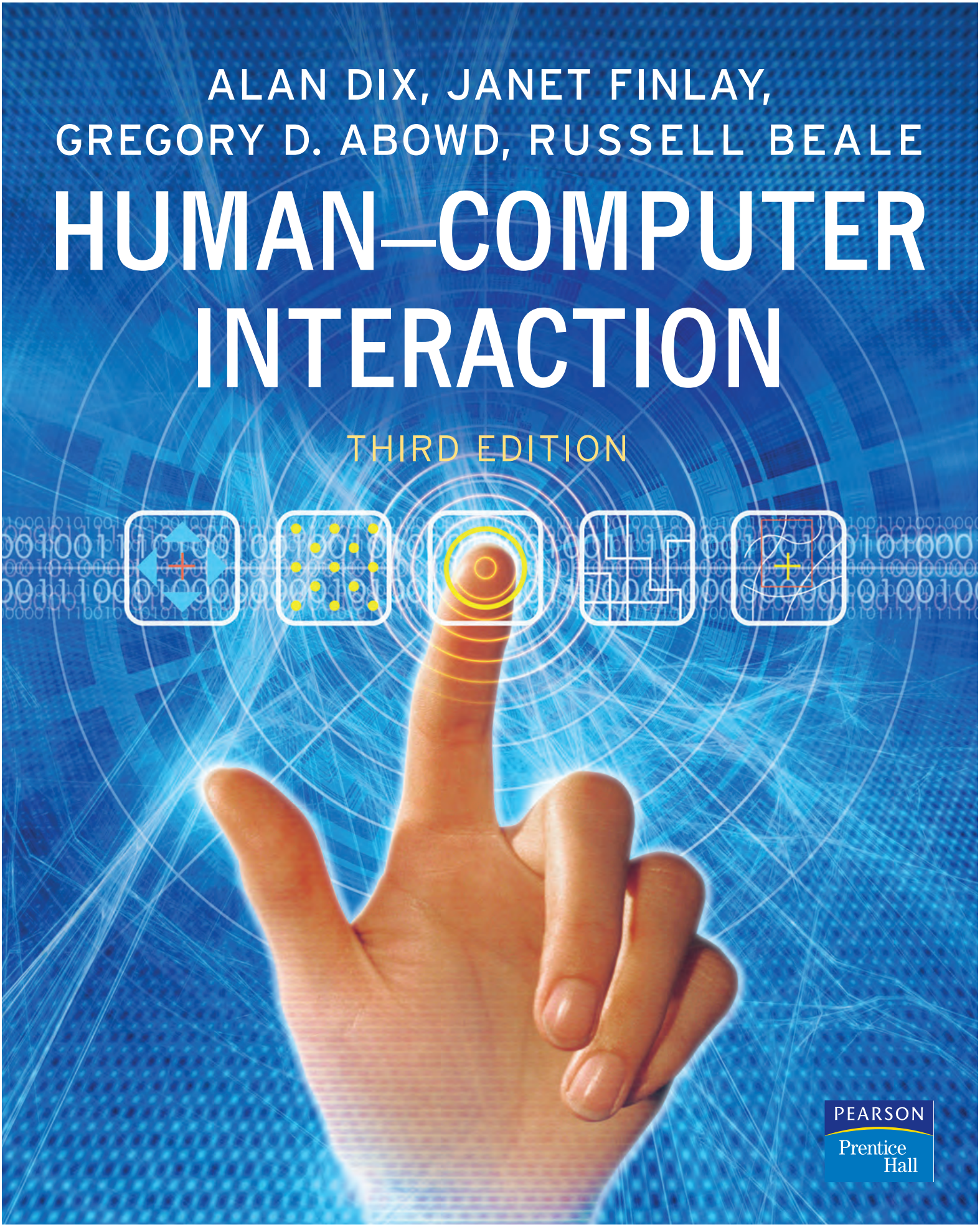
ALAN DIX, JANET FINLAY,
GREGORY D. ABOWD, RUSSELL BEALE

HUMAN–COMPUTER INTERACTION

THIRD EDITION



PEARSON
Prentice Hall

Human–Computer Interaction

We work with leading authors to develop the strongest educational materials in computing, bringing cutting-edge thinking and best learning practice to a global market.

Under a range of well-known imprints, including Prentice Hall, we craft high quality print and electronic publications which help readers to understand and apply their content, whether studying or at work.

To find out more about the complete range of our publishing, please visit us on the world wide web at: www.pearsoned.co.uk

Human–Computer Interaction

Third Edition

Alan Dix, *Lancaster University*

Janet Finlay, *Leeds Metropolitan University*

Gregory D. Abowd, *Georgia Institute of Technology*

Russell Beale, *University of Birmingham*

Harlow, England • London • New York • Boston • San Francisco • Toronto • Sydney • Singapore • Hong Kong
Tokyo • Seoul • Taipei • New Delhi • Cape Town • Madrid • Mexico City • Amsterdam • Munich • Paris • Milan

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

Visit us on the world wide web at:
www.pearsoned.co.uk

First published 1993
Second edition published 1998
Third edition published 2004

ISBN-13: 978-0-13-046109-4
ISBN-10: 0-13-046109-1

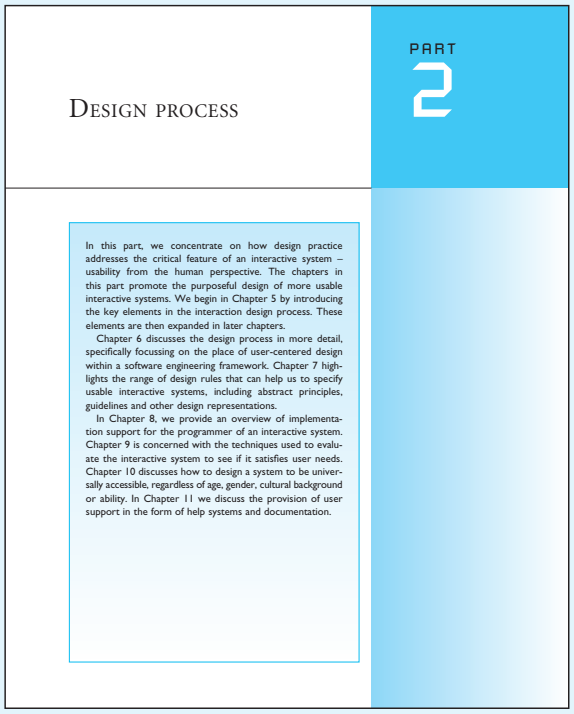
British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3
10 09 08 07 06

Typeset in 10/12½pt Minion by 35
Printed and bound by Scotprint, Haddington

Brief contents

Contents


PART 2

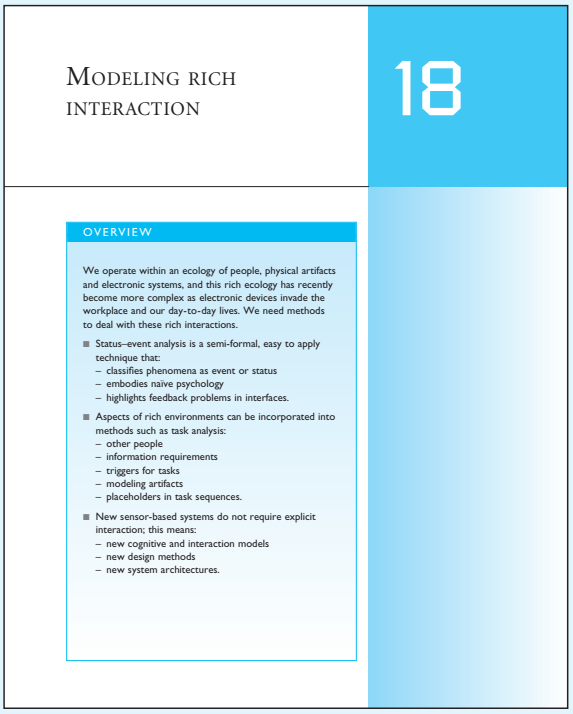
DESIGN PROCESS

In this part, we concentrate on how design practice addresses the critical feature of an interactive system – usability from the human perspective. The chapters in this part promote the purposeful design of more usable interactive systems. We begin in Chapter 5 by introducing the key elements in the interaction design process. These elements are then expanded in later chapters.

Chapter 6 discusses the design process in more detail, specifically focussing on the place of user-centered design within a sofware engineering framework. Chapter 7 highlights the range of design rules that can help us to specify usable interactive systems, including abstract principles, guidelines and other design representations.

In Chapter 8, we provide an overview of implementation support for the programmer of an interactive system. Chapter 9 is concerned with the techniques used to evaluate the interactive system to see if it satisfies user needs. Chapter 10 discusses how to design a system to be universally accessible, regardless of age, gender, cultural background or ability. In Chapter 11 we discuss the provision of user support in the form of help systems and documentation.

The part structure separates out introductory and more advanced material, with each part opener giving a simple description of what its constituent chapters cover


MODELING RICH INTERACTION

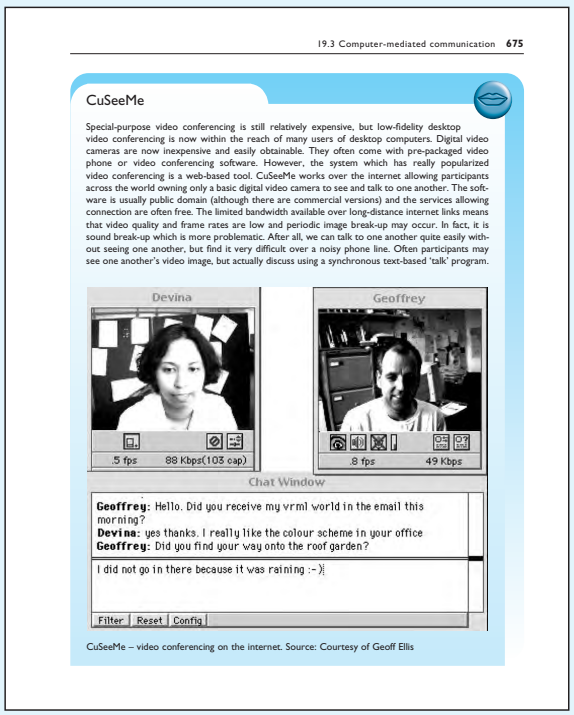
18

OVERVIEW

We operate within an ecology of people, physical artifacts and electronic systems, and this rich ecology has recently become more complex as electronic devices invade the workplace and our day-to-day lives. We need methods to deal with these rich interactions.

- Status–event analysis is a semi-formal, easy to apply technique that:
  - classifies phenomena as event or status
  - embodies naive psychology
  - highlights feedback problems in interfaces.
- Aspects of rich environments can be incorporated into methods such as task analysis:
  - other people
  - information requirements
  - triggers for tasks
  - modeling artifacts
  - placeholders in task sequences.
- New sensor-based systems do not require explicit interaction; this means:
  - new cognitive and interaction models
  - new design methods
  - new system architectures.

Bullet points at the start of each chapter highlight the core coverage

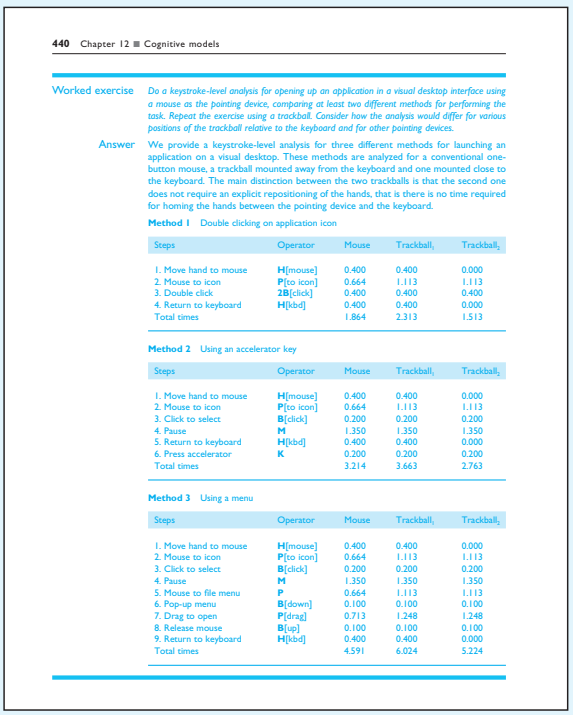


CuSeeMe

Special-purpose video conferencing is still relatively expensive, but low-fidelity desktop video conferencing is now within the reach of many users of desktop computers. Digital video cameras are now inexpensive and easily obtainable. They often come with pre-packaged video phone or video conferencing software. However, the system which has really popularized video conferencing is a web-based tool. CuSeeMe works over the internet allowing participants across the world owning only a basic digital video camera to see and talk to one another. The software is usually public domain (although there are commercial versions) and the services allowing connection are often free. The limited bandwidth available over long-distance internet links means that video quality and frame rates are low and periodic image break-up may occur. In fact, it is sound break-up which is more problematic. After all, we can talk to one another quite easily without seeing one another, but find it very difficult over a noisy phone line. Often participants may see one another's video image, but actually discuss using a synchronous text-based 'talk' program.

CuSeeMe – video conferencing on the internet. Source: Courtesy of Geoff Ellis

Boxed asides contain descriptions of particular tasks or technologies for additional interest, experimentation and discussion



Worked exercise *Do a keystroke-level analysis for opening up an application in a visual desktop interface using a mouse as the pointing device, comparing at least two different methods for performing the task. Repeat the exercise using a trackball. Consider how the analysis would differ for various positions of the trackball relative to the keyboard and for other pointing devices.*

Answer We provide a keystroke-level analysis for three different methods for launching an application on a visual desktop. These methods are analyzed for a conventional one-button mouse, a trackball mounted away from the keyboard and one mounted close to the keyboard. The main distinction between the two trackballs is that the second one does not require an explicit repositioning of the hands, that is there is no time required for homing the hands between the pointing device and the keyboard.

Method 1 Double clicking on application icon

Steps	Operator	Mouse	Trackball$_1$	Trackball$_2$
1. Move hand to mouse	**H**[mouse]	0.400	0.400	0.000
2. Mouse to icon	**P**[to icon]	0.664	1.113	1.113
3. Double click	**2B**[click]	0.400	0.400	0.400
4. Return to keyboard	**H**[kbd]	0.400	0.400	0.000
Total times		1.864	2.313	1.513

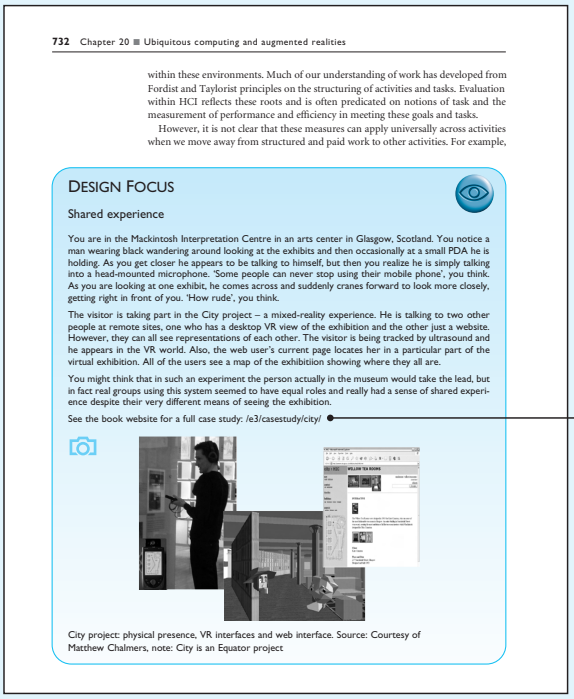
Method 2 Using an accelerator key

Steps	Operator	Mouse	Trackball$_1$	Trackball$_2$
1. Move hand to mouse	**H**[mouse]	0.400	0.400	0.000
2. Mouse to icon	**P**[to icon]	0.664	1.113	1.113
3. Click to select	**B**[click]	0.200	0.200	0.200
4. Pause	**M**	1.350	1.350	1.350
5. Return to keyboard	**H**[kbd]	0.400	0.400	0.000
6. Press accelerator	**K**	0.200	0.200	0.200
Total times		3.214	3.663	2.763

Method 3 Using a menu

Steps	Operator	Mouse	Trackball$_1$	Trackball$_2$
1. Move hand to mouse	**H**[mouse]	0.400	0.400	0.000
2. Mouse to icon	**P**[to icon]	0.664	1.113	1.113
3. Click to select	**B**[click]	0.200	0.200	0.200
4. Pause	**M**	1.350	1.350	1.350
5. Mouse to file menu	**P**	0.664	1.113	1.113
6. Pop-up menu	**B**[down]	0.100	0.100	0.100
7. Drag to open	**P**[drag]	0.713	1.248	1.248
8. Release mouse	**B**[up]	0.100	0.100	0.100
9. Return to keyboard	**H**[kbd]	0.400	0.400	0.000
Total times		4.591	6.024	5.224

Worked exercises within chapters provide step-by-step guidelines to demonstrate problem-solving techniques



732 Chapter 20 ■ Ubiquitous computing and augmented realities

within these environments. Much of our understanding of work has developed from Fordist and Taylorist principles on the structuring of activities and tasks. Evaluation within HCI reflects these roots and is often predicated on notions of task and the measurement of performance and efficiency in meeting these goals and tasks.

However, it is not clear that these measures can apply universally across activities when we move away from structured and paid work to other activities. For example,

DESIGN FOCUS

Shared experience

You are in the Mackintosh Interpretation Centre in an arts center in Glasgow, Scotland. You notice a man wearing black wandering around looking at the exhibits and then occasionally at a small PDA he is holding. As you get closer he appears to be talking to himself, but then you realize he is simply talking into a head-mounted microphone. 'Some people can never stop using their mobile phone', you think. As you are looking at one exhibit, he comes across and suddenly cranes forward to look more closely, getting right in front of you. 'How rude', you think.

The visitor is taking part in the City project – a mixed-reality experience. He is talking to two other people at remote sites, one who has a desktop VR view of the exhibition and the other just a website. However, they can all see representations of each other. The visitor is being tracked by ultrasound and he appears in the VR world. Also, the web user's current page locates her in a particular part of the virtual exhibition. All of the users see a map of the exhibition showing where they all are.

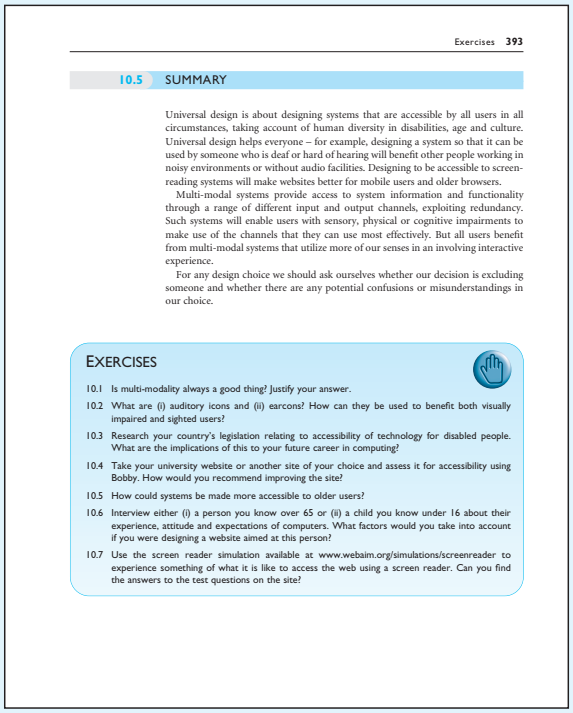
You might think that in such an experiment the person actually in the museum would take the lead, but in fact real groups using this system seemed to have equal roles and really had a sense of shared experience despite their very different means of seeing the exhibition.

See the book website for a full case study: /e3/casestudy/city/

City project: physical presence, VR interfaces and web interface. Source: Courtesy of Matthew Chalmers, note: City is an Equator project

Frequent links to the book website for further information

Design Focus mini case studies highlight practical applications of HCI concepts



Exercises 393

10.5 SUMMARY

Universal design is about designing systems that are accessible by all users in all circumstances, taking account of human diversity in disabilities, age and culture. Universal design helps everyone – for example, designing a system so that it can be used by someone who is deaf or hard of hearing will benefit other people working in noisy environments or without audio facilities. Designing to be accessible to screen-reading systems will make websites better for mobile users and older browsers.

Multi-modal systems provide access to system information and functionality through a range of different input and output channels, exploiting redundancy. Such systems will enable users with sensory, physical or cognitive impairments to make use of the channels that they can use most effectively. But all users benefit from multi-modal systems that utilize more of our senses in an involving interactive experience.

For any design choice we should ask ourselves whether our decision is excluding someone and whether there are any potential confusions or misunderstandings in our choice.

EXERCISES

10.1 Is multi-modality always a good thing? Justify your answer.

10.2 What are (i) auditory icons and (ii) earcons? How can they be used to benefit both visually impaired and sighted users?

10.3 Research your country's legislation relating to accessibility of technology for disabled people. What are the implications of this to your future career in computing?

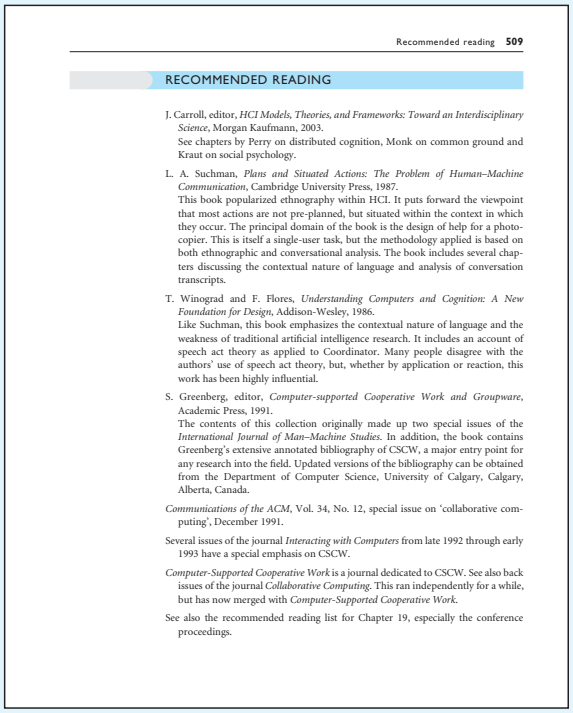
10.4 Take your university website or another site of your choice and assess it for accessibility using Bobby. How would you recommend improving the site?

10.5 How could systems be made more accessible to older users?

10.6 Interview either (i) a person you know over 65 or (ii) a child you know under 16 about their experience, attitude and expectations of computers. What factors would you take into account if you were designing a website aimed at this person?

10.7 Use the screen reader simulation available at www.webaim.org/simulations/screenreader to experience something of what it is like to access the web using a screen reader. Can you find the answers to the test questions on the site?

Chapter summaries reinforce student learning. Exercises at the end of chapters can be used by teachers or individuals to test understanding



Recommended reading 509

RECOMMENDED READING

J. Carroll, editor, *HCI Models, Theories, and Frameworks: Toward an Interdisciplinary Science*, Morgan Kaufmann, 2003.
See chapters by Perry on distributed cognition, Monk on common ground and Kraut on social psychology.

L. A. Suchman, *Plans and Situated Actions: The Problem of Human–Machine Communication*, Cambridge University Press, 1987.
This book popularized ethnography within HCI. It puts forward the viewpoint that most actions are not pre-planned, but situated within the context in which they occur. The principal domain of the book is the design of help for a photocopier. This is itself a single-user task, but the methodology applied is based on both ethnographic and conversational analysis. The book includes several chapters discussing the contextual nature of language and analysis of conversation transcripts.

T. Winograd and F. Flores, *Understanding Computers and Cognition: A New Foundation for Design*, Addison-Wesley, 1986.
Like Suchman, this book emphasizes the contextual nature of language and the weakness of traditional artificial intelligence research. It includes an account of speech act theory as applied to Coordinator. Many people disagree with the authors' use of speech act theory, but, whether by application or reaction, this work has been highly influential.

S. Greenberg, editor, *Computer-supported Cooperative Work and Groupware*, Academic Press, 1991.
The contents of this collection originally made up two special issues of the *International Journal of Man–Machine Studies*. In addition, the book contains Greenberg's extensive annotated bibliography of CSCW, a major entry point for any research into the field. Updated versions of the bibliography can be obtained from the Department of Computer Science, University of Calgary, Calgary, Alberta, Canada.

Communications of the ACM, Vol. 34, No. 12, special issue on 'collaborative computing', December 1991.

Several issues of the journal *Interacting with Computers* from late 1992 through early 1993 have a special emphasis on CSCW.

Computer-Supported Cooperative Work is a journal dedicated to CSCW. See also back issues of the journal *Collaborative Computing*. This ran independently for a while, but has now merged with *Computer-Supported Cooperative Work*.

See also the recommended reading list for Chapter 19, especially the conference proceedings.

Annotated further reading encourages readers to research topics in depth

Foreword

Human–computer interaction is a difficult endeavor with glorious rewards. Designing interactive computer systems to be effective, efficient, easy, and enjoyable to use is important, so that people and society may realize the benefits of computation-based devices. The subtle weave of constraints and their trade-offs – human, machine, algorithmic, task, social, aesthetic, and economic – generates the difficulty. The reward is the creation of digital libraries where scholars can find and turn the pages of virtual medieval manuscripts thousands of miles away; medical instruments that allow a surgical team to conceptualize, locate, and monitor a complex neuro-surgical operation; virtual worlds for entertainment and social interaction, responsive and efficient government services, from online license renewal to the analysis of parliamentary testimony; or smart telephones that know where they are and understand limited speech. Interaction designers create interaction in virtual worlds and embed interaction in physical worlds.

Human–computer interaction is a specialty in many fields, and is therefore multidisciplinary, but it has an intrinsic relationship as a subfield to computer science. Most interactive computing systems are for some human purpose and interact with humans in human contexts. The notion that computer science is the study of algorithms has virtue as an attempt to bring foundational rigor, but can lead to ignoring constraints foundational to the design of successful interactive computer systems. A lesson repeatedly learned in engineering is that a major source of failure is the narrow optimization of a design that does not take sufficient account of contextual factors. Human users and their contexts are major components of the design problem that cannot be wished away simply because they are complex to address. In fact, that largest part of program code in most interactive systems deals with user interaction. Inadequate attention to users and task context not only leads to bad user interfaces, it puts entire systems at risk.

