fig. 7.7).

Another implication of long-term memory characteristics for interactive systems is that learning and long-term retention are enhanced by user-interface consistency.



Select your password security question:

Choose a question from the list:

OR

Create your own question: (i.e. What street did I grow up on?)

FIGURE 7.7

NetworkSolutions.com lets users create a security question if none on the menu works for them.

**Table 7.1** Which UI Design will be Easiest to Learn and Remember? Which One will be Hardest?

Object	Document Editor Keyboard Shortcuts: Alternative Designs					
	Design A		Design B		Design C	
	Cut	Paste	Cut	Paste	Cut	Paste
<b>Text</b>	CNTRL-X	CNTRL-V	CNTRL-X	CNTRL-V	CNTRL-X	CNTRL-V
<b>Sketch</b>	CNTRL-X	CNTRL-V	CNTRL-C	CNTRL-P	CNTRL-X	CNTRL-V
<b>Table</b>	CNTRL-X	CNTRL-V	CNTRL-Z	CNTRL-Y	CNTRL-X	CNTRL-V
<b>Image</b>	CNTRL-X	CNTRL-V	CNTRL-M	CNTRL-N	CNTRL-X	CNTRL-V
<b>Video</b>	CNTRL-X	CNTRL-V	CNTRL-Q	CNTRL-R	CNTRL-E	CNTRL-R

The more consistent the operation of different functions, or the more consistent the actions on different types of objects, the less users have to learn.<sup>4</sup> User interfaces that have many exceptions and little consistency from one function or object to another require users to store in long-term memory many features about each function or object and its correct usage-context. The need to encode more features makes such user interfaces harder to learn. It also makes it more likely that a user's memory will drop essential features during storage or retrieval, increasing the chances that the user will fail to remember, misremember, or make other memory errors.

Consider three alternative designs for the keyboard shortcuts for Cut and Paste in a hypothetical multimedia document editor. The document editor supports the creation of documents containing text, sketches, tables, images, and videos. In Design A, Cut and Paste have the same two keyboard shortcuts regardless of what type of content is being edited. In Design B, the keyboard shortcuts for Cut and Paste are different for every type of content. In Design C, all types of content *except* videos have the same Cut and Paste keyboard shortcuts. (see Table 7.1).

<sup>4</sup>See also Chapter 11.



The first question is: which of these designs is easiest to learn? It is fairly clear that Design A is easiest.

The second question is: which design is hardest to learn? That is a tougher question. It is tempting to say “Design B” because that one seems to be the least consistent of the three. However, the answer really depends on what we mean by “hardest to learn.”

If we mean “the design for which users will require the most time to become productive”, that is certainly Design B. It will take most users a long time to learn all the different Cut and Paste keyboard shortcuts for the different types of content. But people are remarkably adaptable if sufficiently motivated—they can learn amazingly arbitrary things if, say, using the software is required for their job. Eventually—maybe in a month—users would be comfortable and even quick with Design B. In contrast, users of Design C would begin to be productive in about the same short time as users of Design A—probably a matter of minutes.

However, if we interpret “hardest to learn” as meaning “the design for which users will take the longest to be error-free,” that is Design C. All the types of document content use the same shortcut keys for Cut and Paste except videos. Although users of Design C will be productive quickly, they would continue to make the error of trying to use CNTRL-X and CNTRL-V with videos for at least several months—perhaps forever.

Even though some have criticized the concept of consistency as ill-defined and easy to apply badly (Grudin, 1989), the fact is that consistency in a user-interface greatly reduces the burden on users’ long-term memory. Mark Twain once wrote: “If you tell the truth, you never have to remember anything.” One could also say “If everything worked the same way, you would not have to remember much.” We will return to the issue of consistency in Chapter 11.

# Limits on Attention, Shape, Thought and Action

# 8

When people interact purposefully with the world around them, including computer systems, some aspects of their behavior follow predictable patterns, some of which result from the limited capacity of attention and short-term memory. When interactive systems are designed to recognize and support those patterns, they fit better with the way people operate. Some user interface design rules, then, are based directly on the patterns and thus indirectly on the limits of short-term memory and attention. This chapter describes six important patterns.

## WE FOCUS ON OUR GOALS AND PAY LITTLE ATTENTION TO OUR TOOLS

As Chapter 7 explained, our attention has very limited capacity. When people are doing a task—trying to accomplish a goal—most of their attention is focused on the goals and data related to that task. Normally, people devote very little attention to the tools they are using to perform a task, whether they are using computer applications, online services, or interactive appliances. Instead, people think about their tools only superficially, and then only when necessary.

We are of course *capable* of attending to our tools. However, attention (i.e., short-term memory) is limited in capacity. When people refocus their attention on their tools, it is pulled away from the details of the task. This shift increases the chances of users losing track of what they were doing or exactly where they were in doing it.

For example, if your lawn mower stops running while you are mowing your lawn, you will immediately stop and focus on the mower. Restarting the mower becomes your primary task, with the mower itself as the focus. You pay scant attention to any tools you use to restart the mower, just as you paid scant attention to the mower when your primary focus was the lawn. After you restart the mower and resume mowing the lawn, you probably won't remember where you were in mowing the lawn, but the lawn itself shows you.

Other tasks—e.g., reading a document, measuring a table, counting goldfish in a fish tank—might not provide such a clear reminder of your interrupted task and your position in it. You might have to start over from the beginning. You might even forget what you were doing altogether and go off to do something else.

That is why most software design guidelines state that software applications and most Web sites should not call attention to themselves; they should fade into the background and allow users to focus on their own goals. That design guideline is even the title of a popular Web design book: *Don't Make Me Think* (Krug, 2005). The title means: if your software or Web site makes me think about *it*, rather than what I am trying to do, you've lost me.

---

## WE USE EXTERNAL AIDS TO KEEP TRACK OF WHAT WE ARE DOING

Because our short-term memory and attention are so limited, we learn not to rely on them. Instead, we mark up our environment to show us where we are in a task. Examples include these:

- **Counting objects:** If possible, we move already counted objects into a different pile to indicate which objects have already been counted. If we cannot move an object, we point to the last object counted. To keep track of the number we are on, we count on our fingers, draw marks, or write numbers.
- **Reading books:** When we stop reading, we insert bookmarks to show what page we were on.
- **Arithmetic:** We learn methods of doing arithmetic on paper, or we use a calculator.
- **Checklists:** We use checklists to aid both our long-term and short-term memory. In critical or rarely performed tasks, checklists help us remember everything that needs to be done. In that way, they augment our faulty *long-term* memory. While doing the task, we check off items as we complete them. That is a short-term memory aid. A checklist that we can't mark up is hard to use, so we copy it and mark the copy.
- **Editing documents:** People often keep to-be-edited documents, documents that are currently being edited, and already edited documents in separate folders.

One implication of this pattern is that interactive systems should indicate what users have done versus what they have not yet done. Most email applications do this by marking already-read versus unread messages, most Web sites do it by marking visited versus unvisited links, and many applications do it by marking completed steps of a multipart task (see Fig. 8.1).

A second design implication is that interactive systems should allow users to mark or move objects to indicate which ones they have worked on versus which ones they



**FIGURE 8.1**

Mac OS Software Update shows which updates are done (green check) versus which are in progress (rotating circle).



**FIGURE 8.2**

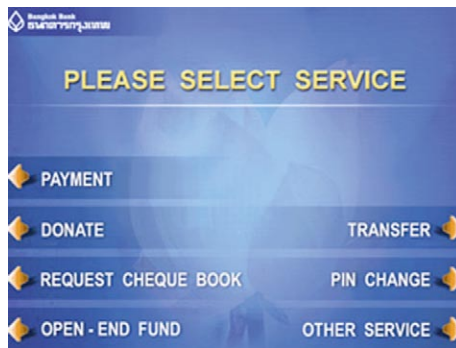
Mac OS lets users assign colors to files or folders; users can use the colors to track their work.

have not worked on. Mac OS lets users assign colors to files. Like moving files between folders, this technique can be used to keep track of where one is in a task (see Fig. 8.2).

## WE FOLLOW INFORMATION “SCENT” TOWARD OUR GOAL

Focusing our attention on our goals makes us interpret what we see on a display or hear in a telephone menu in a very *literal* way. People don't think deeply about instructions, command names, option labels, icons, navigation bar items, or any other aspect of the user interface of computer-based tools. If the goal in their head is to make a flight reservation, their attention will be attracted by anything displaying the words “buy,” “flight,” “ticket,” or “reservation.” Other items that a designer or marketer might think will attract customers, such as “bargain hotels,” will not attract the attention of people who are trying to book a flight, although they might be noticed by people who are looking for bargains.

This tendency of people to notice only things on a computer display that match their goal, and the literal thinking that they exhibit when performing a task on a



**FIGURE 8.3**

ATM screen. Our attention is drawn initially toward items that match our goal literally.

For each goal below, what on the screen would attract your attention?

- Pay a bill
- Transfer money to your savings account
- Pay your dentist by funds transfer
- Change your PIN
- Open a new account
- Purchase travelers' cheques

computer has been called “following the *scent* of information toward the goal” (Chi, Pirolli, Chen, & Pitkow, 2001; Nielsen, 2003). Consider the ATM machine display shown in [Figure 8.3](#). What is the first thing on the screen that gets your attention when you are given each of the goals listed?

You probably noticed that some of the listed goals direct your attention initially to the wrong option. Is “Pay your dentist by funds transfer” under “Payment” or “Transfer”? “Open a new account” probably sent your eyes briefly to “Open-End Fund,” even though it is actually under “Other Service.” Did the goal “Purchase traveler’s cheques” make you glance at “Request cheque book” because of the word they share?

The goal-seeking strategy of following information scent, observed across a wide variety of situations and systems, suggests that interactive systems should be designed so that the scent is strong and really does lead users to their goals. To do that, designers need to understand the goals that users are likely to have at each decision point in a task, and ensure that each choice point in the software provides options for every important user goal and clearly indicates which option leads to which goal.

For example, imagine that you want to cancel a reservation you made or a payment you scheduled. You tell the system to cancel it, and a confirmation dialog box appears asking if you really want to do that. How should the options be labeled? Given that people interpret words literally in following information scent toward their goal, the standard confirmation button labels “OK” (for yes) and “Cancel” (for no) would give a misleading scent. If we compare a cancellation confirmation dialog box from Marriott.com to one from Quicken.com, we see that Marriott.com’s labeling provides clearer scent than Quicken.com’s (see [Fig. 8.4](#)).

As a second example, imagine that you forgot that a certain document was already open, and you tried to open it again. The designers of Microsoft Excel did a better job than the designers of Microsoft Word did in anticipating this situation, understanding the goals you might have at this point, and presenting you with instructions and options that make it clear what to do (see [Fig. 8.5](#)).

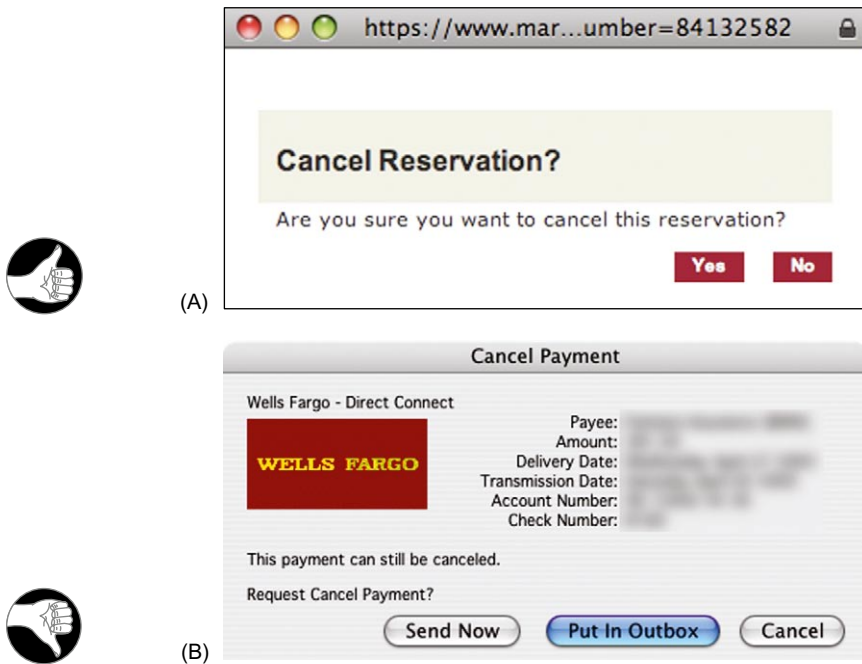


FIGURE 8.4

Marriott's cancellation confirmation (A) provides clearer scent than Quicken's (B).

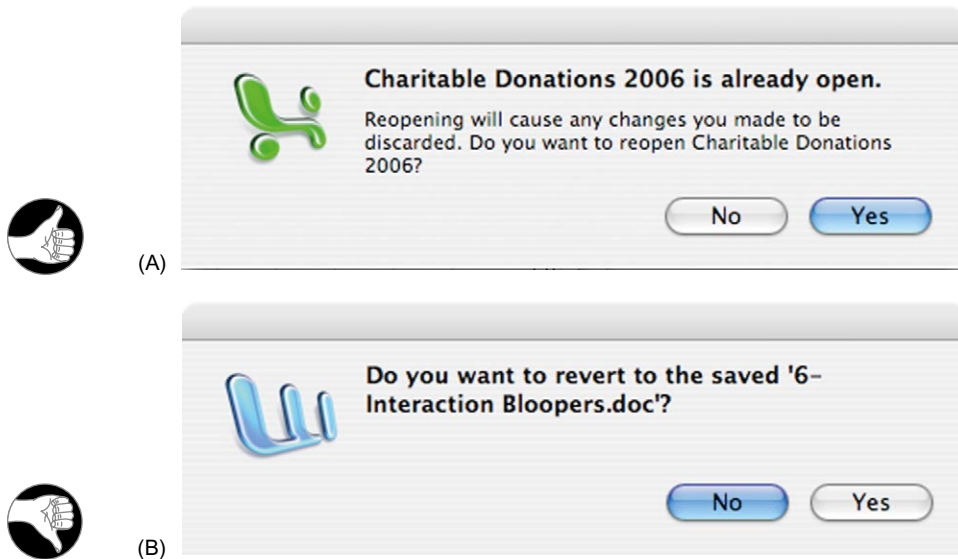


FIGURE 8.5

Microsoft Excel's warning (A) when users try to open an already open file is clearer than Word's (B).

---

## WE PREFER FAMILIAR PATHS

People know that their attention is limited, and they act accordingly. While pursuing a goal, they take *familiar* paths whenever possible rather than exploring *new* ones, especially when working under deadlines. As is explained more fully in Chapter 10, exploring new paths is problem solving, which places a heavy load on attention and short-term memory. In contrast, taking familiar, well-learned routes can be done fairly automatically and does not consume attention and short-term memory.

Years ago, in a usability test session, a test participant in the middle of a task said to me:

---

**I'm in a hurry, so I'll do it the long way.**

---

He knew there probably was a more efficient way to do what he was doing, but he also knew that learning the shorter way would require time and thought, which he was unwilling to spend.

Once we learn one way to perform a certain task using a software application, we may continue to do it that way and *never* discover a more efficient way. Even if we discover or are told that there is a “better” way, we may stick with the old way because it is familiar, comfortable, and, most important, requires little thought. Avoiding thought when using computers is important. People are willing to type *more* in order to think *less*.

This preference for familiar, relatively mindless paths has several design implications for interactive systems:

- ***Sometimes mindlessness trumps keystrokes.*** With software intended for casual use or infrequent use, such as bank ATM machines or household accounting applications, allowing users to become productive quickly and reducing their need to problem-solve while working is more important than saving keystrokes. Such software simply isn't used enough for keystrokes per task to matter much. On the other hand, in software that is used all day by highly trained users in intensive work environments, such as airline telephone reservation operators, every extra keystroke in a task adds high cumulative costs.
- ***Guide users to the best paths.*** From its first screen or home page, software should show users the way to their goals. This is basically the guideline that software should provide clear information scent.
- ***Help experienced users speed up.*** Make it easy for users to switch to faster paths after they have gained experience. The slower paths for newcomers should show users faster paths if there are any. This is why most applications show the keyboard accelerators for frequently used functions in the menu bar menus.

## OUR THOUGHT CYCLE: GOAL, EXECUTE, EVALUATE

Over many decades, scientists studying human behavior have found a cyclical pattern that seems to hold across a wide variety of activities:

- Form a **goal**, e.g., open a bank account, or eat a peach, or delete a word from a document
- Choose and **execute** actions to try to make progress toward the goal
- **Evaluate** whether the actions worked, i.e., whether the goal has been reached or is nearer than before
- Repeat until the goal is reached (or appears unreachable)

People cycle through this pattern constantly (Card, Moran, & Newell, 1983). In fact, we run through it at many different levels simultaneously. For example, we might be trying to insert a picture into a document, which is part of a larger task of writing a term paper, which is part of a higher-level task of passing a history course, which is part of a higher task of completing college, which is part of a higher-level goal of getting a good job, which we want in order to achieve our top-level goal of having a comfortable life.

As an example, let's run through the cycle for a typical computer task: buying an airline ticket online. The person first forms the primary goal of the task and then begins to break that down into actions that appear to lead toward the goal. Promising actions are selected for execution, executed, and then evaluated to determine if they have moved the person closer to the goal.

- **Goal:** Buy airline ticket to Berlin, using your favorite travel Web site.
- **Step 1:** Go to travel Web site. You are still far from the goal.
- **Step 2:** Search for suitable flights. This is a very normal, predictable step at travel Web sites.
- **Step 3:** Look at search results. Choose a flight from those listed. If no flights on the results list are suitable, return to Step 2 with new search criteria. You are not at the goal yet, but you feel confident of getting there.
- **Step 4:** Go to checkout. Now you are getting so close to your goal that you can almost smell it.
- **Step 5:** Confirm flight details. Check it—all correct? If no, back up; otherwise proceed. Almost done.
- **Step 6:** Purchase ticket with credit card. Check credit card information. Everything look OK?
- **Step 7:** Print e-ticket. Goal achieved.



In the airline ticket example, to keep the example short, we didn't get down into the details of each step. If we had, we would have seen substeps that followed the same *goal-execute-evaluate* cycle.

Let's try another example, this time examining the details of some of the high-level steps. This time the task is sending flowers to a friend. If we simply look at the top level, we see the task like this:

Send flowers to friend.

If we want to examine the *goal-execute-evaluate* cycle for this task, we must break this task down a bit. We must ask, *how* do we send flowers to a friend? To do that, we break the top-level task down into subtasks.

Send flowers to friend.

Find flower delivery Web site.

Order flowers to be delivered to friend.

For many purposes, the two steps we have identified are enough detail. After we execute each step, we evaluate whether we are closer to our goal. But *how* is each step executed? To see that, we have to treat each major step as a subgoal, and break it down into substeps.

Send flowers to friend.

Find flower delivery Web site.

Open Web browser.

Go to Google Web search page.

Type "flower delivery" into Google.

Scan the first page of search results.

Visit some of the listed links.

Choose a flower delivery service.

Order flowers to be delivered to friend.

Review service's flower selection.

Choose flowers.

Specify delivery address and date.

Pay for flowers and delivery.

After each substep is executed, we evaluate to see if it is getting us closer to the subgoal of which it is part. If we want to examine how a substep is executed and evaluated, we have to treat it as a sub-subgoal and break it into its component steps.

Send flowers to friend.

Find flower delivery Web site.

Open Web browser.

Click browser icon on taskbar, startup menu, or desktop.

Go to Google Web search page.

If Google isn't browser's starting page, choose Google from favorites list.

If Google is not on favorites list, type "Google.com" into browser's address box.

Type “flower delivery” into Google.  
 Set text-insertion point in search box.  
 Type the text.  
 Correct typo: “floowers” to “flowers”.  
 Visit some of the resulting links.  
 Move screen pointer to link.  
 Click on link.  
 Look at resulting Web page.  
 Choose a flower delivery service.  
 Enter chosen service’s URL into browser.

...

You get the idea. We could keep expanding, down to the level of individual key-strokes and individual mouse movements, but we rarely need that level of detail to be able to understand the task well enough to design software to fit its steps and the goal-execute-evaluate cycle that is applied to each step.

How can software support users in carrying out the goal-execute-evaluate cycle? Any of these ways:

- **Goal:** Provide clear paths—including initial steps—for the user goals that the software is intended to support.
- **Execute:** Software concepts (objects and actions) should be based on the task rather than the implementation (see Chapter 11). Don’t force users to figure out how the software’s objects and actions map to those of the task. Provide clear information about choice points to guide users to their goals. Don’t make them choose actions that seem to take them away from their goal in order to achieve it.
- **Evaluate:** Provide feedback and status information to show users their progress toward the goal. Allow users to back out of tasks that didn’t take them toward their goal.

An example of the “Evaluate” guideline—clear feedback about the user’s progress through a series of steps—is provided by ITN’s flight reservation system (see Fig. 8.6). By the way, does the figure seem familiar? If so, it is because you saw it in Chapter 5 (see Fig. 5.15B, page 63), and your brain recognized it.



FIGURE 8.6

ITN’s flight reservation system clearly shows users’ progress toward making a reservation.