The problem is how to take into account the human and contextual part of a system with anything like the rigor with which other parts of the system are understood and designed – how to go beyond fuzzy platitudes like 'know the user' that are true, but do not give a method for doing or a test for having done. This is difficult to do, but inescapable, and, in fact, capable of progress. Over the years, the need to take into account human aspects of technical systems has led to the creation of new fields of study: applied psychology, industrial engineering, ergonomics, human factors,

man–machine systems. Human–computer interaction is the latest of these, more complex in some ways because of the breadth of user populations and applications, the reach into cognitive and social constraints, and the emphasis on interaction. The experiences with other human-technical disciplines lead to a set of conclusions about how a discipline of human–computer interaction should be organized if it is to be successful.

First, design is where the action is. An effective discipline of human–computer interaction cannot be based largely on 'usability analysis', important though that may be. Usability analysis happens too late; there are too few degrees of freedom; and most importantly, it is not generative. Design thrives on understanding constraints, on insight into the design space, and on deep knowledge of the materials of the design, that is, the user, the task, and the machine. The classic landmark designs in human–computer interaction, such as the Xerox Star and the Apple Lisa/Macintosh, were not created from usability analysis (although usability analysis had important roles), but by generative principles for their designs by user interface designers who had control of the design and implementation.

Second, although the notion of 'user-centered design' gets much press, we should really be emphasizing 'task-centered design'. Understanding the purpose and context of a system is key to allocating functions between people and machines and to designing their interaction. It is only in deciding what a human–machine system should do and the constraints on this goal that the human and technical issues can be resolved. The need for task-centered design brings forward the need for methods of task analysis as a central part of system design.

Third, human–computer interaction needs to be structured to include both analytic and implementation methods together in the same discipline and taught together as part of the core. Practitioners of the discipline who can only evaluate, but not design and build are under a handicap. Builders who cannot reason analytically about the systems they build or who do not understand the human information processing or social contexts of their designs are under a handicap. Of course, there will be specialists in one or another part of human–computer interaction, but for there to be a successful field, there must be a common core.

Finally, what makes a discipline is a set of methods for doing something. A field must have results that can be taught and used by people other than their originators to do something. Historically, a field naturally evolves from a set of point results to a set of techniques to a set of facts, general abstractions, methods, and theories. In fact, for a field to be cumulative, there must be compaction of knowledge by crunching the results down into methods and theories; otherwise the field becomes fad-driven and a collection of an almost unteachably large set of weak results. The most useful methods and theories are generative theories: from some task analysis it is possible to compute some insightful property that constrains the design space of a system. In a formula: task analysis, approximation, and calculation. For example, we can predict that if a graphics system cannot update the display faster than 10 times/second then the illusion of animation will begin to break down. This constraint worked backwards has architectural implications for how to guarantee the needed display rate under variable computational load. It can be designed against.

This textbook, by Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale, represents how far human–computer interaction has come in developing and organizing technical results for the design and understanding of interactive systems. Remarkably, by the light of their text, it is pretty far, satisfying all the just-enumerated conclusions. This book makes an argument that by now there are many teachable results in human–computer interaction by weight alone! It makes an argument that these results form a cumulative discipline by its structure, with sections that organize the results systematically, characterizing human, machine, interaction, and the design process. There are analytic models, but also code implementation examples. It is no surprise that methods of task analysis play a prominent role in the text as do theories to help in the design of the interaction. Usability evaluation methods are integrated in their proper niche within the larger framework.

In short, the codification of the field of human–computer interaction in this text is now starting to look like other subfields of computer science. Students by studying the text can learn how to understand and build interactive systems. Human–computer interaction as represented by the text fits together with other parts of computer science. Moreover, human–computer interaction as presented is a challenge problem for advancing theory in cognitive science, design, business, or social-technical systems. Given where the field was just a few short years ago, the creation of this text is a monumental achievement. The way is open to reap the glorious rewards of interactive systems through a markedly less difficult endeavor, both for designer and for user.

Stuart K. Card
Palo Alto Research Center, Palo Alto, California

Preface to the third edition

It is ten years since the first edition of this book was published and much has changed. Ubiquitous computing and rich sensor-filled environments are finding their way out of the laboratory, not just into films and fiction, but also into our workplaces and homes. Now the computer really has broken its bounds of plastic and glass: we live in networked societies where personal computing devices from mobile phones to smart cards fill our pockets, and electronic devices surround us at home and at work. The web too has grown from a largely academic network into the hub of business and everyday lives. As the distinctions between physical and digital, work and leisure start to break down, human–computer interaction is also radically changing.

We have tried to capture some of the excitement of these changes in this revised edition, including issues of physical devices in Chapters 2 and 3, discussion of web interfaces in Chapter 21, ubiquitous computing in Chapters 4 and 20, and new models and paradigms for interaction in these new environments in Chapters 17 and 18. We have reflected aspects of the shift in use of technology from work to leisure in the analysis of user experience in Chapter 3, and in several of the boxed examples and case studies in the text. This new edition of *Human–Computer Interaction* is not just tracking these changes but looking ahead at emerging areas.

However, it is also rooted in strong principles and models that are not dependent on the passing technologies of the day. We are excited both by the challenges of the new and by the established foundations, as it is these foundations that will be the means by which today's students understand tomorrow's technology. So we make no apology for continuing the focus of previous editions on the theoretical and conceptual models that underpin our discipline. As the use of technology has changed, these models have expanded. In particular, the insular individual focus of early work is increasingly giving way to include the social and physical context. This is reflected in the expanded treatment of social and organizational analysis, including ethnography, in Chapter 13, and the analysis of artifacts in the physical environment in Chapter 18.

STRUCTURE

The structure of the new edition has been completely revised. This in part reflects the growth of the area: ten years ago HCI was as often as not a minority optional subject, and the original edition was written to capture the core material for a standard course. Today HCI is much expanded: some areas (like CSCW) are fully fledged disciplines in their own right, and HCI is studied from a range of perspectives and at different levels of detail. We have therefore separated basic material suitable for introductory courses into the first two parts, including a new chapter on interaction design, which adds new material on scenarios and navigation design and provides an overview suitable for a first course. In addition, we have included a new chapter on universal design, to reflect the growing emphasis on design that is inclusive of all, regardless of ability, age or cultural background. More advanced material focussing on different HCI models and theories is presented in Part 3, with extended coverage of social and contextual models and rich interaction. It is intended that these sections will be suitable for more advanced HCI courses at undergraduate and postgraduate level, as well as for researchers new to the field. Detailed coverage of the particular domains of web applications, ubiquitous computing and CSCW is given in Part 4.

New to this edition is a full color plate section. Images flagged with a camera icon in the text can be found in color in the plate section.

WEBSITE AND SUPPORT MATERIALS

We have always believed that support materials are an essential part of a textbook of this kind. These are designed to supplement and enhance the printed book – physical and digital integration in practice. Since the first edition we have had exercises, mini-case studies and presentation slides for all chapters available electronically. For the second edition these were incorporated into a website including links and an online search facility that acts as an exhaustive index to the book and mini-encyclopedia of HCI. For visually disabled readers, access to a full online electronic text has also been available. The website is continuing to develop, and for the third edition provides all these features plus more, including WAP search, multi-choice questions, and extended case study material (see also color plate section). We will use the book website to bring you new exercises, information and other things, so do visit us at www.hcibook.com (also available via www.booksites.net/dix). Throughout the book you will find shorthand web references of the form /e3/a-page-url/. Just prepend http://www.hcibook.com to find further information. To assist users of the second edition, a mapping between the structures of the old and new editions is available on the web at: http://www.hcibook.com/e3/contents/map2e/

STYLISTIC CONVENTION

As with all books, we have had to make some global decisions regarding style and terminology. Specifically, in a book in which the central characters are 'the user' and 'the designer', it is difficult to avoid the singular pronoun. We therefore use the pronoun 'he' when discussing the user and 'she' when referring to the designer. In other cases we use 'she' as a generic term. This should not be taken to imply anything about the composition of any actual population.

Similarly, we have adopted the convention of referring to the field of 'Human–Computer Interaction' and the notion of 'human–computer interaction'. In many cases we will also use the abbreviation HCI.

ACKNOWLEDGEMENTS

In a book of this size, written by multiple authors, there will always be myriad people behind the scenes who have aided, supported and abetted our efforts. We would like to thank all those who provided information, pictures and software that have enhanced the quality of the final product. In particular, we are indebted to Wendy Mackay for the photograph of EVA; Wendy Hall and her colleagues at the University of Southampton for the screen shot of Microcosm; Saul Greenberg for the reactive keyboard; Alistair Edwards for Soundtrack; Christina Engelbart for the photographs of the early chord keyset and mouse; Geoff Ellis for the screen shot of Devina and himself using CuSeeMe; Steve Benford for images of the Internet Foyer; and Tony Renshaw who provided photographs of the eye tracking equipment. Thanks too to Simon Shum for information on design rationale, Robert Ward who gave us material on psycho-physiology, and Elizabeth Mynatt and Tom Rodden who worked with Gregory on material adapted in Chapter 20. Several of the boxed case studies are based on the work of multi-institution projects, and we are grateful to all those from the project teams of CASCO, thePooch SMART-ITS, TOWER, AVATAR-Conference and TEAM-HOS for boxes and case studies based on their work; and also to the EQUATOR project from which we drew material for the boxes on cultural probes, 'Ambient Wood' and 'City'. We would also like to thank all the reviewers and survey respondents whose feedback helped us to select our subject matter and improve our coverage; and our colleagues at our respective institutions and beyond who offered insight, encouragement and tolerance throughout the revision. We are indebted to all those who have contributed to the production process at Pearson Education and elsewhere, especially Keith Mansfield, Anita Atkinson, Lynette Miller, Sheila Chatten and Robert Chaundy.

Personal thanks must go to Fiona, Esther, Miriam, Rachel, Tina, Meghan, Aidan and Blaise, who have all endured 'The Book' well beyond the call of duty and over

many years, and Bruno and 'the girls' who continue to make their own inimitable contribution.

Finally we all owe huge thanks to Fiona for her continued deep personal support and for tireless proofreading, checking of figures, and keeping us all moving. We would never have got beyond the first edition without her.

The efforts of all of these have meant that the book is better than it would otherwise have been. Where it could still be better, we take full responsibility.

Publisher's acknowledgements

We are grateful to the following for permission to reproduce copyright material:

Figure p. 2, Figures 3.14, 3.15, 3.16 and 5.13 and Exercise 8.4 screen shots reprinted by permission from Apple Computer, Inc.; Figure 2.11 reprinted by permission of Keith Cheverst; Figure 3.13 from The WebBook and Web Forager: An information workspace for the world-wide web in *CHI Conference Proceedings*, © 1996 ACM, Inc., reprinted by permission (Card, S. K., Robertson, G. G. and York, W. 1996); Figures 3.9, 3.19, 5.5, Chapter 14, Design Focus: Looking real – Avatar Conference screen shots, Figures 21.3, 21.10, 21.11 screen shot frames reprinted by permission from Microsoft Corporation; Tables 6.2 and 6.3 adapted from Usability engineering: our experience and evolution in *Handbook for Human–Computer Interaction* edited by M. Helander, Copyright 1988, with permission from Elsevier (Whiteside, J., Bennett, J. and Hotzblatt, K. 1988); Figure 7.1 adapted from The alternate reality kit – an animated environment for creating interactive simulations in *Proceedings of Workshop on Visual Languages*, © 1986 IEEE, reprinted by permission of IEEE (Smith, R. B. 1986); Figure 7.2 from Guidelines for designing user interface software in *MITRE Corporation Report MTR-9420*, reprinted by permission of The MITRE Corporation (Smith, S. L. and Mosier, J. N. 1986); Figure 7.3 reprinted by permission of Jenifer Tidwell; Figures 8.6 and 8.9 from *Xview Programming Manual*, Volume 7 of *The X Window System*, reprinted by permission of O'Reilly and Associates, Inc. (Heller, D. 1990); Figure 9.8 screen shot reprinted by permission of Dr. R. D. Ward; Figure 10.2 after Earcons and icons: their structure and common design principles in *Human-Computer Interaction*, 4(1), published and reprinted by permission of Lawrence Erlbaum Associates, Inc. (Blattner, M., Sumikawa, D. and Greenberg, R. 1989); Figure 10.5 reprinted by permission of Alistair D. N. Edwards; Figure 10.7 reprinted by permission of Saul Greenberg; Figure 11.2 screen shot reprinted by permission of Macromedia, Inc.; Table 12.1 adapted from *The Psychology of Human Computer Interaction*, published and reprinted by permission of Lawrence Erlbaum Associates, Inc. (Card, S. K., Moran, T. P. and Newell, A. 1983); Table 12.2 after Table in A comparison of input devices in elemental pointing and dragging tasks in *Reaching through technology – CHI'91 Conference Proceedings, Human Factors in Computing Systems*, April, edited by S. P. Robertson, G. M. Olson and J. S. Olson, © 1991 ACM, Inc., reprinted by permission (Mackenzie,

I. S., Sellen, A. and Buxton, W. 1991); Figure 14.1 from *Understanding Computers and Cognition: A New Foundation for Design*, published by Addison-Wesley, reprinted by permission of Pearson Education, Inc. (Winograd, T. and Flores, F. 1986); Figure 14.5 from *Theories of multi-party interaction. Technical report*, Social and Computer Sciences Research Group, University of Surrey and Queen Mary and Westfield Colleges, University of London, reprinted by permission of Nigel Gilbert (Hewitt, B., Gilbert, N., Jirotka, M. and Wilbur, S. 1990); Figure 14.6 from *Dialogue processes in computer-mediated communication: a study of letters in the com system. Technical report,* Linköping Studies in Arts and Sciences, reprinted by permission of Kerstin Severinson Eklundh (Eklundh, K. S. 1986); Chapter 14, Design Focus: Looking real – Avatar Conference, screen shots reprinted by permission of AVATAR-Conference project team; Figure 16.17 screen shot reprinted by permission of Harold Thimbleby; Figure 17.5 based on Verifying the behaviour of virtual world objects in *DSV-IS 2000 Interactive Systems: Design, Specification and Verification.* LNCS 1946, edited by P. Palanque and F. Paternò, published and reprinted by permission of Spinger-Verlag GmbH & Co. KG (Willans, J. S. and Harrison, M. D. 2001); Figure 18.4 icons reprinted by permission of Fabio Paternò; Chapter 19, p.675 CuSeeMe screen shot reprinted by permission of Geoff Ellis; Chapter 19, Design Focus: TOWER – workspace awareness, screen shots reprinted by permission of Wolfgang Prinz; Figure 20.1 reprinted by permission of Mitsubishi Electric Research Laboratories, Inc.; Figure 20.4 (right) reprinted by permission of Sony Computer Science Laboratories, Inc; Figure 20.9 from Cone trees. Animated 3d visualisation of hierarchical information in *Proceedings of the CH'91 Conference of Human Factors in Computing Systems*, © 1991 ACM, Inc., reprinted by permission (Robertson, G. G., Card, S. K., and Mackinlay, J. D. 1991); Figure 20.10 from Lifelines: visualising personal histories in *Proceedings of CH'96*, © 1996 ACM, Inc., reprinted by permission (Plaisant, C., Milash, B., Rose, A., Widoff, S. and Shneiderman, B. 1996); Figure 20.11 from Browsing anatomical image databases: a case study of the Visible Human in *CH'96 Conference Companion*, © 1996 ACM, Inc., reprinted by permission (North, C. and Korn, F. 1996); Figure 20.12 from Externalising abstract mathematical models in *Proceedings of CH'96*, © 1996 ACM, Inc., reprinted by permission (Tweedie, L., Spence, R., Dawkes, H. and Su, H. 1996); Figure 21.2 from The impact of Utility and Time on Distributed Information Retrieval in *People and Computers XII: Proceedings of HCI'97*, edited by H. Thimbleby, B. O'Conaill and P. Thomas, published and reprinted by permission of Spinger-Verlag GmbH & Co. KG (Johnson, C. W. 1997); Figure 21.4 screen shot reprinted by permission of the Departments of Electronics and Computer Science and History at the University of Southampton; Figure 21.6 Netscape browser window © 2002 Netscape Communications Corporation. Used with permission. Netscape has not authorized, sponsored, endorsed, or approved this publication and is not responsible for its content.

We are grateful to the following for permission to reproduce photographs:

Chapter 1, p. 50, Popperfoto.com; Chapter 2, p. 65, PCD Maltron Ltd; Figure 2.2 Electrolux; Figures 2.6 and 19.6 photos courtesy of Douglas Engelbart and Bootstrap Institute; Figure 2.8 (left) British Sky Broadcasting Limited; Figure 2.13 (bottom

right) Sony (UK) Ltd; Chapter 2, Design Focus: Feeling the Road, BMW AG; Chapter 2, Design Focus: Smart-Its – making using sensors easy, Hans Gellersen; Figures 4.1 (right) and 20.2 (left) Palo Alto Research Center; Figure 4.2 and 20.3 (left) François Guimbretière; Figure 4.3 (bottom left) Franklin Electronic Publishers; Figure 5.2 (top plate and middle plate) Kingston Museum and Heritage Service, (bottom plate) V&A Images, The Victoria and Albert Museum, London; Chapter 5, Design Focus: Cultural probes, William W. Gaver, Anthony Boucher, Sarah Pennington and Brendan Walker, Equator IRC, Royal College of Art; Chapter 6, p. 245, from The 1984 Olympic Message System: a text of behavioural principle of system design in *Communications of the ACM*, 30(9), © 1987 ACM, Inc., reprinted by permission (Gould, J. D., Boies, S. J., Levy, S., Richards, J. T. and Schoonard, J. 1987); Figures 9.5 and 9.6 J. A. Renshaw; Figure 9.7 Dr. R. D. Ward; Figure 10.3 SensAble Technologies; Chapter 13, Design Focus: Tomorrow's hospital – using participatory design, Professor J. Artur Vale Serrano; Chapter 18, p. 650, Michael Beigl; Chapter 19, p. 678, Steve Benford, The Mixed Reality Laboratory, University of Nottingham; Chapter 19, Design Focus: SMS in action, Mark Rouncefield; Figure 20.2 (right) Ken Hinckley; Figure 20.3 (right) MIT Media Lab; Figure 20.4 (left) from Interacting with paper on the digital desk in *Communications of the ACM*, 36(7), © 1993 ACM, Inc., reprinted by permission (Wellner, P. 1993); Chapter 20, p. 726, Peter Phillips; Chapter 20, Design Focus: Ambient wood – augmenting the physical, Yvonne Rogers; Chapter 20, Design Focus: Shared experience, Matthew Chalmers.

We are grateful to the following for permission to reproduce text extracts:

Pearson Education, Inc. Publishing as Pearson Addison Wesley for an extract adapted from *Designing the User Interface: Strategies for Effective Human–Computer Interaction 3/e* by B. Shneiderman © 1998, Pearson Education, Inc; Perseus Books Group for an extract adapted from *The Design of Everyday Things* by D. Norman, 1998; and Wiley Publishing, Inc. for extracts adapted from 'Heuristic Evaluation' by Jakob Nielson and Robert L. Mack published in *Usability Inspection Methods* © 1994 Wiley Publishing, Inc.; IEEE for permission to base chapter 20 on 'The human experience' by Gregory Abowd, Elizabeth Mynatt and Tom Rodden which appeared in *IEEE Pervasive Computing Magazine*, Special Inaugural Issue on Reaching for Weiser's Vision, Vol. 1, Issue 1, pp. 48–58, Jan–March 2002. © 2002 IEEE.

In some instances we have been unable to trace the owners of copyright material, and we would appreciate any information that would enable us to do so.

Introduction

WHY HUMAN–COMPUTER INTERACTION?

In the first edition of this book we wrote the following:

> This is the authors' second attempt at writing this introduction. Our first attempt fell victim to a design quirk coupled with an innocent, though weary and less than attentive, user. The word-processing package we originally used to write this introduction is menu based. Menu items are grouped to reflect their function. The 'save' and 'delete' options, both of which are correctly classified as file-level operations, are consequently adjacent items in the menu. With a cursor controlled by a trackball it is all too easy for the hand to slip, inadvertently selecting delete instead of save. Of course, the delete option, being well thought out, pops up a confirmation box allowing the user to cancel a mistaken command. Unfortunately, the save option produces a very similar confirmation box – it was only as we hit the 'Confirm' button that we noticed the word 'delete' at the top . . .

Happily this word processor no longer has a delete option in its menu, but unfortunately, similar problems to this are still an all too common occurrence. Errors such as these, resulting from poor design choices, happen every day. Perhaps they are not catastrophic: after all nobody's life is endangered nor is there environmental damage (unless the designer happens to be nearby or you break something in frustration!). However, when you lose several hours' work with no written notes or backup and a publisher's deadline already a week past, 'catastrophe' is certainly the word that springs to mind.

Why is it then that when computers are marketed as 'user friendly' and 'easy to use', simple mistakes like this can still occur? Did the designer of the word processor actually try to use it with the trackball, or was it just that she was so expert with the system that the mistake never arose? We hazard a guess that no one tried to use it when tired and under pressure. But these criticisms are not levied only on the designers of traditional computer software. More and more, our everyday lives involve programmed devices that do not sit on our desk, and these devices are just as unusable. Exactly how many VCR designers understand the universal difficulty people have trying to set their machines to record a television program? Do car radio designers

actually think it is safe to use so many knobs and displays that the driver has to divert attention away from the road completely in order to tune the radio or adjust the volume?

Computers and related devices have to be designed with an understanding that people with specific tasks in mind will want to use them in a way that is seamless with respect to their everyday work. To do this, those who design these systems need to know how to think in terms of the eventual users' tasks and how to translate that knowledge into an executable system. But there is a problem with trying to teach the notion of designing computers for people. All designers *are* people and, most probably, they are users as well. Isn't it therefore intuitive to design for the user? Why does it need to be taught when we all know what a good interface looks like? As a result, the study of human–computer interaction (HCI) tends to come late in the designer's training, if at all. The scenario with which we started shows that this is a mistaken view; it is not at all intuitive or easy to design consistent, robust systems

DESIGN FOCUS

Things don't change

It would be nice to think that problems like those described at the start of the Introduction would never happen now. Think again! Look at the MacOS X 'dock' below. It is a fast launch point for applications; folders and files can be dragged there for instant access; and also, at the right-hand side, there sits the trash can. Imagine what happens as you try to drag a file into one of the folders. If your finger accidentally slips whilst the icon is over the trash can – oops!

Happily this is not quite as easy in reality as it looks in the screen shot, since the icons in the dock constantly move around as you try to drag a file into it. This is to make room for the file in case you want to place it in the dock. However, it means you have to concentrate very hard when dragging a file over the dock. We assume this is not a deliberate feature, but it does have the beneficial side effect that users are less likely to throw away a file by accident – whew!

In fact it is quite fun to watch a new user trying to throw away a file. The trash can keeps moving as if it didn't want the file in it. Experienced users evolve coping strategies. One user always drags files into the trash from the right-hand side as then the icons in the dock don't move around. So two lessons:

- designs don't always get better
- but at least users are clever.

Screen shot reprinted by permission from Apple Computer, Inc.

that will cope with all manner of user carelessness. The interface is not something that can be plugged in at the last minute; its design should be developed integrally with the rest of the system. It should not just present a 'pretty face', but should support the tasks that people actually want to do, and forgive the careless mistakes. We therefore need to consider how HCI fits into the design process.

Designing usable systems is not simply a matter of altruism towards the eventual user, or even marketing; it is increasingly a matter of law. National health and safety standards constrain employers to provide their workforce with usable computer systems: not just safe but *usable*. For example, EC Directive 90/270/EEC, which has been incorporated into member countries' legislation, requires employers to ensure the following when designing, selecting, commissioning or modifying software:

- that it is suitable for the task
- that it is easy to use and, where appropriate, adaptable to the user's knowledge and experience
- that it provides feedback on performance
- that it displays information in a format and at a pace that is adapted to the user
- that it conforms to the 'principles of software ergonomics'.

Designers and employers can no longer afford to ignore the user.

WHAT IS HCI?

The term *human–computer interaction* has only been in widespread use since the early 1980s, but has its roots in more established disciplines. Systematic study of human performance began in earnest at the beginning of the last century in factories, with an emphasis on manual tasks. The Second World War provided the impetus for studying the interaction between humans and machines, as each side strove to produce more effective weapons systems. This led to a wave of interest in the area among researchers, and the formation of the Ergonomics Research Society in 1949. Traditionally, ergonomists have been concerned primarily with the physical characteristics of machines and systems, and how these affect user performance. Human Factors incorporates these issues, and more cognitive issues as well. The terms are often used interchangeably, with Ergonomics being the preferred term in the United Kingdom and Human Factors in the English-speaking parts of North America. Both of these disciplines are concerned with user performance in the context of any system, whether computer, mechanical or manual. As computer use became more widespread, an increasing number of researchers specialized in studying the interaction between people and computers, concerning themselves with the physical, psychological and theoretical aspects of this process. This research originally went under the name *man–machine interaction*, but this became *human–computer interaction* in recognition of the particular interest in computers and the composition of the user population!

Another strand of research that has influenced the development of HCI is information science and technology. Again the former is an old discipline, pre-dating the introduction of technology, and is concerned with the management and manipulation

of information within an organization. The introduction of technology has had a profound effect on the way that information can be stored, accessed and utilized and, consequently, a significant effect on the organization and work environment. Systems analysis has traditionally concerned itself with the influence of technology in the workplace, and fitting the technology to the requirements and constraints of the job. These issues are also the concern of HCI.

HCI draws on many disciplines, as we shall see, but it is in computer science and systems design that it must be accepted as a central concern. For all the other disciplines it can be a specialism, albeit one that provides crucial input; for systems design it is an essential part of the design process. From this perspective, HCI involves the design, implementation and evaluation of interactive systems in the context of the user's task and work.

However, when we talk about human–computer interaction, we do not necessarily envisage a single user with a desktop computer. By *user* we may mean an individual user, a group of users working together, or a sequence of users in an organization, each dealing with some part of the task or process. The user is whoever is trying to get the job done using the technology. By *computer* we mean any technology ranging from the general desktop computer to a large-scale computer system, a process control system or an embedded system. The system may include non-computerized parts, including other people. By *interaction* we mean any communication between a user and computer, be it direct or indirect. Direct interaction involves a dialog with feedback and control throughout performance of the task. Indirect interaction may involve batch processing or intelligent sensors controlling the environment. The important thing is that the user is interacting with the computer in order to accomplish something.

WHO IS INVOLVED IN HCI?

HCI is undoubtedly a multi-disciplinary subject. The ideal designer of an interactive system would have expertise in a range of topics: psychology and cognitive science to give her knowledge of the user's perceptual, cognitive and problem-solving skills; ergonomics for the user's physical capabilities; sociology to help her understand the wider context of the interaction; computer science and engineering to be able to build the necessary technology; business to be able to market it; graphic design to produce an effective interface presentation; technical writing to produce the manuals, and so it goes on. There is obviously too much expertise here to be held by one person (or indeed four!), perhaps even too much for the average design team. Indeed, although HCI is recognized as an interdisciplinary subject, in practice people tend to take a strong stance on one side or another. However, it is not possible to design effective interactive systems from one discipline in isolation. Input is needed from all sides. For example, a beautifully designed graphic display may be unusable if it ignores dialog constraints or the psychological limitations of the user.

In this book we want to encourage the multi-disciplinary view of HCI but we too have our 'stance', as computer scientists. We are interested in answering a particular question. How do principles and methods from each of these contributing disciplines in HCI help us to design better systems? In this we must be pragmatists rather than theorists: we want to know how to apply the theory to the problem rather than just acquire a deep understanding of the theory. Our goal, then, is to be multi-disciplinary but practical. We concentrate particularly on computer science, psychology and cognitive science as core subjects, and on their application to design; other disciplines are consulted to provide input where relevant.

THEORY AND HCI

Unfortunately for us, there is no general and unified theory of HCI that we can present. Indeed, it may be impossible ever to derive one; it is certainly out of our reach today. However, there is an underlying principle that forms the basis of our own views on HCI, and it is captured in our claim that people use computers to accomplish work. This outlines the three major issues of concern: the people, the computers and the tasks that are performed. The system must support the user's task, which gives us a fourth focus, usability: if the system forces the user to adopt an unacceptable mode of work then it is not usable.

There are, however, those who would dismiss our concentration on the task, saying that we do not even know enough about a theory of human tasks to support them in design. There is a good argument here (to which we return in Chapter 15). However, we can live with this confusion about what real tasks are because our understanding of tasks at the moment is sufficient to give us direction in design. The user's current tasks are studied and then supported by computers, which can in turn affect the nature of the original task and cause it to evolve. To illustrate, word processing has made it easy to manipulate paragraphs and reorder documents, allowing writers a completely new freedom that has affected writing styles. No longer is it vital to plan and construct text in an ordered fashion, since free-flowing prose can easily be restructured at a later date. This evolution of task in turn affects the design of the ideal system. However, we see this evolution as providing a motivating force behind the system development cycle, rather than a refutation of the whole idea of supportive design.

This word 'task' or the focus on accomplishing 'work' is also problematic when we think of areas such as domestic appliances, consumer electronics and e-commerce. There are three 'use' words that must all be true for a product to be successful; it must be:

useful – accomplish what is required: play music, cook dinner, format a document;

usable – do it easily and naturally, without danger of error, etc.;

used – make people want to use it, be attractive, engaging, fun, etc.

The last of these has not been a major factor until recently in HCI, but issues of motivation, enjoyment and experience are increasingly important. We are certainly even further from having a unified theory of experience than of task.

The question of whether HCI, or more importantly the design of interactive systems and the user interface in particular, is a science or a craft discipline is an interesting one. Does it involve artistic skill and fortuitous insight or reasoned methodical science? Here we can draw an analogy with architecture. The most impressive structures, the most beautiful buildings, the innovative and imaginative creations that provide aesthetic pleasure, all require inventive inspiration in design and a sense of artistry, and in this sense the discipline is a craft. However, these structures also have to be able to stand up to fulfill their purpose successfully, and to be able to do this the architect has to use science. So it is for HCI: beautiful and/or novel interfaces are artistically pleasing *and* capable of fulfilling the tasks required – a marriage of art and science into a successful whole. We want to reuse lessons learned from the past about how to achieve good results and avoid bad ones. For this we require both craft and science. Innovative ideas lead to more usable systems, but in order to maximize the potential benefit from the ideas, we need to understand not only that they work, but how and why they work. This scientific rationalization allows us to reuse related concepts in similar situations, in much the same way that architects can produce a bridge and know that it will stand, since it is based upon tried and tested principles.

The craft–science tension becomes even more difficult when we consider novel systems. Their increasing complexity means that our personal ideas of good and bad are no longer enough; for a complex system to be well designed we need to rely on something more than simply our intuition. Designers may be able to think about how one user would want to act, but how about groups? And what about new media? Our ideas of how best to share workloads or present video information are open to debate and question even in non-computing situations, and the incorporation of one version of good design into a computer system is quite likely to be unlike anyone else's version. Different people work in different ways, whilst different media color the nature of the interaction; both can dramatically change the very nature of the original task. In order to assist designers, it is unrealistic to assume that they can rely on artistic skill and perfect insight to develop usable systems. Instead we have to provide them with an understanding of the concepts involved, a scientific view of the reasons why certain things are successful whilst others are not, and then allow their creative nature to feed off this information: creative flow, underpinned with science; or maybe scientific method, accelerated by artistic insight. The truth is that HCI is required to be both a craft and a science in order to be successful.

HCI IN THE CURRICULUM

If HCI involves both craft and science then it must, in part at least, be taught. Imagination and skill may be qualities innate in the designer or developed through experience, but the underlying theory must be learned. In the past, when computers

were used primarily by expert specialists, concentration on the interface was a luxury that was often relinquished. Now designers cannot afford to ignore the interface in favour of the functionality of their systems: the two are too closely intertwined. If the interface is poor, the functionality is obscured; if it is well designed, it will allow the system's functionality to support the user's task.

Increasingly, therefore, computer science educators cannot afford to ignore HCI. We would go as far as to claim that HCI should be integrated into every computer science or software engineering course, either as a recurring feature of other modules or, preferably, as a module itself. It should not be viewed as an 'optional extra' (although, of course, more advanced HCI options can complement a basic core course). This view is shared by the ACM SIGCHI curriculum development group, who propose a curriculum for such a core course [9]. The topics included in this book, although developed without reference to this curriculum, cover the main emphases of it, and include enough detail and coverage to support specialized options as well.

In courses other than computer science, HCI may well be an option specializing in a particular area, such as cognitive modeling or task analysis. Selected use of the relevant chapters of this book can also support such a course.

HCI must be taken seriously by designers and educators if the requirement for additional complexity in the system is to be matched by increased clarity and usability in the interface. In this book we demonstrate how this can be done in practice.

DESIGN FOCUS

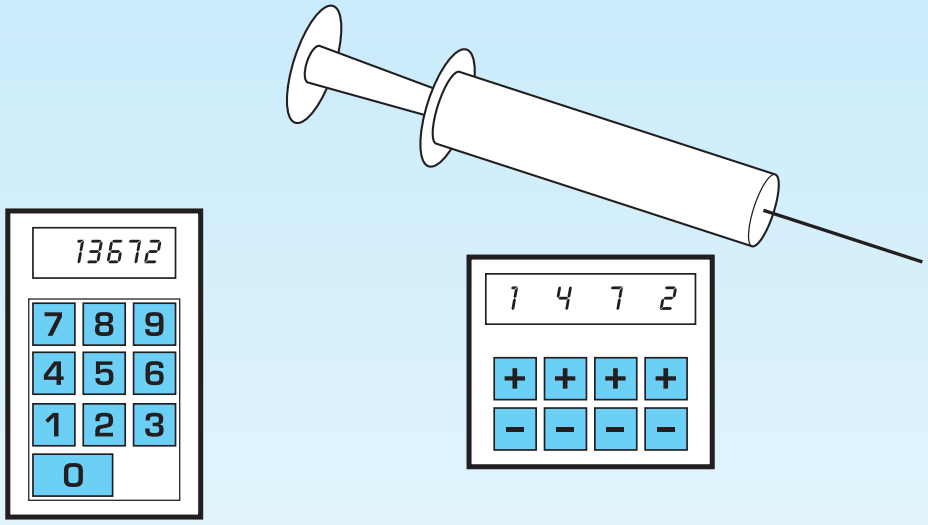
Quick fixes

You should expect to spend both time and money on interface design, just as you would with other parts of a system. So in one sense there are no quick fixes. However, a few simple steps can make a dramatic improvement.

Think 'user'
Probably 90% of the value of any interface design technique is that it forces the designer to remember that someone (and in particular someone else) will use the system under construction.

Try it out
Of course, many designers will build a system that they find easy and pleasant to use, and they find it incomprehensible that anyone else could have trouble with it. Simply sitting someone down with an early version of an interface (without the designer prompting them at each step!) is enormously valuable. Professional usability laboratories will have video equipment, one-way mirrors and other sophisticated monitors, but a notebook and pencil and a home-video camera will suffice (more about evaluation in Chapter 9).

Involve the users
Where possible, the eventual users should be involved in the design process. They have vital knowledge and will soon find flaws. A mechanical syringe was once being developed and a prototype was demonstrated to hospital staff. Happily they quickly noticed the potentially fatal flaw in its interface.



Before consulting users

After consulting users

Figure 0.1 Automatic syringe: setting the dose to 1372. The effect of one key slip before and after user involvement

The doses were entered via a numeric keypad: an accidental keypress and the dose could be out by a factor of 10! The production version had individual increment/decrement buttons for each digit (more about participatory design in Chapter 13).

Iterate

People are complicated, so you won't get it right first time. Programming an interface can be a very difficult and time-consuming business. So, the result becomes precious and the builder will want to defend it and minimize changes. Making early prototypes less precious and easier to throw away is crucial. Happily there are now many interface builder tools that aid this process. For example, mock-ups can be quickly constructed using HyperCard on the Apple Macintosh or Visual Basic on the PC. For visual and layout decisions, paper designs and simple models can be used (more about iterative design in Chapter 5).

PART 1

FOUNDATIONS

In this part we introduce the fundamental components of an interactive system: the human user, the computer system itself and the nature of the interactive process. We then present a view of the history of interactive systems by looking at key interaction paradigms that have been significant.

Chapter 1 discusses the psychological and physiological attributes of the user, providing us with a basic overview of the capabilities and limitations that affect our ability to use computer systems. It is only when we have an understanding of the user at this level that we can understand what makes for successful designs. Chapter 2 considers the computer in a similar way. Input and output devices are described and explained and the effect that their individual characteristics have on the interaction highlighted. The computational power and memory of the computer is another important component in determining what can be achieved in the interaction, whilst due attention is also paid to paper output since this forms one of the major uses of computers and users' tasks today. Having approached interaction from both the human and the computer side, we then turn our attention to the dialog between them in Chapter 3, where we look at models of interaction. In Chapter 4 we take a historical perspective on the evolution of interactive systems and how they have increased the usability of computers in general.

1

The human

OVERVIEW

- Humans are limited in their capacity to process information. This has important implications for design.
- Information is received and responses given via a number of input and output channels:
  - visual channel
  - auditory channel
  - haptic channel
  - movement.
- Information is stored in memory:
  - sensory memory
  - short-term (working) memory
  - long-term memory.
- Information is processed and applied:
  - reasoning
  - problem solving
  - skill acquisition
  - error.
- Emotion influences human capabilities.
- Users share common capabilities but are individuals with differences, which should not be ignored.

1.1 INTRODUCTION

This chapter is the first of four in which we introduce some of the 'foundations' of HCI. We start with the human, the central character in any discussion of interactive systems. The human, the *user*, is, after all, the one whom computer systems are designed to assist. The requirements of the user should therefore be our first priority.

In this chapter we will look at areas of human psychology coming under the general banner of *cognitive psychology*. This may seem a far cry from designing and building interactive computer systems, but it is not. In order to design something for someone, we need to understand their capabilities and limitations. We need to know if there are things that they will find difficult or, even, impossible. It will also help us to know what people find easy and how we can help them by encouraging these things. We will look at aspects of cognitive psychology which have a bearing on the use of computer systems: how humans perceive the world around them, how they store and process information and solve problems, and how they physically manipulate objects.

We have already said that we will restrict our study to those things that are relevant to HCI. One way to structure this discussion is to think of the user in a way that highlights these aspects. In other words, to think of a simplified *model* of what is actually going on. Many models have been proposed and it useful to consider one of the most influential in passing, to understand the context of the discussion that is to follow. In 1983, Card, Moran and Newell [56] described the *Model Human Processor*, which is a simplified view of the human processing involved in interacting with computer systems. The model comprises three subsystems: the perceptual system, handling sensory stimulus from the outside world, the motor system, which controls actions, and the cognitive system, which provides the processing needed to connect the two. Each of these subsystems has its own processor and memory, although obviously the complexity of these varies depending on the complexity of the tasks the subsystem has to perform. The model also includes a number of *principles of operation* which dictate the behavior of the systems under certain conditions.

We will use the analogy of the user as an information processing system, but in our model make the analogy closer to that of a conventional computer system. Information comes in, is stored and processed, and information is passed out. We will therefore discuss three components of this system: input–output, memory and processing. In the human, we are dealing with an intelligent information-processing system, and processing therefore includes problem solving, learning, and, consequently, making mistakes. This model is obviously a simplification of the real situation, since memory and processing are required at all levels, as we have seen in the Model Human Processor. However, it is convenient as a way of grasping how information is handled by the human system. The human, unlike the computer, is also influenced by external factors such as the social and organizational environment, and we need to be aware of these influences as well. We will ignore such factors for now and concentrate on the human's information processing capabilities only. We will return to social and organizational influences in Chapter 3 and, in more detail, in Chapter 13.

In this chapter, we will first look at the human's input–output channels, the senses and responders or effectors. This will involve some low-level processing. Secondly, we will consider human memory and how it works. We will then think about how humans perform complex problem solving, how they learn and acquire skills, and why they make mistakes. Finally, we will discuss how these things can help us in the design of computer systems.

1.2 INPUT–OUTPUT CHANNELS

A person's interaction with the outside world occurs through information being received and sent: input and output. In an interaction with a computer the user receives information that is output by the computer, and responds by providing input to the computer – the user's output becomes the computer's input and vice versa. Consequently the use of the terms input and output may lead to confusion so we shall blur the distinction somewhat and concentrate on the channels involved. This blurring is appropriate since, although a particular channel may have a primary role as input or output in the interaction, it is more than likely that it is also used in the other role. For example, sight may be used primarily in receiving information from the computer, but it can also be used to provide information to the computer, for example by fixating on a particular screen point when using an eyegaze system.

Input in the human occurs mainly through the senses and output through the motor control of the effectors. There are five major senses: sight, hearing, touch, taste and smell. Of these, the first three are the most important to HCI. Taste and smell do not currently play a significant role in HCI, and it is not clear whether they could be exploited at all in general computer systems, although they could have a role to play in more specialized systems (smells to give warning of malfunction, for example) or in augmented reality systems. However, vision, hearing and touch are central.

Similarly there are a number of effectors, including the limbs, fingers, eyes, head and vocal system. In the interaction with the computer, the fingers play the primary role, through typing or mouse control, with some use of voice, and eye, head and body position.

Imagine using a personal computer (PC) with a mouse and a keyboard. The application you are using has a graphical interface, with menus, icons and windows. In your interaction with this system you receive information primarily by sight, from what appears on the screen. However, you may also receive information by ear: for example, the computer may 'beep' at you if you make a mistake or to draw attention to something, or there may be a voice commentary in a multimedia presentation. Touch plays a part too in that you will feel the keys moving (also hearing the 'click') or the orientation of the mouse, which provides vital feedback about what you have done. You yourself send information to the computer using your hands, either by hitting keys or moving the mouse. Sight and hearing do not play a direct role in sending information in this example, although they may be used to receive

information from a third source (for example, a book, or the words of another person) which is then transmitted to the computer.

In this section we will look at the main elements of such an interaction, first considering the role and limitations of the three primary senses and going on to consider motor control.

1.2.1 Vision

Human vision is a highly complex activity with a range of physical and perceptual limitations, yet it is the primary source of information for the average person. We can roughly divide visual perception into two stages: the physical reception of the stimulus from the outside world, and the processing and interpretation of that stimulus. On the one hand the physical properties of the eye and the visual system mean that there are certain things that cannot be seen by the human; on the other the interpretative capabilities of visual processing allow images to be constructed from incomplete information. We need to understand both stages as both influence what can and cannot be perceived visually by a human being, which in turn directly affects the way that we design computer systems. We will begin by looking at the eye as a physical receptor, and then go on to consider the processing involved in basic vision.

The human eye

Vision begins with light. The eye is a mechanism for receiving light and transforming it into electrical energy. Light is reflected from objects in the world and their image is focussed upside down on the back of the eye. The receptors in the eye transform it into electrical signals which are passed to the brain.

The eye has a number of important components (see Figure 1.1) which we will look at in more detail. The *cornea* and *lens* at the front of the eye focus the light into a sharp image on the back of the eye, the *retina*. The retina is light sensitive and contains two types of *photoreceptor*: *rods* and *cones*.

Rods are highly sensitive to light and therefore allow us to see under a low level of illumination. However, they are unable to resolve fine detail and are subject to light saturation. This is the reason for the temporary blindness we get when moving from a darkened room into sunlight: the rods have been active and are saturated by the sudden light. The cones do not operate either as they are suppressed by the rods. We are therefore temporarily unable to see at all. There are approximately 120 million rods per eye which are mainly situated towards the edges of the retina. Rods therefore dominate peripheral vision.

Cones are the second type of receptor in the eye. They are less sensitive to light than the rods and can therefore tolerate more light. There are three types of cone, each sensitive to a different wavelength of light. This allows color vision. The eye has approximately 6 million cones, mainly concentrated on the *fovea*, a small area of the retina on which images are fixated.

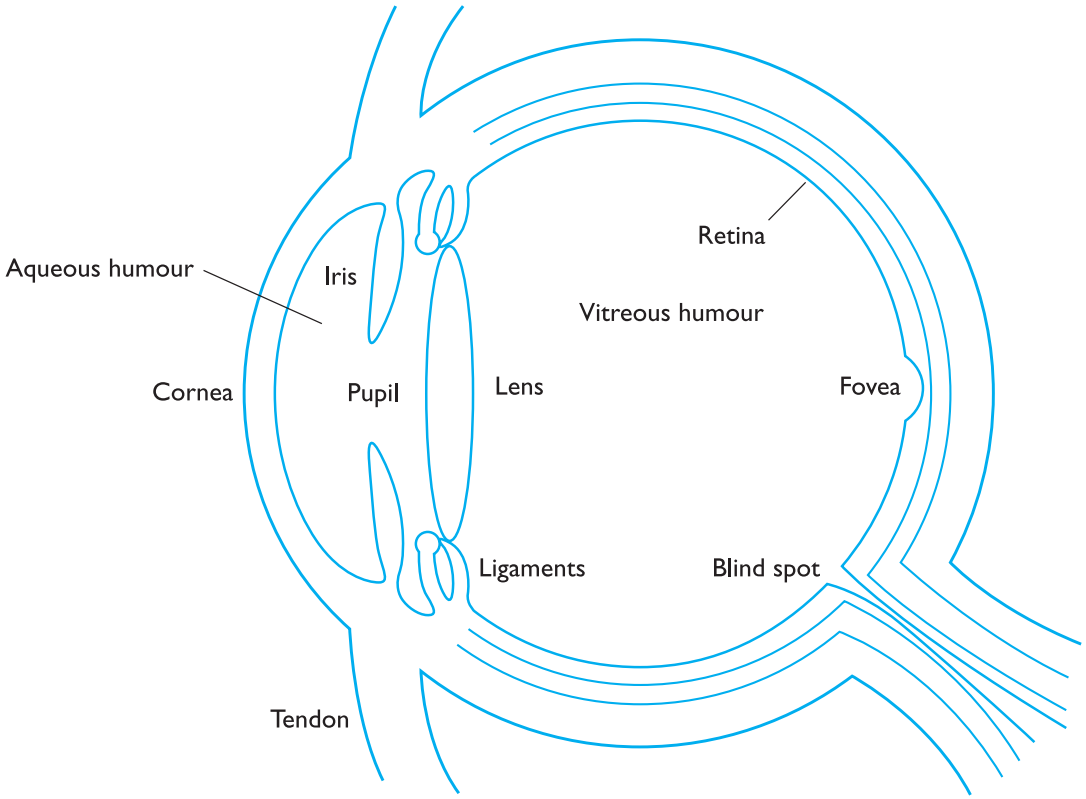


Figure 1.1 The human eye

Although the retina is mainly covered with photoreceptors there is one *blind spot* where the optic nerve enters the eye. The blind spot has no rods or cones, yet our visual system compensates for this so that in normal circumstances we are unaware of it.

The retina also has specialized nerve cells called *ganglion cells*. There are two types: X-cells, which are concentrated in the fovea and are responsible for the early detection of pattern; and Y-cells which are more widely distributed in the retina and are responsible for the early detection of movement. The distribution of these cells means that, while we may not be able to detect changes in pattern in peripheral vision, we can perceive movement.

Visual perception

Understanding the basic construction of the eye goes some way to explaining the physical mechanisms of vision but visual perception is more than this. The information received by the visual apparatus must be filtered and passed to processing elements which allow us to recognize coherent scenes, disambiguate relative distances and differentiate color. We will consider some of the capabilities and limitations of visual processing later, but first we will look a little more closely at how we perceive size and depth, brightness and color, each of which is crucial to the design of effective visual interfaces.

DESIGN FOCUS

Getting noticed

The extensive knowledge about the human visual system can be brought to bear in practical design. For example, our ability to read or distinguish falls off inversely as the distance from our point of focus increases. This is due to the fact that the cones are packed more densely towards the center of our visual field. You can see this in the following image. Fixate on the dot in the center. The letters on the left should all be equally readable, those on the right all equally harder.

A B C D E F • H I J K

This loss of discrimination sets limits on the amount that can be seen or read without moving one's eyes. A user concentrating on the middle of the screen cannot be expected to read help text on the bottom line.

However, although our ability to discriminate static text diminishes, the rods, which are concentrated more in the outer parts of our visual field, are very sensitive to changes; hence we see movement well at the edge of our vision. So if you want a user to see an error message at the bottom of the screen it had better be flashing! On the other hand clever moving icons, however impressive they are, will be distracting even when the user is not looking directly at them.

Perceiving size and depth Imagine you are standing on a hilltop. Beside you on the summit you can see rocks, sheep and a small tree. On the hillside is a farmhouse with outbuildings and farm vehicles. Someone is on the track, walking toward the summit. Below in the valley is a small market town.

Even in describing such a scene the notions of size and distance predominate. Our visual system is easily able to interpret the images which it receives to take account of these things. We can identify similar objects regardless of the fact that they appear to us to be of vastly different sizes. In fact, we can use this information to judge distances.

So how does the eye perceive size, depth and relative distances? To understand this we must consider how the image appears on the retina. As we noted in the previous section, reflected light from the object forms an upside-down image on the retina. The size of that image is specified as a *visual angle*. Figure 1.2 illustrates how the visual angle is calculated.

If we were to draw a line from the top of the object to a central point on the front of the eye and a second line from the bottom of the object to the same point, the visual angle of the object is the angle between these two lines. Visual angle is affected by both the size of the object and its distance from the eye. Therefore if two objects are at the same distance, the larger one will have the larger visual angle. Similarly, if two objects of the same size are placed at different distances from the eye, the

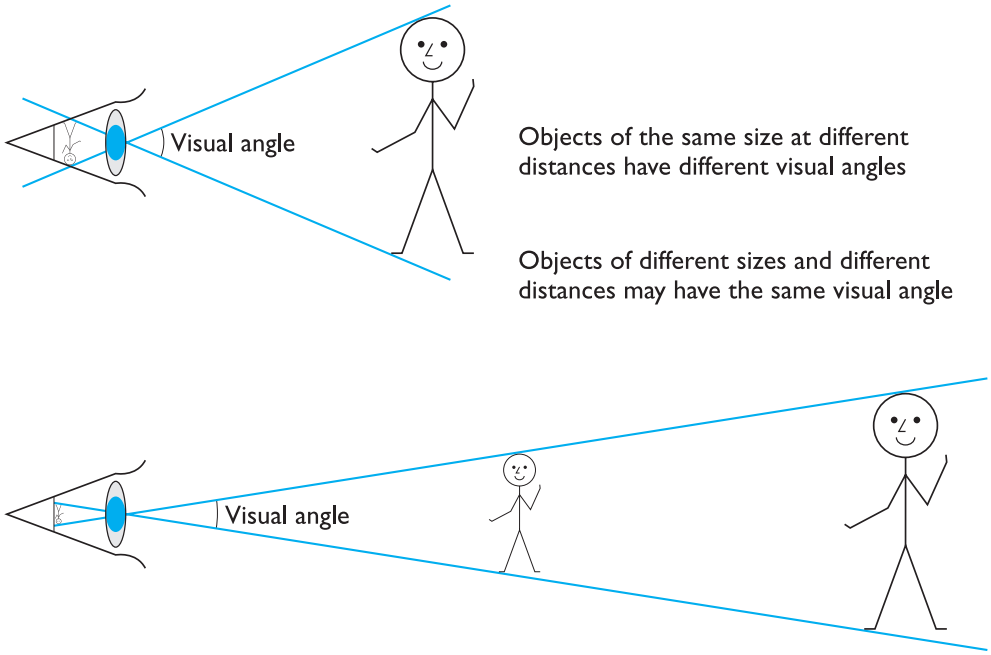


Figure 1.2 Visual angle

furthest one will have the smaller visual angle. The visual angle indicates how much of the field of view is taken by the object. The visual angle measurement is given in either degrees or *minutes of arc*, where 1 degree is equivalent to 60 minutes of arc, and 1 minute of arc to 60 seconds of arc.

So how does an object's visual angle affect our perception of its size? First, if the visual angle of an object is too small we will be unable to perceive it at all. *Visual acuity* is the ability of a person to perceive fine detail. A number of measurements have been established to test visual acuity, most of which are included in standard eye tests. For example, a person with normal vision can detect a single line if it has a visual angle of 0.5 seconds of arc. Spaces between lines can be detected at 30 seconds to 1 minute of visual arc. These represent the limits of human visual acuity.

Assuming that we can perceive the object, does its visual angle affect our perception of its size? Given that the visual angle of an object is reduced as it gets further away, we might expect that we would perceive the object as smaller. In fact, our perception of an object's size remains constant even if its visual angle changes. So a person's height is perceived as constant even if they move further from you. This is the *law of size constancy*, and it indicates that our perception of size relies on factors other than the visual angle.

One of these factors is our perception of depth. If we return to the hilltop scene there are a number of *cues* which we can use to determine the relative positions and distances of the objects which we see. If objects overlap, the object which is partially covered is perceived to be in the background, and therefore further away. Similarly, the size and height of the object in our field of view provides a cue to its distance.