## AFTER WE ACHIEVE A TASK'S PRIMARY GOAL, WE OFTEN FORGET CLEANUP STEPS

The *goal-execute-evaluate* cycle interacts strongly with short-term-memory. This interaction makes perfect sense: short-term memory is really just what the focus of attention is at any given moment. Part of the focus of attention is our current goal. The rest of our attentional resources are directed toward obtaining the information needed to achieve our current goal. The focus shifts as tasks are executed and the current goal shifts from high-level goals to lower-level ones, then back to the next high-level goal.

Attention is a very scarce resource. Our brain does not waste it by keeping it focused on anything that is no longer important. Therefore, when we complete a task, the attentional resources focused on that task's main goal are freed to be re-focused on other information that is now more important. The impression we get is that once we achieve a goal, everything related to it often immediately "falls out" of our short-term memory, i.e., we forget about it.

One result is that people often forget loose ends of tasks. For example, people often forget to do these things:

- Turn car headlights OFF after arrival, to prevent draining the battery
- Remove last pages of documents from copiers and scanners
- Turn stove burners and ovens OFF after use
- Add closing parentheses and quotation marks after typing text passages
- Turn OFF turn signals after completing turns
- Take books they were reading on a flight with them when they exit the plane
- Log out of public computers when finished using them
- Set devices and software back into normal mode after putting them into a special mode

These end-of-task short-term memory lapses are completely predictable and avoidable. When they happen to us, we call ourselves "absent-minded," but they are the result of how the brain works (or doesn't), combined with a lack of support from our devices.

To avoid such lapses, interactive systems can and should be designed to remind people that loose-end steps remain. In some cases, it may even be possible for the system to complete the task itself. For example:

- Cars already turn turn-signals OFF after a turn.
- Cars should (and now do) turn off headlights automatically when the car is no longer in use, or at least remind drivers that the lights are still ON.
- Copiers and scanners should automatically eject all documents when tasks are finished, or at least signal that a page has been left behind.
- Stoves should signal when a burner is left ON with no pot present for longer than some suitable interval, and ovens should do likewise when left ON with nothing in them.

- Computers should issue warnings if users try to power them down or put them to sleep before the computer has finished a background task, e.g., saving files or sending a document to a printer.
- Special software modes should revert to “normal” automatically, either by timing out—as some appliances do—or through the use of spring-loaded mode controls, which must be physically held in the non-normal state and revert to normal when released (Johnson, 1990).

Software designers should consider whether the tasks supported by a system they are designing have cleanup steps that users are likely to forget, and if so, they should design the system either to help users remember or to eliminate the need for users to remember.

# Recognition is Easy; Recall is Hard

# 9

Chapter 7 described the strengths and limitations of long-term memory and their implications for the design of interactive systems. This chapter extends that discussion by describing important differences between two functions of long-term memory: recognition and recall.

## RECOGNITION IS EASY

The human brain was “designed,” through millions of years of natural selection and evolution, to recognize things quickly. By contrast, recalling memories, i.e., retrieving them without perceptual support, must not have been as crucial for survival, because our brains are much worse at that.

Remember how our long-term memory works (see Chapter 7): perceptions enter through our sensory systems, and their signals, on reaching the brain, cause complex patterns of neural activity. The neural pattern resulting from a perception is determined not only by the features of the perception, but also by the context in which it occurs. Similar perceptions in similar contexts cause similar patterns of neural activity. Repeated activation of a particular neural pattern makes that pattern easier to reactivate in the future. Over time, connections between neural patterns develop in such a way that activating one pattern activates the other. Roughly speaking, each pattern of neural activity constitutes a different memory.

Patterns of neural activity, which is what memories are, can be activated in two different ways: (a) by more perceptions coming in from the senses, and (b) by other brain activity. If a perception comes in that is similar to an earlier one and the context is close enough, it easily stimulates a similar pattern of neural activity, resulting in a sense of *recognition*. Recognition is essentially perception and long-term memory working in concert.

As a result, we assess situations very quickly. Our distant ancestors on the East African savannah had only a second or two to decide whether an animal emerging

from the tall grasses was something they would regard as food or something that would regard *them* as food (see Fig. 9.1). Their survival depended on it.

Similarly, people recognize human faces very quickly—usually in a fraction of a second (see Fig. 9.2). Until recently, the workings of this process were considered a mystery. However, that was when scientists assumed that recognition was a process in which perceived faces were stored in a separate short-term memory



**FIGURE 9.1**

Early hominids had to recognize quickly whether animals they spotted were prey or predators.



**FIGURE 9.2**

How long did it take you to recognize these faces?<sup>1</sup>

---

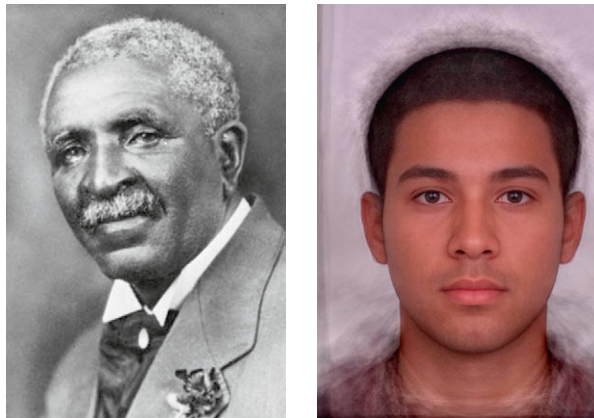
<sup>1</sup>Barack Obama and Bill Clinton.

and compared with those in long-term memory. Because of the speed with which the brain recognizes faces, cognitive scientists assumed that the brain must search many parts of long-term memory simultaneously, via what computer scientists call “parallel processing.” However, even a massively parallel search process could not account for the astounding rapidity of facial recognition.

Nowadays, perception and long-term memory are considered closely linked, which demystifies the speed of facial recognition somewhat. A perceived face stimulates activity in millions of neurons in distinct patterns. Individual neurons and groups of neurons that make up the pattern respond to specific features of the face and the context in which the face is perceived. Different faces stimulate different patterns of neural response. If a face was perceived previously, its corresponding neural pattern will already have been activated. The same face perceived again reactivates the same pattern of neural activity, only more easily than before. That *is* the recognition. There is no need to *search* long-term memory: the new perception reactivates the same pattern of neural activity, more or less, as the previous one. Reactivation of a pattern *is* the reactivation of the corresponding long-term memory.

In computer jargon, we could say that the information in human long-term memory is *addressed by its content*, but the word “addressed” wrongly suggests that each memory is located at a specific spot in the brain. In fact, each memory corresponds to a pattern of neural activity extending over a wide area of the brain.

That explains why, when presented with faces we have not seen before and asked if they are familiar, we don’t spend a long time searching through our memories to try to see if that face is stored in there somewhere (see [Fig. 9.3](#)). There is no search. A new face stimulates a pattern of neural activity that has not been activated



**FIGURE 9.3**

How long did it take you to realize that you do not recognize these faces?<sup>2</sup>

<sup>2</sup> George Washington Carver (American scientist, educator, and inventor) and average male face (FaceResearch.org).





**FIGURE 9.4**

We can recognize complex patterns quickly.

before, so no sense of recognition results. Of course, a new face may be so similar to a face we have seen that it triggers a *misrecognition*, or it may be just similar enough that the neural pattern it activates triggers a familiar pattern, causing a feeling that the new face *reminds* us of someone we know.

The same mechanisms that make our visual system amazingly fast at recognizing faces also make it fast at recognizing complex patterns. Anyone with at least a high-school education quickly and easily recognizes a map of Europe and a chessboard (see Fig. 9.4). Chess masters who have studied chess history may even recognize the chess position as Kasparov versus Karpov 1986.

## RECALL IS HARD

In contrast, *recall* is long-term memory reactivating old neural patterns without immediate similar perceptual input. That is much harder than reactivating a neural pattern with the same or similar perceptions. People *can* recall memories, so it obviously *is* possible for activity in other neural patterns or input from other areas of the brain to reactivate a pattern of neural activity corresponding to a memory. However, the coordination and timing required to recall a memory increase the likelihood that the wrong pattern or only a subset of the right pattern will be activated, resulting in a failure to recall.

Whatever the evolutionary reasons, our brain did not evolve to recall facts. Many schoolchildren dislike history class because it demands that they remember facts, such as the year the English Magna Carta was signed, the capital city of Argentina, and the names of all 50 U.S. states. Their dislike is not surprising; the human brain is not well suited for that sort of task.

Because people are bad at recall, they develop methods and technologies to help them remember facts and procedures (see Chapter 7). Orators in ancient Greece



used the “method of loci” to memorize the main points of long speeches. They imagined a large building or plaza and mentally placed their talking points in spots around it. When presenting the speech, they mentally “walked” through the site, picking up their talking points as they passed.

Today we rely more on external recall aids than on internal methods. Modern-day speakers remember their talking points by writing them down on paper or displaying them in overhead slides or presentation software. Businesses keep track of how much money they have, owe, or are owed by keeping account books. To remember contact information of friends and relatives, we use address books. To remember appointments, birthdays, anniversaries, and other events, we use calendars and alarm clocks. Electronic calendars are best for remembering appointments, because they actively remind us; we don’t have to remember to look at them.

## RECOGNITION VERSUS RECALL: IMPLICATIONS FOR UI DESIGN

The relative ease with which we can recognize things rather than recall them is the basis of the graphical user interface (GUI) (Johnson *et al.*, 1989). The GUI is based on two well-known user interface design rules:

- **See and choose is easier than recall and type.** Show users their options and let them choose among them, rather than force users to recall their options and tell the computer what they want. This rule is the reason GUIs have almost replaced command-line user interfaces (CLIs) in personal computers (see Fig. 9.5). “Recognition rather than recall” is one of Nielsen and Molich’s (1990) widely used heuristics for evaluating user interfaces. By contrast, using language to control a software application sometimes allows more expressiveness and efficiency than a GUI would. Thus, *recall and type* remains a useful

*Remember and type:*

> copy doc1 doc2

> remove olddoc

*See and choose*

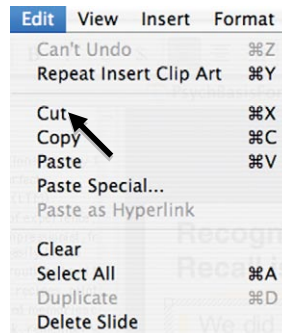


FIGURE 9.5

The main design rule behind today’s GUI: “See and choose is easier than remember and type.”

approach, especially in cases where users can easily recall what to type, such as when entering target keywords into a search box.

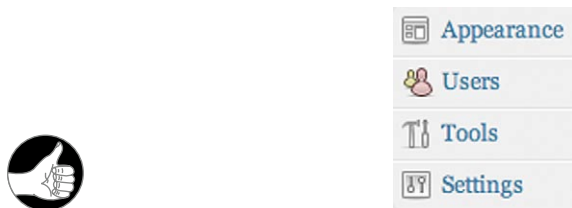
- **Use pictures where possible to convey function.** People recognize pictures very quickly, and recognizing a picture also stimulates the recall of associated information. For this reason, today's user interfaces often use pictures to convey function (see Figs. 9.6 and 9.7), such as desktop or toolbar icons, error symbols, and graphically depicted choices. Pictures that people recognize from the physical world are useful because they can be recognized without needing to be taught. This recognition is good as long as the familiar meaning matches the intended meaning in the computer system (Johnson, 1987). However, using familiar pictures from the physical world is not absolutely crucial. Computer users can learn to associate new icons and symbols with their intended meaning if these graphics are well designed. Memorable icons and symbols hint at their meaning, are distinguishable from others, and consistently mean the same thing, even across applications.

The GUI originated in the mid-1970s and became widespread decades ago—in the 1980s and 1990s. Since then, additional design rules have arisen that are based on human perception in general and on recognition and recall in particular. This chapter ends with a few of these newer rules.



**FIGURE 9.6**

Desktop icons convey function via recognition—by analogy with physical objects or by experience.



**FIGURE 9.7**

Wordpress.com uses symbols plus text to label functional pages on the Dashboard.

## Use thumbnail images to depict full-sized images compactly

Recognition is fairly insensitive to the size in which objects and events are displayed. After all, we have to be able to recognize things independently of their distance from us. What is important are features: as long as most of the same features are present in the new picture as were in the original one, the new perception stimulates the same neural pattern, resulting in recognition.

Therefore, a great way to display pictures people have *already* seen is to present them as small “thumbnail” images. The more familiar a picture, the smaller the thumbnails of it can be and still be recognizable. Displaying small thumbnails instead of full-sized images allows people to see more of their options, their data, their history, etc., at once.

Photo management and presentation applications use thumbnail images to give users an overview of their images or slides (see Fig. 9.8). Web browsers use thumbnails to show pages a user has recently visited (see Fig. 9.9).

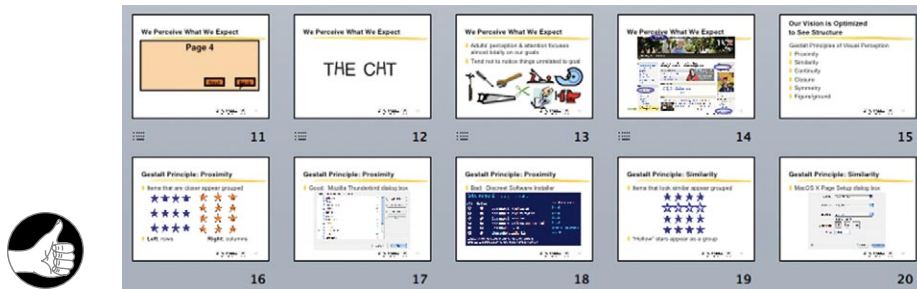


FIGURE 9.8

Microsoft PowerPoint can show slides as thumbnails, providing an overview based on recognition.

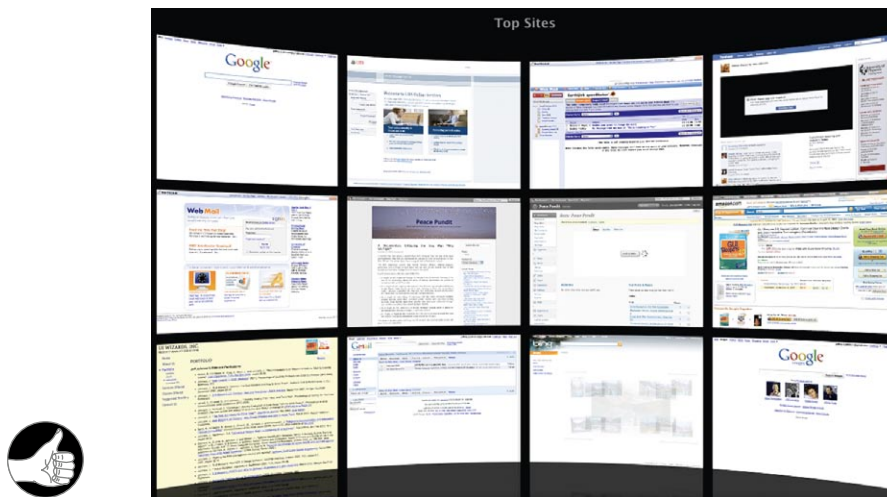


FIGURE 9.9

Apple Safari can show recently visited pages as thumbnail images, for quick recognition and choice.

### The larger the number of people who will use a function, the more visible the function should be

For the reasons described above, recall often fails. If a software application hides its functionality and requires its users to recall what to do, some percentage of users will fail. If the software has a lot of users, that percentage who fail to recall—even if it is small—adds up to a significant number. Software designers obviously don't want a significant number of users to fail in using the product.

The solution is to make functions that many people need highly visible, so users see and *recognize* their options rather than having to *recall* them. By contrast, functionality that few people will use—especially when those few people are highly trained—can be hidden, e.g., behind “Details” panels, in right-click menus, or via special key combinations.

### Use visual cues to let users recognize where they are

Visual recognition is fast and reliable, so designers can use visual cues to show users instantly where they are. For example, it is a well-known Web design rule that all pages in a Web site should have a common distinctive visual style so people can easily tell whether they are still on the site or have gone to a different one. Slight but systematic variations on a site's visual style can show users which section of the site they are in.

Some desktop operating systems allow users to set up multiple desktops (“rooms” or “workspaces”) as locations for different categories of work. Each has its own background graphic to allow easy recognition.

Some corporate Web sites use pictures to assure users that they are on a secure site. Users choose a picture as a personal account logo, and the site displays the logo whenever it recognizes the user from cookies or after the user has entered a valid login name but not yet a password (see Fig. 9.10). This lets users know they are at the real company site and not a fake site hosted by someone running a phishing scam.

### Make authentication information easy to recall

People know that it is hard to recall arbitrary facts, words, and sequences of letters or digits. That is why they often write passwords and challenge question answers down and keep the information in places that are easy to reach and thus insecure. Or they base passwords on their children's initials, their birthdates, their street address, and other information they know they can recall. Unfortunately, such passwords are too often easy for other people to guess (Schrage, 2005). How can designers help users avoid such unsafe behavior?

For starters, we can at least not make it *hard* for people to recall their login information, like the systems cited in Chapter 7 that impose burdensome password restrictions or offer a limited choice of challenge questions.

Instead, we can give users the freedom to create passwords they can remember and challenge questions for which they can remember the correct response. We can


Bank of America Online Banking  
En Español

**Confirm that your SiteKey is correct**

If you recognize your SiteKey image, you'll know for sure that you are at the valid Bank of America site. Confirming your SiteKey image is also how you'll know that it's safe to enter your Passcode.

\* = required

**Your SiteKey:**



If you don't recognize your personalized SiteKey, don't enter your Passcode.

\* **Passcode:**

(8 - 20 Characters, case sensitive)

[Forgot your SiteKey?](#)  
[Incorrect SiteKey showing?](#)  
[Reset passcode](#)



**FIGURE 9.10**

BankOfAmerica.com shows recognized customers their self-selected account logo (SiteKey), to assure them that it is the real bank's site.

also let users supply password *hints* that the system can present to them, under the assumption that users can devise hints that will serve as a recall probe for them but not identify the password to third parties.

Authentication methods that do not rely on users to recall the authentication data would seem to be a solution. Biometric authentication methods such as iris scans, digital fingerprint scans, and voice identification fall into this category. However, many people regard such methods as privacy threats because they require the collection and storage of individuals' biometric data, creating the potential for information leaks and abuse. Therefore, while biometric authentication does not burden users' memory, it would have to be implemented in a way that meets stringent privacy requirements in order to be widely accepted.

# Learning from Experience and Performing Learned Actions are Easy; Problem Solving and Calculation are Hard

# 10

As we saw in the previous chapter's comparison of recognition and recall, the human brain is good at some things and not so good at others. In this chapter, we compare several additional functions of the brain to show which functions it is good and bad at, and to see how to design computer systems accordingly. But first, a bit more about the brain and the mind.

## WE HAVE THREE BRAINS

We really have three brains, or if you prefer, a brain with three main parts, each of which affects different aspects of our thought and behavior (Weinschenk, 2009):

- **The old brain:** This is mainly the brain stem, where the spinal cord enters the base of the brain. It has been around since the first fish evolved. (Insects and mollusks, which appeared before fish, don't have brains in the usual sense of the word.) The old brain classifies everything into three categories: edible, dangerous, or sexy. It also regulates the body's automatic functions, such as digestion, breathing, and reflexive movement. Reptiles, amphibians, and most fish have only the old brain.
- **The midbrain:** This part of the brain is "middle" in two senses: (a) physically, because it is located above the old brain and beneath the cortex, and (b) evolutionarily, because it evolved after the old brain and before the new brain. The midbrain controls emotions; it reacts to things with joy, sadness, fear, aggressiveness, apprehensiveness, anger, etc. Birds<sup>1</sup> and lower mammals have only an old brain and a midbrain.

<sup>1</sup>Corvids (ravens, crows, and magpies) and some types of parrots (e.g., New Zealand kīa) have no cortex, but they do have large brains compared to other birds. They often exhibit intelligence rivaling elephants, porpoises, and monkeys. In these birds, other parts of their brains apparently serve functions that the cortex serves in mammals.

- **The new brain:** This part of the brain mainly consists of the cerebral cortex. It controls intentional, purposeful, conscious activity, including planning. Most mammals have a cortex in addition to their old brain and midbrain, but only a few highly evolved mammals—elephants; porpoises, dolphins, and whales; and monkeys, apes, and humans—have a sizable one.

The human mind is not fully rational and conscious—some experts claim that it isn't even *mostly* rational and conscious. Our thoughts and behavior are affected at least as much by the midbrain and old brain as they are by the new brain. When we perceive something—an object or an event—all three “brains” react and contribute to our thought and behavior. In fact, the old and midbrains tend to react faster than the new brain does, so we sometimes act based on what they tell us before our cortex reaches a decision or even knows that action is required.

---

## LEARNING FROM EXPERIENCE IS (USUALLY) EASY

People are pretty good at generalizing from specific experiences and observations to extract conclusions. We generalize constantly throughout our lives.

The neural basis of behavioral learning is not as well understood as that of recognition and recall (Liang *et al.*, 2007). However, people learn from their experiences constantly, often without awareness that they are doing it. From this fact we can postulate that the human brain evolved the ability to learn quickly and easily from experience because there were evolutionary advantages to being able to do so. Thus, most people, if given the necessary experience, easily learn such lessons as these:

- Stay away from leopards
- Don't eat bad-smelling food
- Ice cream tastes good, but it melts quickly in hot weather
- Wait a day before replying to an email that makes you mad
- Don't open attachments from unfamiliar senders
- LinkedIn is useful, but Facebook is a waste of time (or vice versa, depending on your preference)

However, our ability to learn from experience is not perfect for several reasons. First, complex situations that involve many variables or that are subject to a wide variety of forces are difficult for people to predict, learn from, and generalize about. For example:

- Experienced stock market investors still aren't sure what stocks to sell or buy now.
- People who have lived in Denver for years still have trouble predicting the weather there.
- Even after interacting with your sister's boyfriend on several occasions, you may still not be sure he is a good guy.