A third cue is familiarity: if we expect an object to be of a certain size then we can judge its distance accordingly. This has been exploited for humour in advertising: one advertisement for beer shows a man walking away from a bottle in the foreground. As he walks, he bumps into the bottle, which is in fact a giant one in the background!

Perceiving brightness A second aspect of visual perception is the perception of *brightness*. Brightness is in fact a subjective reaction to levels of light. It is affected by *luminance* which is the amount of light emitted by an object. The luminance of an object is dependent on the amount of light falling on the object's surface and its reflective properties. Luminance is a physical characteristic and can be measured using a *photometer*. *Contrast* is related to luminance: it is a function of the luminance of an object and the luminance of its background.

Although brightness is a subjective response, it can be described in terms of the amount of luminance that gives a *just noticeable difference* in brightness. However, the visual system itself also compensates for changes in brightness. In dim lighting, the rods predominate vision. Since there are fewer rods on the fovea, objects in low lighting can be seen less easily when fixated upon, and are more visible in peripheral vision. In normal lighting, the cones take over.

Visual acuity increases with increased luminance. This may be an argument for using high display luminance. However, as luminance increases, *flicker* also increases. The eye will perceive a light switched on and off rapidly as constantly on. But if the speed of switching is less than 50 Hz then the light is perceived to flicker. In high luminance flicker can be perceived at over 50 Hz. Flicker is also more noticeable in peripheral vision. This means that the larger the display (and consequently the more peripheral vision that it occupies), the more it will appear to flicker.

Perceiving color A third factor that we need to consider is perception of color. Color is usually regarded as being made up of three components: *hue*, *intensity* and *saturation*. Hue is determined by the spectral wavelength of the light. Blues have short wavelengths, greens medium and reds long. Approximately 150 different hues can be discriminated by the average person. Intensity is the brightness of the color, and saturation is the amount of whiteness in the color. By varying these two, we can perceive in the region of 7 million different colors. However, the number of colors that can be identified by an individual without training is far fewer (in the region of 10).

The eye perceives color because the cones are sensitive to light of different wavelengths. There are three different types of cone, each sensitive to a different color (blue, green and red). Color vision is best in the fovea, and worst at the periphery where rods predominate. It should also be noted that only 3–4% of the fovea is occupied by cones which are sensitive to blue light, making blue acuity lower.

Finally, we should remember that around 8% of males and 1% of females suffer from color blindness, most commonly being unable to discriminate between red and green.

The capabilities and limitations of visual processing

In considering the way in which we perceive images we have already encountered some of the capabilities and limitations of the human visual processing system. However, we have concentrated largely on low-level perception. Visual processing involves the transformation and interpretation of a complete image, from the light that is thrown onto the retina. As we have already noted, our expectations affect the way an image is perceived. For example, if we know that an object is a particular size, we will perceive it as that size no matter how far it is from us.

Visual processing compensates for the movement of the image on the retina which occurs as we move around and as the object which we see moves. Although the retinal image is moving, the image that we perceive is stable. Similarly, color and brightness of objects are perceived as constant, in spite of changes in luminance.

This ability to interpret and exploit our expectations can be used to resolve ambiguity. For example, consider the image shown in Figure 1.3. What do you perceive? Now consider Figure 1.4 and Figure 1.5. The context in which the object appears

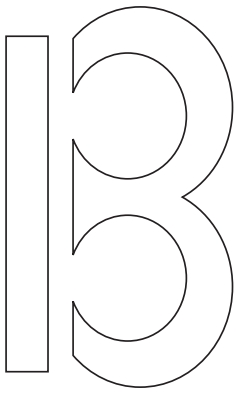


Figure 1.3 An ambiguous shape?

Figure 1.4 ABC

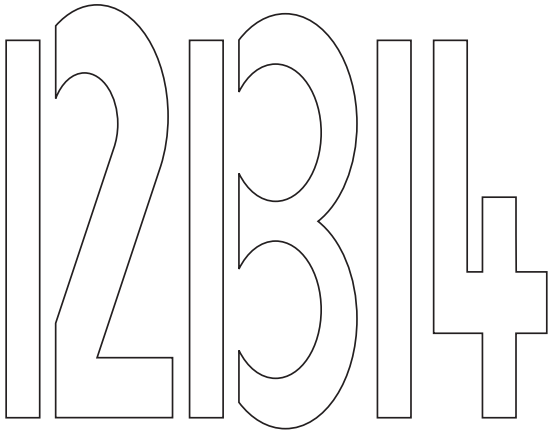


Figure 1.5 12 13 14

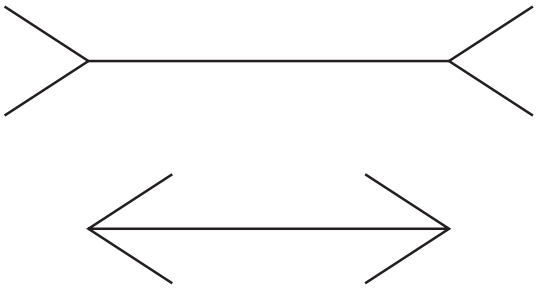


Figure 1.6 The Muller–Lyer illusion – which line is longer?

allows our expectations to clearly disambiguate the interpretation of the object, as either a B or a 13.

However, it can also create optical illusions. For example, consider Figure 1.6. Which line is longer? Most people when presented with this will say that the top line is longer than the bottom. In fact, the two lines are the same length. This may be due to a false application of the law of size constancy: the top line appears like a concave edge, the bottom like a convex edge. The former therefore seems further away than the latter and is therefore scaled to appear larger. A similar illusion is the Ponzo illusion (Figure 1.7). Here the top line appears longer, owing to the distance effect, although both lines are the same length. These illusions demonstrate that our perception of size is not completely reliable.

Another illusion created by our expectations compensating an image is the proofreading illusion. Read the text in Figure 1.8 quickly. What does it say? Most people reading this rapidly will read it correctly, although closer inspection shows that the word 'the' is repeated in the second and third line.

These are just a few examples of how the visual system compensates, and sometimes overcompensates, to allow us to perceive the world around us.

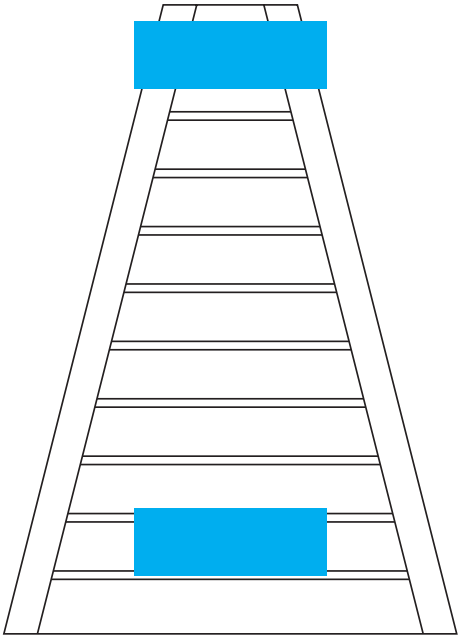


Figure 1.7 The Ponzo illusion – are these the same size?

The quick brown fox jumps over the the lazy dog.

Figure 1.8 Is this text correct?

DESIGN FOCUS

Where's the middle?

Optical illusions highlight the differences between the way things are and the way we perceive them – and in interface design we need to be aware that we will not always perceive things exactly as they are. The way that objects are composed together will affect the way we perceive them, and we do not perceive geometric shapes exactly as they are drawn. For example, we tend to magnify horizontal lines and reduce vertical. So a square needs to be slightly increased in height to appear square and lines will appear thicker if horizontal rather than vertical.

Optical illusions also affect page symmetry. We tend to see the center of a page as being a little above the actual center – so if a page is arranged symmetrically around the actual center, we will see it as too low down. In graphic design this is known as the *optical center* – and bottom page margins tend to be increased by 50% to compensate.

Reading

So far we have concentrated on the perception of images in general. However, the perception and processing of text is a special case that is important to interface design, which invariably requires some textual display. We will therefore end this section by looking at *reading*. There are several stages in the reading process. First, the visual pattern of the word on the page is perceived. It is then decoded with reference to an internal representation of language. The final stages of language processing include syntactic and semantic analysis and operate on phrases or sentences.

We are most concerned with the first two stages of this process and how they influence interface design. During reading, the eye makes jerky movements called *saccades* followed by fixations. Perception occurs during the fixation periods, which account for approximately 94% of the time elapsed. The eye moves backwards over the text as well as forwards, in what are known as *regressions*. If the text is complex there will be more regressions.

Adults read approximately 250 words a minute. It is unlikely that words are scanned serially, character by character, since experiments have shown that words can be recognized as quickly as single characters. Instead, familiar words are recognized using word shape. This means that removing the word shape clues (for example, by capitalizing words) is detrimental to reading speed and accuracy.

The speed at which text can be read is a measure of its legibility. Experiments have shown that standard font sizes of 9 to 12 points are equally legible, given proportional spacing between lines [346]. Similarly line lengths of between 2.3 and 5.2 inches (58 and 132 mm) are equally legible. However, there is evidence that reading from a computer screen is slower than from a book [244]. This is thought to be due to a number of factors including a longer line length, fewer words to a page,

orientation and the familiarity of the medium of the page. These factors can of course be reduced by careful design of textual interfaces.

A final word about the use of contrast in visual display: a negative contrast (dark characters on a light screen) provides higher luminance and, therefore, increased acuity, than a positive contrast. This will in turn increase legibility. However, it will also be more prone to flicker. Experimental evidence suggests that in practice negative contrast displays are preferred and result in more accurate performance [30].

1.2.2 Hearing

The sense of hearing is often considered secondary to sight, but we tend to underestimate the amount of information that we receive through our ears. Close your eyes for a moment and listen. What sounds can you hear? Where are they coming from? What is making them? As I sit at my desk I can hear cars passing on the road outside, machinery working on a site nearby, the drone of a plane overhead and bird song. But I can also tell *where* the sounds are coming from, and estimate how far away they are. So from the sounds I hear I can tell that a car is passing on a particular road near my house, and which direction it is traveling in. I know that building work is in progress in a particular location, and that a certain type of bird is perched in the tree in my garden.

The auditory system can convey a lot of information about our environment. But how does it work?

The human ear

Just as vision begins with light, hearing begins with vibrations in the air or *sound waves*. The ear receives these vibrations and transmits them, through various stages, to the auditory nerves. The ear comprises three sections, commonly known as the *outer ear*, *middle ear* and *inner ear*.

The outer ear is the visible part of the ear. It has two parts: the *pinna*, which is the structure that is attached to the sides of the head, and the *auditory canal*, along which sound waves are passed to the middle ear. The outer ear serves two purposes. First, it protects the sensitive middle ear from damage. The auditory canal contains wax which prevents dust, dirt and over-inquisitive insects reaching the middle ear. It also maintains the middle ear at a constant temperature. Secondly, the pinna and auditory canal serve to amplify some sounds.

The middle ear is a small cavity connected to the outer ear by the *tympanic membrane*, or ear drum, and to the inner ear by the *cochlea*. Within the cavity are the *ossicles*, the smallest bones in the body. Sound waves pass along the auditory canal and vibrate the ear drum which in turn vibrates the ossicles, which transmit the vibrations to the cochlea, and so into the inner ear. This 'relay' is required because, unlike the air-filled outer and middle ears, the inner ear is filled with a denser cochlean liquid. If passed directly from the air to the liquid, the transmission of the sound waves would be poor. By transmitting them via the ossicles the sound waves are concentrated and amplified.

The waves are passed into the liquid-filled cochlea in the inner ear. Within the cochlea are delicate hair cells or *cilia* that bend because of the vibrations in the cochlean liquid and release a chemical transmitter which causes impulses in the auditory nerve.

Processing sound

As we have seen, sound is changes or vibrations in air pressure. It has a number of characteristics which we can differentiate. *Pitch* is the frequency of the sound. A low frequency produces a low pitch, a high frequency, a high pitch. *Loudness* is proportional to the amplitude of the sound; the frequency remains constant. *Timbre* relates to the type of the sound: sounds may have the same pitch and loudness but be made by different instruments and so vary in timbre. We can also identify a sound's location, since the two ears receive slightly different sounds, owing to the time difference between the sound reaching the two ears and the reduction in intensity caused by the sound waves reflecting from the head.

The human ear can hear frequencies from about 20 Hz to 15 kHz. It can distinguish frequency changes of less than 1.5 Hz at low frequencies but is less accurate at high frequencies. Different frequencies trigger activity in neurons in different parts of the auditory system, and cause different rates of firing of nerve impulses.

The auditory system performs some filtering of the sounds received, allowing us to ignore background noise and concentrate on important information. We are selective in our hearing, as illustrated by the *cocktail party effect*, where we can pick out our name spoken across a crowded noisy room. However, if sounds are too loud, or frequencies too similar, we are unable to differentiate sound.

As we have seen, sound can convey a remarkable amount of information. It is rarely used to its potential in interface design, usually being confined to warning sounds and notifications. The exception is multimedia, which may include music, voice commentary and sound effects. However, the ear can differentiate quite subtle sound changes and can recognize familiar sounds without concentrating attention on the sound source. This suggests that sound could be used more extensively in interface design, to convey information about the system state, for example. This is discussed in more detail in Chapter 10.

Worked exercise *Suggest ideas for an interface which uses the properties of sound effectively.*

Answer You might approach this exercise by considering how sound could be added to an application with which you are familiar. Use your imagination. This is also a good subject for a literature survey (starting with the references in Chapter 10).

Speech sounds can obviously be used to convey information. This is useful not only for the visually impaired but also for any application where the user's attention has to be divided (for example, power plant control, flight control, etc.). Uses of non-speech sounds include the following:

- Attention – to attract the user's attention to a critical situation or to the end of a process, for example.

- Status information – continuous background sounds can be used to convey status information. For example, monitoring the progress of a process (without the need for visual attention).
- Confirmation – a sound associated with an action to confirm that the action has been carried out. For example, associating a sound with deleting a file.
- Navigation – using changing sound to indicate where the user is in a system. For example, what about sound to support navigation in hypertext?

1.2.3 Touch

The third and last of the senses that we will consider is touch or *haptic perception*. Although this sense is often viewed as less important than sight or hearing, imagine life without it. Touch provides us with vital information about our environment. It tells us when we touch something hot or cold, and can therefore act as a warning. It also provides us with feedback when we attempt to lift an object, for example. Consider the act of picking up a glass of water. If we could only see the glass and not feel when our hand made contact with it or feel its shape, the speed and accuracy of the action would be reduced. This is the experience of users of certain *virtual reality* games: they can see the computer-generated objects which they need to manipulate but they have no physical sensation of touching them. Watching such users can be an informative and amusing experience! Touch is therefore an important means of feedback, and this is no less so in using computer systems. Feeling buttons depress is an important part of the task of pressing the button. Also, we should be aware that, although for the average person, haptic perception is a secondary source of information, for those whose other senses are impaired, it may be vitally important. For such users, interfaces such as braille may be the primary source of information in the interaction. We should not therefore underestimate the importance of touch.

The apparatus of touch differs from that of sight and hearing in that it is not localized. We receive stimuli through the skin. The skin contains three types of sensory receptor: *thermoreceptors* respond to heat and cold, *nociceptors* respond to intense pressure, heat and pain, and *mechanoreceptors* respond to pressure. It is the last of these that we are concerned with in relation to human–computer interaction.

There are two kinds of mechanoreceptor, which respond to different types of pressure. *Rapidly adapting mechanoreceptors* respond to immediate pressure as the skin is indented. These receptors also react more quickly with increased pressure. However, they stop responding if continuous pressure is applied. *Slowly adapting mechanoreceptors* respond to continuously applied pressure.

Although the whole of the body contains such receptors, some areas have greater sensitivity or acuity than others. It is possible to measure the acuity of different areas of the body using the *two-point threshold test*. Take two pencils, held so their tips are about 12 mm apart. Touch the points to your thumb and see if you can feel two points. If you cannot, move the points a little further apart. When you can feel two points, measure the distance between them. The greater the distance, the lower the sensitivity. You can repeat this test on different parts of your body. You should find

that the measure on the forearm is around 10 times that of the finger or thumb. The fingers and thumbs have the highest acuity.

A second aspect of haptic perception is *kinesthesis*: awareness of the position of the body and limbs. This is due to receptors in the joints. Again there are three types: rapidly adapting, which respond when a limb is moved in a particular direction; slowly adapting, which respond to both movement and static position; and positional receptors, which only respond when a limb is in a static position. This perception affects both comfort and performance. For example, for a touch typist, awareness of the relative positions of the fingers and feedback from the keyboard are very important.

Handling the goods

E-commerce has become very successful in some areas of sales, such as travel services, books and CDs, and food. However, in some retail areas, such as clothes shopping, e-commerce has been less successful. Why?

When buying train and airline tickets and, to some extent, books and food, the experience of shopping is less important than the convenience. So, as long as we know what we want, we are happy to shop online. With clothes, the experience of shopping is far more important. We need to be able to handle the goods, feel the texture of the material, check the weight to test quality. Even if we know that something will fit us we still want to be able to handle it before buying.

Research into haptic interaction (see Chapter 2 and Chapter 10) is looking at ways of solving this problem. By using special force feedback and tactile hardware, users are able to feel surfaces and shape. For example, a demonstration environment called TouchCity allows people to walk around a virtual shopping mall, pick up products and feel their texture and weight. A key problem with the commercial use of such an application, however, is that the haptic experience requires expensive hardware not yet available to the average e-shopper. However, in future, such immersive e-commerce experiences are likely to be the norm. (See www.novint.com/)

1.2.4 Movement

Before leaving this section on the human's input–output channels, we need to consider motor control and how the way we move affects our interaction with computers. A simple action such as hitting a button in response to a question involves a number of processing stages. The stimulus (of the question) is received through the sensory receptors and transmitted to the brain. The question is processed and a valid response generated. The brain then tells the appropriate muscles to respond. Each of these stages takes time, which can be roughly divided into reaction time and movement time.

Movement time is dependent largely on the physical characteristics of the subjects: their age and fitness, for example. Reaction time varies according to the sensory channel through which the stimulus is received. A person can react to an auditory

signal in approximately 150 ms, to a visual signal in 200 ms and to pain in 700 ms. However, a combined signal will result in the quickest response. Factors such as skill or practice can reduce reaction time, and fatigue can increase it.

A second measure of motor skill is accuracy. One question that we should ask is whether speed of reaction results in reduced accuracy. This is dependent on the task and the user. In some cases, requiring increased reaction time reduces accuracy. This is the premise behind many arcade and video games where less skilled users fail at levels of play that require faster responses. However, for skilled operators this is not necessarily the case. Studies of keyboard operators have shown that, although the faster operators were up to twice as fast as the others, the slower ones made 10 times the errors.

Speed and accuracy of movement are important considerations in the design of interactive systems, primarily in terms of the time taken to move to a particular target on a screen. The target may be a button, a menu item or an icon, for example. The time taken to hit a target is a function of the size of the target and the distance that has to be moved. This is formalized in *Fitts' law* [135]. There are many variations of this formula, which have varying constants, but they are all very similar. One common form is

$$\text{Movement time} = a + b \log_2(\text{distance}/\text{size} + 1)$$

where a and b are empirically determined constants.

This affects the type of target we design. Since users will find it more difficult to manipulate small objects, targets should generally be as large as possible and the distance to be moved as small as possible. This has led to suggestions that pie-chart-shaped menus are preferable to lists since all options are equidistant. However, the trade-off is increased use of screen estate, so the choice may not be so simple. If lists are used, the most frequently used options can be placed closest to the user's start point (for example, at the top of the menu). The implications of Fitts' law in design are discussed in more detail in Chapter 12.

1.3 HUMAN MEMORY

Have you ever played the memory game? The idea is that each player has to recount a list of objects and add one more to the end. There are many variations but the objects are all loosely related: 'I went to the market and bought a lemon, some oranges, bacon . . .' or 'I went to the zoo and saw monkeys, and lions, and tigers . . .' and so on. As the list grows objects are missed out or recalled in the wrong order and so people are eliminated from the game. The winner is the person remaining at the end. Such games rely on our ability to store and retrieve information, even seemingly arbitrary items. This is the job of our memory system.

Indeed, much of our everyday activity relies on memory. As well as storing all our factual knowledge, our memory contains our knowledge of actions or procedures.

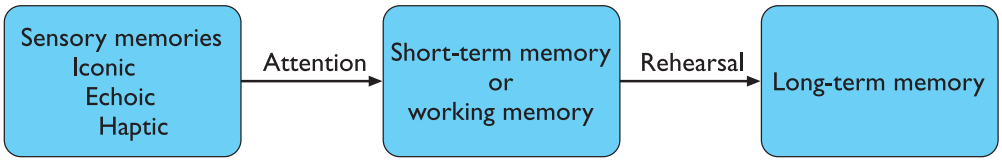


Figure 1.9 A model of the structure of memory

It allows us to repeat actions, to use language, and to use new information received via our senses. It also gives us our sense of identity, by preserving information from our past experiences.

But how does our memory work? How do we remember arbitrary lists such as those generated in the memory game? Why do some people remember more easily than others? And what happens when we forget?

In order to answer questions such as these, we need to understand some of the capabilities and limitations of human memory. Memory is the second part of our model of the human as an information-processing system. However, as we noted earlier, such a division is simplistic since, as we shall see, memory is associated with each level of processing. Bearing this in mind, we will consider the way in which memory is structured and the activities that take place within the system.

It is generally agreed that there are three types of memory or memory function: *sensory buffers*, *short-term memory* or *working memory*, and *long-term memory*. There is some disagreement as to whether these are three separate systems or different functions of the same system. We will not concern ourselves here with the details of this debate, which is discussed in detail by Baddeley [21], but will indicate the evidence used by both sides as we go along. For our purposes, it is sufficient to note three separate types of memory. These memories interact, with information being processed and passed between memory stores, as shown in Figure 1.9.

1.3.1 Sensory memory

The sensory memories act as buffers for stimuli received through the senses. A sensory memory exists for each sensory channel: *iconic memory* for visual stimuli, *echoic memory* for aural stimuli and *haptic memory* for touch. These memories are constantly overwritten by new information coming in on these channels.

We can demonstrate the existence of iconic memory by moving a finger in front of the eye. Can you see it in more than one place at once? This indicates a persistence of the image after the stimulus has been removed. A similar effect is noticed most vividly at firework displays where moving sparklers leave a persistent image. Information remains in iconic memory very briefly, in the order of 0.5 seconds.

Similarly, the existence of echoic memory is evidenced by our ability to ascertain the direction from which a sound originates. This is due to information being received by both ears. However, since this information is received at different times, we must store the stimulus in the meantime. Echoic memory allows brief 'play-back'

of information. Have you ever had someone ask you a question when you are reading? You ask them to repeat the question, only to realize that you know what was asked after all. This experience, too, is evidence of the existence of echoic memory.

Information is passed from sensory memory into short-term memory by attention, thereby filtering the stimuli to only those which are of interest at a given time. Attention is the concentration of the mind on one out of a number of competing stimuli or thoughts. It is clear that we are able to focus our attention selectively, choosing to attend to one thing rather than another. This is due to the limited capacity of our sensory and mental processes. If we did not selectively attend to the stimuli coming into our senses, we would be overloaded. We can choose which stimuli to attend to, and this choice is governed to an extent by our *arousal*, our level of interest or need. This explains the cocktail party phenomenon mentioned earlier: we can attend to one conversation over the background noise, but we may choose to switch our attention to a conversation across the room if we hear our name mentioned. Information received by sensory memories is quickly passed into a more permanent memory store, or overwritten and lost.

1.3.2 Short-term memory

Short-term memory or working memory acts as a 'scratch-pad' for temporary recall of information. It is used to store information which is only required fleetingly. For example, calculate the multiplication 35×6 in your head. The chances are that you will have done this calculation in stages, perhaps 5×6 and then 30×6 and added the results; or you may have used the fact that $6 = 2 \times 3$ and calculated $2 \times 35 = 70$ followed by 3×70. To perform calculations such as this we need to store the intermediate stages for use later. Or consider reading. In order to comprehend this sentence you need to hold in your mind the beginning of the sentence as you read the rest. Both of these tasks use short-term memory.

Short-term memory can be accessed rapidly, in the order of 70 ms. However, it also decays rapidly, meaning that information can only be held there temporarily, in the order of 200 ms.

Short-term memory also has a limited capacity. There are two basic methods for measuring memory capacity. The first involves determining the length of a sequence which can be remembered in order. The second allows items to be freely recalled in any order. Using the first measure, the average person can remember 7 ± 2 digits. This was established in experiments by Miller [234]. Try it. Look at the following number sequence:

265397620853

Now write down as much of the sequence as you can remember. Did you get it all right? If not, how many digits could you remember? If you remembered between five and nine digits your *digit span* is average.

Now try the following sequence:

44 113 245 8920

Did you recall that more easily? Here the digits are grouped or *chunked*. A generalization of the 7 ± 2 rule is that we can remember 7 ± 2 *chunks* of information. Therefore chunking information can increase the short-term memory capacity. The limited capacity of short-term memory produces a subconscious desire to create chunks, and so optimize the use of the memory. The successful formation of a chunk is known as *closure*. This process can be generalized to account for the desire to complete or close tasks held in short-term memory. If a subject fails to do this or is prevented from doing so by interference, the subject is liable to lose track of what she is doing and make consequent errors.

DESIGN FOCUS

Cashing in

Closure gives you a nice 'done it' when we complete some part of a task. At this point our minds have a tendency to flush short-term memory in order to get on with the next job. Early automatic teller machines (ATMs) gave the customer money before returning their bank card. On receiving the money the customer would reach closure and hence often forget to take the card. Modern ATMs return the card first!

The sequence of chunks given above also makes use of pattern abstraction: it is written in the form of a UK telephone number which makes it easier to remember. We may even recognize the first sets of digits as the international code for the UK and the dialing code for Leeds – chunks of information. Patterns can be useful as aids

to memory. For example, most people would have difficulty remembering the following sequence of chunks:

HEC ATR ANU PTH ETR EET

However, if you notice that by moving the last character to the first position, you get the statement 'the cat ran up the tree', the sequence is easy to recall.

In experiments where subjects were able to recall words freely, evidence shows that recall of the last words presented is better than recall of those in the middle [296]. This is known as the *recency effect*. However, if the subject is asked to perform another task between presentation and recall (for example, counting backwards) the recency effect is eliminated. The recall of the other words is unaffected. This suggests that short-term memory recall is damaged by interference of other information. However, the fact that this interference does not affect recall of earlier items provides some evidence for the existence of separate long-term and short-term memories. The early items are held in a long-term store which is unaffected by the recency effect.