Second, experiences from our own lives or those of relatives and friends influence our conclusions more than experiences we read or hear about. For example, we may have read and seen reports, consumer reviews, and statistics indicating



that the Toyota Prius is a great car, but if our sister or our uncle had a bad experience with one, we will probably have a negative assessment of the car. We do this because our midbrain considers family members to be like us and therefore more trustworthy than data about thousands of anonymous car buyers, even though from a rational standpoint the statistics are more reliable (Weinschenk, 2009).

Third, when people make a mistake, they don't always learn the right lesson from it. By the time they realize they are in a bad situation, they may not remember their recent actions well enough to be able to connect their situation with the true cause or causes.

A fourth problem people have in learning from experience is that they often *overgeneralize*, i.e., make generalizations based on incomplete data. For example, many people assume all crows are black because all the crows they have seen are black. In fact, there are crows that are not black (see Fig. 10.1).

However, it can be argued that overgeneralizing isn't a problem—it's a *feature*. It is rare that one can see all possible examples of something. For example, a person can never see all crows, but it may still be useful in daily life (although not in scientific research) to assume that the many crows one has seen are enough evidence to conclude that all crows are black. Overgeneralization therefore seems like a necessary adaptation for life in the real world. It is primarily when we overgeneralize in extreme ways—e.g., making generalizations on the basis of *one* example or *atypical* examples—that we get ourselves into trouble.

The ability to learn from experience has a long evolutionary history. A creature does not need a cerebral cortex (new brain) to be able to do it. Both the old brain and midbrain can learn from experience. Even insects, mollusks, and worms, without even an old brain—just a few neuron clusters—can learn from experience. However, only creatures with a cortex or brain structures serving similar functions<sup>2</sup> can learn



**FIGURE 10.1**

The common belief that all crows are black is false. Left: African pied crow (Photograph by Thomas Schoch). Right: white (nonalbino) crow, Ohio.

<sup>2</sup>The reason for the caveat is that some birds can learn from watching other birds.



from the experiences of *others*. A cortex is certainly necessary to be *aware* that one has learned from experience, and only creatures with the largest new brains (relative to body size)—possibly only humans—can *articulate* what they have learned from experience.

Even though there are limits on how well we learn from direct experience and from the experience of others, the bottom line is that learning and generalizing from experience are relatively easy for the human mind.

---

## PERFORMING LEARNED ACTIONS IS EASY

When we go somewhere we have been many times before, or do something we have done many times before, we do it almost automatically, without much conscious thought. The route, the routine, the recipe, the procedure, the action, has become semiautomatic or fully automatic. Here are some examples:

- Riding a bicycle after many years of practice
- Backing out of your driveway and driving to work for the 300th time
- Brushing your teeth as an adult
- Playing a tune that you have played hundreds of times on a musical instrument
- Using a mouse or a touch pad to move a cursor on a computer display after a few days of practice
- Entering a banking transaction into your old familiar bank account software
- Reading and then deleting a text message from your longtime mobile phone

In fact, “automatic” is how cognitive psychologists refer to routine, well-learned behavior (Schneider & Shiffrin, 1977). Researchers have determined that performing this type of action consumes few or no conscious cognitive resources, i.e., it is not subject to the limits of attention and short-term memory described in Chapter 7.

Automatic activities can even be done in parallel with other activities. Thus, you can tap your foot while humming a familiar song while beating an egg, while still leaving your mind “free” to keep an eye on your children or plan your upcoming vacation.

How does an activity become automatic? The same way you get to Carnegie Hall (as the old joke goes): practice, practice, practice.

When a person first tries to drive a car—especially a car with a stick shift—every part of the activity requires conscious attention. Am I in the right gear? Which foot do I use to press the accelerator pedal, the brake pedal, and the clutch pedal? How hard should I press on each of these pedals? How hard am I pressing on the clutch pedal now? Which way am I headed? How fast am I going? What is ahead of me, behind me, beside me? Where are the mirrors I should be checking? Is that my street coming up ahead? “Objects in mirror are closer than they appear”—what does that mean? And what is that light blinking on the dashboard?

When everything involved in driving a car is still conscious, keeping track of it all *far* exceeds our attention capacity—remember that it is four items, plus or minus two (see Chapter 7). People who are still learning to drive often feel overwhelmed. That is

why they often practice driving in parking lots, parks, rural areas, and quiet neighborhoods, where traffic is light: to reduce the number of things they have to attend to.

After a lot of practice, all the actions involved in driving a car become automatic. They no longer compete for attention and they recede from consciousness. We may not even be fully aware of doing them. For example, which foot do you use to push the accelerator pedal? To remember, you probably had to pump your feet briefly.

Similarly, when music teachers teach students to play a musical instrument, they don't make students monitor and control every aspect of their playing at once. That would overwhelm the students' attention capacity. Instead, teachers focus students' attention narrowly on one or two aspects of their playing: the correct notes, the rhythm, the tone, the articulation, or the tempo. Only after students learn to control some aspects of their playing without thinking about them do music teachers require their students to control more aspects simultaneously.

To demonstrate to yourself the difference in conscious attention required by well-learned (automatic) versus novel (controlled) tasks, try these:

- Recite the letters of the alphabet from A to M. Then recite the letters of the alphabet from M to A.
- Count down from 10 to 0—think of a rocket launch. Then count down from 21 to 1 by odd numbers.
- Drive to work, using your normal route. The next day, use a very different, unfamiliar route.
- Throw a ball with your usual ball-throwing hand. Then throw one using the opposite hand.
- Enter your phone number using a standard 12-key telephone pad. Then enter your phone number using the number keys at the top of your computer keyboard.
- Type your full name on a computer keyboard. Then cross your hands on the keyboard and type your full name again. (I was going to suggest riding a bicycle with your hands crossed, but that is actually dangerous, so I do *not* recommend trying it.)

Most real-world tasks have a mixture of automatic and controlled components. Driving to work along your usual route is mostly automatic, allowing you to focus on the radio news or think about your evening dinner plans. But if another vehicle near you does something unexpected or a child appears on the road ahead of you, your attention will be yanked back to the task of driving.

Similarly, if you check your email using your usual email program, the way you retrieve and view your email is well practiced and mostly automatic, and reading text is well practiced and automatic, but the content of any newly arrived email messages is new and therefore requires your conscious attention. If while on vacation you go into an Internet cafe and try to check your email using an unfamiliar computer, operating system, or email program, less of the task will be automatic, so it will require more conscious thought, take more time, and be more prone to error.

When people want to get something done—as opposed to challenging themselves mentally—they prefer to use methods that are automatic or at least semi-automatic in order to save time and mental effort, and to reduce the chance of error. If you are in a hurry to pick up your child from school, you take your tried-and-true route, even if your neighbor just told you yesterday about a faster route. Remember what the usability test subject said (previously mentioned in Chapter 8):

---

**I'm in a hurry, so I'll do it the long way.**

---

How can designers of interactive systems make the tasks that they support faster, easier, and less error prone? By designing them to become automatic quickly. How does one do that? Chapter 11 describes some of the ways.

---

## PROBLEM SOLVING AND CALCULATION ARE HARD

Reptiles, amphibians, and most birds get along in their world quite well with just an old brain and a midbrain.<sup>3</sup> Insects, spiders, and mollusks survive in their environments with even less. Animals without a cortex (or its equivalent, as in a few birds) can learn from experience, but it usually takes a lot of experience and they can only learn minor adjustments to their behavior. Most of their behavior is stereotyped, repetitive, and predictable once we understand the demands of their environment (Simon, 1969). That may be just fine when their environment requires only the behaviors they already have automated.

But what if the environment throws a curve ball: it requires new behavior, and requires it *right now*? What if a creature faces a situation it has never encountered before, and may never encounter again? In short, what if it is faced with a *problem*? In such cases, creatures with no cortex or its equivalent cannot cope.

Having a cerebral cortex (new brain) frees creatures from relying solely on instinctive, reactive, automatic, well-practiced behaviors. The cortex is where conscious reasoning happens (Monti, Osherson, Martinez, & Parsons, 2007). Generally speaking, the larger a creature's cerebral cortex relative to the rest of its brain, the greater its ability to interpret and analyze situations on-the-fly, plan or find strategies and procedures to cope with those situations, execute those strategies and procedures, and monitor their progress.

Expressed in computer jargon, having a large cortex gives us the ability to devise programs for ourselves on the fly and run them in an emulated, highly monitored mode rather than a compiled or native mode. That is essentially what we are doing when we are following a cooking recipe, playing bridge, calculating income taxes, following instructions in a software manual, or figuring out why no sound is coming out of the computer when we play a video.

---

<sup>3</sup>For example, salamanders choose a jar containing four fruit flies over one with two or three fruit flies (Sohn, 2003).

## THE NEW BRAIN ALSO ACTS AS A BRAKE ON IMPULSIVE BEHAVIOR

The new brain—specifically the frontal cortex—also acts to inhibit reflexive and impulsive behavior coming from the midbrain and old brain that could interfere with the execution of the new brain’s carefully worked-out plans (Sapolsky, 2002). It keeps us from jumping up and getting off of a subway car when a smelly person boards, because after all, we do have to get to work on time. It keeps us sitting quietly in our seats in classical music concerts, but lets us stand up and hoot and holler in rock concerts. It helps keep us out of fights (usually). It tries to stop us from buying that red sports car because preserving our marriage is a higher goal than having the car. And whereas the old and midbrains are tempted by the email that proposes a BUSINESS OPPORTUNITY WORTH \$12.5 MILLION, the new brain stops us from clicking, saying “It’s a spammer and a scammer; you know that, don’t you?”

Although having a large new brain gives us the flexibility to deal with problems on short notice, that flexibility has a price. Learning from experience and performing well-learned actions are easy largely because they don’t require constant awareness or focused attention and because they can occur in parallel. In contrast, controlled processing—including problem solving and calculation—requires focused attention and constant conscious monitoring, and executes relatively slowly and serially (Schneider & Shiffrin, 1977). It strains the limits of our short-term memory because all the chunks of information needed to execute a given procedure compete with each other for scarce attention resources. It requires conscious mental effort, as you saw when you tried to recite the alphabet backward from M to A.

In computer jargon, the human mind has only one serial processor for emulation mode, controlled execution of processes. That processor is severely limited in its temporary storage capacity and its clock is an order of magnitude slower than that of the brain’s highly parallelized and compiled automatic processing.

Modern humans evolved from earlier hominids between 200,000 and 50,000 years ago, but numbers and numerical calculation did not exist until about 3400 BC, when people in Mesopotamia (modern-day Iraq) invented and started using a number system in commerce. By then, the human brain was more or less as it is today. Since the modern human brain evolved before numerical calculation existed, it is not optimized for calculation.

Calculation is done mainly in the brain's controlled, monitored mode. It is a task that consumes scarce resources of attention and short-term memory, so when we try to perform calculations entirely in our heads, we have trouble. The exception is that some steps in a calculation may be memorized and therefore are automatic. For example, the overall process of multiplying  $479 \times 832$  is controlled, but certain substeps of the process may be automatic if we have memorized the multiplication tables for single-digit numbers.

Problems and calculations that involve only one or two steps, or in which some steps are memorized (automatic), or that don't involve much information, or in which all the relevant information is immediately available—and therefore need not be kept in short-term memory—are easy for most people to work out in their heads. For example:

- $9 \times 10 = ?$
- I need to move the washing machine out of the garage, but the car is in the way, and my car keys are in my pocket. What to do?
- My girlfriend has two brothers, Bob and Fred. I have met Fred, and the one here now isn't Fred, so it must be Bob.

However, problems that exceed our short-term memory limits, or that require that certain information be retrieved from long-term memory, or in which we encounter distractions, strain our brains. For example:

- I need to move the washing machine out of the garage, but the car is in the way, and my car keys are ... hmmm ... they're not in my pocket. Where are they? ... [Search car.] They're not in the car. Maybe I left them in my jacket. ... Now where did I leave my jacket? [Search house; eventually find jacket in bedroom.] OK, found the keys. ... Boy is this bedroom messy—must clean it before wife gets home. ... Hmmm. Why did I need the car keys? [Return to garage, see washer.] Oh, yeah: to move the car so I can move the washing machine out of the garage. (*Higher-level goal was pushed out of short-term memory by interim subgoals.*)
- Chapter 8 gave examples of tasks in which people have to remember to complete cleanup steps after achieving their primary goal, for example, remembering to turn your car headlights OFF after arrival at your destination or to remove the last page of a document from a copier after you have the copy.
- John's cat is not black and likes milk. Sue's cat is not brown and doesn't like milk. Sam's cat is not white and doesn't like milk. Mary's cat is not yellow and likes milk. Someone found a cat that is yellow and likes milk. Whose cat is it?<sup>4</sup> (*The negations create more chunks of information than most people's short-term memory can hold at once.*)

---

<sup>4</sup>Answers provided at the end of this chapter.

- A man built a four-sided house. All four walls faced south. A bear walked by. What color was the bear? (*Requires deduction and knowing and retrieving specific facts about the world and its wildlife.*)
- You have to measure exactly four liters of water, but you only have a three-liter bottle and a five-liter bottle. How do you do it? (*Requires mentally simulating a series of pours until the right series is found, straining short-term memory and perhaps exceeding mental simulation abilities.*)

When solving such problems, people often use external memory aids, such as writing down interim results, sketching diagrams, and manipulating models of the problem. Such tools augment our limited short-term memory and our limited ability to imagine manipulating problem elements.

Problem solving and calculation are also difficult if they require a cognitive strategy, solution method, or procedure that we don't know and cannot devise or find. For example:

- $93.3 \times 102.1 = ?$  (*Requires arithmetic that exceeds short-term memory capacity, so must be done with a calculator or on paper. The latter requires knowing how to multiply multidigit decimal numbers on paper.*)
- A farmer has cows and chickens—30 animals total. The animals have a total of 74 legs. How many of each animal does the farmer have? (*Requires translation to two equations and then solving using algebra.*)
- A Zen master blindfolded three of his students. He told them that he would paint either a red dot or a blue dot on each one's forehead. In fact, he painted red dots on all three foreheads. Then he said "In a minute I will remove your blindfolds. When I do, look at each other and if you see at least one red dot, raise your hand. Then guess which color your own dot is." Then he removed the blindfolds. The three students looked at each other, then all three raised a hand. After a minute, one of the students said "My dot is red." How did she know? (*Requires reasoning by contradiction, a specialized method taught in logic and mathematics.*)
- You play a YouTube video on your computer, but there is no sound even though you can see people speaking. Is the problem in the video, the video player, your computer, your speaker cables, or your speakers? (*Requires devising and executing a series of diagnostic tests that successively narrow the possible causes of the problem, which requires computer and electronics domain knowledge.*)

These made-up examples demonstrate that certain problems and calculations require training that many people do not have. The sidebar gives real examples of people being unable to resolve technical problems because they lack training in effective diagnosis in the technical problem domain and are not interested in learning how to do it.

## SOLVING TECHNICAL PROBLEMS REQUIRES TECHNICAL INTEREST AND TRAINING

Software engineers are trained to do systematic diagnosis of problems. It is part of their job to know how to devise and execute a series of tests to eliminate possible causes of a fault until they find the cause. Engineers often design technology-based products as if the intended users were as skilled as engineers in diagnosing technical problems. However, most people who are not software engineers have not been trained in that sort of problem diagnosis, and therefore cannot do it effectively. Here are real examples of non-technical people facing problems they could not solve without help:

- Ann wanted to book a flight, but couldn't because the airline Web site wouldn't let her. It demanded a password but she didn't have one. She called a computer-engineer friend, who asked several questions to learn her situation. It turned out that the Web site assumed that she was her husband, because he had previously bought tickets from that airline on that computer. The site wanted his username and password. She didn't know his password and he was out of town. The engineer told her to log out of the Web site, then return as a *new* customer and create her own account.
- At a church, one of two stage monitor speakers stopped working. The Assistant Music Director assumed that the monitor had failed and said he would replace it. A musician who also is an engineer wasn't sure the monitor was bad, so he swapped the two monitor cables at the speaker end. Now the "bad" speaker worked and the "good" one didn't, showing that the problem was *not* a bad speaker. The Assistant Music Director concluded that one speaker cable was bad and said he would buy a new one. Before he did, the engineer-musician swapped the monitor cables where they connect to the monitor amplifier, to see if the problem was a faulty monitor

amplifier output rather than the cable. The problem turned out to be a loose connection in the monitor amplifier output jack.

Even when people know they *could* solve a problem or perform a calculation if they put effort into it, sometimes they don't do it because they don't consider the potential reward worth the effort. This response is especially common when solving a problem is not required by one's job or otherwise. Here are some real examples:

- A posting on San Francisco Freecycle Network: “Free: Epson Stylus C86. Was working fine, and then suddenly it couldn't recognize the new full ink cartridge. Not sure if it's the cartridge or the printer. So I bought a new printer and am giving the old one away.”
- Fred and Alice, a schoolteacher and a nurse who are married, never install or update software on their home computer. They don't know how, and they don't want to know. They use only the software that came with the computer. If their computer says updates are available, they ignore it. If an application—e.g., a Web browser—stops working because it is outdated, they stop using it. When necessary, they buy a new computer.
- Another couple, Ted and Sue, have a television, a videotape player, and a DVD player. Remote controls for the devices lie in a pile near the TV, unused. Ted and Sue control the devices by getting up and walking across the room. They say it's too much trouble to learn to work the remotes and remember which one is for which device. Yet they use computers daily, for email and Web.

The people in these examples are not stupid. Many have college degrees, putting them in the top 30% of educational attainment in the United States. Some are even trained to diagnose problems in different domains, such as medicine. They just have no training or interest in solving technical problems in computers and computer-based devices.



People invented calculators and computers mainly as tools for performing calculations and solving problems that humans cannot easily solve on their own. Computers and calculators do calculation and problem solving much more easily and reliably than we do, at least when the problems are well defined.

## IMPLICATIONS FOR USER INTERFACE DESIGN

People often intentionally challenge and entertain themselves by creating or solving puzzles that strain—or “exercise”—their minds (see Fig. 10.2). However, that fact does not imply that people will happily accept mind-straining problems foisted upon them by someone or something else. People have their own goals. They are using a computer to help them achieve a goal. They want—and need—to focus their attention on that goal. Interactive systems—and designers of them—should respect that and not distract users by imposing technical problems and goals that users don’t want.

Here are some examples of technical problems that computers and Web services impose upon their users:

- “It wants my ‘member ID.’ Is that the same as my ‘username’? It must be.”
- “Huh? It charged me the full price! It didn’t give me my discount. What now?”
- “It says that the software may be incompatible with a plug-in already on my computer. ‘May be’? Is it or isn’t it? And if it is, which plug-in is the culprit? What should I do?”
- “I want page numbers in the chapter to start at 23 instead of 1, but I don’t see a command to do that. I’ve tried *Page Setup*, *Document Layout*, and *View Header and Footer*, but it isn’t there. All that’s left is this *Insert Page Numbers* command. But I don’t want to *insert* page numbers: the chapter already has page numbers. I just want to change the starting number.”
- “Hmmm. This checkbox is labeled *Align icons horizontally*. I wonder what happens if I uncheck it. Will my icons be aligned vertically, or will they simply not be aligned?”

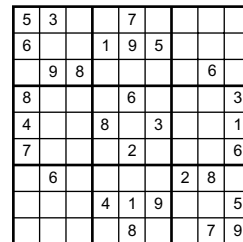
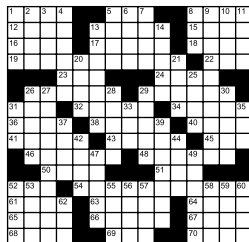


FIGURE 10.2

We challenge ourselves by creating and solving puzzles that tax our mental abilities.

Interactive systems should minimize the amount of attention users must devote to operating them (Krug, 2005), because that pulls precious cognitive resources away from the task a user came to the computer to do. Here are some design rules:

- **Prominently indicate system status and users' progress toward their goal.** If users can always check their status easily by direct perception, using the system will not strain their attention and short-term memory.
- **Guide users toward their goals.** Designers can do this implicitly, by making sure every choice-point provides clear information “scent” that leads users toward their goal, or explicitly, by using a wizard (multistep dialog box). Don't just display a bunch of options that appear equally likely and expect users to know how to start and how to get to their goal, especially if they won't perform the task very often.
- **Tell users explicitly and exactly what they need to know.** Don't expect them to deduce information. Don't require them to figure things out by a process of elimination.
- **Don't make users diagnose system problems,** such as a faulty network connection. Such diagnosis requires technical training, which most users don't have.
- **Minimize the number and complexity of settings.** Don't expect people to optimize combinations of many interacting settings or parameters. People are *really* bad at that.
- **Let people use perception rather than calculation.** Some problems that might seem to require calculation can be represented graphically, allowing people to achieve their goals with quick perceptual estimates instead of calculation. A simple example: suppose you want to go to the middle of a document. Document editing software of the 1970s and early 1980s forced you to look at the document's length, divide that in half, and issue a command to go to the middle page number. With modern-day document editing software, you just drag the scrollbar “elevator” to the middle of the bar, and you are there. Similarly, snap-to grids and alignment guides in drawing tools eliminate the need for users to determine, match, and compute coordinates of existing graphic elements when adding new ones.
- **Make the system familiar.** Use concepts, terminology, and graphics that users already know to make the system as familiar to them as possible, requiring them to think about it less. Designers can use this approach to a certain extent even if the system provides functionality that users have not seen before. One way to do it is to follow industry conventions and standards (e.g., Apple Computer, 2009; Microsoft Corporation, 2009). A second way is to make new software applications work like older ones that users have used before. A third approach is to base the design on metaphors, such as the desktop metaphor (Johnson *et al.*, 1989). Finally, designers can study users to learn what is and is not familiar to them.
- **Let the computer do the math.** Don't make people calculate things the computer can calculate itself (see Fig. 10.3).



33. List the names of **all of the employers** you worked for in the last 18 months, the dates you worked for each employer, the wages you earned from each, and how you were paid. Please also indicate the employer you worked for longest by selecting the radio button next to that employer. [Help](#)

	Employer Name <a href="#">Help</a>	From Date (mm/dd/yyyy)	To Date (mm/dd/yyyy)	Earnings	How Paid
<input type="radio"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="radio"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="radio"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="radio"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="radio"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="radio"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="radio"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="radio"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

34. Regarding the employer in question 33 that you indicated you worked for the longest, please answer the following:  
 34a. How long did you work for that employer?      Years  Months

FIGURE 10.3

California online unemployment form asks for data it could calculate itself in both of these questions.

## ANSWERS TO PUZZLES ON PAGES 124 AND 125

- The cat is John's.
- The bear was white, because to have four south-facing walls, the house must be on the North Pole.
- To end up with four liters of water, fill the three-liter bottle and pour it into the five-liter bottle, then fill the three-liter bottle again and pour as much as will fit from it into the five-liter bottle. That leaves one liter in the three-liter bottle. Empty the five-liter bottle, and pour the one liter from the three-liter bottle into the five-liter bottle. Then fill the three-liter bottle again and pour it into the five-liter bottle.
- Let  $A$  = the number of cows, and  $B$  = the number of chickens. "A farmer has cows and chickens—30 animals total" translates to " $A + B = 30$ ." "The animals have a total of 74 legs" translates to " $4A + 2B = 74$ ." Solving for  $A$  and  $B$  gives:  $A = 7$  and  $B = 23$ , so the farmer has 7 cows and 23 chickens.
- The Zen student saw three hands up and red dots on both other students. From this information, she didn't know whether her dot was red or blue. She started out assuming it was blue, and waited. She reasoned that the other students would see her (assumed) blue dot and one other red dot, realize that two red dots were required for all three hands to be up, and quickly figure out that their own dot had to be red. But after a minute neither of the other students had said anything, which told the Zen student that the other students couldn't figure out what color their dot was, which meant that her own dot was *not* blue; it had to be red.

# Many Factors Affect Learning

# 11

Chapter 10 contrasted the “automatic” processes that our brain uses to carry out well-learned activities with the conscious, highly monitored, controlled processes that we use to solve novel problems and perform calculations. Automatic processes consume little or no short-term memory (attention) resources and can operate in parallel with each other, while controlled processes place high demands on short-term memory and operate one at a time (Schneider & Shiffrin, 1977).

The first time or even the first several times we perform an activity, we do it in a highly controlled and conscious manner, but with practice it becomes more and more automatic. Examples include peeling an apple, driving a car, juggling balls, riding a bicycle, reading, playing a musical instrument. Even an activity that might seem to require our attention, such as sorting good cherries from bad ones, can become automated to the point that we can do it as a background task, with plenty of cognitive resources left over for having a conversation, watching the news on television, etc.

This progression from controlled to automatic raises an obvious question for designers of interactive applications, online services, and electronic appliances: How can we design them so that using them becomes automatic within a reasonable amount of time?

This chapter explains and demonstrates factors that affect how quickly people can learn to use interactive systems. To preview the factors, We learn faster under the following conditions:

- Operation is task-focused, simple, and consistent
- Vocabulary is task-focused, familiar, and consistent
- Risk is low

## **WE LEARN FASTER WHEN OPERATION IS TASK-FOCUSED, SIMPLE, AND CONSISTENT**

When we use a tool—whether it is computer-based or not—to do a task, we have to translate what we want to do into the operations provided by the tool. Some examples:

- Imagine that you are an astronomer. You want to point your telescope at the star Alpha Centauri. Most telescopes don't let you just specify what star you want to observe. Instead, you have to translate that goal into how the telescope's positioning controls operate: in terms of a vertical angle (azimuth) and a horizontal angle, or perhaps even the *difference* between where the telescope is pointing now and where you *want* it to point.
- Assume you have a telephone that doesn't have speed dial. To call a person, you have to translate the *person* into a *telephone number* and give *that* to the phone.
- You want to create an organization chart for your company, using a generic drawing program. To indicate organizations, suborganizations, and their managers, you have to draw boxes, label them with organization and manager names, and connect them with lines.

Cognitive psychologists call the gap between what a tool user wants and the operations the tool provides “the gulf of execution” (Norman & Draper, 1986). A person using a tool must expend cognitive effort to translate what she wants into the tool's available operations and vice versa. That cognitive effort pulls the person's attention away from her task and refocuses it on the requirements of the tool. The smaller the gulf between the operations that a tool provides and what its users want to do, the less the users need to think about the tool and the more they can concentrate on their task. As a result, the tool becomes automatic more quickly.

The way to reduce the gulf is to design the tool to provide operations that match what users are trying to do. To build on the examples above:

- A telescope's control system could have a database of celestial objects, so users could simply indicate which object they want to observe, perhaps by pointing to it on a display.
- Telephones with speed dial allow users to simply specify the person or organization they want to call, rather than having to translate that to a number first.
- A special-purpose organization chart editing application would let users simply enter the names of organizations and managers, freeing users from having to create boxes and connect them.

To design software, services, and appliances to provide operations matching users' goals and tasks, designers must thoroughly understand the user goals and tasks the tool is intended to support. Gaining that understanding requires three steps:

1. Perform a task analysis
2. Design a task-focused conceptual model, consisting mainly of an objects/actions analysis
3. Design a user interface based strictly on the task analysis and conceptual model

## Task analysis

Describing in detail how to analyze users' goals and tasks is beyond the scope of this book. Entire chapters—even whole books—have been written about it (Beyer & Holtzblatt, 1997; Hackos & Redish, 1998; Johnson, 2007). For now, it is enough to say that a good task analysis answers these questions:

- What goals do users want to achieve by using the application?
- What set of human tasks is the application intended to support?
- Which tasks are common, and which ones are rare?
- Which tasks are most important, and which ones are least important?
- What are the steps of each task?
- What are the result and output of each task?
- Where does the information for each task come from?
- How is the information that results from each task used?
- Which people do which tasks?
- What tools are used to do each task?
- What problems do people have performing each task? What sorts of mistakes are common? What causes them? How damaging are mistakes?
- What terminology do people who do these tasks use?
- What communication with other people is required to do the tasks?
- How are different tasks related?

Once these questions are answered (by observing and/or interviewing people who do the tasks that the tool will support), the next step is *not* to start sketching possible user interfaces. The next step is to design a conceptual model for the tool that focuses on the users' tasks and goals (Johnson & Henderson, 2002).

A conceptual model explains the function of the software and what concepts people need to be aware of in order to use it. Ideally, the concepts should be those that came out of the task analysis. The more direct the mapping between the tool's concepts and those of the tasks it is intended to support, the less translating users will have to do, and the easier the tool will be to learn.

After you have designed a conceptual model that is task-focused, as simple as possible, and as consistent as possible, you can design a user interface for it that minimizes the time and experience required for using the application to become an automatic process.

## Objects/actions analysis

The most important component of a conceptual model is an objects/actions analysis. This specifies all of the conceptual objects that an application will expose to users, the actions that users can perform on each object, the attributes (user-visible settings) of each type of object, and the relationships between objects (Card, 1996; Johnson & Henderson, 2002).

The software's implementation may include objects other than those listed in the conceptual model, but, if so, those extra objects should be invisible to users. Objects and actions that are related purely to implementation—such as a text buffer, a hash table, or a database record—do not belong in a conceptual model.

The objects/actions analysis, then, is a *declaration* of the concepts that are exposed to users. Follow this rule: “If it isn't in the objects/actions analysis, users shouldn't know about it.”

If we were designing software for managing checking accounts, a task-based objects/actions analysis would include objects like *transaction*, *check*, and *account*. It would exclude non-task-related objects like *buffer*, *dialog box*, *mode*, *database*, *table*, and *text string*.

A task-based conceptual model would include actions like *writing* and *voiding* checks, *depositing* and *withdrawing* funds, and *balancing* accounts, while excluding non-task-related actions like *clicking* buttons, *loading* databases, *editing* table rows, *flushing* buffers, and *switching* modes.

In a task-focused conceptual model, the attributes might be as follows:

- **Checks** have a *payee*, a *number*, an *amount*, *memo* text, and a *date*
- **Accounts** have an *owner* and a *balance*
- **Transactions** (deposits and withdrawals) have an *amount* and a *date*

If the model included attributes from computer technology, such as *transaction record format*, it would not be task-focused. Users wouldn't care what internal format the application used for storing transaction records. Forcing them to care would detract from the learnability and usability of the software, no matter how much effort went into designing the user interface.

The objects, actions, and attributes for checkbook management may seem obvious, so let's consider a task for which the objects/actions analysis may seem less clear-cut: customers posting comments about products at an online store.

Suitable objects for a conceptual model might include *customers*, *products*, *customer comments*, and *responses* to comments. Unsuitable objects would include *databases*, *tables*, and *persistent cookies*.

Actions on products would include *viewing* and *adding* comments. Actions on comments would include *viewing* and *responding*, and, for a user's own comments, *editing*. The attributes of a comment might include the *title*, the customer's *name*, and the posting *date*.

Notice that for both the checkbook management application and the customer commenting system, important conceptual design issues can be decided before the user interface is designed, or even before we know whether the user interface is presented on a personal computer screen or via voice menus on a telephone.

### As simple as possible

In addition to being focused on users' tasks, a conceptual model should be as simple as possible. Simpler means fewer concepts. The fewer concepts a model has for

users to master, the better, as long as it provides the required functionality. Less is more, provided that what is there fits well with users' goals and tasks.

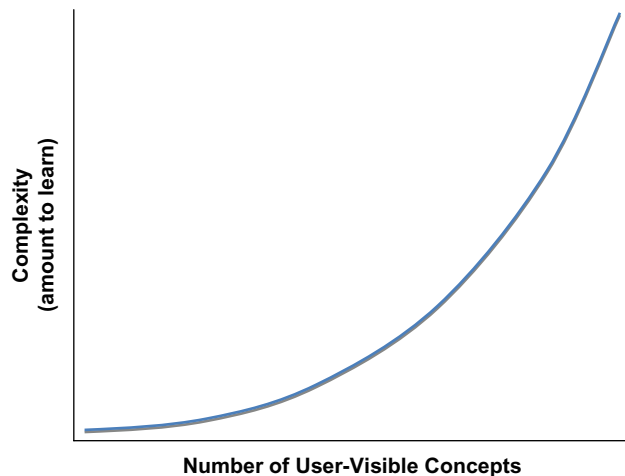
For example:

- In a To-Do List application, do users need to be able to assign priorities of 1-10 to items, or are two priority levels – low and high – enough?
- Does a Search function need to allow users to enter full Boolean expressions? If it allowed that, would a significant number of people use it? If not, leave it out.
- Does a ticket machine in a train station need to be able to offer tickets for train routes other than the routes that this station is on?

In most development efforts, there is pressure to add extra functionality “in case a user might want it.” Resist such pressure unless there is considerable evidence that a significant number of potential customers and users really need the extra functionality. Why? Because every extra concept increases the complexity of the software. It is one more thing users have to learn. But actually it is not just *one* more thing. Each concept in an application interacts with most of the other concepts, and those interactions result in more complexity. Therefore, as concepts are added to an application, the application's complexity grows not just linearly, but multiplicatively (see Fig. 11.1).

## Consistency

The *consistency* of an interactive system strongly affects how quickly its users progress from controlled, consciously monitored, slow operation to automatic, unmonitored, faster operation (Schneider & Shiffrin, 1977). The more predictable the operation of



**FIGURE 11.1**

The complexity of an application increases nonlinearly as concepts are added.



### EXCESS COMPLEXITY DUE TO SEPARATE CONCEPTS BEING TOO SIMILAR

Some software applications are too complex because they have concepts that overlap in meaning or functionality. For example, one company's customer-support Web site presented four concepts that the developers considered quite different:

- **Membership:** whether a company had paid for the customer-support service.
- **Subscription:** whether a company had subscribed to a customer-support newsletter.
- **Access:** which areas of the customer-support Web site users in a company could access.
- **Entitlements:** services provided for each membership level.

Users confused these four concepts. The four concepts should have been collapsed into *one*, or at least fewer than four.

Another company developed a Web site for people seeking to buy a home. There were two ways to start looking for a home: (a) name the state, county, or town; and (b) point to a location on a map. The site called these two methods "by location" or "by map," respectively, and required users to choose one. A usability test found that many users did not think of those as different ways of finding a home. To them, both methods were *by location*; they just differed in how the location was specified.

a system's different functions, the more consistent it is. In a highly consistent system, the operation of a particular function is predictable from its *type*, so people quickly learn how everything in the system works and its use quickly becomes habitual. In an inconsistent system, users cannot predict how its different functions work, so they must learn each one anew, which slows their learning of the overall system and keeps their use of it a controlled, attention-consuming process.

Interactive systems can be consistent or inconsistent on at least two different levels: the conceptual level and the keystroke level. Consistency at the conceptual level is determined by the mapping between the objects, actions, and attributes of the conceptual model (see above). Do most objects in the system have the same actions and attributes, or not? Consistency at the keystroke level is determined by the mapping between the conceptual actions and the physical movements required

to execute them. Are all conceptual actions of a certain type initiated and controlled by the same physical movements, or not?

### The objects/actions matrix

An optional but sometimes useful step in designing an interactive system is to illustrate its conceptual model as a matrix of objects and actions. Objects are listed down the left edge; actions are listed across the top (see Fig. 11.2). For now, we are ignoring the object *type* hierarchy and simply listing the objects. The more objects, the taller the matrix; the more actions, the wider.

Constructing an objects/actions matrix lets you visualize the simplicity or complexity of your interactive system's conceptual model. The larger the matrix, the *more* concepts there are to learn. A tall matrix indicates many *objects* to master. A wide one indicates many *actions* to learn. The matrix also illustrates how consistent or inconsistent the conceptual model is—how easy it is for users to transfer what they have learned about one part of the system to another.

A small, dense matrix indicates a design that will be easy to learn: few objects, few actions, and the operations on every type of object are the same (see Fig. 11.3A). For example, the conceptual objects in a simple drawing program would be graphical elements: lines, ellipses, arcs, rectangles, triangles, text labels, etc. The applicable actions on graphical objects would presumably be create, delete, view/edit attributes, move, copy, resize, rotate, flip, etc. The objects/actions matrix for such a simple drawing application would have a row for each object type and a column for each action. All the actions would apply to every object type, so the matrix would be densely packed, like that in Figure 11.3A.

A large, sparse matrix reflects an inconsistent design that will be hard to learn and remember because every conceptual object has different actions (see Fig. 11.3B). Such a design will be hard to learn and remember, no matter what user interface is plastered on it.

A good rule of thumb is to simplify the conceptual model so that the matrix representing it is as small and dense as possible. However, a small matrix reflects limited functionality. Achieving a small matrix is difficult when the application is anything

		Actions										
		A	B	C	D	E	F	G	H	I	...	
Objects	1		X			X	X				X	
	2			X			X	X				
	3								X			
	4	X			X	X	X		X			
	5	X			X		X		X			
	6		X			X		X	X	X		
	...											

**FIGURE 11.2**

Objects/actions matrix shows the actions for each object.

		Actions					
		A	B	C	D	E	F
Objects	1	✓	✓	✓	✓	✓	✓
	2	✓	✓	✓	✓	✓	✓
	3	✓	✓	✓	✓	✓	✓
	4	✓	✓	✓	✓	✓	✓
	5	✓	✓	✓	✓	✓	✓

**A - Good**

		Actions												
		A	B	C	D	E	F	G	H	I	J	K	L	M
Objects	1		✓	✓				✓						
	2					✓	✓					✓		
	3								✓		✓			
	4	✓			✓	✓	✓		✓					
	5			✓							✓		✓	
	6					✓			✓			✓		
	7		✓						✓					✓
	8				✓	✓	✓						✓	
	9	✓				✓								

**B - Bad**

FIGURE 11.3

Objects/actions matrices representing easy-to-learn vs. hard-to-learn conceptual designs.

more functional than, say, a simple drawing program, a personal phone directory, or a Web site for looking up postage rates. Consider, for example, the objects and actions an intensive-care patient-monitoring system would have. Even a typical word processing application—e.g., Microsoft Word or Apple Pages—embodies a nontrivial array of conceptual objects and actions.

However, for any desired functionality, a designer can develop conceptual models of varying complexity. For example, most personal bank account tracking applications have similar functionality, but Intuit's Quicken has an ultra-simple conceptual model, which may be one reason that it is so popular. Software designers should just aim for the simplest conceptual model (with the most compact objects/actions matrix) for the required functionality.

Although easy-to-learn, easy-to-use systems often have small, dense object/action matrices, they can have other matrix configurations as well. Consider an application in which all functionality is accessed through five or six generic actions that apply to all objects. Such a system could have a large number of objects without much negative impact on learnability, because all objects operate in a totally consistent way. The object/action matrix for such a system, although tall, would be narrow and dense. This approach has been used to design some highly functional systems, such as the Xerox Star office workstation (Johnson *et al.*, 1989). In Star, the same six commands—move, copy, open, delete, show properties, and copy properties—applied to all objects: characters, words, paragraphs, table rows, tables, charts, email messages, documents, folders, printers, etc.

If we include the object *type* hierarchy in the matrix, we can see another sort of conceptual model that is easy to learn: one in which objects fall into clear *categories*, each one having its own actions, perhaps with a few actions that apply to *all* objects (see Fig. 11.4). The matrix for such a model isn't small or dense, but it also isn't scattered. It has a regularity and consistency that aid learning and retention.

An example would be a real estate service offering both commercial and residential properties, with different actions for each as well as actions that apply to both types of properties.

		Type-Specific Actions										Common Actions		
		A	B	C	D	E	F	G	H	I	J	K	L	M
Objects	X	1	X	X	X	X								X
	2	X	X	X	X									X
	3	X	X	X	X									X
	Y	4				X	X	X	X					X
	5					X	X	X	X					X
	6					X	X	X	X					X
	7									X	X	X	X	X
	Z	8								X	X	X	X	X
	9									X	X	X	X	X

FIGURE 11.4

Objects/actions matrix representing a more realistic easy-to-learn conceptual design.

I regard creating an objects/actions matrix as an optional design step for two reasons:

- Experienced interaction designers rarely need to actually draw the matrix to know whether the conceptual model underlying their design is simple or complex, consistent or inconsistent.
- Usability testing can reveal aspects of an application’s conceptual model that designers did not know needed to be simplified.

It may be enough for designers, as they design the conceptual model for an application, to imagine what its objects/actions matrix *would look like if they drew it*.

The goal is to devise a conceptual model that is task-focused, as simple as possible, and as consistent as possible. From such a model, one can design a user interface for it that minimizes the time and experience required for using the application to become automatic.

### Keystroke consistency

When a designer moves from conceptual design to actual user interface design, keystroke-level consistency becomes important.

Keystroke-level consistency is harder to illustrate and measure, but it is *at least* as important as conceptual consistency in determining how quickly the operation of an interactive system becomes automatic. The goal is to foster the growth of what is often called “muscle memory,” meaning motor habits.

Achieving keystroke-level consistency requires standardizing the physical actions for all activities of the same type. An example of a type of activity is editing text. Keystroke-level consistency for text editing requires the keystrokes (and pointer movements) to be the same regardless of the context in which text is being edited—documents, form fields, filenames, etc. Other types of activities for which keystroke-level consistency is desirable are opening documents, following links, choosing from a menu, choosing from a displayed set of options, clicking buttons, scrolling a display, etc.

A system that is inconsistent at the keystroke level does *not* let people quickly fall into “muscle memory” motor habits but, rather, keeps them guessing about what keystrokes to use in each context, even when contexts differ only slightly.

A common way that developers promote keystroke-level consistency is to follow look-and-feel standards. Such standards can be presented in style guides or they can be built into common user interface construction tools and component sets. Style guides exist for the entire industry and they exist separately for desktop software (Apple Computer, 2009; Microsoft Corporation, 2009) and Web design (Koyani, Bailey, & Nall, 2006). Ideally, companies also have internal style guides that augment the industry style guides to define a look and feel for their own products.

However conventions are encapsulated, the goal is to stick to conventions at the keystroke level while perhaps innovating at the conceptual and task levels. We as designers *really* don't want our software's users to have to keep thinking about their keystroke-level actions as they work, and users don't want to think about them either.