Interference does not necessarily impair recall in short-term memory. Baddeley asked subjects to remember six-digit numbers and attend to sentence processing at the same time [21]. They were asked to answer questions on sentences, such as 'A precedes B: AB is true or false?'. Surprisingly, this did not result in interference, suggesting that in fact short-term memory is not a unitary system but is made up of a number of components, including a visual channel and an articulatory channel. The task of sentence processing used the visual channel, while the task of remembering digits used the articulatory channel, so interference only occurs if tasks utilize the same channel.

These findings led Baddeley to propose a model of working memory that incorporated a number of elements together with a central processing executive. This is illustrated in Figure 1.10.

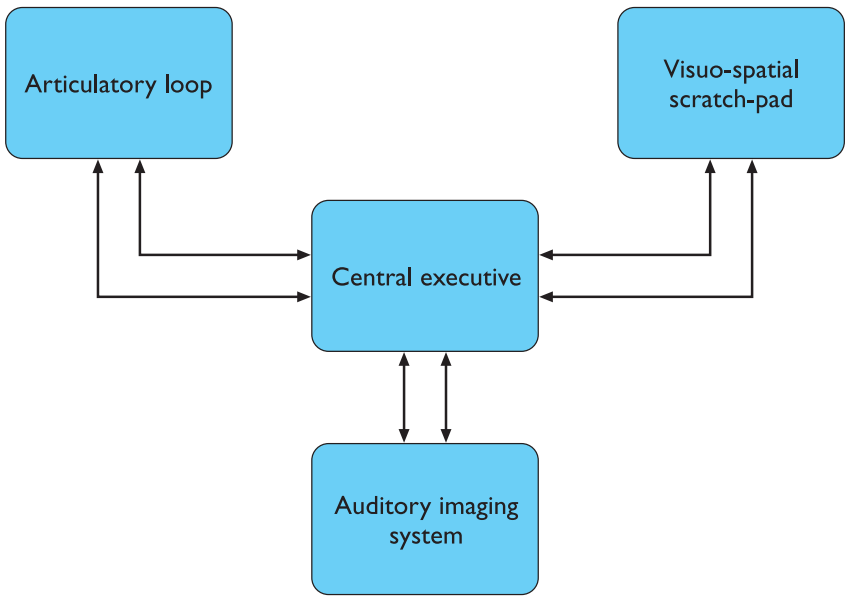


Figure 1.10 A more detailed model of short-term memory

DESIGN FOCUS

7 ± 2 revisited

When we looked at short-term memory, we noted the general rule that people can hold 7 ± 2 items or chunks of information in short-term memory. It is a principle that people tend to remember but it can be misapplied. For example, it is often suggested that this means that lists, menus and other groups of items should be designed to be no more than 7 items long. But use of menus and lists of course has little to do with short-term memory – they are available in the environment as cues and so do not need to be remembered.

On the other hand the 7 ± 2 rule would apply in command line interfaces. Imagine a scenario where a UNIX user looks up a command in the manual. Perhaps the command has a number of parameters of options, to be applied in a particular order, and it is going to be applied to several files that have long path names. The user then has to hold the command, its parameters and the file path names in short-term memory while he types them in. Here we could say that the task may cause problems if the number of items or chunks in the command line string is more than 7.

1.3.3 Long-term memory

If short-term memory is our working memory or 'scratch-pad', long-term memory is our main resource. Here we store factual information, experiential knowledge, procedural rules of behavior – in fact, everything that we 'know'. It differs from short-term memory in a number of significant ways. First, it has a huge, if not unlimited, capacity. Secondly, it has a relatively slow access time of approximately a tenth of a second. Thirdly, forgetting occurs more slowly in long-term memory, if at all. These distinctions provide further evidence of a memory structure with several parts.

Long-term memory is intended for the long-term storage of information. Information is placed there from working memory through rehearsal. Unlike working memory there is little decay: long-term recall after minutes is the same as that after hours or days.

Long-term memory structure

There are two types of long-term memory: *episodic memory* and *semantic memory*. Episodic memory represents our memory of events and experiences in a serial form. It is from this memory that we can reconstruct the actual events that took place at a given point in our lives. Semantic memory, on the other hand, is a structured record of facts, concepts and skills that we have acquired. The information in semantic memory is derived from that in our episodic memory, such that we can learn new facts or concepts from our experiences.

Semantic memory is structured in some way to allow access to information, representation of relationships between pieces of information, and inference. One model for the way in which semantic memory is structured is as a network. Items are

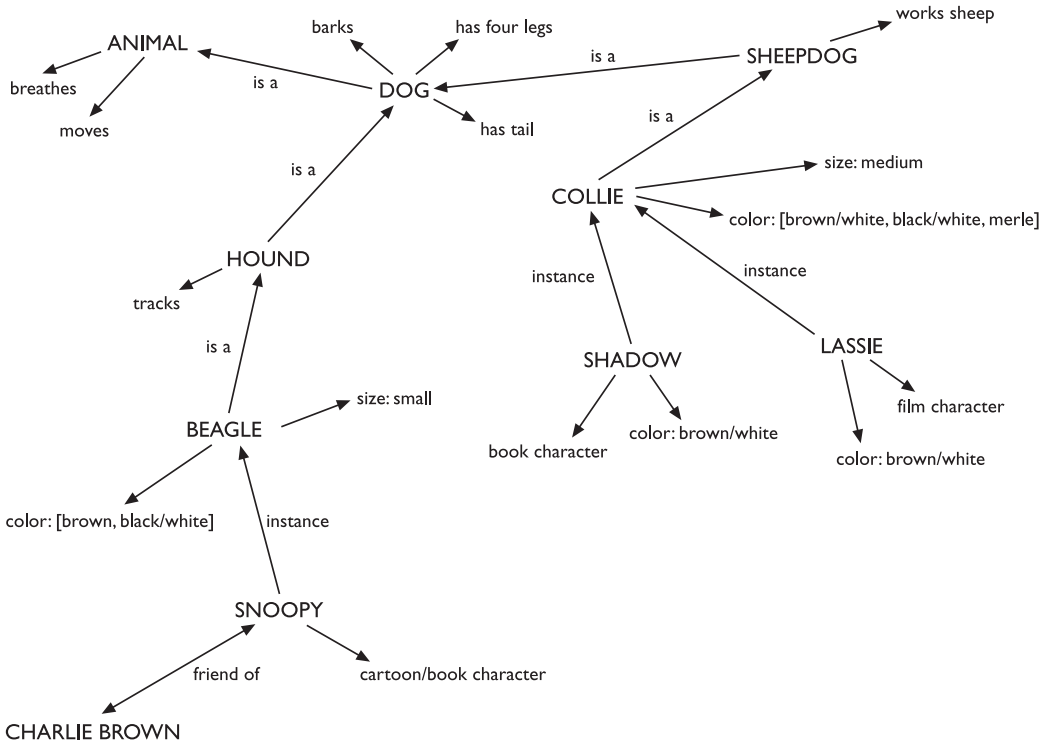


Figure 1.11 Long-term memory may store information in a semantic network

associated to each other in classes, and may inherit attributes from parent classes. This model is known as a *semantic network*. As an example, our knowledge about dogs may be stored in a network such as that shown in Figure 1.11.

Specific breed attributes may be stored with each given breed, yet general dog information is stored at a higher level. This allows us to generalize about specific cases. For instance, we may not have been told that the sheepdog Shadow has four legs and a tail, but we can infer this information from our general knowledge about sheepdogs and dogs in general. Note also that there are connections within the network which link into other domains of knowledge, for example cartoon characters. This illustrates how our knowledge is organized by association.

The viability of semantic networks as a model of memory organization has been demonstrated by Collins and Quillian [74]. Subjects were asked questions about different properties of related objects and their reaction times were measured. The types of question asked (taking examples from our own network) were 'Can a collie breathe?', 'Is a beagle a hound?' and 'Does a hound track?' In spite of the fact that the answers to such questions may seem obvious, subjects took longer to answer questions such as 'Can a collie breathe?' than ones such as 'Does a hound track?' The reason for this, it is suggested, is that in the former case subjects had to search further through the memory hierarchy to find the answer, since information is stored at its most abstract level.

A number of other memory structures have been proposed to explain how we represent and store different types of knowledge. Each of these represents a different

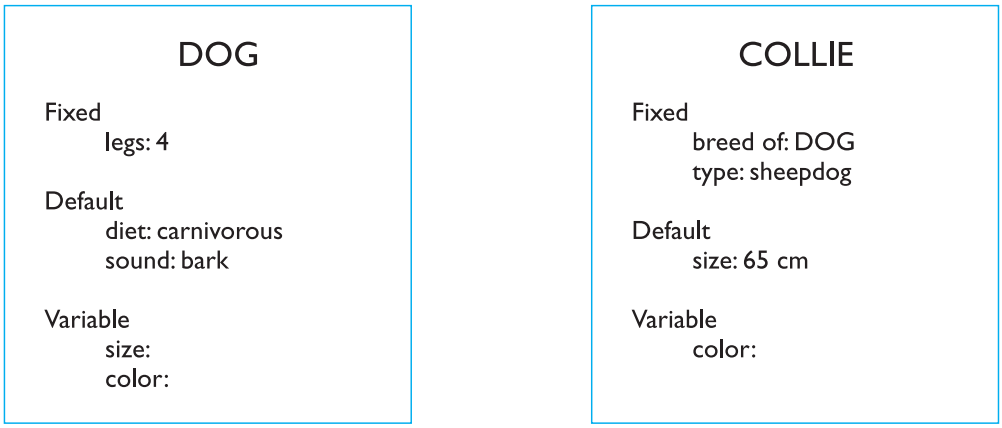


Figure 1.12 A frame-based representation of knowledge

aspect of knowledge and, as such, the models can be viewed as complementary rather than mutually exclusive. Semantic networks represent the associations and relationships between single items in memory. However, they do not allow us to model the representation of more complex objects or events, which are perhaps composed of a number of items or activities. Structured representations such as *frames* and *scripts* organize information into data structures. *Slots* in these structures allow attribute values to be added. Frame slots may contain default, fixed or variable information. A frame is instantiated when the slots are filled with appropriate values. Frames and scripts can be linked together in networks to represent hierarchical structured knowledge.

Returning to the 'dog' domain, a frame-based representation of the knowledge may look something like Figure 1.12. The fixed slots are those for which the attribute value is set, default slots represent the usual attribute value, although this may be overridden in particular instantiations (for example, the Basenji does not bark), and variable slots can be filled with particular values in a given instance. Slots can also contain procedural knowledge. Actions or operations can be associated with a slot and performed, for example, whenever the value of the slot is changed.

Frames extend semantic nets to include structured, hierarchical information. They represent knowledge items in a way which makes explicit the relative importance of each piece of information.

Scripts attempt to model the representation of stereotypical knowledge about situations. Consider the following sentence:

> John took his dog to the surgery. After seeing the vet, he left.

From our knowledge of the activities of dog owners and vets, we may fill in a substantial amount of detail. The animal was ill. The vet examined and treated the animal. John paid for the treatment before leaving. We are less likely to assume the alternative reading of the sentence, that John took an instant dislike to the vet on sight and did not stay long enough to talk to him!

Script for a visit to the vet

Entry conditions:	*dog ill* *vet open* *owner has money*	Roles:	*vet examines* *diagnoses* *treats* *owner brings dog in* *pays* *takes dog out*
Result:	*dog better* *owner poorer* *vet richer*	Scenes:	*arriving at reception* *waiting in room* *examination* *paying*
Props:	*examination table* *medicine* *instruments*	Tracks:	*dog needs medicine* *dog needs operation*

Figure 1.13 A script for visiting the vet

A script represents this default or stereotypical information, allowing us to interpret partial descriptions or cues fully. A script comprises a number of elements, which, like slots, can be filled with appropriate information:

Entry conditions Conditions that must be satisfied for the script to be activated.

Result Conditions that will be true after the script is terminated.

Props Objects involved in the events described in the script.

Roles Actions performed by particular participants.

Scenes The sequences of events that occur.

Tracks A variation on the general pattern representing an alternative scenario.

An example script for going to the vet is shown in Figure 1.13.

A final type of knowledge representation which we hold in memory is the representation of procedural knowledge, our knowledge of how to do something. A common model for this is the production system. Condition–action rules are stored in long-term memory. Information coming into short-term memory can match a condition in one of these rules and result in the action being executed. For example, a pair of production rules might be

IF dog is wagging tail
THEN pat dog

IF dog is growling
THEN run away

If we then meet a growling dog, the condition in the second rule is matched, and we respond by turning tail and running. (Not to be recommended by the way!)

Long-term memory processes

So much for the structure of memory, but what about the processes which it uses? There are three main activities related to long-term memory: storage or remembering of information, forgetting and information retrieval. We shall consider each of these in turn.

First, how does information get into long-term memory and how can we improve this process? Information from short-term memory is stored in long-term memory by rehearsal. The repeated exposure to a stimulus or the rehearsal of a piece of information transfers it into long-term memory.

This process can be optimized in a number of ways. Ebbinghaus performed numerous experiments on memory, using himself as a subject [117]. In these experiments he tested his ability to learn and repeat nonsense syllables, comparing his recall minutes, hours and days after the learning process. He discovered that the amount learned was directly proportional to the amount of time spent learning. This is known as the *total time hypothesis*. However, experiments by Baddeley and others suggest that learning time is most effective if it is distributed over time [22]. For example, in an experiment in which Post Office workers were taught to type, those whose training period was divided into weekly sessions of one hour performed better than those who spent two or four hours a week learning (although the former obviously took more weeks to complete their training). This is known as the *distribution of practice effect*.

However, repetition is not enough to learn information well. If information is not meaningful it is more difficult to remember. This is illustrated by the fact that it is more difficult to remember a set of words representing concepts than a set of words representing objects. Try it. First try to remember the words in list A and test yourself.

List A: Faith Age Cold Tenet Quiet Logic Idea Value Past Large

Now try list B.

List B: Boat Tree Cat Child Rug Plate Church Gun Flame Head

The second list was probably easier to remember than the first since you could visualize the objects in the second list.

Sentences are easier still to memorize. Bartlett performed experiments on remembering meaningful information (as opposed to meaningless such as Ebbinghaus used) [28]. In one such experiment he got subjects to learn a story about an unfamiliar culture and then retell it. He found that subjects would retell the story replacing unfamiliar words and concepts with words which were meaningful to them. Stories were effectively translated into the subject's own culture. This is related to the semantic structuring of long-term memory: if information is meaningful and familiar, it can be related to existing structures and more easily incorporated into memory.

Memorable or secure?

As online activities become more widespread, people are having to remember more and more access information, such as passwords and security checks. The average active internet user may have separate passwords and user names for several email accounts, mailing lists, e-shopping sites, e-banking, online auctions and more! Remembering these passwords is not easy.

From a security perspective it is important that passwords are random. Words and names are very easy to crack, hence the recommendation that passwords are frequently changed and constructed from random strings of letters and numbers. But in reality these are the hardest things for people to commit to memory. Hence many people will use the same password for all their online activities (rarely if ever changing it) and will choose a word or a name that is easy for them to remember, in spite of the obviously increased security risks. Security here is in conflict with memorability!

A solution to this is to construct a nonsense password out of letters or numbers that will have meaning to you but will not make up a word in a dictionary (e.g. initials of names, numbers from significant dates or postcodes, and so on). Then what is remembered is the meaningful rule for constructing the password, and not a meaningless string of alphanumeric characters.

So if structure, familiarity and concreteness help us in learning information, what causes us to lose this information, to forget? There are two main theories of forgetting: *decay* and *interference*. The first theory suggests that the information held in long-term memory may eventually be forgotten. Ebbinghaus concluded from his experiments with nonsense syllables that information in memory decayed logarithmically, that is that it was lost rapidly to begin with, and then more slowly. *Jost's law*, which follows from this, states that if two memory traces are equally strong at a given time the older one will be more durable.

The second theory is that information is lost from memory through interference. If we acquire new information it causes the loss of old information. This is termed *retroactive interference*. A common example of this is the fact that if you change telephone numbers, learning your new number makes it more difficult to remember your old number. This is because the new association masks the old. However, sometimes the old memory trace breaks through and interferes with new information. This is called *proactive inhibition*. An example of this is when you find yourself driving to your old house rather than your new one.

Forgetting is also affected by emotional factors. In experiments, subjects given emotive words and non-emotive words found the former harder to remember in the short term but easier in the long term. Indeed, this observation tallies with our experience of selective memory. We tend to remember positive information rather than negative (hence nostalgia for the 'good old days'), and highly emotive events rather than mundane.

It is debatable whether we ever actually forget anything or whether it just becomes increasingly difficult to access certain items from memory. This question is in some ways moot since it is impossible to prove that we *do* forget: appearing to have forgotten something may just be caused by not being able to retrieve it! However, there is evidence to suggest that we may not lose information completely from long-term memory. First, proactive inhibition demonstrates the recovery of old information even after it has been 'lost' by interference. Secondly, there is the 'tip of the tongue' experience, which indicates that some information is present but cannot be satisfactorily accessed. Thirdly, information may not be recalled but may be recognized, or may be recalled only with prompting.

This leads us to the third process of memory: information retrieval. Here we need to distinguish between two types of information retrieval, recall and recognition. In recall the information is reproduced from memory. In recognition, the presentation of the information provides the knowledge that the information has been seen before. Recognition is the less complex cognitive activity since the information is provided as a cue.

However, recall can be assisted by the provision of retrieval cues, which enable the subject quickly to access the information in memory. One such cue is the use of categories. In an experiment subjects were asked to recall lists of words, some of which were organized into categories and some of which were randomly organized. The words that were related to a category were easier to recall than the others [38]. Recall is even more successful if subjects are allowed to categorize their own lists of words during learning. For example, consider the following list of words:

child red plane dog friend blood cold tree big angry

Now make up a story that links the words using as vivid imagery as possible. Now try to recall as many of the words as you can. Did you find this easier than the previous experiment where the words were unrelated?

The use of vivid imagery is a common cue to help people remember information. It is known that people often visualize a scene that is described to them. They can then answer questions based on their visualization. Indeed, subjects given a description of a scene often embellish it with additional information. Consider the following description and imagine the scene:

> The engines roared above the noise of the crowd. Even in the blistering heat people rose to their feet and waved their hands in excitement. The flag fell and they were off. Within seconds the car had pulled away from the pack and was careering round the bend at a desperate pace. Its wheels momentarily left the ground as it cornered. Coming down the straight the sun glinted on its shimmering paint. The driver gripped the wheel with fierce concentration. Sweat lay in fine drops on his brow.

Without looking back to the passage, what color is the car?

If you could answer that question you have visualized the scene, including the car's color. In fact, the color of the car is not mentioned in the description at all.

Improve your memory

Many people can perform astonishing feats of memory: recalling the sequence of cards in a pack (or multiple packs – up to six have been reported), or recounting π to 1000 decimal places, for example. There are also adverts to 'Improve Your Memory' (usually leading to success, or wealth, or other such inducement), and so the question arises: can you improve your memory abilities? The answer is yes; this exercise shows you one technique.

Look at the list below of numbers and associated words:

1	bun	6	sticks
2	shoe	7	heaven
3	tree	8	gate
4	door	9	wine
5	hive	10	hen

Notice that the words sound similar to the numbers. Now think about the words one at a time and visualize them, in as much detail as possible. For example, for '1', think of a large, sticky iced bun, the base spiralling round and round, with raisins in it, covered in sweet, white, gooey icing. Now do the rest, using as much visualization as you can muster: imagine how things would look, smell, taste, sound, and so on.

This is your reference list, and you need to know it off by heart.

Having learnt it, look at a pile of at least a dozen odd items collected together by a colleague. The task is to look at the collection of objects for only 30 seconds, and then list as many as possible without making a mistake or viewing the collection again. Most people can manage between five and eight items, if they do not know any memory-enhancing techniques like the following.

Mentally pick one (say, for example, a paper clip), and call it number one. Now visualize it interacting with the bun. It can get stuck into the icing on the top of the bun, and make your fingers all gooey and sticky when you try to remove it. If you ate the bun without noticing, you'd get a crunched tooth when you bit into it – imagine how that would feel. When you've really got a graphic scenario developed, move on to the next item, call it number two, and again visualize it interacting with the reference item, shoe. Continue down your list, until you have done 10 things.

This should take you about the 30 seconds allowed. Then hide the collection and try and recall the numbers in order, the associated reference word, and then the image associated with that word. You should find that you can recall the 10 associated items practically every time. The technique can be easily extended by extending your reference list.

1.4 THINKING: REASONING AND PROBLEM SOLVING

We have considered how information finds its way into and out of the human system and how it is stored. Finally, we come to look at how it is processed and manipulated. This is perhaps the area which is most complex and which separates

humans from other information-processing systems, both artificial and natural. Although it is clear that animals receive and store information, there is little evidence to suggest that they can use it in quite the same way as humans. Similarly, artificial intelligence has produced machines which can see (albeit in a limited way) and store information. But their ability to use that information is limited to small domains.

Humans, on the other hand, are able to use information to reason and solve problems, and indeed do these activities when the information is partial or unavailable. Human thought is conscious and self-aware: while we may not always be able to identify the processes we use, we can identify the products of these processes, our thoughts. In addition, we are able to think about things of which we have no experience, and solve problems which we have never seen before. How is this done?

Thinking can require different amounts of knowledge. Some thinking activities are very directed and the knowledge required is constrained. Others require vast amounts of knowledge from different domains. For example, performing a subtraction calculation requires a relatively small amount of knowledge, from a constrained domain, whereas understanding newspaper headlines demands knowledge of politics, social structures, public figures and world events.

In this section we will consider two categories of thinking: reasoning and problem solving. In practice these are not distinct since the activity of solving a problem may well involve reasoning and vice versa. However, the distinction is a common one and is helpful in clarifying the processes involved.

1.4.1 Reasoning

Reasoning is the process by which we use the knowledge we have to draw conclusions or infer something new about the domain of interest. There are a number of different types of reasoning: *deductive*, *inductive* and *abductive*. We use each of these types of reasoning in everyday life, but they differ in significant ways.

Deductive reasoning

Deductive reasoning derives the logically necessary conclusion from the given premises. For example,

If it is Friday then she will go to work
It is Friday
Therefore she will go to work.

It is important to note that this is the *logical* conclusion from the premises; it does not necessarily have to correspond to our notion of truth. So, for example,

If it is raining then the ground is dry
It is raining
Therefore the ground is dry.

is a perfectly valid deduction, even though it conflicts with our knowledge of what is true in the world.

Deductive reasoning is therefore often misapplied. Given the premises

Some people are babies
Some babies cry

many people will infer that 'Some people cry'. This is in fact an invalid deduction since we are not told that all babies are people. It is therefore logically possible that the babies who cry are those who are not people.

It is at this point, where truth and validity clash, that human deduction is poorest. One explanation for this is that people bring their world knowledge into the reasoning process. There is good reason for this. It allows us to take short cuts which make dialog and interaction between people informative but efficient. We assume a certain amount of shared knowledge in our dealings with each other, which in turn allows us to interpret the inferences and deductions implied by others. If validity rather than truth was preferred, all premises would have to be made explicit.

Inductive reasoning

Induction is generalizing from cases we have seen to infer information about cases we have not seen. For example, if every elephant we have ever seen has a trunk, we infer that all elephants have trunks. Of course, this inference is unreliable and cannot be proved to be true; it can only be proved to be false. We can disprove the inference simply by producing an elephant without a trunk. However, we can never prove it true because, no matter how many elephants with trunks we have seen or are known to exist, the next one we see may be trunkless. The best that we can do is gather evidence to support our inductive inference.

In spite of its unreliability, induction is a useful process, which we use constantly in learning about our environment. We can never see all the elephants that have ever lived or will ever live, but we have certain knowledge about elephants which we are prepared to trust for all practical purposes, which has largely been inferred by induction. Even if we saw an elephant without a trunk, we would be unlikely to move from our position that 'All elephants have trunks', since we are better at using positive than negative evidence. This is illustrated in an experiment first devised by Wason [365]. You are presented with four cards as in Figure 1.14. Each card has a number on one side and a letter on the other. Which cards would you need to pick up to test the truth of the statement 'If a card has a vowel on one side it has an even number on the other'?

A common response to this (was it yours?) is to check the E and the 4. However, this uses only positive evidence. In fact, to test the truth of the statement we need to check negative evidence: if we can find a card which has an odd number on one side and a vowel on the other we have disproved the statement. We must therefore check E and 7. (It does not matter what is on the other side of the other cards: the statement does not say that all even numbers have vowels, just that all vowels have even numbers.)

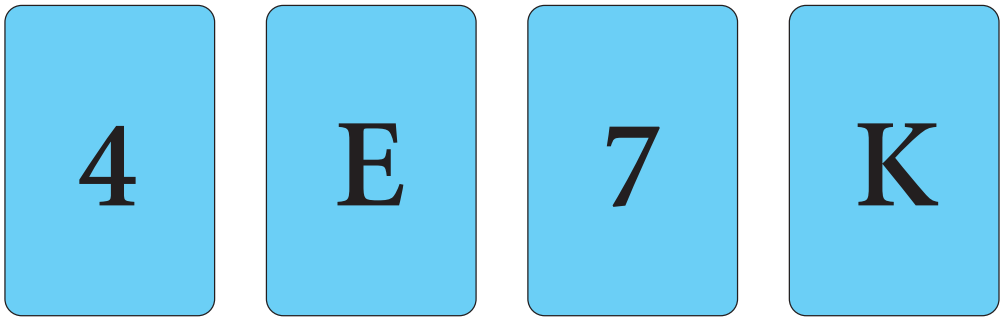


Figure 1.14 Wason's cards

Filling the gaps

Look again at Wason's cards in Figure 1.14. In the text we say that you only need to check the E and the 7. This is correct, but only because we very carefully stated in the text that 'each card has a number on one side and a letter on the other'. If the problem were stated without that condition then the K would also need to be examined in case it has a vowel on the other side. In fact, when the problem is so stated, even the most careful subjects ignore this possibility. Why? Because the nature of the problem implicitly suggests that each card has a number on one side and a letter on the other.

This is similar to the embellishment of the story at the end of Section 1.3.3. In fact, we constantly fill in gaps in the evidence that reaches us through our senses. Although this can lead to errors in our reasoning it is also essential for us to function. In the real world we rarely have all the evidence necessary for logical deductions and at all levels of perception and reasoning we fill in details in order to allow higher levels of reasoning to work.