---

## WE LEARN FASTER WHEN VOCABULARY IS TASK-FOCUSED, FAMILIAR, AND CONSISTENT

Ensuring that an application, Web service, or appliance exposes a small, consistent, and task-appropriate set of concepts to its users is a big first step, but it is not enough to minimize the time it takes for people to learn an interactive system. You also have to make sure that the *vocabulary*—what concepts are called—fits the task, is familiar, and is consistent.

### Terminology should be task-focused

Just as the user-visible concepts in an interactive system should be task-focused, so should the *names* for the concepts. Usually, task-focused terms for concepts emerge from the interviews and observations of users that designers conduct as part of the task analysis. Occasionally, software needs to expose a concept that is new to users; the challenge for a designer is keeping such concepts and their names focused on the task, not on the technology.

Some examples of interactive software systems using terminology that is not task focused:

- A company developed a desktop software application for performing investment transactions. The application let users create and save templates for common transactions. It gave users the option of saving templates either on their own PC or on a network server. Templates stored on the PC were private. Templates stored on the server were accessible to other people. The developers used the term “database” for templates on the server because they were kept in a database. They used “local” for templates on the users' own PC because that's what “local” meant to them. Terms that would be more task focused are “shared” or “public” instead of “database”, and “private” instead of “local.”
- iCasualties.org provides up-to-date tallies of the number of Coalition military personnel killed or injured in the Iraq and Afghanistan wars. It starts by asking



FIGURE 11.5

iCasualties.org uses language that is not task-focused (“database”) in its instructions.

site visitors to select a “database.” However, visitors to this site don’t care or need to know that the Web site’s data is stored in multiple databases. Task-focused instructions would ask them to select a *country* in which there is an ongoing conflict, not a *database* (see Fig. 11.5).

### Terminology should be familiar

To reduce the time it takes for people to master your application, Web site, or appliance, so that using it becomes automatic or nearly so, don’t force them to learn a whole new vocabulary. Chapter 4 explained that familiar words are easier to read and understand because they can be recognized automatically. Unfamiliar words cause people to use more conscious decoding methods, which consumes scarce short-term memory resources and thereby lowers comprehension.

Unfortunately, many computer-based products and services present users with unfamiliar terms from computer engineering—often called “geek speak”—and require them to master those terms (see Fig. 11.6). Why? Operating a stove doesn’t require us to master terminology about the pressure and chemical composition of natural gas, or terminology about the production and delivery of electricity. Why should shopping on the Web, sharing photographs, or checking email require us to learn geek speak such as *USB*, *TIFF*, or *broadband*? But in many cases, it does.

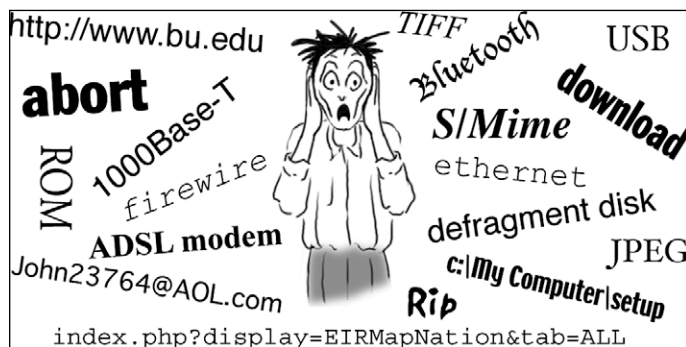


FIGURE 11.6

Unfamiliar computer jargon (aka “geek speak”) slows learning and frustrates users.

Some examples of interactive software systems using unfamiliar terminology:

- A development team was designing a video-on-demand system for schoolteachers to use in classrooms. The purpose of the system was to allow teachers to find videos offered by their school district, download them, and show them in their classrooms. The developers' initial plan was to organize the videos into a hierarchy of "categories" and "subcategories." Interviews with teachers showed, however, that they use the terms "subject" and "unit" to organize instructional content, including videos. If the system had used the developers' terminology, teachers who used it would have to *learn* that "category" meant "subject" and "subcategory" meant "unit," making the system harder to master.
- Continental Airlines' Web site displays several error messages that speak "geek" (see Fig. 11.7). Most are attempts to tell the Web site user about a problem, but because they use an unfamiliar jargon, few users understand what the site is saying and so are unsure what to do. Such error messages are more appropriate for reporting the problem to system engineers. Error messages like these should either be rewritten in terms users understand, or they should be displayed to the Web site administrators who *monitor* the operation of the site rather than to the users.
- Windows Media Player sometimes displays error messages that use familiar terms in unfamiliar, "geeky" ways (see Fig. 11.8). The error message in the figure is referring to the state of the *software*, but the average Media Player user is likely to interpret it as referring to the *state* in which he or she lives.

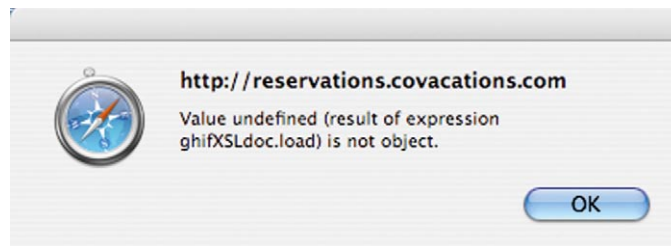


FIGURE 11.7

Error message at Continental.com uses "geek speak" (computer jargon).

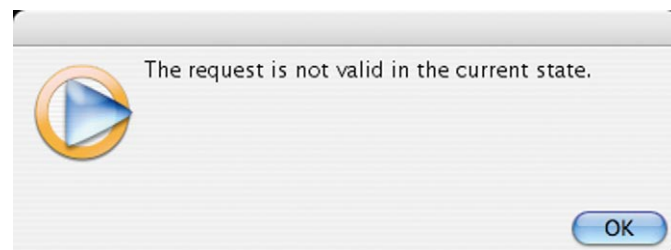
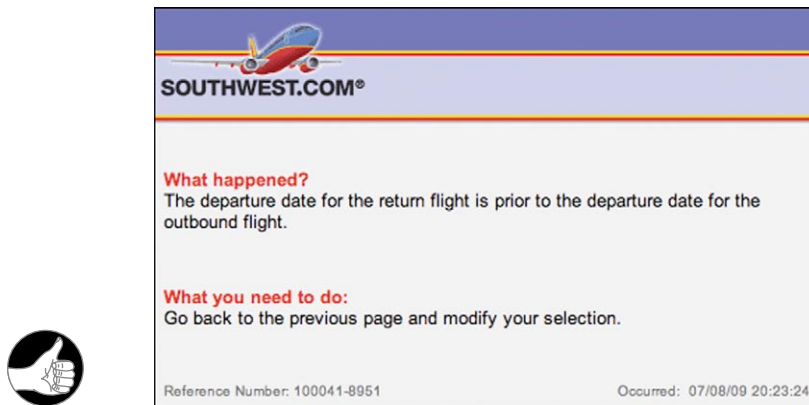


FIGURE 11.8

Error message in Windows Media Player uses a familiar term ("current state") in an unfamiliar way.



**FIGURE 11.9**

Error messages at Southwest Airlines' Web site are task-focused and clear, fostering learning.

In contrast to these examples, Southwest Airlines' Web site tries to prevent errors from occurring, but when they do occur, it explains the problem using task-focused, familiar language (see Fig. 11.9).

### Terminology should be consistent

People want to focus their cognitive resources on their own goals and tasks, not on the software they are using. They just want to accomplish their goal, whatever it is. They are not interested in the software. They interpret what the system presents only superficially and very literally. Their limited attentional resources are so focused on their goal that if they are looking for a Search function but it is labeled "Query" on the current screen or page, they may miss it. Therefore, the terminology in an interactive system should be designed for maximum consistency.

The terminology used in an interactive system is consistent when each concept has *one and only one* name. Caroline Jarrett, an authority on user interface and forms design, provides this rule:

---

Same name, same thing; different name, different thing. (FormsThatWork.com)

---

This means that terms and concepts should map strictly 1:1. Never use *different* terms for the *same* concept, or the *same* term for *different* concepts. Even terms that are ambiguous in the real world should mean only one thing *in the system*. Otherwise, the system will be harder to learn and remember.

An example of different terms for the same concepts is provided by Earthlink's frequently asked questions (FAQ) page in the Web-hosting section of its site (see Fig. 11.10). In the *question*, the two available Web-hosting platforms are called "Windows-based" and "UNIX-based," but in the *table* they are referred to as "Standard" and "ASP." Customers have to stop and try to figure out which one is which. Do you know?



**What are the differences between Windows-based and UNIX-based platforms?**  
 For detailed information on choosing between these two operating systems, please visit our Windows or UNIX page.

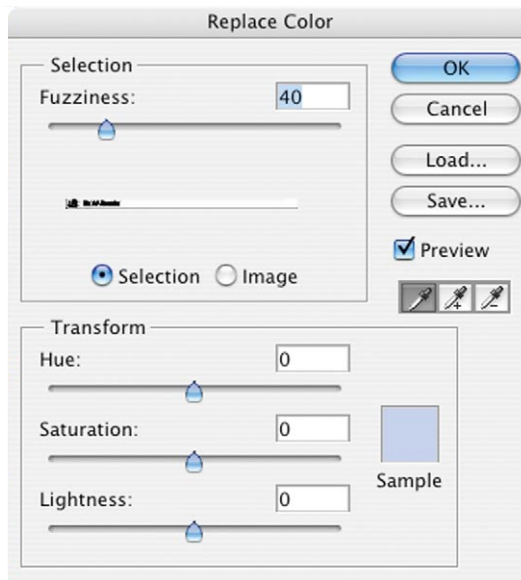
For a quick look at which features and programs each platform supports, please consult the chart below:

Feature	Standard	ASP
Microsoft FrontPage Extensions	YES	YES
RealVideo and RealAudio	YES	NO
ASP (Active Server Pages)	NO	YES
ADO (ActiveX Data Objects)	NO	YES
ODBC Data Sources	NO	YES
Windows Media Server	NO	YES
PHP	YES	NO
MySQL Databases	YES	NO
Free ready-to run scripts, such as a hit counter, forum, e-mail form, blog, and guestbook	YES	NO



FIGURE 11.10

Earthlink's Web-hosting FAQ uses different terms for the same options in the question and in the table.



(A)



(B)

FIGURE 11.11

Photoshop uses different names for the tolerance parameter in two color-replacement functions: (A) “Fuzziness” in Replace Color; (B) “Tolerance” in Paint Bucket.

An example from Adobe Photoshop shows that inconsistent terminology can impede learning. Photoshop has two functions for replacing a target color in an image: *Replace Color*, which replaces the target color throughout an image with a new color, and *PaintBucket*, which replaces the target color in an enclosed area with a new color. Both functions have a parameter that specifies how similar a color in the image must be to the target color before it will be replaced. The inconsistency is that the Replace Color function calls this parameter “Fuzziness,” but the Paint Bucket function calls it “Tolerance” (see Fig. 11.11). Photoshop’s online Help documentation for Replace Color even says “Adjust the *tolerance* of the mask by dragging the *Fuzziness* slider or entering a value” [emphasis added]. If the parameter were simply called “Tolerance” in *both* color replacement functions, people who learned one function could quickly transfer that learning to the other. But it isn’t, so people have to learn the two functions separately.

Finally, WordPress.com provides an example of the *same* term for *different* concepts—also called *overloading* a term. For administering a blog, WordPress provides each blogger with a *Dashboard* consisting of monitoring and administrative functions organized into several pages. The problem is that one of the administrative function pages in the Dashboard is also called the “Dashboard,” so the same name refers to both the whole Dashboard and one page of it (see Fig. 11.12). Therefore, when new bloggers are learning to use WordPress, they have to discover and remember that sometimes “Dashboard” means the entire administrative area and sometimes it means the Dashboard *page* of the administrative area.

### Developing task-focused, familiar, consistent terminology is easier with a good conceptual model

The good news is that when you perform a task analysis and develop a task-focused conceptual model, you also get the vocabulary your target user population uses to talk about the tasks. You don’t have to make up new terms for the user-visible concepts in your application—you can use the terms that people who do the task *already* use. In fact, you *shouldn’t* assign new names for those concepts, because any names you assign will likely be computer technology concepts, foreign to the task domain.<sup>1</sup>

From the conceptual model, a software development team should create a product *lexicon*. The lexicon gives a name and definition for each object, action, and attribute that the product—including its documentation—exposes to users. The lexicon should map terms onto concepts 1:1. It should not assign multiple terms to a single concept, or a single term to multiple concepts.

Terms in the lexicon should come from the software’s supported *tasks*, not its implementation. Terms should fit well into the users’ normal task vocabulary, even if

---

<sup>1</sup>Unless you are designing software development tools.

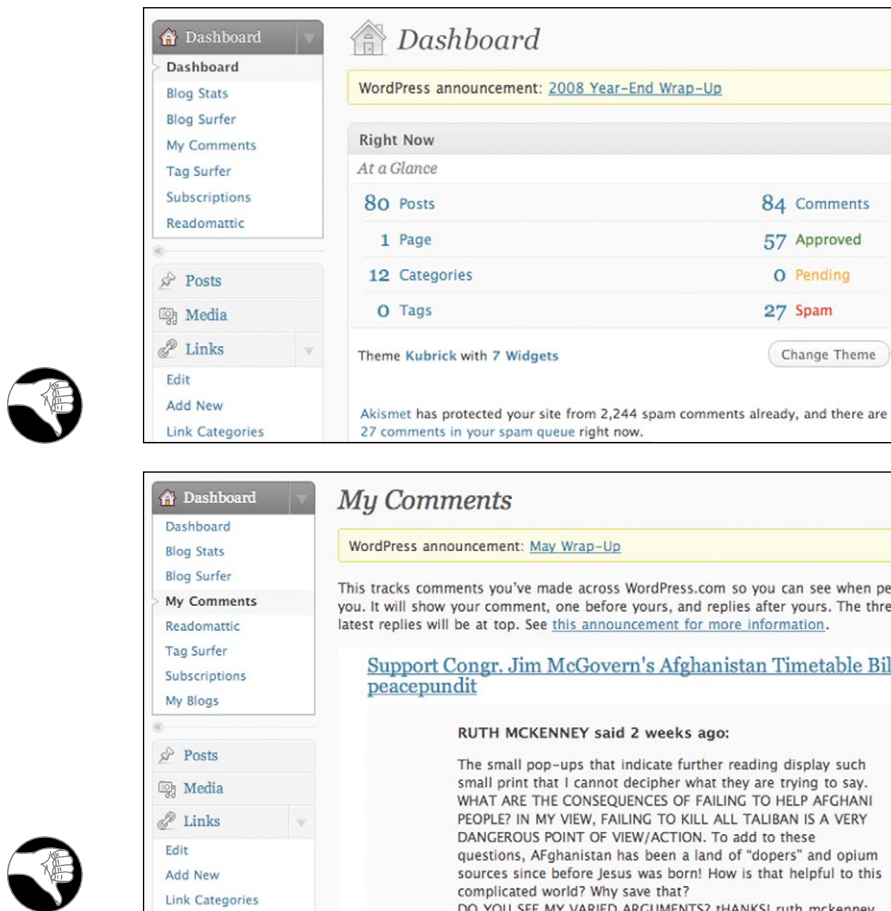


FIGURE 11.12

At WordPress, “Dashboard” means both a blog’s entire administrative area and a certain page in it.

they are new. Typically, technical writers, user interface designers, developers, managers, and users all help create the lexicon.

Certain concepts in GUIs have industry-standard names. These are the GUI equivalents of “reserved words” in programming languages. If you rename such concepts or assign new meanings to the standard names, you will confuse users.

Follow the product lexicon consistently throughout the software, user manuals, and marketing literature. Treat it as a living document: As the product evolves, the lexicon changes based on the basis of new design insights, changes in functionality, usability test results, and market feedback.

## WE LEARN FASTER WHEN RISK IS LOW

Imagine you are visiting a foreign city on business for a week or two. You have spare time after your work duties are finished in the evenings and on weekends. Compare two possible cities:

- You have been told that this city is easy to get around in: it is laid out in a consistent grid of streets and avenues with clear street and subway signs written in a language you understand, and the residents and police speak your language and are friendly and eager to help tourists.
- You have been warned that this city has a convoluted, confusing layout, with winding, poorly marked streets; the few street and subway signs are in a language you cannot read, and residents don't speak your language and are generally contemptuous of tourists.

In which city are you more likely to go out exploring?

Most interactive systems—desktop software, Web services, electronic appliances—have *far* more functionality than most of their users ever try. Often people don't even *know* about most of the functionality provided by software or gadgets they use every day. One reason for this is fear of being “burned.”

People make mistakes. Many interactive systems make it too easy for users to make mistakes, do not allow users to correct mistakes, or make it costly or time-consuming to correct mistakes. People won't be very productive in using such systems: they will waste too much time correcting or recovering from mistakes.

Even more important than the impact on *time* is the impact on *learning*. A high-risk system, in which mistakes are easy to make and costly, discourages exploration: people who are anxious and afraid of making mistakes will tend to stick to familiar, safe paths and functions. When exploration is discouraged and anxiety is high, learning is severely hampered.

In contrast, a low-risk system, in which mistakes are hard to make, low in cost, and easy to correct, reduces stress and encourages exploration, and therefore greatly fosters learning. With such systems, users are more willing to try new paths: “Hmmm, I wonder what *that* does.”

To foster learning, interactive systems should be low-risk environments, so users are not afraid to explore and try new things. Designing software this way means doing the following:

- Prevent errors where possible
- Deactivate invalid commands
- Make errors easy to detect by showing users clearly what they have done (e.g., deleting a paragraph by mistake)
- Allow users to undo, reverse, or correct errors easily

---

## SUMMARY

The goal of this chapter is to explain and demonstrate the factors that affect how quickly people can learn to use interactive systems so proficiently that operating the system is handled largely by automatic cognitive processes. We learn to use interactive systems faster under the following conditions:

- Their operation is based on users' goals and tasks (not on the implementation of the system), conceptually simple, and consistent
- The vocabulary they employ is familiar to users, is based on that of the task domain, and is used consistently in the sense that it maps terms onto concepts 1:1
- They provide a low-risk environment, in which errors are difficult to make and, when users do make errors, they are low in cost and easy to correct.

# We Have Time Requirements

# 12

Events in the world take time to play out. *Perceiving* objects and events also takes time. So does remembering perceived events, thinking about past and future events, learning from events, acting on plans, and reacting to perceived and remembered events. How much time? And how does knowing the duration of perceptual and cognitive processes help us design interactive systems?

This chapter provides answers to those questions. It presents the *durations* of perceptual and cognitive processes, and based on those, provides some real-time *deadlines* that interactive systems must meet in order to synchronize well with human users. Systems that don't synchronize well with users' time requirements are less effective tools and they are perceived as unresponsive.

The second issue, perceived responsiveness, may seem less important than effectiveness, but in fact it is more important. Over the past four decades, researchers have found consistently that an interactive system's responsiveness—its ability to keep up with users, keep them informed about its status, and not make them wait unexpectedly—is the *most* important factor in determining user satisfaction. It is not just *one* of the most important factors; it is *the* most important factor.<sup>1</sup> It is more important than ease of learning. It is more important than ease of use. Study after study has confirmed this finding (Barber & Lucas, 1983; Carroll & Rosson, 1984; Lambert, 1984; Miller, 1968; Rushinek & Rushinek, 1986; Shneiderman, 1984; Thadhani, 1981).

This chapter first defines responsiveness. It then enumerates some important time constants of human perception and cognition. It ends with real-time guidelines for interactive system design, including examples.

---

<sup>1</sup>Some researchers have suggested that for users' perception of the loading speed of Web sites, the causality may go the other way: the more success people have at a site, the faster they think it is, even when their ratings don't correlate with the sites' actual speed (Perfetti & Landesman, 2001).

## RESPONSIVENESS DEFINED

Responsiveness is related to performance, but it is different. Performance is measured in terms of computations per unit of time. Responsiveness is measured in terms of compliance with human time requirements and, as described above, user satisfaction.

Interactive systems can be responsive despite low performance. If you call someone to ask a question, he can be responsive even if he can't answer your question immediately: he can acknowledge the question and promise to call back. He can be even more responsive by saying *when* he will call back.

Responsive systems keep a user informed even if they cannot fulfill the user's requests immediately. They provide feedback about what the user has done and what is happening, and they prioritize the feedback based on *human* perceptual, motor, and cognitive deadlines (Duis & Johnson, 1990). Specifically, they do the following:

- Let you know immediately that your input was received
- Provide some indication of how long operations will take (see Fig. 12.1)
- Free you to do other things while waiting
- Manage queued events intelligently
- Perform housekeeping and low-priority tasks in the background
- Anticipate your most common requests

Software can have poor responsiveness even if it is fast. Even if a watch repairman is very fast at fixing watches, he is unresponsive if you walk into his shop and he ignores you until he finishes working on another watch. He is unresponsive if you hand him your watch and he silently walks away without saying whether he is going to fix it now or go to lunch. Even if he starts working on your watch immediately, he is unresponsive if he doesn't tell you whether fixing it will take five minutes or five hours.

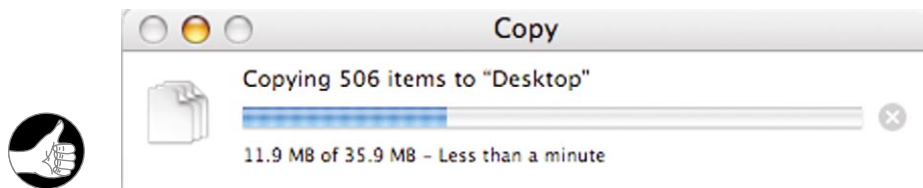


FIGURE 12.1

MacOS X file transfer: good progress indicator, useful time estimate, cancel button (circled X).