Abductive reasoning

The third type of reasoning is abduction. Abduction reasons from a fact to the action or state that caused it. This is the method we use to derive explanations for the events we observe. For example, suppose we know that Sam always drives too fast when she has been drinking. If we see Sam driving too fast we may infer that she has been drinking. Of course, this too is unreliable since there may be another reason why she is driving fast: she may have been called to an emergency, for example.

In spite of its unreliability, it is clear that people do infer explanations in this way, and hold onto them until they have evidence to support an alternative theory or explanation. This can lead to problems in using interactive systems. If an event always follows an action, the user will infer that the event is caused by the action unless evidence to the contrary is made available. If, in fact, the event and the action are unrelated, confusion and even error often result.

1.4.2 Problem solving

If reasoning is a means of inferring new information from what is already known, problem solving is the process of finding a solution to an unfamiliar task, using the knowledge we have. Human problem solving is characterized by the ability to adapt the information we have to deal with new situations. However, often solutions seem to be original and creative. There are a number of different views of how people solve problems. The earliest, dating back to the first half of the twentieth century, is the *Gestalt* view that problem solving involves both reuse of knowledge and insight. This has been largely superseded but the questions it was trying to address remain and its influence can be seen in later research. A second major theory, proposed in the 1970s by Newell and Simon, was the *problem space theory*, which takes the view that the mind is a limited information processor. Later variations on this drew on the earlier theory and attempted to reinterpret Gestalt theory in terms of information-processing theories. We will look briefly at each of these views.

Gestalt theory

Gestalt psychologists were answering the claim, made by behaviorists, that problem solving is a matter of reproducing known responses or trial and error. This explanation was considered by the Gestalt school to be insufficient to account for human problem-solving behavior. Instead, they claimed, problem solving is both *productive* and *reproductive*. Reproductive problem solving draws on previous experience as the behaviorists claimed, but productive problem solving involves insight and restructuring of the problem. Indeed, reproductive problem solving could be a hindrance to finding a solution, since a person may 'fixate' on the known aspects of the problem and so be unable to see novel interpretations that might lead to a solution.

Gestalt psychologists backed up their claims with experimental evidence. Kohler provided evidence of apparent insight being demonstrated by apes, which he observed joining sticks together in order to reach food outside their cages [202]. However, this was difficult to verify since the apes had once been wild and so could have been using previous knowledge.

Other experiments observed human problem-solving behavior. One well-known example of this is Maier's *pendulum problem* [224]. The problem was this: the subjects were in a room with two pieces of string hanging from the ceiling. Also in the room were other objects including pliers, poles and extensions. The task set was to tie the pieces of string together. However, they were too far apart to catch hold of both at once. Although various solutions were proposed by subjects, few chose to use the weight of the pliers as a pendulum to 'swing' the strings together. However, when the experimenter brushed against the string, setting it in motion, this solution presented itself to subjects. Maier interpreted this as an example of productive restructuring. The movement of the string had given insight and allowed the subjects to see the problem in a new way. The experiment also illustrates fixation: subjects were initially unable to see beyond their view of the role or use of a pair of pliers.

Although Gestalt theory is attractive in terms of its description of human problem solving, it does not provide sufficient evidence or structure to support its theories. It does not explain when restructuring occurs or what insight is, for example. However, the move away from behaviorist theories was helpful in paving the way for the information-processing theory that was to follow.

Problem space theory

Newell and Simon proposed that problem solving centers on the problem space. The problem space comprises *problem states*, and problem solving involves generating these states using legal state transition operators. The problem has an initial state and a goal state and people use the operators to move from the former to the latter. Such problem spaces may be huge, and so *heuristics* are employed to select appropriate operators to reach the goal. One such heuristic is *means–ends analysis*. In means–ends analysis the initial state is compared with the goal state and an operator chosen to reduce the difference between the two. For example, imagine you are reorganizing your office and you want to move your desk from the north wall of the room to the window. Your initial state is that the desk is at the north wall. The goal state is that the desk is by the window. The main difference between these two is the location of your desk. You have a number of operators which you can apply to moving things: you can carry them or push them or drag them, etc. However, you know that to carry something it must be light and that your desk is heavy. You therefore have a new subgoal: to make the desk light. Your operators for this may involve removing drawers, and so on.

An important feature of Newell and Simon's model is that it operates within the constraints of the human processing system, and so searching the problem space is limited by the capacity of short-term memory, and the speed at which information can be retrieved. Within the problem space framework, experience allows us to solve problems more easily since we can structure the problem space appropriately and choose operators efficiently.

Newell and Simon's theory, and their *General Problem Solver* model which is based on it, have largely been applied to problem solving in well-defined domains, for example solving puzzles. These problems may be unfamiliar but the knowledge that is required to solve them is present in the statement of the problem and the expected solution is clear. In real-world problems finding the knowledge required to solve the problem may be part of the problem, or specifying the goal may be difficult. Problems such as these require significant domain knowledge: for example, to solve a programming problem you need knowledge of the language and the domain in which the program operates. In this instance specifying the goal clearly may be a significant part of solving the problem.

However, the problem space framework provides a clear theory of problem solving, which can be extended, as we shall see when we look at skill acquisition in the next section, to deal with knowledge-intensive problem solving. First we will look briefly at the use of analogy in problem solving.

Worked exercise *Identify the goals and operators involved in the problem 'delete the second paragraph of the document' on a word processor. Now use a word processor to delete a paragraph and note your actions, goals and subgoals. How well did they match your earlier description?*

Answer Assume you have a document open and you are at some arbitrary position within it. You also need to decide which operators are available and what their preconditions and results are. Based on an imaginary word processor we assume the following operators (you may wish to use your own WP package):

Operator	Precondition	Result
delete_paragraph	Cursor at start of paragraph	Paragraph deleted
move_to_paragraph	Cursor anywhere in document	Cursor moves to start of next paragraph (except where there is no next paragraph when no effect)
move_to_start	Cursor anywhere in document	Cursor at start of document

Goal: *delete second paragraph in document*
Looking at the operators an obvious one to resolve this goal is delete_paragraph which has the precondition 'cursor at start of paragraph'. We therefore have a new subgoal: *move_to_paragraph*. The precondition is 'cursor anywhere in document' (which we can meet) but we want the second paragraph so we must initially be in the first.

We set up a new subgoal, move_to_start, with precondition 'cursor anywhere in document' and result 'cursor at start of document'. We can then apply *move_to_paragraph* and finally *delete_paragraph*.

We assume some knowledge here (that the second paragraph is the paragraph after the first one).

Analogy in problem solving

A third element of problem solving is the use of analogy. Here we are interested in how people solve novel problems. One suggestion is that this is done by mapping knowledge relating to a similar known domain to the new problem – called *analogical mapping*. Similarities between the known domain and the new one are noted and operators from the known domain are transferred to the new one.

This process has been investigated using analogous stories. Gick and Holyoak [149] gave subjects the following problem:

> A doctor is treating a malignant tumor. In order to destroy it he needs to blast it with high-intensity rays. However, these will also destroy the healthy tissue surrounding the tumor. If he lessens the rays' intensity the tumor will remain. How does he destroy the tumor?

The solution to this problem is to fire low-intensity rays from different directions converging on the tumor. That way, the healthy tissue receives harmless low-intensity rays while the tumor receives the rays combined, making a high-intensity dose. The investigators found that only 10% of subjects reached this solution without help. However, this rose to 80% when they were given this analogous story and told that it may help them:

> A general is attacking a fortress. He can't send all his men in together as the roads are mined to explode if large numbers of men cross them. He therefore splits his men into small groups and sends them in on separate roads.

In spite of this, it seems that people often miss analogous information, unless it is semantically close to the problem domain. When subjects were not told to use the story, many failed to see the analogy. However, the number spotting the analogy rose when the story was made semantically close to the problem, for example a general using rays to destroy a castle.

The use of analogy is reminiscent of the Gestalt view of productive restructuring and insight. Old knowledge is used to solve a new problem.

1.4.3 Skill acquisition

All of the problem solving that we have considered so far has concentrated on handling unfamiliar problems. However, for much of the time, the problems that we face are not completely new. Instead, we gradually acquire skill in a particular domain area. But how is such skill acquired and what difference does it make to our problem-solving performance? We can gain insight into how skilled behavior works, and how skills are acquired, by considering the difference between novice and expert behavior in given domains.

Chess: of human and artificial intelligence

A few years ago, Deep Blue, a chess-playing computer, beat Gary Kasparov, the world's top Grand Master, in a full tournament. This was the long-awaited breakthrough for the artificial intelligence (AI) community, who have traditionally seen chess as the ultimate test of their art. However, despite the fact that computer chess programs can play at Grand Master level against human players, this does not mean they play in the same way. For each move played, Deep Blue investigated many millions of alternative moves and counter-moves. In contrast, a human chess player will only consider a few dozen. But, if the human player is good, these will usually be the right few dozen. The ability to spot patterns allows a human to address a problem with far less effort than a brute force approach. In chess, the number of moves is such that finally brute force, applied fast enough, has overcome human pattern-matching skill. In Go, which has far more possible moves, computer programs do not even reach a good club level of play. Many models of the mental processes have been heavily influenced by computation. It is worth remembering that although there are similarities, computer 'intelligence' is very different from that of humans.

A commonly studied domain is chess playing. It is particularly suitable since it lends itself easily to representation in terms of problem space theory. The initial state is the opening board position; the goal state is one player checkmating the other; operators to move states are legal moves of chess. It is therefore possible to examine skilled behavior within the context of the problem space theory of problem solving.

Studies of chess players by DeGroot, Chase and Simon, among others, produced some interesting observations [64, 65, 88, 89]. In all the experiments the behavior of chess masters was compared with less experienced chess players. The first observation was that players did not consider large numbers of moves in choosing their move, nor did they look ahead more than six moves (often far fewer). Masters considered no more alternatives than the less experienced, but they took less time to make a decision and produced better moves.

So what makes the difference between skilled and less skilled behavior in chess? It appears that chess masters remember board configurations and good moves associated with them. When given actual board positions to remember, masters are much better at reconstructing the board than the less experienced. However, when given random configurations (which were unfamiliar), the groups of players were equally bad at reconstructing the positions. It seems therefore that expert players 'chunk' the board configuration in order to hold it in short-term memory. Expert players use larger chunks than the less experienced and can therefore remember more detail.

This behavior is also seen among skilled computer programmers. They can also reconstruct programs more effectively than novices since they have the structures available to build appropriate chunks. They acquire plans representing code to solve particular problems. When that problem is encountered in a new domain or new program they will recall that particular plan and reuse it.

Another observed difference between skilled and less skilled problem solving is in the way that different problems are grouped. Novices tend to group problems according to superficial characteristics such as the objects or features common to both. Experts, on the other hand, demonstrate a deeper understanding of the problems and group them according to underlying conceptual similarities which may not be at all obvious from the problem descriptions.

Each of these differences stems from a better encoding of knowledge in the expert: information structures are fine tuned at a deep level to enable efficient and accurate retrieval. But how does this happen? How is skill such as this acquired? One model of skill acquisition is Anderson's *ACT** model [14]. ACT* identifies three basic levels of skill:

1. The learner uses general-purpose rules which interpret facts about a problem. This is slow and demanding on memory access.
2. The learner develops rules specific to the task.
3. The rules are tuned to speed up performance.

General mechanisms are provided to account for the transitions between these levels. For example, *proceduralization* is a mechanism to move from the first to the second. It removes the parts of the rule which demand memory access and replaces

variables with specific values. *Generalization*, on the other hand, is a mechanism which moves from the second level to the third. It generalizes from the specific cases to general properties of those cases. Commonalities between rules are condensed to produce a general-purpose rule.

These are best illustrated by example. Imagine you are learning to cook. Initially you may have a general rule to tell you how long a dish needs to be in the oven, and a number of explicit representations of dishes in memory. You can instantiate the rule by retrieving information from memory.

```
IF cook[type, ingredients, time]
THEN
    cook for: time
cook[casserole, [chicken,carrots,potatoes], 2 hours]
cook[casserole, [beef,dumplings,carrots], 2 hours]
cook[cake, [flour,sugar,butter,eggs], 45 mins]
```

Gradually your knowledge becomes proceduralized and you have specific rules for each case:

```
IF type is casserole
AND ingredients are [chicken,carrots,potatoes]
THEN
    cook for: 2 hours
IF type is casserole
AND ingredients are [beef,dumplings,carrots]
THEN
    cook for: 2 hours
IF type is cake
AND ingredients are [flour,sugar,butter,eggs]
THEN
    cook for: 45 mins
```

Finally, you may generalize from these rules to produce general-purpose rules, which exploit their commonalities:

```
IF type is casserole
AND ingredients are ANYTHING
THEN
    cook for: 2 hours
```

The first stage uses knowledge extensively. The second stage relies upon known procedures. The third stage represents skilled behavior. Such behavior may in fact become automatic and as such be difficult to make explicit. For example, think of an activity at which you are skilled, perhaps driving a car or riding a bike. Try to describe to someone the exact procedure which you go through to do this. You will find this quite difficult. In fact experts tend to have to rehearse their actions mentally in order to identify exactly what they do. Such skilled behavior is efficient but may cause errors when the context of the activity changes.

1.4.4 Errors and mental models

Human capability for interpreting and manipulating information is quite impressive. However, we do make mistakes. Some are trivial, resulting in no more than temporary inconvenience or annoyance. Others may be more serious, requiring substantial effort to correct. Occasionally an error may have catastrophic effects, as we see when 'human error' results in a plane crash or nuclear plant leak.

Why do we make mistakes and can we avoid them? In order to answer the latter part of the question we must first look at what is going on when we make an error. There are several different types of error. As we saw in the last section some errors result from changes in the context of skilled behavior. If a pattern of behavior has become automatic and we change some aspect of it, the more familiar pattern may break through and cause an error. A familiar example of this is where we intend to stop at the shop on the way home from work but in fact drive past. Here, the activity of driving home is the more familiar and overrides the less familiar intention.

Other errors result from an incorrect understanding, or model, of a situation or system. People build their own theories to understand the causal behavior of systems. These have been termed *mental models*. They have a number of characteristics. Mental models are often partial: the person does not have a full understanding of the working of the whole system. They are unstable and are subject to change. They can be internally inconsistent, since the person may not have worked through the logical consequences of their beliefs. They are often unscientific and may be based on superstition rather than evidence. Often they are based on an incorrect interpretation of the evidence.

DESIGN FOCUS

Human error and false memories

In the second edition of this book, one of the authors added the following story:

> During the Second World War a new cockpit design was introduced for Spitfires. The pilots were trained and flew successfully during training, but would unaccountably bail out when engaged in dog fights. The new design had exchanged the positions of the gun trigger and ejector controls. In the heat of battle the old responses resurfaced and the pilots ejected. Human error, yes, but the designer's error, not the pilot's.

It is a good story, but after the book was published we got several emails saying 'Spitfires didn't have ejector seats'. It was Kai-Mikael Jää-Aro who was able to find what may have been the original to the story (and incidentally inform us what model of Spitfire was in our photo and who the pilot was!). He pointed us to and translated the story of Sierra 44, an S35E Draken reconnaissance aircraft.[1] The full story involves just about every perceptual and cognitive error imaginable, but the point that links to

1. Pej Kristoffersson, 1984. Sigurd 44 – Historien om hur man gör bort sig så att det märks by, *Flygrevyn* 2/1984, pp. 44–6.

the (false) Spitfire story is that in the Draken the red buttons for releasing the fuel 'drop' tanks and for the canopy release differed only in very small writing. In an emergency (burning fuel tanks) the pilot accidentally released the canopy and so ended up flying home cabriolet style.

There is a second story of human error here – the author's memory. When the book was written he could not recall where he had come across the story but was convinced it was to do with a Spitfire. It may be that he had been told the story by someone else who had got it mixed up, but it is as likely that he simply remembered the rough outline of the story and then 'reconstructed' the rest. In fact that is exactly how our memories work. Our brains do not bother to lay down every little detail, but when we 'remember' we rebuild what the incident 'must have been' using our world knowledge. This process is completely unconscious and can lead to what are known as *false memories*. This is particularly problematic in witness statements in criminal trials as early questioning by police or lawyers can unintentionally lead to witnesses being sure they have seen things that they have not. Numerous controlled psychological experiments have demonstrated this effect which furthermore is strongly influenced by biasing factors such as the race of supposed criminals.

To save his blushes we have not said here which author's failing memory was responsible for the Spitfire story, but you can read more on this story and also find who it was on the book website at: /e3/online/spitfire/

Courtesy of popperfoto.com

Assuming a person builds a mental model of the system being dealt with, errors may occur if the actual operation differs from the mental model. For example, on one occasion we were staying in a hotel in Germany, attending a conference. In the lobby of the hotel was a lift. Beside the lift door was a button. Our model of the system, based on previous experience of lifts, was that the button would call the lift. We pressed the button and the lobby light went out! In fact the button was a light switch and the lift button was on the inside rim of the lift, hidden from view.

Although both the light switch and the lift button were inconsistent with our mental models of these controls, we would probably have managed if they had been encountered separately. If there had been no button beside the lift we would have looked more closely and found the one on the inner rim. But since the light switch reflected our model of a lift button we looked no further. During our stay we observed many more new guests making the same error.

This illustrates the importance of a correct mental model and the dangers of ignoring conventions. There are certain conventions that we use to interpret the world and ideally designs should support these. If these are to be violated, explicit support must be given to enable us to form a correct mental model. A label on the button saying 'light switch' would have been sufficient.

1.5 EMOTION

So far in this chapter we have concentrated on human perceptual and cognitive abilities. But human experience is far more complex than this. Our emotional response to situations affects how we perform. For example, positive emotions enable us to think more creatively, to solve complex problems, whereas negative emotion pushes us into narrow, focussed thinking. A problem that may be easy to solve when we are relaxed, will become difficult if we are frustrated or afraid.

Psychologists have studied emotional response for decades and there are many theories as to what is happening when we feel an emotion and why such a response occurs. More than a century ago, William James proposed what has become known as the James–Lange theory (Lange was a contemporary of James whose theories were similar): that emotion was the interpretation of a physiological response, rather than the other way around. So while we may feel that we respond *to* an emotion, James contended that we respond physiologically to a stimulus and interpret that as emotion:

> Common sense says, we lose our fortune, are sorry and weep; we meet a bear, are frightened and run; we are insulted by a rival, are angry and strike. The hypothesis here . . . is that we feel sorry because we cry, angry because we strike, afraid because we tremble.
>
> (W. James, *Principles of Psychology*, page 449. Henry Holt, New York, 1890.)

Others, however, disagree. Cannon [54a], for example, argued that our physiological processes are in fact too slow to account for our emotional reactions, and that the physiological responses for some emotional states are too similar (e.g. anger and fear), yet they can be easily distinguished. Experience in studies with the use of drugs that stimulate broadly the same physiological responses as anger or fear seems to support this: participants reported physical symptoms but not the emotion, which suggests that emotional response is more than a recognition of physiological changes.

Schachter and Singer [312a] proposed a third interpretation: that emotion results from a person evaluating physical responses in the light of the whole situation. So whereas the same physiological response can result from a range of different situations, the emotion that is felt is based on a cognitive evaluation of the circumstance and will depend on what the person attributes this to. So the same physiological response of a pounding heart will be interpreted as excitement if we are in a competition and fear if we find ourselves under attack.

Whatever the exact process, what is clear is that emotion involves both physical and cognitive events. Our body responds biologically to an external stimulus and we interpret that in some way as a particular emotion. That biological response – known as *affect* – changes the way we deal with different situations, and this has an impact on the way we interact with computer systems. As Donald Norman says:

> Negative affect can make it harder to do even easy tasks; positive affect can make it easier to do difficult tasks.
>
> (D. A. Norman, Emotion and design: attractive things work better. *Interactions Magazine*, ix(4): 36–42, 2002.)

So what are the implications of this for design? It suggests that in situations of stress, people will be less able to cope with complex problem solving or managing difficult interfaces, whereas if people are relaxed they will be more forgiving of limitations in the design. This does not give us an excuse to design bad interfaces but does suggest that if we build interfaces that promote positive responses – for example by using aesthetics or reward – then they are likely be more successful.

1.6 INDIVIDUAL DIFFERENCES

In this chapter we have been discussing humans in general. We have made the assumption that everyone has similar capabilities and limitations and that we can therefore make generalizations. To an extent this is true: the psychological principles and properties that we have discussed apply to the majority of people. Notwithstanding this, we should remember that, although we share processes in common, humans, and therefore users, are not all the same. We should be aware of individual differences so that we can account for them as far as possible within our designs. These differences may be long term, such as sex, physical capabilities and intellectual capabilities. Others are shorter term and include the effect of stress or fatigue on the user. Still others change through time, such as age.

These differences should be taken into account in our designs. It is useful to consider, for any design decision, if there are likely to be users within the target group who will be adversely affected by our decision. At the extremes a decision may exclude a section of the user population. For example, the current emphasis on visual interfaces excludes those who are visually impaired, unless the design also makes use of the other sensory channels. On a more mundane level, designs should allow for

users who are under pressure, feeling ill or distracted by other concerns: they should not push users to their perceptual or cognitive limits.

We will consider the issues of universal accessibility in more detail in Chapter 10.

1.7 PSYCHOLOGY AND THE DESIGN OF INTERACTIVE SYSTEMS

So far we have looked briefly at the way in which humans receive, process and store information, solve problems and acquire skill. But how can we apply what we have learned to designing interactive systems? Sometimes, straightforward conclusions can be drawn. For example, we can deduce that recognition is easier than recall and allow users to select commands from a set (such as a menu) rather than input them directly. However, in the majority of cases, application is not so obvious or simple. In fact, it may be dangerous, leading us to make generalizations which are not valid. In order to apply a psychological principle or result properly in design, we need to understand its context, both in terms of where it fits in the wider field of psychology and in terms of the details of the actual experiments, the measures used and the subjects involved, for example. This may appear daunting, particularly to the novice designer who wants to acknowledge the relevance of cognitive psychology but does not have the background to derive appropriate conclusions. Fortunately, principles and results from research in psychology have been distilled into guidelines for design, models to support design and techniques for evaluating design. Parts 2 and 3 of this book include discussion of a range of guidelines, models and techniques, based on cognitive psychology, which can be used to support the design process.

1.7.1 Guidelines

Throughout this chapter we have discussed the strengths and weaknesses of human cognitive and perceptual processes but, for the most part, we have avoided attempting to apply these directly to design. This is because such an attempt could only be partial and simplistic, and may give the impression that this is all psychology has to offer.

However, general design principles and guidelines can be and have been derived from the theories we have discussed. Some of these are relatively straightforward: for instance, recall is assisted by the provision of retrieval cues so interfaces should incorporate recognizable cues wherever possible. Others are more complex and context dependent. In Chapter 7 we discuss principles and guidelines further, many of which are derived from psychological theory. The interested reader is also referred to Gardiner and Christie [140] which illustrates how guidelines can be derived from psychological theory.

1.7.2 Models to support design

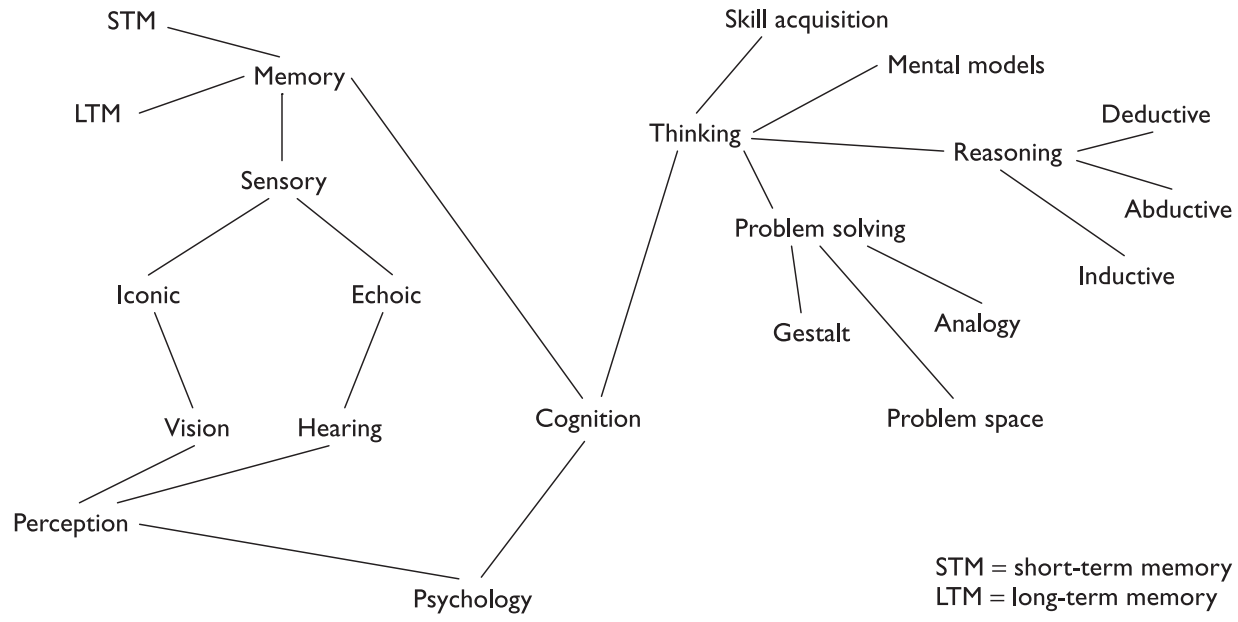
As well as guidelines and principles, psychological theory has led to the development of analytic and predictive models of user behavior. Some of these include a specific model of human problem solving, others of physical activity, and others attempt a more comprehensive view of cognition. Some predict how a typical computer user would behave in a given situation, others analyze why particular user behavior occurred. All are based on cognitive theory. We discuss these models in detail in Chapter 12.

1.7.3 Techniques for evaluation

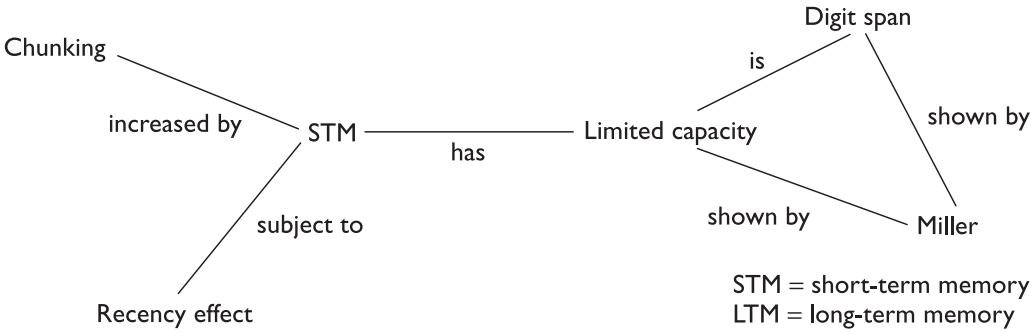
In addition to providing us with a wealth of theoretical understanding of the human user, psychology also provides a range of empirical techniques which we can employ to evaluate our designs and our systems. In order to use these effectively we need to understand the scope and benefits of each method. Chapter 9 provides an overview of these techniques and an indication of the circumstances under which each should be used.

Worked exercise *Produce a semantic network of the main information in this chapter.*

Answer This network is potentially huge so it is probably unnecessary to devise the whole thing! Be selective. One helpful way to tackle the exercise is to approach it in both a top-down and a bottom-up manner. Top-down will give you a general overview of topics and how they relate; bottom-up can fill in the details of a particular field. These can then be



Top-down view



Bottom-up view

'glued' together to build up the whole picture. You may be able to tackle this problem in a group, each taking one part of it. We will not provide the full network here but will give examples of the level of detail anticipated for the overview and the detailed versions. In the overview we have not included labels on the arcs for clarity.

1.8 SUMMARY

In this chapter we have considered the human as an information processor, receiving inputs from the world, storing, manipulating and using information, and reacting to the information received. Information is received through the senses, particularly, in the case of computer use, through sight, hearing and touch. It is stored in memory, either temporarily in sensory or working memory, or permanently in long-term memory. It can then be used in reasoning and problem solving. Recurrent familiar situations allow people to acquire skills in a particular domain, as their information structures become better defined. However, this can also lead to error, if the context changes.

Human perception and cognition are complex and sophisticated but they are not without their limitations. We have considered some of these limitations in this chapter. An understanding of the capabilities and limitations of the human as information processor can help us to design interactive systems which support the former and compensate for the latter. The principles, guidelines and models which can be derived from cognitive psychology and the techniques which it provides are invaluable tools for the designer of interactive systems.

EXERCISES

1.1 Devise experiments to test the properties of (i) short-term memory, (ii) long-term memory, using the experiments described in this chapter to help you. Try out your experiments on your friends. Are your results consistent with the properties described in this chapter?

1.2 Observe skilled and novice operators in a familiar domain, for example touch and 'hunt-and-peck' typists, expert and novice game players, or expert and novice users of a computer application. What differences can you discern between their behaviors?

1.3 From what you have learned about cognitive psychology devise appropriate guidelines for use by interface designers. You may find it helpful to group these under key headings, for example visual perception, memory, problem solving, etc., although some may overlap such groupings.

1.4 What are *mental models*, and why are they important in interface design?

1.5 What can a system designer do to minimize the memory load of the user?

1.6 Human short-term memory has a limited span. This is a series of experiments to determine what that span is. (You will need some other people to take part in these experiments with you – they do not need to be studying the course – try it with a group of friends.)

(a) *Kim's game*
Divide into groups. Each group gathers together an assortment of objects – pens, pencils, paperclips, books, sticky notes, etc. The stranger the object, the better! You need a large number of them – at least 12 to 15. Place them in some compact arrangement on a table, so that all items are visible. Then, swap with another group for 30 seconds only and look at their pile. Return to your table, and on your own try to write down all the items in the other group's pile.

Compare your list with what they actually have in their pile. Compare the number of things you remembered with how the rest of your group did. Now think introspectively: what helped you remember certain things? Did you recognize things in their pile that you had in yours? Did that help? Do not pack the things away just yet.

Calculate the average score for your group. Compare that with the averages from the other group(s).

Questions: What conclusions can you draw from this experiment? What does this indicate about the capacity of short-term memory? What does it indicate that helps improve the capacity of short-term memory?

(b) *'I went to market…'*
In your group, one person starts off with 'I went to market and I bought a fish' (or some other produce, or whatever!). The next person continues 'I went to market and I bought a fish and I bought a bread roll as well'. The process continues, with each person adding some item to the list each time. Keep going around the group until you cannot remember the list accurately. Make a note of the first time someone gets it wrong, and then record the number of items that you can successfully remember. Some of you will find it hard to remember more than a few, others will fare much better. Do this a few more times with different lists, and then calculate your average score, and your group's average score.

Questions: What does this tell you about short-term memory? What do you do that helps you remember? What do you estimate is the typical capacity of human short-term memory? Is this a good test for short-term memory?

(c) *Improving your memory*
Try experiment 1.6(a) again, using the techniques on page 39.

Has your recall ability improved? Has your group's average improved? What does this show you about memory?

1.7 Locate one source (through the library or the web) that reports on empirical evidence on human limitations. Provide a full reference to the source. In one paragraph, summarize what the result of the research states in terms of a physical human limitation.

In a separate paragraph, write your thoughts on how you think this evidence on human capabilities impacts interactive system design.

RECOMMENDED READING

E. B. Goldstein, *Sensation and Perception*, 6th edition, Wadsworth, 2001.
A textbook covering human senses and perception in detail. Easy to read with many home experiments to illustrate the points made.

A. Baddeley, *Human Memory: Theory and Practice*, revised edition, Allyn & Bacon, 1997.
The latest and most complete of Baddeley's texts on memory. Provides up-to-date discussion on the different views of memory structure as well as a detailed survey of experimental work on memory.

M. W. Eysenck and M. T. Keane, *Cognitive Psychology: A Student's Handbook*, 4th edition, Psychology Press, 2000.
A comprehensive and readable textbook giving more detail on cognitive psychology, including memory, problem solving and skill acquisition.

S. K. Card, T. P. Moran and A. Newell, *The Psychology of Human–Computer Interaction*, Lawrence Erlbaum Associates, 1983.
A classic text looking at the human as an information processor in interaction with the computer. Develops and describes the Model Human Processor in detail.

A. Newell and H. Simon, *Human Problem Solving*, Prentice Hall, 1972.
Describes the problem space view of problem solving in more detail.

M. M. Gardiner and B. Christie, editors, *Applying Cognitive Psychology to User-Interface Design*, John Wiley, 1987.
A collection of essays on the implications of different aspects of cognitive psychology to interface design. Includes memory, thinking, language and skill acquisition. Provides detailed guidelines for applying psychological principles in design practice.

A. Monk, editor, *Fundamentals of Human Computer Interaction*, Academic Press, 1985.
A good collection of articles giving brief coverage of aspects of human psychology including perception, memory, thinking and reading. Also contains articles on experimental design which provide useful introductions.

ACT-R site. Website of resources and examples of the use of the cognitive architecture ACT-R, which is the latest development of Anderson's ACT model, http://act-r.psy.cmu.edu/

THE COMPUTER

2

OVERVIEW

A computer system comprises various elements, each of which affects the user of the system.

- Input devices for interactive use, allowing text entry, drawing and selection from the screen:
  - text entry: traditional keyboard, phone text entry, speech and handwriting
  - pointing: principally the mouse, but also touchpad, stylus and others
  - 3D interaction devices.
- Output display devices for interactive use:
  - different types of screen mostly using some form of bitmap display
  - large displays and situated displays for shared and public use
  - digital paper may be usable in the near future.
- Virtual reality systems and 3D visualization which have special interaction and display devices.
- Various devices in the physical world:
  - physical controls and dedicated displays
  - sound, smell and haptic feedback
  - sensors for nearly everything including movement, temperature, bio-signs.
- Paper output and input: the paperless office and the less-paper office:
  - different types of printers and their characteristics, character styles and fonts
  - scanners and optical character recognition.
- Memory:
  - short-term memory: RAM
  - long-term memory: magnetic and optical disks
  - capacity limitations related to document and video storage
  - access methods as they limit or help the user.
- Processing:
  - the effects when systems run too slow or too fast, the myth of the infinitely fast machine
  - limitations on processing speed
  - networks and their impact on system performance.

2.1 INTRODUCTION

In order to understand how humans interact with computers, we need to have an understanding of both parties in the interaction. The previous chapter explored aspects of human capabilities and behavior of which we need to be aware in the context of human–computer interaction; this chapter considers the computer and associated input–output devices and investigates how the technology influences the nature of the interaction and style of the interface.

We will concentrate principally on the traditional computer but we will also look at devices that take us beyond the closed world of keyboard, mouse and screen. As well as giving us lessons about more traditional systems, these are increasingly becoming important application areas in HCI.

Exercise: how many computers?

In a group or class do a quick survey:

- How many computers do you have in your home?
- How many computers do you normally carry with you in your pockets or bags?

Collate the answers and see who the techno-freaks are!

Discuss your answers.

After doing this look at /e3/online/how-many-computers/

When we interact with computers, what are we trying to achieve? Consider what happens when we interact with each other – we are either passing information to other people, or receiving information from them. Often, the information we receive is in response to the information that we have recently imparted to them, and we may then respond to that. Interaction is therefore a process of information transfer. Relating this to the electronic computer, the same principles hold: interaction is a process of information transfer, from the user to the computer and from the computer to the user.

The first part of this chapter concentrates on the transference of information from the user to the computer and back. We begin by considering a current typical computer interface and the devices it employs, largely variants of keyboard for text entry (Section 2.2), mouse for positioning (Section 2.3) and screen for displaying output (Section 2.4).

Then we move on to consider devices that go beyond the keyboard, mouse and screen: entering deeper into the electronic world with virtual reality and 3D interaction

(Section 2.5) and outside the electronic world looking at more physical interactions (Section 2.6).

In addition to direct input and output, information is passed to and fro via paper documents. This is dealt with in Section 2.7, which describes printers and scanners. Although not requiring the same degree of user interaction as a mouse or keyboard, these are an important means of input and output for many current applications.

We then consider the computer itself, its processor and memory devices and the networks that link them together. We note how the technology drives and empowers the interface. The details of computer processing should largely be irrelevant to the end-user, but the interface designer needs to be aware of the limitations of storage capacity and computational power; it is no good designing on paper a marvellous new interface, only to find it needs a Cray to run. Software designers often have high-end machines on which to develop applications, and it is easy to forget what a more typical configuration feels like.

Before looking at these devices and technology in detail we'll take a quick bird's-eye view of the way computer systems are changing.

2.1.1 A typical computer system

Consider a typical computer setup as shown in Figure 2.1. There is the computer 'box' itself, a keyboard, a mouse and a color screen. The screen layout is shown alongside it. If we examine the interface, we can see how its various characteristics are related to the devices used. The details of the interface itself, its underlying principles and design, are discussed in more depth in Chapter 3. As we shall see there are variants on these basic devices. Some of this variation is driven by different hardware configurations: desktop use, laptop computers, PDAs (personal digital assistants). Partly the diversity of devices reflects the fact that there are many different types of

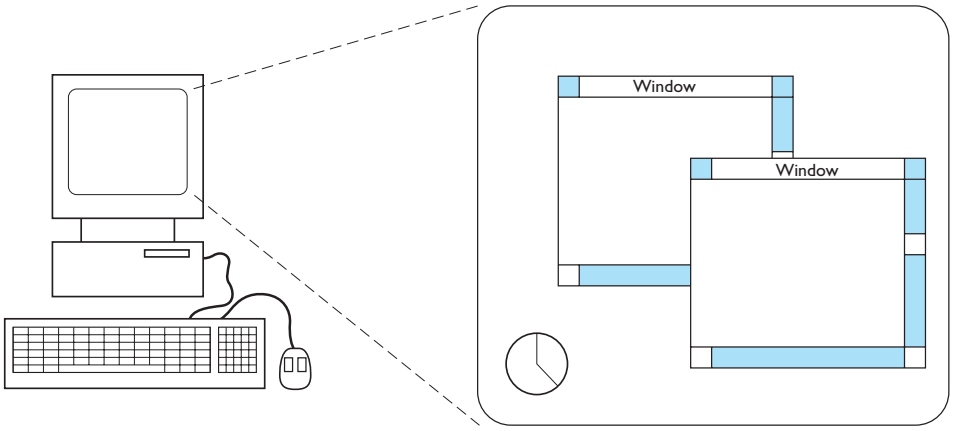


Figure 2.1 A typical computer system

data that may have to be entered into and obtained from a system, and there are also many different types of user, each with their own unique requirements.

2.1.2 Levels of interaction – batch processing

In the early days of computing, information was entered into the computer in a large mass – batch data entry. There was minimal interaction with the machine: the user would simply dump a pile of punched cards onto a reader, press the start button, and then return a few hours later. This still continues today although now with pre-prepared electronic files or possibly machine-read forms. It is clearly the most appropriate mode for certain kinds of application, for example printing pay checks or entering the results from a questionnaire.

With batch processing the interactions take place over hours or days. In contrast the typical desktop computer system has interactions taking seconds or fractions of a second (or with slow web pages sometimes minutes!). The field of Human–Computer Interaction largely grew due to this change in interactive pace. It is easy to assume that faster means better, but some of the paper-based technology discussed in Section 2.7 suggests that sometimes slower paced interaction may be better.

2.1.3 Richer interaction – everywhere, everywhen

Computers are coming out of the box! Information appliances are putting internet access or dedicated systems onto the fridge, microwave and washing machine: to automate shopping, give you email in your kitchen or simply call for maintenance when needed. We carry with us WAP phones and smartcards, have security systems that monitor us and web cams that show our homes to the world. Is Figure 2.1 really the typical computer system or is it really more like Figure 2.2?

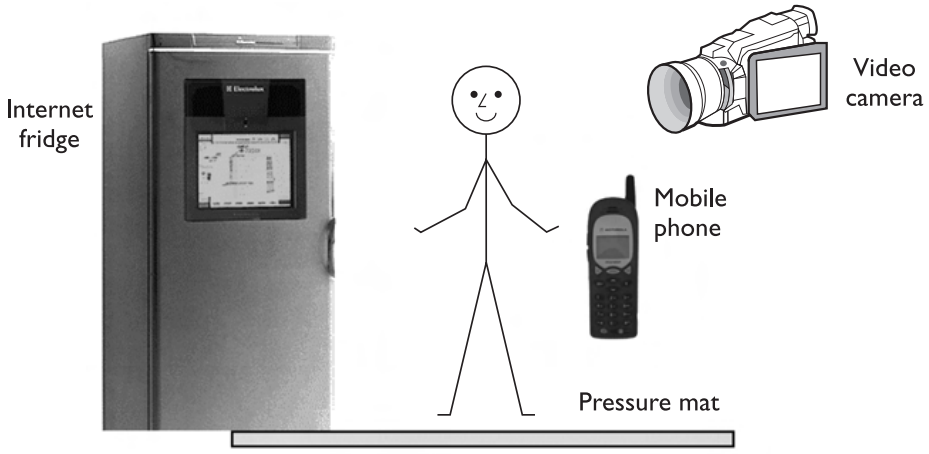


Figure 2.2 A typical computer system? Photo courtesy Electrolux

2.2 TEXT ENTRY DEVICES

Whether writing a book like this, producing an office memo, sending a thank you letter after your birthday, or simply sending an email to a friend, entering text is one of our main activities when using the computer. The most obvious means of text entry is the plain keyboard, but there are several variations on this: different keyboard layouts, 'chord' keyboards that use combinations of fingers to enter letters, and phone key pads. Handwriting and speech recognition offer more radical alternatives.

2.2.1 The alphanumeric keyboard

The keyboard is still one of the most common input devices in use today. It is used for entering textual data and commands. The vast majority of keyboards have a standardized layout, and are known by the first six letters of the top row of alphabetical keys, QWERTY. There are alternative designs which have some advantages over the QWERTY layout, but these have not been able to overcome the vast technological inertia of the QWERTY keyboard. These alternatives are of two forms: 26 key layouts and chord keyboards. A 26 key layout rearranges the order of the alphabetic keys, putting the most commonly used letters under the strongest fingers, or adopting simpler practices. In addition to QWERTY, we will discuss two 26 key layouts, alphabetic and DVORAK, and chord keyboards.

The QWERTY keyboard

The layout of the digits and letters on a QWERTY keyboard is fixed (see Figure 2.3), but non-alphanumeric keys vary between keyboards. For example, there is a difference between key assignments on British and American keyboards (in particular, above the 3 on the UK keyboard is the pound sign £, whilst on the US keyboard there is a dollar sign $). The standard layout is also subject to variation in the placement of brackets, backslashes and suchlike. In addition different national keyboards include accented letters and the traditional French layout places the main letters in different locations – the top line starts AZERTY.

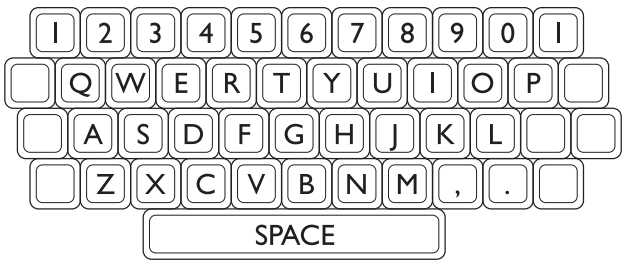


Figure 2.3 The standard QWERTY keyboard

The QWERTY arrangement of keys is not optimal for typing, however. The reason for the layout of the keyboard in this fashion can be traced back to the days of mechanical typewriters. Hitting a key caused an arm to shoot towards the carriage, imprinting the letter on the head on the ribbon and hence onto the paper. If two arms flew towards the paper in quick succession from nearly the same angle, they would often jam – the solution to this was to set out the keys so that common combinations of consecutive letters were placed at different ends of the keyboard, which meant that the arms would usually move from alternate sides. One appealing story relating to the key layout is that it was also important for a salesman to be able to type the word 'typewriter' quickly in order to impress potential customers: the letters are all on the top row!

The electric typewriter and now the computer keyboard are not subject to the original mechanical constraints, but the QWERTY keyboard remains the dominant layout. The reason for this is social – the vast base of trained typists would be reluctant to relearn their craft, whilst the management is not prepared to accept an initial lowering of performance whilst the new skills are gained. There is also a large investment in current keyboards, which would all have to be either replaced at great cost, or phased out, with the subsequent requirement for people to be proficient on both keyboards. As whole populations have become keyboard users this technological inertia has probably become impossible to change.

How keyboards work

Current keyboards work by a keypress closing a connection, causing a character code to be sent to the computer. The connection is usually via a lead, but wireless systems also exist. One aspect of keyboards that is important to users is the 'feel' of the keys. Some keyboards require a very hard press to operate the key, much like a manual typewriter, whilst others are featherlight. The distance that the keys travel also affects the tactile nature of the keyboard. The keyboards that are currently used on most notebook computers are 'half-travel' keyboards, where the keys travel only a small distance before activating their connection; such a keyboard can feel dead to begin with, but such qualitative judgments often change as people become more used to using it. By making the actual keys thinner, and allowing them a much reduced travel, a lot of vertical space can be saved on the keyboard, thereby making the machine slimmer than would otherwise be possible.

Some keyboards are even made of touch-sensitive buttons, which require a light touch and practically no travel; they often appear as a sheet of plastic with the buttons printed on them. Such keyboards are often found on shop tills, though the keys are not QWERTY, but specific to the task. Being fully sealed, they have the advantage of being easily cleaned and resistant to dirty environments, but have little feel, and are not popular with trained touch-typists. Feedback is important even at this level of human–computer interaction! With the recent increase of repetitive strain injury (RSI) to users' fingers, and the increased responsibilities of employers in these circumstances, it may be that such designs will enjoy a resurgence in the near future. RSI in fingers is caused by the tendons that control the movement of the fingers becoming inflamed owing to overuse and making repeated unnatural movements.

There are a variety of specially shaped keyboards to relieve the strain of typing or to allow people to type with some injury (e.g. RSI) or disability. These may slope the keys towards the hands to improve the ergonomic position, be designed for single-handed use, or for no hands at all. Some use bespoke key layouts to reduce strain of finger movements. The keyboard illustrated is produced by PCD Maltron Ltd. for left-handed use. See www.maltron.com/

Source: www.maltron.com, reproduced courtesy of PCD Maltron Ltd.

Ease of learning – alphabetic keyboard

One of the most obvious layouts to be produced is the alphabetic keyboard, in which the letters are arranged alphabetically across the keyboard. It might be expected that such a layout would make it quicker for untrained typists to use, but this is not the case. Studies have shown that this keyboard is not faster for properly trained typists, as we may expect, since there is no inherent advantage to this layout. And even for novice or occasional users, the alphabetic layout appears to make very little difference to the speed of typing. These keyboards are used in some pocket electronic personal organizers, perhaps because the layout looks simpler to use than the QWERTY one. Also, it dissuades people from attempting to use their touch-typing skills on a very small keyboard and hence avoids criticisms of difficulty of use!

Ergonomics of use – DVORAK keyboard and split designs

The DVORAK keyboard uses a similar layout of keys to the QWERTY system, but assigns the letters to different keys. Based upon an analysis of typing, the keyboard is designed to help people reach faster typing speeds. It is biased towards right-handed people, in that 56% of keystrokes are made with the right hand. The layout of the keys also attempts to ensure that the majority of keystrokes alternate between hands, thereby increasing the potential speed. By keeping the most commonly used keys on the home, or middle, row, 70% of keystrokes are made without the typist having to stretch far, thereby reducing fatigue and increasing keying speed. The layout also

aims to minimize the number of keystrokes made with the weak fingers. Many of these requirements are in conflict, and the DVORAK keyboard represents one possible solution. Experiments have shown that there is a speed improvement of between 10 and 15%, coupled with a reduction in user fatigue due to the increased ergonomic layout of the keyboard [230].

Other aspects of keyboard design have been altered apart from the layout of the keys. A number of more ergonomic designs have appeared, in which the basic tilted planar base of the keyboard is altered. Moderate designs curve the plane of the keyboard, making it concave, whilst more extreme ones split the keys into those for the left and right hand and curve both halves separately. Often in these the keys are also moved to bring them all within easy reach, to minimize movement between keys. Such designs are supposed to aid comfort and reduce RSI by minimizing effort, but have had practically no impact on the majority of systems sold.

2.2.2 Chord keyboards

Chord keyboards are significantly different from normal alphanumeric keyboards. Only a few keys, four or five, are used (see Figure 2.4) and letters are produced by pressing one or more of the keys at once. For example, in the *Microwriter*, the pattern of multiple keypresses is chosen to reflect the actual letter shape.

Such keyboards have a number of advantages. They are extremely compact: simply reducing the size of a conventional keyboard makes the keys too small and close together, with a correspondingly large increase in the difficulty of using it. The

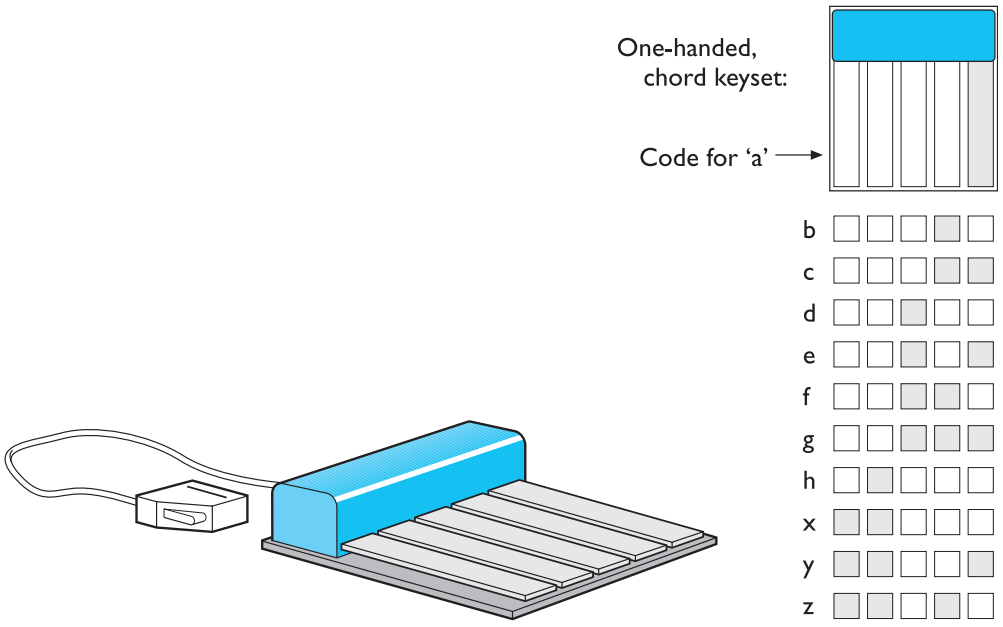


Figure 2.4 A very early chord keyboard (left) and its lettercodes (right)

learning time for the keyboard is supposed to be fairly short – of the order of a few hours – but social resistance is still high. Moreover, they are capable of fast typing speeds in the hands (or rather hand!) of a competent user. Chord keyboards can also be used where only one-handed operation is possible, in cramped and confined conditions.

Lack of familiarity means that these are unlikely ever to be a mainstream form of text entry, but they do have applications in niche areas. In particular, courtroom stenographers use a special form of two-handed chord keyboard and associated shorthand to enter text at full spoken speed. Also it may be that the compact size and one-handed operation will find a place in the growing wearables market.

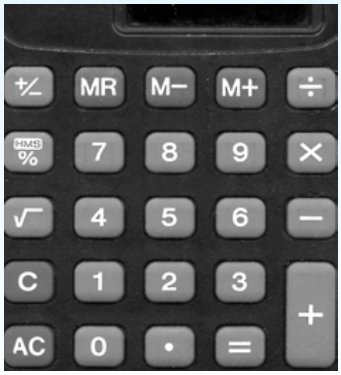
DESIGN FOCUS

Numeric keypads

Alphanumeric keyboards (as the name suggests) include numbers as well as letters. In the QWERTY layout these are in a line across the top of the keyboard, but in most larger keyboards there is also a separate number pad to allow faster entry of digits. Number keypads occur in other contexts too, including calculators, telephones and ATM cash dispensers. Many people are unaware that there are two different layouts for numeric keypads: the calculator style that has '123' on the bottom and the telephone style that has '123' at the top.

It is a demonstration of the amazing adaptability of humans that we move between these two styles with such ease. However, if you need to include a numeric keypad in a device you must consider which is most appropriate for your potential users. For example, computer keyboards use calculator-style layout, as they are primarily used for entering numbers for calculations.