Systems that display poor responsiveness do not meet human time deadlines. They don't keep up with users. They don't provide timely feedback for user actions, so users are unsure of what they have done or what the system is doing. They

make users wait at unpredictable times and for unpredictable periods. They limit users' work pace—sometimes severely. Here are some specific examples of poor responsiveness:

- Delayed feedback for button presses, scrollbar movement, or object manipulations
- Time-consuming operations that block other activity and cannot be aborted (see Fig. 12.2)
- Providing no clue how long lengthy operations will take (see Fig. 12.2)
- Jerky, hard-to-follow animations
- Ignoring user input while performing “housekeeping” tasks users did not request

These problems impede users' productivity and frustrate and annoy them. Unfortunately, despite all of the research showing that responsiveness is critical to user satisfaction and productivity, a lot of today's interactive systems have poor responsiveness (Johnson, 2007).



**FIGURE 12.2**

MacOS X: No progress bar (just a busy bar) and no cancel. (A) MacOS X, (B) iMovie.

## THE MANY TIME CONSTANTS OF THE HUMAN BRAIN

To understand the time requirements of human users of interactive systems, let's start with neurophysiology.

The human brain and nervous system are not really a single organ; rather, they are made up of a collection of neuron-based organs that appeared at vastly different points in the evolutionary chain from worms to people. This collection provides a large variety of sensory, regulatory, motor, and cognitive functions. Not surprisingly, these functions operate at different speeds. Some work very fast, executing functions in small fractions of a second, while others are many, many times slower, executing over many seconds, minutes, hours, or even longer time spans.

For example, Chapter 10 explained that automatic processing, such as *playing* a memorized musical piece, operates on a “clock” that is at least 10 times faster than the one governing highly monitored, controlled processing, such as *composing* a



### HOW LONG DOES OUR BRAIN TAKE TO...?

Listed below are measured durations for perceptual and cognitive brain functions that affect our perceptions of system responsiveness. The times are listed from shortest to longest (Card *et al.*, 1991; Johnson, 2007; Sousa, 2005; Stafford & Webb, 2005):

Perceptual and Cognitive Functions	Duration
Shortest gap of silence that we can detect in a sound	1 millisecond (0.001 second)
Minimum time between spikes in auditory neurons, the fastest neurons in the brain	milliseconds (0.002 second)
Shortest time a visual stimulus can be shown and still affect us (perhaps unconsciously)	5 milliseconds (0.005 second)
Minimum noticeable lag in ink as someone draws with a stylus	10 milliseconds (0.01 second)
Maximum interval for auditory fusion of successive sound pulses into a pitched tone	20 milliseconds (0.02 second)
Maximum interval for visual fusion of successive images	50 milliseconds (0.05 second)
Speed of flinch reflex (involuntary motor response to possible danger)	80 milliseconds (0.08 second)
Time lag between a visual event and our full perception of it (or perceptual cycle time)	100 milliseconds (0.1 second)
Duration of <i>saccade</i> (involuntary eye movement), during which vision is suppressed	100 milliseconds (0.1 second)
Maximum interval between events for perception that one event caused another event	140 milliseconds (0.14 second)
Time required for a skilled reader's brain to comprehend a printed word	150 milliseconds (0.15 second)

Time to <i>subitize</i> (determine the number of) up to four to five items in our visual field	200 milliseconds (0.2 second; 50 milliseconds/item)
Editorial “window” for events that reach consciousness	200 milliseconds (0.2 second)
Time to identify (i.e., name) a visually presented object	250 milliseconds (0.25 second)
Time required to mentally count <i>each</i> item in a scene when there are <i>more than four items</i>	300 milliseconds (0.3 second)
Attentional “blink” (inattentiveness to other input) following recognition of an object	500 milliseconds (0.5 second)
Visual-motor reaction time (intentional response to unexpected event)	700 milliseconds (0.7 second)
Maximum duration of silent gap between turns in person-to-person conversation	About 1 second
Duration of unbroken attention to a single task (“unit task”)	6–30 second
Time to make critical decisions in emergency situations, e.g., medical triage	1–5 minutes
Duration of important purchase decision, e.g., buying a car	1–10 days
Time to choose a lifetime career	20 years

musical piece. Another example is the flinch reflex: a region of the brain called the superior colliculus—part of the evolutionarily ancient old brain—can “see” a rapidly approaching object and can make you flinch or raise your arms long before your cortex has perceived and identified the object.

The sidebar gives the durations of some important perceptual and cognitive brain functions. Most are self-explanatory, but a few require additional explanation.

### **Shortest gap of silence that we can detect in a sound: 1 millisecond (0.001 second)**

Our hearing is much more sensitive to short events and small differences than our vision is. Our ears operate using mechanical sound transducers, not electrochemical

neural circuitry. The eardrum transmits vibration to the ossicles (middle-ear bones), which in turn transmit vibration to the cochlea's hair cells, which, when vibrated, trigger electrical pulses that go to the brain. Because the connection is mechanical, our ears respond to sound much faster than the rods and cones in our retina respond to light. This speed allows our auditory system to detect very small differences in the time when a sound arrives at our two different ears, from which our brain calculates the direction of the sound's source.

### **Shortest time a visual stimulus can be shown and still affect us (perhaps unconsciously): 5 milliseconds (0.005 second)**

This is the basis of so-called subliminal perception. If you are shown an image for 5–10 milliseconds, you won't be aware of seeing it, but low-level parts of your visual system will register it. One effect of such exposure to an image is that your familiarity with it will increase: if you see it again later, it will seem familiar. Brief exposure to an image or a looming object can also trigger responses from your old brain and midbrain—avoidance, fear, anger, sadness, joy—even if the image disappears before the conscious mind identifies it. However, contrary to popular myth, subliminal perception is not a strong determinant of behavior. It cannot make you do things you wouldn't otherwise do or want things you wouldn't otherwise want (Stafford & Web, 2005).

### **Speed of flinch reflex (involuntary motor response to possible danger): 80 milliseconds (0.08 second)**

When an object—even a shadow—approaches you rapidly, or if you hear a loud sound nearby, or if something suddenly pushes, jabs, or grabs you, your reflex is to flinch: pull away, close your eyes, throw up your hands in defense, etc. This is the flinch reflex. It is very fast compared to intentional reaction to a perceived event: about 10 times faster. Evidence of the speed of the flinch reflex has been seen not only experimentally but also in examining the injuries that occur when people are attacked or involved in vehicle accidents: often their arms and hands are injured in ways that indicate that they managed to get their hands up in a split second (Blauer, 2007).

### **Time lag between a visual event and our full perception of it: 100 milliseconds (0.1 seconds)**

From the time that light from an external event hits your retina to the time that neural impulses from that event reach your cerebral cortex, about 0.1 second elapses. Suppose our conscious awareness of the world lagged behind the real world by a tenth of a second. That lag would not be conducive to our survival: a tenth of a second is a long time when a rabbit you are hunting darts across a meadow. Our brain compensates by extrapolating the position of moving objects by 0.1 second.

Therefore, as a rabbit runs across your visual field, you see it where your brain estimates it is *now*, not where it was 0.1 second ago (Stafford & Web, 2005).

### **Maximum interval between events for perception that one event caused another event: 140 milliseconds (0.14 second)**

This interval is the deadline for perception of cause and effect. If an interactive system takes longer than 0.14 second to respond to your action, you won't perceive your action as having caused the response. For example, if the echoing of characters you type lags more than 140 milliseconds behind your typing, then you will lose the perception that you are typing those characters. Your attention will be diverted away from the meaning of the text and toward the act of typing, which slows you down, pulls typing out of automatic processing and into conscious processing, and increases your chances of making an error.

### **Time to subitize (determine the number of) up to four to five items in our visual field: 200 milliseconds (0.2 second; 50 milliseconds/item)**

If someone tosses two coins onto the table and asks how many coins there are, it takes only a glance for you to see that there are two. You don't have to explicitly count them. You can do the same with three coins, or four. Some people can do it with five. This function is called *subitizing*. Beyond four or five, it gets harder: now you are starting to have to count, or, if the coins happen to fall into separate groups on the table, you can subitize each subgroup and add the results. This phenomenon is why when we count objects using tick-marks, we write the tick-marks in groups of four, then draw the fifth tick-mark across the group, like this: + + + + + |. Subitizing feels instantaneous, but it isn't. It takes about 50 milliseconds per item (Card *et al.*, 1983; Stafford & Webb, 2005). However, that's much less time per item than explicit counting, which takes about 300 milliseconds per item.

### **Editorial “window” for events that reach consciousness: 200 milliseconds (0.2 second)**

The order in which we perceive events is not necessarily the order in which they occur. The brain apparently has a moving “editorial window” of about 200 milliseconds, during which perceived and recalled items vie for promotion to consciousness. Within that time window, events and objects that might otherwise have made it to consciousness may be superseded by others—even ones that occurred later in time (within the window). Within the window, events can also be re-sequenced on the way to consciousness. An example: we see a dot that disappears and immediately reappears in a new position as moving. Why? Our brain certainly does not do it by “guessing” the second object's position and making us see “phantom” motion in that direction, because we see motion in the correct direction *regardless* of where the new object appears. Answer: We don't actually perceive motion until the dot

appears in the new position. The second dot must appear within 0.2 second of the disappearance of the first dot in order for the brain to resequence the events.

### **Attentional “blink” (inattentiveness to other objects) following recognition of an object: 500 milliseconds (0.5 second)**

Imagine you are riding a subway, gliding slowly through a station. You pass dozens of strangers who are standing on the platform, but you pay little attention to them. Then you spot a friend on the platform, but the train keeps moving and the friend goes out of view. Your attention snaps to thinking about that friend: all sorts of thoughts and feelings about her are triggered. In that moment, your window passes another friend on the platform. Chances are that you would miss the second friend, because your mind was still on the first. That’s the attentional blink (Stafford & Webb, 2005). With a colleague’s help, you can demonstrate it. Choose two target words. Tell the colleague the two words. Then explain that you will read a list of words and at the end you want to know if either of the two target words was in the list. Quickly read a long list of words at a rate of three words per second. Somewhere in the list, include one target word. If the second target word is presented right after the first—within one or two items—your colleague probably won’t hear it.

### **Visual-motor reaction time (intentional response to unexpected event): 700 milliseconds (0.7 second)**

This interval is the combined time for your visual system to notice something in the environment and initiate a conscious motor action, and for the motor system to execute the action. If you are driving your car toward an intersection and the light turns red, this is the time required for you to notice the red light, decide to stop, and put your foot on the brake pedal. How long it takes your car to actually *stop* is not included in the 700 milliseconds. The vehicle stopping time depends on how fast the car is going, the condition of the pavement under the wheels, etc.

This reaction time is *not* the flinch reflex—the old brain responding to rapidly approaching objects, making you automatically close your eyes, dodge, or throw your hands up to protect yourself. That reflex operates about 10 times faster (see above).

The visual-motor intentional reaction time is approximate. It varies a bit among people. It also increases with distractions, drowsiness, blood-alcohol level, and possibly age.

### **Maximum duration of silent gap between turns in person-to-person conversation: ~1 second**

This is the approximate normal length of gaps in a conversation. When gaps exceed this limit, participants—either speakers or listeners—often say something to keep the conversation going: they interject “uh” or “uh-huh,” or take over as speaker. Listeners respond to such pauses by turning their attention to the speaker to see what caused it. The precise length of such gaps varies by culture, but it is always in the range of 0.5–2 seconds.

### Duration of unbroken attention to a single task (“unit task”): ~10 seconds

When people perform a task, they break it down into little pieces: subtasks. For example, buying airline tickets online consists of: (1) going to a travel or airline Web site, (2) entering the trip details, (3) scanning the results, (4) selecting a set of flights, (5) providing credit card information, (6) reviewing the purchase, and (7) finalizing the purchase. Some subtasks are broken down further, for example entering the trip details consists of entering the trip origin, destination, dates, time, etc., piece by piece. This breaking down of tasks into subtasks ends with small subtasks that can be completed without a break in concentration, with the subgoal and all necessary information either held in working memory or directly perceivable in the environment. These bottom-level subtasks are called “unit tasks” (Card *et al.*, 1983). Between unit tasks, people typically look up from their work, check to see if anything else requires attention, perhaps look out the window or take a sip of coffee, etc. Unit tasks have been observed in activities as diverse as editing documents, entering checkbook transactions, designing electronic circuits, and maneuvering fighter jet planes in dog-fights, and they always last somewhere in the range of 6–30 seconds.

---

## ENGINEERING APPROXIMATIONS OF TIME CONSTANTS: ORDERS OF MAGNITUDE

Interactive systems should be designed to meet the temporal requirements of their human users. However, trying to design interactive systems for the wide variety of perceptual and cognitive time constants would be nearly impossible.

But people who design interactive systems are engineers, not scientists. We don't have to account for the full variety of brain-related time constants and clock-cycle times. We just have to design interactive systems that work for human beings. This more approximate requirement gives us the freedom to consolidate the many perceptual and cognitive time constants into a smaller set that is easier to teach, remember, and use in design.

Examining the list of critical durations presented above yields some useful groupings. Times related to sound perception are all on the order of a millisecond, so we can consolidate them all into that value. Whether they are really 1 millisecond or 2, or 3—we don't care. We only care about factors of 10.

Similarly, there are groups of durations around 10 milliseconds, 100 milliseconds, 1 second, 10 seconds, and 100 seconds. Above 100 seconds, we are beyond durations that most interaction designers care about. Thus, for designing interactive systems, these consolidated deadlines provide the required accuracy:

- 0.001 second (1 millisecond): Shortest detectable silent audio gap
- 0.01 second (10 milliseconds): Preconscious (“subliminal”) visual perception, shortest noticeable pen-ink lag, auditory fusion

- 0.1 second (100 milliseconds): Subitizing one to four items, involuntary eye movement (saccade), perception of cause-effect, perceptual-motor feedback, visual fusion, flinch reflex, object identification, editorial “window” of consciousness, one “moment” in conscious awareness
- 1.0 second: Average conversation gap, visual-motor intentional reaction time, attentional blink
- 10 seconds: Unit task, unbroken attention to a task, one step of a complex task
- 100 seconds (1.6 minutes): Critical decision in emergency situation

Notice that these deadlines form a convenient series: each successive deadline is 10 times—one order of magnitude—greater than the previous one. That makes the series fairly easy for designers to remember, although remembering what each deadline is *for* may be challenging.

---

## DESIGNING TO MEET REAL-TIME HUMAN INTERACTION DEADLINES

To be perceived by users as responsive, interactive software must follow these guidelines:

- Acknowledge user actions instantly, even if returning the answer will take time; preserve users’ perception of cause and effect
- Let users know when the software is busy and when it isn’t
- Free users to do other things while waiting for a function to finish
- Animate movement smoothly and clearly
- Allow users to abort (cancel) lengthy operations they don’t want
- Allow users to judge how much time lengthy operations will take
- Do its best to let users set their own work pace

In the above guidelines, “instantly” means within about 0.1 second. Much longer than that, and the user interface will have moved out of the realm of cause and effect, reflexes, perceptual-motor feedback, and automatic behavior, into the realm of conversational gaps and intentional behavior see sidebar: “How long does our brain take to ....” After two seconds, a system has exceeded the expected time for turn taking in dialog and has moved into the time range of unit tasks, decision making, and planning.

Now that we have listed time-constants of human perception and cognition, and consolidated them into a simplified set, we can quantify terms such as “instantly,” “take time,” “smoothly,” and “lengthy” in the above guidelines (see also [Table 12.1](#)).

### 0.001 second (1 millisecond)

As described above, the human auditory system is sensitive to very brief intervals between sounds. If an interactive system provides audible feedback or content, its

**Table 12.1** The Time Deadlines for Human Computer Interaction

Deadline	Perceptual and Cognitive Functions	Deadlines for Interactive System Design
0.001 second	<ul style="list-style-type: none"> <li>• Minimum detectable silent audio gap</li> </ul>	<ul style="list-style-type: none"> <li>• Maximum tolerable delay or drop-out time for audio feedback (e.g., tones, “earcons,” music)</li> </ul>
0.01 second	<ul style="list-style-type: none"> <li>• Preconscious perception</li> <li>• Shortest noticeable pen-ink lag</li> </ul>	<ul style="list-style-type: none"> <li>• Inducing unconscious familiarity of images or symbols</li> <li>• Generating tones of various pitches</li> <li>• Electronic ink maximum lag time</li> </ul>
0.1 second	<ul style="list-style-type: none"> <li>• Subitizing 1–4 items</li> <li>• Involuntary eye movement (saccade)</li> <li>• Flinch reflex</li> <li>• Perception of cause-effect</li> <li>• Perceptual-motor feedback</li> <li>• Visual fusion</li> <li>• Object identification</li> <li>• Editorial window of consciousness</li> <li>• The perceptual “moment”</li> </ul>	<ul style="list-style-type: none"> <li>• Assume users can “count” 1–4 screen items in ~100 milliseconds, but more than four take 300 milliseconds/item</li> <li>• Feedback for successful hand-eye coordination, e.g., pointer movement, object movement or resizing, scrolling, drawing with mouse</li> <li>• Feedback for click on button or link</li> <li>• Displaying “busy” indicators</li> <li>• Allowable overlap between speech utterances</li> <li>• Maximum interval between animation frames</li> </ul>
1 second	<ul style="list-style-type: none"> <li>• Max conversational gaps</li> <li>• Visual-motor reaction time for unexpected events</li> <li>• Attentional “blink”</li> </ul>	<ul style="list-style-type: none"> <li>• Displaying progress indicators for long operations</li> <li>• Finishing user-requested operations, e.g., open window</li> <li>• Finishing unrequested operations, e.g., auto-save</li> <li>• Time after info presentation that can be used for other computation, e.g., to make inactive objects active</li> <li>• Required wait time after presenting important info before presenting more</li> </ul>
10 seconds	<ul style="list-style-type: none"> <li>• Unbroken concentration on a task</li> <li>• Unit task: one part of a larger task</li> </ul>	<ul style="list-style-type: none"> <li>• Completing one step of a multistep task, e.g., one edit in a text editor</li> <li>• Completing user input to an operation</li> <li>• Completing one step in a wizard (multipage dialog box)</li> </ul>
100 seconds	<ul style="list-style-type: none"> <li>• Critical decision in emergency situation</li> </ul>	<ul style="list-style-type: none"> <li>• Assure that all info required for decision is provided or can be found within this time</li> </ul>



audio-generation software should be engineered to avoid network bottlenecks, getting swapped out, deadlocks, and other interruptions. Otherwise, it may produce noticeable gaps, clicks, or lack of synchrony between audio tracks. Audio feedback and content should be provided by well-timed processes running with high priority and sufficient resources.

### 0.01 second (10 milliseconds)

“Subliminal” perception is rarely, if ever, used in interactive systems, so we needn’t concern ourselves with that issue. Suffice it to say that if designers wanted to boost the familiarity of certain visual symbols or images without the users’ awareness, the designers could do so by presenting the images or symbols repeatedly for 10 milliseconds at a time. It is also worth mentioning that while extremely brief exposure to an image can increase its familiarity, the effect is weak—certainly not strong enough to make people like or dislike specific products.

One way for software to generate tones is by sounding clicks at various rates. If the clicks are less than 10 milliseconds apart, they will be heard as a single sustained buzz, in which the pitch is determined in part by the click frequency. Users will hear clicks as distinct if the clicks are separated by intervals of more than 10 milliseconds.

Systems that use stylus-based input should ensure that electronic “ink” does not lag behind the stylus by more than 10 milliseconds; otherwise users will notice the lag and be annoyed.

### 0.1 second (100 milliseconds)

If software waits longer than 0.1 second to show a response to a user’s action, the perception of cause and effect is broken: the software’s reaction will not seem to be a result of the user’s action. Therefore, onscreen buttons have 0.1 second to show they’ve been clicked; otherwise users will assume they missed and click again. This does not mean that buttons have to complete their *function* in 0.1 second—only that buttons must show that they have been *pressed* by that deadline.

The main design point about the flinch reflex is that interactive systems should not startle users and cause flinching. Other than that, the flinch reflex and its duration don’t seem very relevant to interactive system design. It is difficult to imagine beneficial uses of the flinch reflex in human-computer interaction, but one can imagine games with loud noises, joysticks with sudden tactile jolts, or three-dimensional virtual environments that cause their users to flinch under some circumstances, perhaps purposefully. For example, if a vehicle detects a pending collision, it could do something to make riders flinch in order to help protect them upon impact.

If an object the user is dragging or resizing lags more than 0.1 second behind the user’s pointer movements, users will have trouble placing or resizing the object as desired. Therefore, interactive systems should prioritize feedback for hand-eye coordination tasks so that the feedback never lags past this deadline. If meeting that deadline is unachievable, then the system should be designed so that the task does not require close hand-eye coordination.

If an operation will take more than a perceptual “moment” (0.1 second) to complete, it should display a busy indicator. If a busy indicator can be displayed within 0.1 second, it can double as the action acknowledgment. If not, the software’s response should come in two parts: a quick acknowledgment within 0.1 second, followed by a busy (or progress) indicator within 1 second. More guidance for displaying busy indicators is given below.

The brain can reorder events within this approximate time window before the events reach consciousness. Human speech is highly prone to such reordering if it occurs out of order. If you listen to several people talking and some people start talking just before the person before them has finished talking (within the time window), your brain automatically “untangles” the utterances so that you seem to hear them sequentially, without perceived overlap. Television and movies sometimes take advantage of this phenomenon to speed up conversations that normally would take too long.

We also regard 10 per second as an approximate minimum frame rate for perception of smooth animation, even though smooth animation really requires a rate more like 20 frames per second.

### 1.0 second

Because 1 second is the maximum gap expected in conversation, and because operating an interactive system is a form of conversation, interactive systems should avoid lengthy gaps in on their side of the conversation. Otherwise, the human user will wonder what is happening. Systems have about 1 second to either do what the user asked or indicate how long it will take. Otherwise, users get impatient.

If an operation will take more than a few seconds, a progress indicator is needed. Progress indicators are an interactive system’s way of keeping its side of the expected conversational protocol: “I’m working on the problem. Here’s how much progress I’ve made and an indication of how much more time it will take.” More guidelines for progress indicators are provided below.

One second is also the approximate minimum time a user needs to respond intentionally to an unanticipated event. Therefore, when information suddenly appears on the screen, designers can assume that users will take at least a second to react to it (unless it causes a flinch response; see above). That lag time can be useful in cases when the system needs to display an interactive object but cannot both render the object and make it interactive within 0.1 second. Instead, the system can display a “fake,” inactive version of the object, and then take its time (1 second) to fill in details and make the object fully interactive. Today’s computers can do a lot in 1 second.

### 10 seconds

Ten seconds is the approximate unit of time into which people usually break down their planning and execution of larger tasks. Examples of unit tasks: completing a single edit in a text editing application, entering a transaction into a bank account program, and executing a maneuver in an airplane dogfight. Software should support segmentation of tasks into these 10-second pieces.

Ten seconds is also roughly the amount of time users are willing to spend setting up “heavyweight” operations like file transfers or searches—if it takes any longer, users start to lose patience. Computing the result can then take longer if the system provides progress feedback.

Similarly, each step in a multipage “wizard” dialog box should have at most about 10 seconds of work for a user to do. If a step of a wizard takes significantly longer than 10 seconds to complete, it probably should be broken up into multiple smaller steps.

### 100 seconds (~1.5 minutes)

Interactive systems that support rapid critical decision making should be designed so that all the necessary information is either already in front of the decision maker or can be easily obtained with minimal browsing or searching. The best user interface for this type of situation is one in which users can obtain all crucial information simply by moving their eyes to where it is displayed<sup>2</sup> (Isaacs & Walendowski, 2001).

---

## ADDITIONAL GUIDELINES FOR ACHIEVING RESPONSIVE INTERACTIVE SYSTEMS

In addition to design guidelines specific to each of the consolidated human-computer interaction deadlines, there are general guidelines for achieving responsiveness in interactive systems.

### Use busy indicators

Busy indicators vary in sophistication. At the low end, we have simple, static wait-cursors (e.g., an hourglass). They provide very little information: only that the software is temporarily occupied and unavailable to the user for other operations.

Next, we have wait-animations. Some of these are animated wait-cursors, such as the MacOS rotating color wheel. Some wait-animations are not cursors but, rather, larger graphics elsewhere on the screen, such as the “downloading data” animations displayed by some Web browsers. Wait animations are more “friendly” to users than static wait-cursors because they show that the system is working, not crashed or hung up waiting for a network connection to open or data to unlock. Of course, busy animations should cycle in response to the actual computations they represent. Busy animations that are simply started by a function but run independently of it are not really busy animations: they keep running even when the process they represent has hung or crashed and thereby potentially misleading users.

A common excuse for not displaying a busy indicator is that the function is supposed to execute quickly and so doesn’t need to display one. But how quickly is “quickly”? What if the function doesn’t always execute quickly? What if the user has a slower computer than the developer, or one that is not optimally configured?

---

<sup>2</sup>Sometimes called “no-click” user interfaces.

What if the function tries to access data that is temporarily locked? What if the function uses network services and the network is hung or overloaded?

Software should display a busy indicator for *any* function that blocks further user actions while it is executing, even if the function normally executes quickly (e.g., in less than 0.1 second). This indicator can be very helpful to a user if for some reason the function gets bogged down or hung. Furthermore, it harms nothing: when the function executes at the normal speed, the busy indicator appears and disappears so quickly that users barely see it.

## Use progress indicators

Progress indicators are better than busy indicators because they let users see how much time remains. Again: the deadline for displaying a progress indicator is 1 second.

Progress indicators can be graphical (e.g., a progress bar), textual (e.g., a count of files yet to be copied), or a combination of graphical and textual. They greatly increase the perceived responsiveness of an application, even though they don't shorten the time to complete operations.

Progress indicators should be displayed for any operation that will take longer than a few seconds. The longer the operation, the more important they are. Many noncomputer devices provide progress indicators, so we often take them for granted. Elevators that don't show the elevator's progress toward your floor are annoying. Most people wouldn't like a microwave oven that didn't show the remaining cooking time.

Here are some guidelines for designing effective progress indicators (McInerney & Li, 2002):

- Show work remaining, not work completed. Bad: "3 files copied." Good: "3 of 4 files copied."
- Show total progress, not progress on the current step. Bad: "5 seconds left on this step." Good: "15 seconds left."
- To show the percentage of an operation that is complete, start at 1%, not 0%. Users worry if the bar stays at 0% for more than a second or two.
- Similarly, display 100% only very briefly at the end of an operation. If the progress bar stays at 100% for more than a second or two, users assume it's wrong.
- Show smooth, linear progress, not erratic bursts of progress.
- Use human-scale precision, not computer precision. Bad: "240 seconds." Good: "About 4 minutes."

## Delays between unit tasks are less bothersome than delays within unit tasks

Unit tasks are useful not only as a way of understanding how (and why) users break down large tasks. They also provide insight into when system response delays are most and least harmful or annoying.

During execution of a unit task, users keep their goal and necessary information in working memory or within their perceptual field. After they complete one unit task, before moving onto the next one, they relax a bit, and then pull the information needed for the next unit task into memory or into view.

Because unit tasks are intervals during which the content of working memory and the perceptual field must remain fairly stable, unexpected system delays *during* a unit task are particularly harmful and annoying. They can cause users to lose track of some or all of what they were doing. By contrast, system delays *between* unit tasks are not as harmful or annoying, even though they may slow the user's overall work rate.

This difference between the impact of system response delays during and between unit tasks is sometimes expressed in user interface design guidelines in terms of task "closure," as in the classic user interface design handbook *Human-Computer Interface Design Guidelines* (Brown, 1988):

---

**A key factor determining acceptable response delays is level of closure. ... A delay after completing a major unit of work may not bother a user or adversely affect performance. Delays between minor steps in a larger unit of work, however, may cause the user to forget the next planned steps. In general, actions with high levels of closure, such as saving a completed document to a file, are less sensitive to response time delays. Actions at the lowest levels of closure, such as typing a character and seeing it echoed on the display, are most sensitive to response time delays.**

---

Bottom line: If a system has to impose delays, it should do so between unit tasks, not during tasks.

### Display important information first

Interactive systems can appear to be operating fast by displaying important information first, then details and auxiliary information later. Don't wait until a display is fully rendered before letting users see it. Give users something to think about and act upon as soon as possible.

This approach has several benefits. It distracts users from the absence of the rest of the information and it fools them into believing that the computer did what they asked quickly. Research indicates that users prefer progressive results to progress indicators (Geelhoed, Toft, Roberts, & Hyland, 1995). Displaying results progressively lets users start planning their next unit task. Finally, because of the aforementioned minimum reaction time for users to respond intentionally to what they see, this approach buys at least one more second for the system to catch up before the user tries to do anything. Here are some examples:

- **Document editing software:** When you open a document, the software shows the first page as soon as it has it, rather than waiting until it has loaded the entire document.

- **Web or Database search function:** When you do a search, the application displays items as soon as it finds them, while continuing to search for more.

High-resolution images sometimes render slowly, especially in Web browsers. To decrease the perceived time for an image to render, the system can display the image quickly at low resolution and then re-render it at a higher resolution. Because the visual system processes images holistically, this appears faster than revealing a full-resolution image slowly from top to bottom (see Fig. 12.3). Exception: For text, rendering a page at low resolution first and then substituting a higher resolution version is *not* recommended: it annoys users (Geelhoed *et al.*, 1995).



**FIGURE 12.3**

If displaying images takes more than two seconds, display the whole image first at low-resolution (A), not at full resolution from the top down (B).

### Fake heavyweight computations during hand-eye coordination tasks

In interactive systems, some user actions require rapid successive adjustments—with hand-eye coordination—until the goal is achieved. Examples include scrolling through a document, moving a game character through a landscape, resizing a window, or dragging an object to a new position. If feedback lags behind user actions by more than 0.1 second, users will have trouble hitting their goal. When your system cannot update its display fast enough to meet this hand-eye-coordination deadline, provide lightweight simulated feedback until the goal is clear and then apply the real operation.

Graphics editors fake feedback when they provide rubberband outlines of objects that a user is trying to move or resize. Some document editing applications make quick-and-dirty changes to internal document data structures to represent the effect of user actions, and then straighten things out later.

### Work ahead

Work ahead of users when possible. Software can use periods of low load to precompute responses to high-probability requests. There will be periods of low load because the users are human. Interactive software typically spends a lot of time waiting for input from the user. Don't waste that time! Use it to prepare something the user will probably want. If the user never wants it, so what? The software did it in "free" time; it didn't take time away from anything else. Here are some examples of using background processing to work ahead of users:

- A text search function looks for the *next* occurrence of the target word while you look at the current one. When you tell the function to find the next occurrence of the word, it already has it and so it seems very fast.
- A document viewer renders the next page while you view the current page. When you ask to see the next page, it is already ready.

### Process user input according to priority, not the order in which it was received

The order in which tasks are done often matters. Blindly doing tasks in the order in which they were requested may waste time and resources or even create extra work. Interactive systems should look for opportunities to reorder tasks in their queue. Sometimes reordering tasks can make completing the entire set more efficient.

Airline personnel use nonsequential input processing when they walk up and down long check-in lines looking for people whose flights are leaving very soon so they can pull them out of line and get them checked in. In Web browsers, clicking the Back or Stop buttons or on a link *immediately* aborts the process of loading and displaying the current page. Given how long it can take to load and display a Web page, the ability to abort a page load is critical to user acceptance.







rotates, the software might not have time to render all details of each frame while maintaining smooth animation. In that case, it might, for example, save time by rendering the filename labels on each folder as black blobs instead of as text. When the user stops rotating the tree, it is again rendered in full detail. Most users don't even notice a degradation of the image during the movement, because they attribute their inability to read the labels to motion blur.

### Provide timely feedback even on the Web

Developers of Web applications may have dismissed the time deadlines presented above as pure fantasy.

It is true that meeting those deadlines on the Web is difficult—often impossible. However, it is also true that those deadlines are psychological time constants, wired into us by millions of years of evolution, governing our perception of responsiveness. They are not arbitrary targets that we can adjust at will to match the limitations of the Web or of any technology platform. If an interactive system does not meet those deadlines, even if it is a Web application, users *will* consider its responsiveness to be poor. That means most Web software has poor responsiveness. The question is: how can designers and developers maximize responsiveness on the Web? Here are some approaches:

- Minimize the size and number of images
- Provide quick-to-display thumbnail images or overviews, with ways to show details only as needed
- When the amount of data is too large or time-consuming to display all at once, design the system to give an overview of *all* the data, and allow users to drill down into specific parts of the data to the level of detail they need
- Style and lay out pages using Cascading Style Sheets (CSS) instead of presentational HTML, frames, or tables
- Use built-in browser components—e.g., error dialog boxes—instead of constructing them in HTML
- Download applets and scripts to the browser; use AJAX methods

---

## ACHIEVING RESPONSIVENESS IS IMPORTANT

By following the guidelines described in this chapter and additional responsiveness guidelines given in Johnson (2007), interaction designers and developers can create systems that meet human real-time deadlines, and that users therefore perceive as responsive.

However, the software industry must first recognize these facts about responsiveness:

- It is of great importance to users

- It is different from performance; responsiveness problems are not solvable merely by tuning performance or making hardware faster
- It is a *design* issue, not just an implementation issue

History shows that faster processors will not solve the problem. Today's personal computers are as fast as supercomputers were 30 years ago, yet people still wait for their computers and grumble about a lack of responsiveness. Ten years from now, when personal computers and electronic appliances are as powerful as *today's* most powerful supercomputers, responsiveness will still be an issue because the software of that day will demand much more from the machines and the networks connecting them. For example, whereas today's text and document editing applications do *spell-checking* in the background, future versions may well do Internet-based *fact-checking* in the background. Additionally, applications 10 years from now will probably be based upon these capabilities and technologies:

- Deductive reasoning
- Image recognition
- Real-time speech generation and recognition
- Downloading terabyte-sized files
- Wireless communication among dozens of household appliances
- Collation of data from thousands of remote databases
- Complex searches of the entire Web

The result will be systems that place a much heavier load on the computer than today's systems do. As computers grow more powerful, history shows that much of that power is eaten up by applications that demand ever more processing power. Therefore, despite ever-increasing performance, responsiveness will never disappear as an issue.

For design flaws (bloopers) that hurt responsiveness, principles for designing responsive systems, and more techniques for achieving responsiveness, see Johnson (2007).

---

## SUMMARY

In the Introduction, I stated that applying interaction design guidelines in real designs is not simple and mindless. Constraints happen and force tradeoffs. Sometimes designers have to violate one guideline to follow another one, so they must be able to determine which guideline takes precedence in that situation.

That's why interaction design is a *skill*, not something that anyone can do by following a recipe. Learning that skill amounts to learning not only what the design guidelines are but also how to recognize which rules to follow in each design situation.

The purpose of this book was to provide a brief background in the human perceptual and cognitive psychology that underlies interaction design guidelines. Now that you have that background knowledge, hopefully any user interface guidelines you have been following will make more sense—they should no longer seem like arbitrary edicts by some user interface guru. It should also now be clearer that the basis of all sets of user interface design guidelines (see the Appendix) is the same. Finally, you are now better equipped to interpret, trade off, and apply user interface design guidelines in real-world design situations.

---

## CAVEAT

Technology—especially computer technology—advances quickly. The state of the art of computer-based interactive systems changes so quickly that it is difficult to get a book out before some of the technologies and designs mentioned in it are obsolete.

On the other hand, the fundamentals of how people perceive, learn, and think do not change quickly. The basic operations of human perception and cognition

have remained fairly stable for tens—perhaps even hundreds—of thousands of years. In the long term, human perceptual and cognitive function will continue to evolve, but not in the time span during which this book will be in circulation. However, people already use technology to improve our perception, memory, and reasoning; that trend will continue. Thus, human perception and thinking *will* change in a matter of decades, as our tools proliferate and improve and our reliance on them increases.

On the *third* hand, humanity's *knowledge* of human perception and cognition is, like computer technology, advancing rapidly. The past 20 years, especially, have seen a tremendous surge in our understanding of how the human brain works, aided by research tools such as functional MRI, eye-tracking systems, and neural network simulations. This has allowed cognitive psychology to move beyond “black box” models that merely *predicted* behavior to ones that *explain* how the brain processes and stores information and produces behavior. In this book, I have tried to digest and present some of these exciting new findings because of their value to designers. I do this knowing that, like the state of the art of computer technology, the state of knowledge of human cognitive/perceptual psychology will continue to advance, possibly rendering some of what the book says obsolete. It is better for designers to proceed using mostly correct knowledge of how people perceive and think than to design with *no* knowledge.

# Well-known User Interface Design Rules

Here is a sampling of user interface design guidelines that have been published.

## NORMAN (1983A)

### Inferences from research

- Mode errors suggest the need for better feedback
- Description errors suggest the need for better system configuration
- Lack of consistency leads to errors
- Capture errors imply the need to avoid overlapping command sequences
- Activation issues suggest the importance of memory reminders
- People will make errors, so make the system insensitive to them

### Lessons

- **Feedback:** The state of the system should be clearly available to the user, ideally in a form that is unambiguous and that makes the set of options readily available so as to avoid mode errors.
- **Similarity of response sequences:** Different classes of actions should have quite dissimilar command sequences (or menu patterns) so as to avoid capture and description errors.
- **Actions should be reversible:** As much as possible and where both irreversible and of relatively high consequence, they should be difficult to do, thereby preventing unintentional performance.
- **Consistency of the system:** The system should be consistent in its structure and design of command so as to minimize memory problems in retrieving the operations.

---

### SHNEIDERMAN (1987); SHNEIDERMAN AND PLAISANT (2009)

- Strive for consistency
- Cater to universal usability
- Offer informative feedback
- Design task flows to yield closure
- Prevent errors
- Permit easy reversal of actions
- Make users feel they are in control
- Minimize short-term memory load

---

### NIELSEN AND MOLICH (1990)

- Consistency and standards
- Visibility of system status
- Match between system and real world
- User control and freedom
- Error prevention
- Recognition rather than recall
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Help users recognize, diagnose, and recover from errors
- Provide online documentation and help

---

### STONE *et al.* (2005)

- **Visibility:** First step to goal should be clear
- **Affordance:** Control suggests how to use it
- **Feedback:** Should be clear what happened or is happening
- **Simplicity:** As simple as possible and task-focused
- **Structure:** Content organized sensibly
- **Consistency:** Similarity for predictability
- **Tolerance:** Prevent errors, help recovery
- **Accessibility:** Usable by all intended users, despite handicap, access device, or environmental conditions

---

### JOHNSON (2007)

#### Principle 1 Focus on the users and their tasks, not on the technology

- Understand the users
- Understand the tasks
- Consider the context in which the software will function

## Principle 2 Consider function first, presentation later

- Develop a conceptual model

## Principle 3 Conform to the users' view of the task

- Strive for naturalness
- Use users' vocabulary, not your own
- Keep program internals inside the program
- Find the correct point on the power/complexity tradeoff

## Principle 4 Design for the common case

- Make common results easy to achieve
- Two types of "common": "how many users" vs. "how often"
- Design for core cases; don't sweat "edge" cases

## Principle 5 Don't complicate the users' task

- Don't give users extra problems
- Don't make users reason by elimination

## Principle 6 Facilitate learning

- Think "outside-in," not "inside-out"
- Consistency, consistency, consistency
- Provide a low-risk environment

## Principle 7 Deliver information, not just data

- Design displays carefully; get professional help
- The screen belongs to the user
- Preserve display inertia

## Principle 8 Design for responsiveness

- Acknowledge user actions instantly
- Let users know when software is busy and when it isn't
- Free users to do other things while waiting
- Animate movement smoothly and clearly
- Allow users to abort lengthy operations they don't want
- Allow users to estimate how much time operations will take
- Try to let users set their own work pace

**Principle 9 Try it out on users; then fix it**

- Test results can surprise even experienced designers
- Schedule time to correct problems found by tests
- Testing has two goals: informational and social
- There are tests for every time and purpose



# Bibliography

- Angier, N. (2008). Blind to change, even as it stares us in the face. *New York Times*. April 1, 2008. [www.nytimes.com/2008/04/01/science/01angi.html](http://www.nytimes.com/2008/04/01/science/01angi.html).
- Arons, B. (1992). A review of the cocktail party effect. *Journal of the American Voice I/O Society*, 12, 35-50.
- Apple Computer (2009). *Apple human interface guidelines*. developer.apple.com/mac/library/documentation/UserExperience/Conceptual/AppleHIGuidelines
- Barber, R., & Lucas, H. (1983). System response time, operator productivity, and job satisfaction. *Communications of the ACM*, 26(11), 972-986.
- Bays, P. M., & Husain, M. (2008). Dynamic shifts of limited working memory resources in human vision. *Science*, 321, 851-854.
- Beyer, H., & Holtzblatt, K. (1997). *Contextual design: A customer-centered approach to systems design*. Morgan-Kaufmann Publishers.
- Blauer, T. (2007). On the startle/flinch response. *Blauer tactical intro to the spear system: Flinching and the first two seconds of an ambush*. YouTube video: [www.youtube.com/watch?v=jk\\_Ai8qT2s4](http://www.youtube.com/watch?v=jk_Ai8qT2s4).
- Broadbent, D. E. (1975). The magical number seven after fifteen years. In A. Kennedy & A. Wilkes (Eds.), *Studies in long-term memory* (pp. 3-18). London: Wiley.
- Brown, C. M. (1988). *Human-computer interface design guidelines*. Norwood, NJ: Ablex Publishing Corporation.
- Boulton, D. (2009). Cognitive science: The conceptual components of reading & what reading does for the mind. Interview of Dr. Keith Stanovich, Children of the Code website: [www.childrenofthecode.org/interviews/stanovich.htm](http://www.childrenofthecode.org/interviews/stanovich.htm)
- Card, S. (1996). Pioneers and settlers: Methods used in successful user interface design. In M. Rudisill, C. Lewis, P. Polson, & T. McKay (Eds.), *Human-computer interface design: Success cases, emerging methods, real-world context*. San Francisco: Morgan Kaufmann.
- Card, S., Moran, T., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Card, S., Robertson, G., & Mackinlay, J. (1991). *The Information Visualizer, an Information Workspace*. Proceedings of ACM CHI'91, 181-188.
- Carroll, J., & Rosson, M. (1984). Beyond MIPS: Performance is not quality. *BYTE*, 168-172.
- Cheriton, D. R. (1976). Man-machine interface design for time-sharing systems. *Proceedings of the ACM National Conference*, 362-380.
- Chi, E. H., Pirolli, P., Chen, K., & Pitkow, J. (2001). Using information scent to model user information needs and actions on the web. *Proceedings of ACM SIGCHI Conference on Computer-Human Interaction (CHI 2001)*, 490-497.
- Clark, A. (1998). *Being there: Putting brain, body, and world together again*. Cambridge, MA: MIT Press.
- Cowan, N., Chen, Z., & Rouders, J. (2004). Constant capacity in an immediate serial-recall task: A logical sequel to Miller (1956). *Psychological Science*, 15(9), 634-640.
- Duis, D., & Johnson, J. (1990). Improving user-interface responsiveness despite performance limitations. *Proceedings of IEEE CompCon'90*, 380-386.

- Geelgoed, E., Toft, P., Roberts, S., & Hyland, P. (1995). To influence time perception. *Proceedings of ACM CHI'95*, 5, 272-273.
- Grudin, J. (1989). The case against user interface consistency. *Communications of the ACM*, 32(10), 1164-1173.
- Hackos, J., & Redish, J. (1998). *User and task analysis for interface design*. New York: Wiley.
- Isaacs, E., & Walendowski, A. (2001). *Designing from both sides of the screen: How designers and engineers can collaborate to build cooperative technology*. Indianapolis, Indiana: SAMS.
- Johnson, J. (1987). How faithfully should the electronic office simulate the real one? *SIGCHI Bulletin*, 21-25.
- Johnson, J. (1990). Modes in non-computer devices. *International Journal of Man-Machine Studies*, 32, 423-438.
- Johnson, J. (2007). *GUI bloopers 2.0: Common user interface design don'ts and dos*. San Francisco, CA: Morgan-Kaufmann Publishers.
- Johnson, J., & Henderson, A. (2002). Conceptual models: Begin by designing what to design. *Interactions*, 25-32.
- Johnson, J., Roberts, T., Verplank, W., Smith, D. C., Irby, C., Beard, M., & Mackey, K. (1989). The xerox star: A retrospective. *IEEE Computer*, September, 11-29.
- Jonides, J., Lewis, R. L., Nee, D. E., Lustig, C. A., Berman, M. G., & Moore, K. S. (2008). The mind and brain of short-term memory. *Annual Review of Psychology*, 59, 193-224.
- Koyani, S. J., Bailey, R. W., & Nall, J. R. (2006). *Research-based web design and usability guidelines*. US Department of Health and Human Services. Website: [usability.gov/pdfs/guidelines.html](http://usability.gov/pdfs/guidelines.html).
- Krug, S. (2005). *Don't make me think: A common sense approach to web usability* (2nd ed.). Indianapolis: New Riders Press.
- Lambert, G. (1984). A comparative study of system response time on program developer productivity. *IBM Systems Journal*, 23(1), 407-423.
- Landauer, T. K. (1986). How much do people remember? Some estimates of the quantity of learned information in long-term memory. *Cognitive Science*, 10, 477-493.
- Liang, P., Zhong, N., Lu, S., Liu, J., Yau, Y., Li, K., & Yang, Y. (2007). *The neural mechanism of human numerical inductive reasoning process: A combined ERP and fMRI study*. Berlin: Springer Verlag.
- Lindsay, P., & Norman, D. A. (1972). *Human information processing*. New York and London: Academic Press.
- Marcus, A. (1992). *Graphic design for electronic documents and user interfaces*. Reading, MA: Addison-Wesley.
- Marr, D. (1982). *Vision*. New York, NY: W. H. Freeman. p. 101, Figure 3-1, attributed to R. C. James.
- McInerney, P., & Li, J. (2002). Progress indication: Concepts, design, and implementation, July, IBM. *Developer Works*. Website: [www-128.ibm.com/developerworks/web/library/us-progind](http://www-128.ibm.com/developerworks/web/library/us-progind).
- Microsoft Corporation (2009), *Windows User Experience Interaction Guidelines*: <http://www.msdn.microsoft.com/en-us/library/aa511258.aspx>
- Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63, 81-97.

- Miller, R. (1968). Response time in man-computer conversational transactions. *Proceedings of IBM Fall Joint Computer Conference Vol. 33*, 267-277.
- Monti, M. M., Osherson, D. N., Martinez, M. J., & Parsons, L. M. (2007). Functional neuroanatomy of deductive inference: A language-independent distributed network. *NeuroImage*, 37(3), 1005-1016.
- Mullet, K., & Sano, D. (1994). *Designing visual interfaces: Communications oriented techniques*. Prentice-Hall.
- Nielsen, J. (2003). Information foraging: Why Google makes people leave your site faster, Alertbox, June 30, 2003.
- Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. *Proceedings of ACM CHI'90 Conference*. (Seattle, WA, 1-5 April), 249-256.
- Norman, D. A. (1983a). Design rules based on analysis of human error. *Communications of the ACM*, 26(4), 254-258.
- Norman, D. A. (1983b). Design principles for human-computer interfaces. In A. Janda (Ed.), *Proceedings of the CHI-83 conference on human factors in computing systems (Boston)*. New York: ACM. Reprinted in *Readings in Human-Computer Interaction*, ed. by Ronald M. Baecker and William A. S. Buxton. San Mateo, CA: Morgan-Kaufmann Press (1987).
- Norman, D. A., & Draper, S. W. (1986). *User centered system design: New perspectives on human-computer interaction*. Hillsdale, NJ: CRC.
- Oracle Corporation/Sun Microsystems (2001). *Java Look and Feel Design Guidelines*, 2nd ed. <http://www.java.sun.com/products/jlf/ed2/book/index.html>
- Perfetti, C., & Landesman, L. (2001). The truth about download time. Web article, User Interface Engineering, Jan 31, 2001, [http://uie.com/articles/download\\_time/](http://uie.com/articles/download_time/)
- Redish, G. (2007). *Letting go of the words: Writing web content that works*. San Francisco: Morgan-Kaufmann Publishers.
- Robertson, G., Card, S., & Mackinlay, J. (1989). The cognitive co-processor architecture for interactive user interfaces. *Proceedings of the ACM Conference on User Interface Software and Technology (UIST'89)*. November 1989, ACM Press, 10-18.
- Robertson, G., Card, S., & Mackinlay, J. (1993). Information visualization using 3D interactive animation. *Communications of the ACM*, 36(4), 56-71.
- Rushinek, A., & Rushinek, S. (1986). What makes users happy? *Communications of the ACM*, 29, 584-598.
- Sapolsky, R. M. (2002). *A primate's memoir: A neuroscientist's unconventional life among the Baboons*. New York: Scribner.
- Schneider, W., & Shiffrin, R. M. (1977). Controlled and automatic human information processing: 1. Detection, search, and attention. *Psychological Review*, 84, 1-66.
- Schrage, M. (2005). The password is fayleyure. *Technology Review, March*. [http://www.technologyreview.com/read\\_article.aspx?ch=specialsections&sc=security&id=16350](http://www.technologyreview.com/read_article.aspx?ch=specialsections&sc=security&id=16350).
- Shneiderman, B. (1984). Response time and display rate in human performance with computers. *ACM Computing Surveys*, 16(4), 265-285.
- Shneiderman, B. (1987). *Designing the user interface: Strategies for effective human-computer interaction* (1st ed.). Reading, MA: Addison-Wesley.

- Shneiderman, B., & Plaisant, C. (2009). *Designing the user interface: Strategies for effective human-computer interaction* (5th ed.). Reading, MA: Addison-Wesley.
- Simon, H. A. (1969). *The sciences of the artificial*. Cambridge, MA: MIT Press.
- Simons, D. J., & Levin, D. T. (1998). Failure to detect changes in people during a real-world interaction. *Psychonomic Bulletin and Review*, 5, 644-669.
- Smith, S. L., & Mosier, J. N. (1986). *Guidelines for designing user interface software*. Springfield, VA: National Technical Information Service. *Technical Report ESD-TR-86-278*.
- Soegaard, M. (2007). Gestalt Principles of form perception. Web article, Interaction-Design.org, [http://www.interaction-design.org/encyclopedia/gestalt\\_principles\\_of\\_form\\_perception.html](http://www.interaction-design.org/encyclopedia/gestalt_principles_of_form_perception.html)
- Sohn, E. (October 8, 2003). It's a math world for animals. *Science News for Kids*. <http://www.sciencenewsforkids.org/articles/20031008/Feature1.asp>.
- Sousa, D. A. (2005). *How the brain learns to read*. Thousand Oaks, CA: Corwin Press.
- Stafford, T., & Webb, M. (2005). *Mind hacks: Tips and tools for using your brain*. Sebastapol, CA: O'Reilly.
- Stone, D., Jarrett, C., Woodroffe, M., & Minocha, S. (2005). *User interface design and evaluation*. San Francisco: Morgan Kaufmann Publishers.
- Thadhani, A. (1981). Interactive user productivity. *IBM Systems Journal*, 20(4), 407-423.
- Thagard, P. (2002). *Coherence in thought and action*. Cambridge, MA: MIT Press.
- Waloszek, G. (2005). Vision and Visual Disabilities: An Introduction, SAP Design Guild. [http://www.sapdesignguild.org/editions/highlight\\_articles\\_01/vision\\_physiology.asp](http://www.sapdesignguild.org/editions/highlight_articles_01/vision_physiology.asp)
- Ware, C. (2008). *Visual thinking for design*. San Francisco, CA: Morgan-Kaufmann Publishers.
- Weinschenk, S. M. (2009). *Neuro web design: What makes them click?* Berkeley, CA: New Riders.
- Wolfmaier, T. (1999). Designing for the color-challenged: A challenge. ITG Publication, March 1999, 2.1. [http://www.internettg.org/newsletter/mar99/accessibility\\_color\\_challenged.html](http://www.internettg.org/newsletter/mar99/accessibility_color_challenged.html)

# Index

## A

Ambiguity, perceptual, 8

Attention

blink following recognition of object, 158

cleanup after goal achievement, 106–107

duration of unbroken attention to unit task, 159

external aids, 98–99

familiar path following, 102

relationship to short-term memory, 82

scent of information following towards goal,  
99–100, 100f, 101f

Automatic versus controlled processing, 123, 126

## B

Background noise, reading disruption, 41–42, 41f,  
42f

Brain

functional divisions, 119–120

impulsive behavior inhibition by frontal cortex,  
125

perceptual and cognitive temporal function,  
154–160

reading and functional imaging, 37–39, 38f

Broca's area, 38f, 39

Busy indicators, 164–165

## C

Calculations

difficulty, 124–130

user interface design implications, 130–131,  
130f, 132f

Capitalization, all-caps and reading disruption, 40f

Captcha, 42f

Centered text, reading disruption, 44–45, 44f, 45f

Closure principle, Gestalt theory, 17–18, 17f, 18f

Color blindness, 58–60, 60f

Color vision

color presentation and discrimination

external factors influencing discrimination,  
60–61

paleness, 56, 57f

patch size, 56, 57f, 58f

separation, 56, 57f

edge contrast versus brightness perception,  
55–56, 55f, 56f

guidelines for color use, 61–62, 61f, 62f, 63f

mechanisms, 53–55, 54f

Command-line user interface, see User interface,  
command line

Common fate principle, Gestalt theory, 22, 22f, 23f

Computer beep, message notification, 74

Conceptual model, 135–136

Cone

distribution across retina, 66f

sensitivity in vision, 54, 54f

Consistency, learning facilitation, 137–139,  
145–147

Consistency, placement, 9

Continuity principle, Gestalt theory, 15–16, 16f, 17f

Contrast, vision optimized for, 55

Conversation, gap length, 158

Cortex, frontal, 124–125

Current content, perception bias, 4–5, 4f, 5f

## D

Data-specific controls, 29, 29f

Deadlines, see Time-deadlines

Display, color discrimination effects, 60–61

## E

Editorial window, temporal resolution, 157–158

EEG, see Electroencephalography

Electroencephalography (EEG), 38

Error messages

peripheral vision problems, 69–72, 70f, 71f

symbol use, 72, 73f

Evaluation, thought cycle, 103–105

Execution, thought cycle, 103–105

Experience

learning from experience, 120–122, 121f

perception bias, 1–4, 1f, 2f, 3f

External cognitive aids, 98

## F

Figure/ground principle, Gestalt theory, 19–22,  
20f, 21f

Flinch reflex, temporal resolution, 156

fMRI, see Functional magnetic resonance imaging

fMRS, see Functional magnetic resonance

spectroscopy

Font, reading disruption

difficult typefaces, 40

tiny fonts, 41, 41f

Fovea, spatial resolution, 65–68, 66f, 67f, 68f  
 Functional magnetic resonance imaging (fMRI), 38  
 Functional magnetic resonance spectroscopy (fMRS), 38

## G

Gap, visual, 68  
 Generalization, 120–122  
 Gestalt theory of perception
 

- closure principle, 17–18, 17f, 18f
- combination of principles, 23, 24f
- common fate principle, 22, 22f, 23f
- continuity principle, 15–16, 16f, 17f
- figure/ground principle, 19–22, 20f, 21f
- overview, 11
- proximity principle, 11–13, 12f, 13f
- similarity principle, 14–15, 14f, 15f
- symmetry principle, 18, 18f, 19f

 Goals
 

- cleanup after achievement, 106–107
- focus with little attention on tools, 97–98
- perception bias, 5–8, 7f
- scent of information towards goal, 99–100, 100f, 101f
- thought cycle, 103–105

 Graphical user interface, see User Interface, graphical  
 Grayscale, use of in design, 60  
 Gulf of execution, 134

## H

Hearing, see Sound  
 Hierarchy, visual, 30

## I

Impulsive behavior, inhibition by frontal cortex, 125  
 Information scent, see Scent of information  
 Instructions, design implications of memory
 

- long-term memory, 93f
- short-term memory, 87–89, 89f

## K

Keystroke consistency, learning facilitation, 141–142

## L

Language, processing versus reading, 33–34  
 Learning
 

- facilitation

consistency, 137–139  
 keystroke consistency, 141–142  
 objects/actions analysis, 135–136  
 objects/actions matrix, 139–141, 140f, 141f  
 overview, 133–142  
 simplicity of concepts, 136–137, 137f, 138  
 task analysis, 135  
 vocabulary factors
 

- conceptual model, 146f, 147–148, 148f
- consistent terminology, 145–147, 146f
- familiar terminology, 143–145, 143f, 144f, 145f
- task-focused terminology, 142–143, 143f

 learning from experience, 120–122, 121f  
 performing learned actions, 122–124  
 user interface design implications, 130–131, 130f, 132f  
 Legal language, reading disruption, 39–40  
 Lexicon, 147–148  
 Long-term memory, see Memory

## M

Memory
 

- external aids, 98–99
- implications for UI design, 93
- long-term memory
  - design implications, 92–95, 93f, 94f, 94t
  - mechanisms, 80–81
  - test, 91
  - weaknesses
    - emotional influences, 91
    - errors, 90
    - retroactive alterations, 91
- short-term memory
  - characteristics, 82–86
  - design implications
    - instructions, 87–89, 89f
    - moded user interface, 86–87
    - search results, 87, 88f
  - mechanisms, 81–82
  - test, 85–86
  - short-term versus long-term, 79, 80f

 Moded user interface, 86–87  
 Motion
 

- perception of, 69
- use of, 75

## N

Numbers, structure in presentation, 28–29, 28f, 29f

**O**

Objects/actions analysis, learning facilitation, 135-136

Objects/actions matrix, learning facilitation, 139-141, 140f, 141f

**P**

Paleness, color presentation and discrimination, 56, 57f

Patch size, color presentation and discrimination, 56, 57f, 58f

**Perception****biases**

current content, 4-5, 4f, 5f

design implications, 8-9

experience, 1-4, 1f, 2f, 3f

goals, 5-8, 7f

color, see Color vision

Gestalt theory, see Gestalt theory of perception

Performance, definition, 152-153

**Peripheral vision**

computer interface problems, 69-72, 70f, 71f

functions, 68-69, 69f

message visibility

accessory techniques, 74-77, 75f, 76f

improvement, 72, 73f

motion sensitivity, 68-69

spatial resolution, 65-68, 66f, 67f, 68f

Pop-up message, error dialog box, 74, 75f, 76f

**Problem solving**

difficulty, 124-130

puzzles, 126-127, 132

technical problem requirements, 128-129

user interface design implications, 130-131,

130f, 132f

Progress indicators, 165

Proximity principle, Gestalt theory, 11-13, 12f, 13f

**R****Reading****disruption**

all-caps, 40f

background noise, 41-42, 41f, 42f

centered text, 44-45, 44f, 45f

combination of disruptors, 46, 46f

design implications, 46-47

**font**

difficult typefaces, 40

tiny fonts, 41, 41f

repetition, 43-44, 44f

vocabulary, 39-40

feature-driven versus context-driven, 35-37

illiteracy experience, 34, 35f

minimization in good design, 50

origins, 33

patterns of recognition, 34

skilled versus unskilled reading and functional

brain imaging, 37-39, 38f

software dialog boxes, 47-50, 48f

top-down reading, 36f

**Recall**

difficulty, 112-113

recognition comparison and user interface

design implications

authentication information and easy recall,

116-117, 117f

choose versus recall and type, 113-114, 113f

function visibility by popularity, 116

pictures to convey function, 114, 114f

thumbnail images, 115, 115f

visual cues, 116

**Recognition**

ease, 109-112, 110f, 111f, 112f

facial, 112

recall comparison and user interface design

implications

authentication information and easy recall,

116-117, 117f

choose versus recall and type, 113-114, 113f

function visibility by popularity, 116

pictures to convey function, 114, 114f

thumbnail images, 115, 115f

visual cues, 116

Red/green color blindness, 58-59, 59f

Repetition, reading disruption, 43-44, 44f

**Responsiveness**

definition, 152-153

design considerations

artificial feedback during eye-hand

coordination tasks, 168

busy indicators, 164-165

delays between tasks versus within tasks,

165-166

important information display, 166-167, 167f

progress indicators, 165

time scales

0.001 seconds, 160-162

0.01 seconds, 162

0.1 seconds, 162-163

1.0 seconds, 163

10 seconds, 163-164

100 seconds, 164

time-compliance monitoring, 169-170

timely feedback, 170

- Responsiveness (*Continued*)
- user input processing by priority, 168
  - working ahead of users, 168
- importance, 170-171
- perceptual and cognitive temporal function, 154-160
- time-deadlines of human-computer interactions, 160-164, 161t
- Retinal gap, 68, 68f
- Risk, effect on learning, 149
- Rod, distribution across retina, 66f
- S**
- Scent of information, following towards goal, 99-100, 100f, 101f
- Scripts and typefaces, hard to read, 40
- Search results, design implications of short-term memory, 87, 88f
- Separation, color presentation and discrimination, 56, 57f
- Short-term memory, see Memory
- Similarity principle, Gestalt theory, 14-15, 14f, 15f
- Simplicity of concepts, learning facilitation, 136-137, 137f, 138
- Software, dialog boxes, 47-50, 48f
- Sound, temporal resolution of perception, 155-156
- Structure
- data-specific controls, 29, 29f
  - examples, 25, 25f, 26f, 27f
  - Gestalt theory, see Gestalt theory of perception
  - long number presentation, 28-29, 28f, 29f
  - visual hierarchy creation, 30-31, 30f, 31f
- Subitizing, temporal resolution, 157
- Symmetry principle, Gestalt theory, 18, 18f, 19f
- T**
- Task analysis, learning facilitation, 135
- Terminology, see Vocabulary
- Text, see Reading
- Thought cycle, elements, 103-105
- Time-deadlines
- design considerations
    - busy indicators, 164-165
    - delays between tasks versus within tasks, 165-166
    - fake feedback during eye-hand coordination tasks, 168
    - important information display, 166-167, 167f
    - progress indicators, 165
    - time scales
      - 0.001 seconds, 160-162
      - 0.01 seconds, 162
      - 0.1 seconds, 162-163
      - 1.0 seconds, 163
      - 10 seconds, 163-164
      - 100 seconds, 164
    - time-compliance monitoring, 169-170
    - timely feedback, 170
    - user input processing by priority, 168
    - working ahead of users, 168
  - human-computer interactions, 160-164, 161t
  - perceptual and cognitive temporal function, 154-160
- Top-down reading, 36f
- U**
- User interface
- command line, 113
  - graphical (GUI), 113-114
- User interface design rules
- Johnson's principles, 176-178
  - Nielsen and Molich, 176
  - Norman, 175-176
  - Shneiderman and Plaisant, 176
  - Stone et al., 176
- V**
- Vision see Color vision; Perception; Peripheral vision
- Visual hierarchy, 30
- Visual stimulus, temporal resolution of perception, 156-157
- Visual structure, see Structure
- Visual-motor reaction time, 158
- Vocabulary
- learning facilitation
    - conceptual model, 146f, 147-148, 148f
    - consistent terminology, 145-147, 146f
    - familiar terminology, 143-145, 143f, 144f, 145f
    - task-focused terminology, 142-143, 143f
    - reading disruption, 39-40
- W**
- Web site
- message visibility
    - accessory techniques, 74-77, 75f, 76f
    - improvement, 72, 73f
    - peripheral vision problems, 69-72, 70f, 71f
    - search results and design implications of short-term memory, 87, 88f
  - Wernicke's area, 38, 38f
  - Wiggle, message notification, 75, 76f
  - Working memory, see Memory