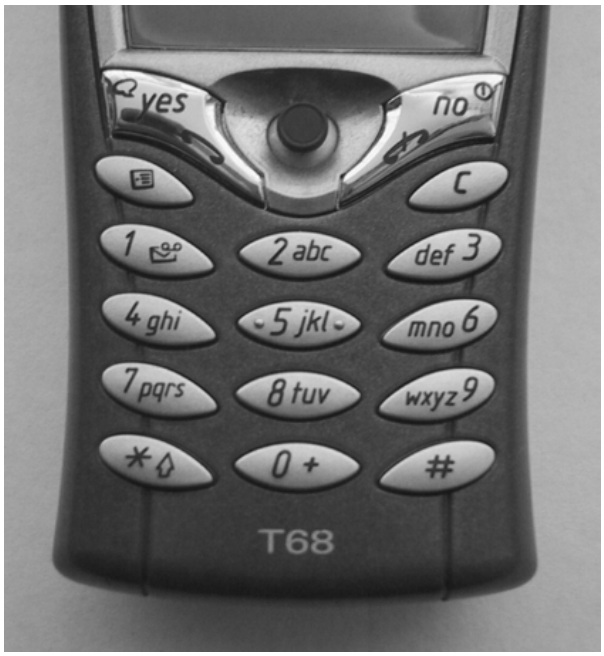
One of the authors was caught out by this once when he forgot the PIN number of his cash card. He half remembered the digits, but also his fingers knew where to type, so he 'practiced' on his calculator. Unfortunately ATMs use telephone-style layout!



calculator

ATM

phone



Typical key mapping:
1 – space, comma, etc. (varies)
2 – a b c
3 – d e f
4 – g h i
5 – j k l
6 – m n o
7 – p q r s
8 – t u v
9 – w x y z
0 – +, &, etc.

Figure 2.5 Mobile phone keypad. Source: Photograph by Alan Dix (Ericsson phone)

2.2.3 Phone pad and T9 entry

With mobile phones being used for SMS text messaging (see Chapter 19) and WAP (see Chapter 21), the phone keypad has become an important form of text input. Unfortunately a phone only has digits 0–9, not a full alphanumeric keyboard.

To overcome this for text input the numeric keys are usually pressed several times – Figure 2.5 shows a typical mapping of digits to letters. For example, the 3 key has 'def' on it. If you press the key once you get a 'd', if you press 3 twice you get an 'e', if you press it three times you get an 'f'. The main number-to-letter mapping is standard, but punctuation and accented letters differ between phones. Also there needs to be a way for the phone to distinguish, say, the 'dd' from 'e'. On some phones you need to pause for a short period between successive letters using the same key, for others you press an additional key (e.g. '#').

Most phones have at least two *modes* for the numeric buttons: one where the keys mean the digits (for example when entering a phone number) and one where they mean letters (for example when typing an SMS message). Some have additional modes to make entering accented characters easier. Also a special mode or setting is needed for capital letters although many phones use rules to reduce this, for example automatically capitalizing the initial letter in a message and letters following full stops, question marks and exclamation marks.

This is all very laborious and, as we will see in Chapter 19, experienced mobile phone users make use of a highly developed shorthand to reduce the number of keystrokes. If you watch a teenager or other experienced txt-er, you will see they

often develop great typing speed holding the phone in one hand and using only their thumb. As these skills spread through society it may be that future devices use this as a means of small format text input. For those who never develop this physical dexterity some phones have tiny plug-in keyboards, or come with fold-out keyboards.

Another technical solution to the problem is the T9 algorithm. This uses a large dictionary to disambiguate words by simply typing the relevant letters once. For example, '3926753' becomes 'example' as there is only one word with letters that match (alternatives like 'ewbosld' that also match are not real words). Where there are ambiguities such as '26', which could be an 'am' or an 'an', the phone gives a series of options to choose from.

2.2.4 Handwriting recognition

Handwriting is a common and familiar activity, and is therefore attractive as a method of text entry. If we were able to write as we would when we use paper, but with the computer taking this form of input and converting it to text, we can see that it is an intuitive and simple way of interacting with the computer. However, there are a number of disadvantages with handwriting recognition. Current technology is still fairly inaccurate and so makes a significant number of mistakes in recognizing letters, though it has improved rapidly. Moreover, individual differences in handwriting are enormous, and make the recognition process even more difficult. The most significant information in handwriting is not in the letter shape itself but in the stroke information – the way in which the letter is drawn. This means that devices which support handwriting recognition must capture the stroke information, not just the final character shape. Because of this, online recognition is far easier than reading handwritten text on paper. Further complications arise because letters within words are shaped and often drawn very differently depending on the actual word; the context can help determine the letter's identity, but is often unable to provide enough information. Handwriting recognition is covered in more detail later in the book, in Chapter 10. More serious in many ways is the limitation on speed; it is difficult to write at more than 25 words a minute, which is no more than half the speed of a decent typist.

The different nature of handwriting means that we may find it more useful in situations where a keyboard-based approach would have its own problems. Such situations will invariably result in completely new systems being designed around the handwriting recognizer as the predominant mode of textual input, and these may bear very little resemblance to the typical system. Pen-based systems that use handwriting recognition are actively marketed in the mobile computing market, especially for smaller pocket organizers. Such machines are typically used for taking notes and jotting down and sketching ideas, as well as acting as a diary, address book and organizer. Using handwriting recognition has many advantages over using a keyboard. A pen-based system can be small and yet still accurate and easy to use, whereas small keys become very tiring, or even impossible, to use accurately. Also the

pen-based approach does not have to be altered when we move from jotting down text to sketching diagrams; pen-based input is highly appropriate for this also.

Some organizer designs have dispensed with a keyboard completely. With such systems one must consider all sorts of other ways to interact with the system that are not character based. For example, we may decide to use *gesture recognition*, rather than commands, to tell the system what to do, for example drawing a line through a word in order to delete it. The important point is that a different input device that was initially considered simply as an alternative to the keyboard opens up a whole host of alternative interface designs and different possibilities for interaction.

Signature authentication

Handwriting recognition is difficult principally because of the great differences between different people's handwriting. These differences can be used to advantage in *signature authentication* where the purpose is to identify the user rather than read the signature. Again this is far easier when we have stroke information as people tend to produce signatures which *look* slightly different from one another in detail, but are formed in a similar fashion. Furthermore, a forger who has a copy of a person's signature may be able to copy the appearance of the signature, but will not be able to reproduce the pattern of strokes.

2.2.5 Speech recognition

Speech recognition is a promising area of text entry, but it has been promising for a number of years and is still only used in very limited situations. There is a natural enthusiasm for being able to talk to the machine and have it respond to commands, since this form of interaction is one with which we are very familiar. Successful recognition rates of over 97% have been reported, but since this represents one letter in error in approximately every 30, or one spelling mistake every six or so words, this is stoll unacceptible (*sic*)! Note also that this performance is usually quoted only for a restricted vocabulary of command words. Trying to extend such systems to the level of understanding natural language, with its inherent vagueness, imprecision and pauses, opens up many more problems that have not been satisfactorily solved even for keyboard-entered natural language. Moreover, since every person speaks differently, the system has to be trained and tuned to each new speaker, or its performance decreases. Strong accents, a cold or emotion can also cause recognition problems, as can background noise. This leads us on to the question of practicality within an office environment: not only may the background level of noise cause errors, but if everyone in an open-plan office were to talk to their machine, the level of noise would dramatically increase, with associated difficulties. Confidentiality would also be harder to maintain.

Despite its problems, speech technology has found niche markets: telephone information systems, access for the disabled, in hands-occupied situations (especially

military) and for those suffering RSI. This is discussed in greater detail in Chapter 10, but we can see that it offers three possibilities. The first is as an alternative text entry device to replace the keyboard within an environment and using software originally designed for keyboard use. The second is to redesign a system, taking full advantage of the benefits of the technique whilst minimizing the potential problems. Finally, it can be used in areas where keyboard-based input is impractical or impossible. It is in the latter, more radical areas that speech technology is currently achieving success.

2.3 POSITIONING, POINTING AND DRAWING

Central to most modern computing systems is the ability to point at something on the screen and thereby manipulate it, or perform some function. There has been a long history of such devices, in particular in *computer-aided design* (CAD), where positioning and drawing are the major activities. Pointing devices allow the user to point, position and select items, either directly or by manipulating a pointer on the screen. Many pointing devices can also be used for free-hand drawing although the skill of drawing with a mouse is very different from using a pencil. The mouse is still most common for desktop computers, but is facing challenges as laptop and hand-held computing increase their market share. Indeed, these words are being typed on a laptop with a touchpad and no mouse.

2.3.1 The mouse

The mouse has become a major component of the majority of desktop computer systems sold today, and is the little box with the tail connecting it to the machine in our basic computer system picture (Figure 2.1). It is a small, palm-sized box housing a weighted ball – as the box is moved over the tabletop, the ball is rolled by the table and so rotates inside the housing. This rotation is detected by small rollers that are in contact with the ball, and these adjust the values of potentiometers. If you remove the ball occasionally to clear dust you may be able to see these rollers. The changing values of these potentiometers can be directly related to changes in position of the ball. The potentiometers are aligned in different directions so that they can detect both horizontal and vertical motion. The relative motion information is passed to the computer via a wire attached to the box, or in some cases using wireless or infrared, and moves a pointer on the screen, called the *cursor*. The whole arrangement tends to look rodent-like, with the box acting as the body and the wire as the tail; hence the term 'mouse'. In addition to detecting motion, the mouse has typically one, two or three buttons on top. These are used to indicate selection or to initiate action. Single-button mice tend to have similar functionality to multi-button mice, and achieve this by instituting different operations for a single and a double button click. A 'double-click' is when the button is pressed twice in rapid succession. Multi-button mice tend to allocate one operation to each particular button.

The mouse operates in a planar fashion, moving around the desktop, and is an indirect input device, since a transformation is required to map from the horizontal nature of the desktop to the vertical alignment of the screen. Left–right motion is directly mapped, whilst up–down on the screen is achieved by moving the mouse away–towards the user. The mouse only provides information on the relative movement of the ball within the housing: it can be physically lifted up from the desktop and replaced in a different position without moving the cursor. This offers the advantage that less physical space is required for the mouse, but suffers from being less intuitive for novice users. Since the mouse sits on the desk, moving it about is easy and users suffer little arm fatigue, although the indirect nature of the medium can lead to problems with hand–eye coordination. However, a major advantage of the mouse is that the cursor itself is small, and it can be easily manipulated without obscuring the display.

The mouse was developed around 1964 by Douglas C. Engelbart, and a photograph of the first prototype is shown in Figure 2.6. This used two wheels that slid across the desktop and transmitted *x*–*y* coordinates to the computer. The housing was carved in wood, and has been damaged, exposing one of the wheels. The original design actually offers a few advantages over today's more sleek versions: by tilting it so that only one wheel is in contact with the desk, pure vertical or horizontal motion can be obtained. Also, the problem of getting the cursor across the large screens that are often used today can be solved by flicking your wrist to get the horizontal wheel spinning. The mouse pointer then races across the screen with no further effort on your behalf, until you stop it at its destination by dropping the mouse down onto the desktop.

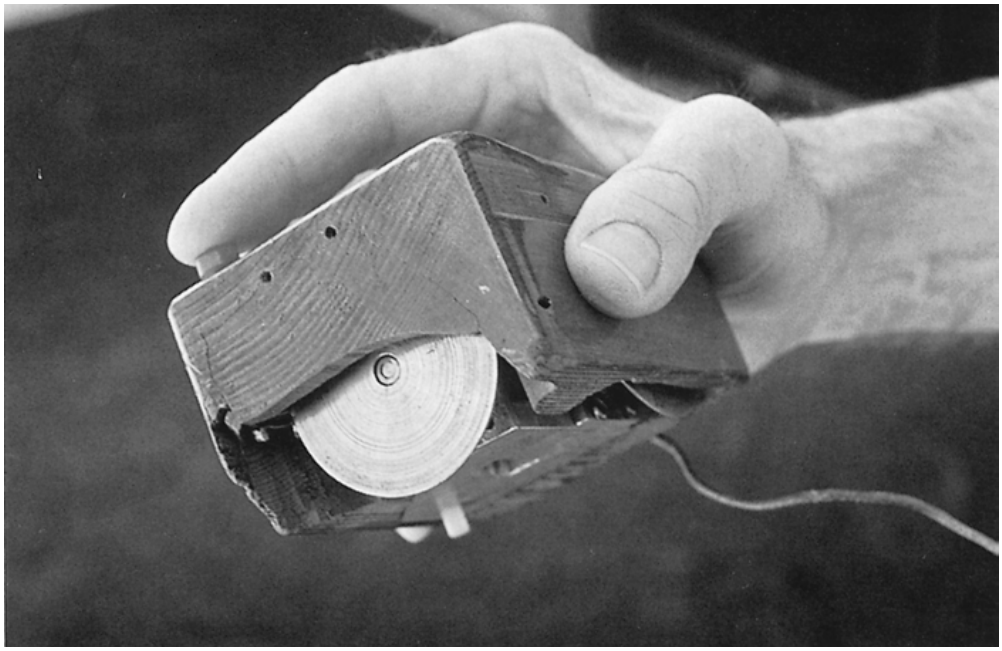


Figure 2.6 The first mouse. Photograph courtesy of Douglas Engelbart and Bootstrap Institute

Optical mice

Optical mice work differently from mechanical mice. A light-emitting diode emits a weak red light from the base of the mouse. This is reflected off a special pad with a metallic grid-like pattern upon which the mouse has to sit, and the fluctuations in reflected intensity as the mouse is moved over the gridlines are recorded by a sensor in the base of the mouse and translated into relative *x*, *y* motion. Some optical mice do not require special mats, just an appropriate surface, and use the natural texture of the surface to detect movement. The optical mouse is less susceptible to dust and dirt than the mechanical one in that its mechanism is less likely to become blocked up. However, for those that rely on a special mat, if the mat is not properly aligned, movement of the mouse may become erratic – especially difficult if you are working with someone and pass the mouse back and forth between you.

Although most mice are hand operated, not all are – there have been experiments with a device called the *footmouse*. As the name implies, it is a foot-operated device, although more akin to an isometric joystick than a mouse. The cursor is moved by foot pressure on one side or the other of a pad. This allows one to dedicate hands to the keyboard. A rare device, the footmouse has not found common acceptance!

Interestingly foot pedals are used heavily in musical instruments including pianos, electric guitars, organs and drums and also in mechanical equipment including cars, cranes, sewing machines and industrial controls. So it is clear that in principle this is a good idea. Two things seem to have limited their use in computer equipment (except simulators and games). One is the practicality of having foot controls in the work environment: pedals under a desk may be operated accidentally, laptops with foot pedals would be plain awkward. The second issue is the kind of control being exercised. Pedals in physical interfaces are used predominantly to control one or more single-dimensional analog controls. It may be that in more specialized interfaces appropriate foot-operated controls could be more commonly and effectively used.

2.3.2 Touchpad

Touchpads are touch-sensitive tablets usually around 2–3 inches (50–75 mm) square. They were first used extensively in Apple Powerbook portable computers but are now used in many other notebook computers and can be obtained separately to replace the mouse on the desktop. They are operated by stroking a finger over their surface, rather like using a simulated trackball. The feel is very different from other input devices, but as with all devices users quickly get used to the action and become proficient.

Because they are small it may require several strokes to move the cursor across the screen. This can be improved by using acceleration settings in the software linking the trackpad movement to the screen movement. Rather than having a fixed ratio of pad distance to screen distance, this varies with the speed of movement. If the finger

moves slowly over the pad then the pad movements map to small distances on the screen. If the finger is moving quickly the same distance on the touchpad moves the cursor a long distance. For example, on the trackpad being used when writing this section a very slow movement of the finger from one side of the trackpad to the other moves the cursor less than 10% of the width of the screen. However, if the finger is moved very rapidly from side to side, the cursor moves the whole width of the screen.

In fact, this form of acceleration setting is also used in other indirect positioning devices including the mouse. Fine settings of this sort of parameter makes a great difference to the 'feel' of the device.

2.3.3 Trackball and thumbwheel

The trackball is really just an upside-down mouse! A weighted ball faces upwards and is rotated inside a static housing, the motion being detected in the same way as for a mechanical mouse, and the relative motion of the ball moves the cursor. Because of this, the trackball requires no additional space in which to operate, and is therefore a very compact device. It is an indirect device, and requires separate buttons for selection. It is fairly accurate, but is hard to draw with, as long movements are difficult. Trackballs now appear in a wide variety of sizes, the most usual being about the same as a golf ball, with a number of larger and smaller devices available. The size and 'feel' of the trackball itself affords significant differences in the usability of the device: its weight, rolling resistance and texture all contribute to the overall effect.

Some of the smaller devices have been used in notebook and portable computers, but more commonly trackpads or nipples are used. They are often sold as alternatives to mice on desktop computers, especially for RSI sufferers. They are also heavily used in video games where their highly responsive behavior, including being able to spin the ball, is ideally suited to the demands of play.

Thumbwheels are different in that they have two orthogonal dials to control the cursor position. Such a device is very cheap, but slow, and it is difficult to manipulate the cursor in any way other than horizontally or vertically. This limitation can sometimes be a useful constraint in the right application. For instance, in CAD the designer is almost always concerned with exact verticals and horizontals, and a device that provides such constraints is very useful, which accounts for the appearance of thumbwheels in CAD systems. Another successful application for such a device has been in a drawing game such as Etch-a-Sketch in which straight lines can be created on a simple screen, since the predominance of straight lines in simple drawings means that the motion restrictions are an advantage rather than a handicap. However, if you were to try to write your signature using a thumbwheel, the limitations would be all too apparent. The appropriateness of the device depends on the task to be performed.

Although two-axis thumbwheels are not heavily used in mainstream applications, single thumbwheels are often included on a standard mouse in order to offer an alternative means to scroll documents. Normally scrolling requires you to grab the scroll bar with the mouse cursor and drag it down. For large documents it is hard to

be accurate and in addition the mouse dragging is done holding a finger down which adds to hand strain. In contrast the small scroll wheel allows comparatively intuitive and fast scrolling, simply rotating the wheel to move the page.

2.3.4 Joystick and keyboard nipple

The joystick is an indirect input device, taking up very little space. Consisting of a small palm-sized box with a stick or shaped grip sticking up from it, the joystick is a simple device with which movements of the stick cause a corresponding movement of the screen cursor. There are two types of joystick: the *absolute* and the *isometric*. In the absolute joystick, movement is the important characteristic, since the position of the joystick in the base corresponds to the position of the cursor on the screen. In the isometric joystick, the pressure on the stick corresponds to the velocity of the cursor, and when released, the stick returns to its usual upright centered position. This type of joystick is also called the velocity-controlled joystick, for obvious reasons. The buttons are usually placed on the top of the stick, or on the front like a trigger. Joysticks are inexpensive and fairly robust, and for this reason they are often found in computer games. Another reason for their dominance of the games market is their relative familiarity to users, and their likeness to aircraft joysticks: aircraft are a favorite basis for games, leading to familiarity with the joystick that can be used for more obscure entertainment ideas.

A smaller device but with the same basic characteristics is used on many laptop computers to control the cursor. Some older systems had a variant of this called the keymouse, which was a single key. More commonly a small rubber nipple projects in the center of the keyboard and acts as a tiny isometric joystick. It is usually difficult for novices to use, but this seems to be related to fine adjustment of the speed settings. Like the joystick the nipple controls the rate of movement across the screen and is thus less direct than a mouse or stylus.

2.3.5 Touch-sensitive screens (touchscreens)

Touchscreens are another method of allowing the user to point and select objects on the screen, but they are much more direct than the mouse, as they detect the presence of the user's finger, or a stylus, on the screen itself. They work in one of a number of different ways: by the finger (or stylus) interrupting a matrix of light beams, or by capacitance changes on a grid overlaying the screen, or by ultrasonic reflections. Because the user indicates exactly which item is required by pointing to it, no mapping is required and therefore this is a direct device.

The touchscreen is very fast, and requires no specialized pointing device. It is especially good for selecting items from menus displayed on the screen. Because the screen acts as an input device as well as an output device, there is no separate hardware to become damaged or destroyed by dirt; this makes touchscreens suitable for use in hostile environments. They are also relatively intuitive to use and have been used successfully as an interface to information systems for the general public.

They suffer from a number of disadvantages, however. Using the finger to point is not always suitable, as it can leave greasy marks on the screen, and, being a fairly blunt instrument, it is quite inaccurate. This means that the selection of small regions is very difficult, as is accurate drawing. Moreover, lifting the arm to point to a vertical screen is very tiring, and also means that the screen has to be within about a meter of the user to enable it to be reached, which can make it too close for comfort. Research has shown that the optimal angle for a touchscreen is about 15 degrees up from the horizontal.

2.3.6 Stylus and light pen

For more accurate positioning (and to avoid greasy screens), systems with touch-sensitive surfaces often emply a stylus. Instead of pointing at the screen directly a small pen-like plastic stick is used to point and draw on the screen. This is particularly popular in PDAs, but they are also being used in some laptop computers.

An older technology that is used in the same way is the light pen. The pen is connected to the screen by a cable and, in operation, is held to the screen and detects a burst of light from the screen phosphor during the display scan. The light pen can therefore address individual pixels and so is much more accurate than the touchscreen.

Both stylus and light pen can be used for fine selection and drawing, but both can be tiring to use on upright displays and are harder to take up and put down when used together with a keyboard. Interestingly some users of PDAs with fold-out keyboards learn to hold the stylus held outwards between their fingers so that they can type whilst holding it. As it is unattached the stylus can easily get lost, but a closed pen can be used in emergencies.

Stylus, light pen and touchscreen are all very direct in that the relationship between the device and the thing selected is immediate. In contrast, mouse, touchpad, joystick and trackball all have to map movements on the desk to cursor movement on the screen.

However, the direct devices suffer from the problem that, in use, the act of pointing actually obscures the display, making it harder to use, especially if complex detailed selections or movements are required in rapid succession. This means that screen designs have to take into account where the user's hand will be. For example, you may want to place menus at the bottom of the screen rather than the top. Also you may want to offer alternative layouts for right-handed and left-handed users.

2.3.7 Digitizing tablet

The digitizing tablet is a more specialized device typically used for freehand drawing, but may also be used as a mouse substitute. Some highly accurate tablets, usually using a puck (a mouse-like device), are used in special applications such as digitizing information for maps.

The tablet provides positional information by measuring the position of some device on a special pad, or *tablet*, and can work in a number of ways. The *resistive*

tablet detects point contact between two separated conducting sheets. It has advantages in that it can be operated without a specialized stylus – a pen or the user's finger is sufficient. The *magnetic tablet* detects current pulses in a magnetic field using a small loop coil housed in a special pen. There are also capacitative and electrostatic tablets that work in a similar way. The *sonic tablet* is similar to the above but requires no special surface. An ultrasonic pulse is emitted by a special pen which is detected by two or more microphones which then triangulate the pen position. This device can be adapted to provide 3D input, if required.

Digitizing tablets are capable of high resolution, and are available in a range of sizes. Sampling rates vary, affecting the resolution of cursor movement, which gets progressively finer as the sampling rate increases. The digitizing tablet can be used to detect relative motion *or* absolute motion, but is an indirect device since there is a mapping from the plane of operation of the tablet to the screen. It can also be used for text input; if supported by character recognition software, handwriting can be interpreted. Problems with digitizing tablets are that they require a large amount of desk space, and may be awkward to use if displaced to one side by the keyboard.

2.3.8 Eyegaze

Eyegaze systems allow you to control the computer by simply looking at it! Some systems require you to wear special glasses or a small head-mounted box, others are built into the screen or sit as a small box below the screen. A low-power laser is shone into the eye and is reflected off the retina. The reflection changes as the angle of the eye alters, and by tracking the reflected beam the eyegaze system can determine the direction in which the eye is looking. The system needs to be calibrated, typically by staring at a series of dots on the screen, but thereafter can be used to move the screen cursor or for other more specialized uses. Eyegaze is a very fast and accurate device, but the more accurate versions can be expensive. It is fine for selection but not for drawing since the eye does not move in smooth lines. Also in real applications it can be difficult to distinguish deliberately gazing at something and accidentally glancing at it.

Such systems have been used in military applications, notably for guiding air-to-air missiles to their targets, but are starting to find more peaceable uses, for disabled users and for workers in environments where it is impossible for them to use their hands. The rarity of the eyegaze is due partly to its novelty and partly to its expense, and it is usually found only in certain domain-specific applications. Within HCI it is particularly useful as part of evaluation as one is able to trace exactly where the user is looking [81]. As prices drop and the technology becomes less intrusive we may see more applications using eyegaze, especially in virtual reality and augmented reality areas (see Chapter 20).

2.3.9 Cursor keys and discrete positioning

All of the devices we have discussed are capable of giving near continuous 2D positioning, with varying degrees of accuracy. For many applications we are only

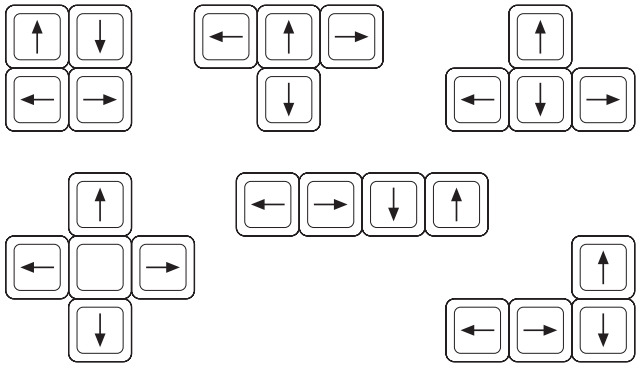


Figure 2.7 Various cursor key layouts

interested in positioning within a sequential list such as a menu or amongst 2D cells as in a spreadsheet. Even for moving within text discrete up/down left/right keys can sometimes be preferable to using a mouse.

Cursor keys are available on most keyboards. Four keys on the keyboard are used to control the cursor, one each for up, down, left and right. There is no standardized layout for the keys. Some layouts are shown in Figure 2.7, but the most common now is the inverted 'T'.

Cursor keys used to be more heavily used in character-based systems before windows and mice were the norm. However, when logging into remote machines such as web servers, the interface is often a virtual character-based terminal within a telnet window. In such applications it is common to find yourself in a 1970s world of text editors controlled sometimes using cursor keys and sometimes by more arcane combinations of control keys!

Small devices such as mobile phones, personal entertainment and television remote controls often require discrete control, either dedicated to a particular function such as volume, or for use as general menu selection. Figure 2.8 shows examples of these. The satellite TV remote control has dedicated '+/–' buttons for controlling volume and stepping between channels. It also has a central cursor pad that is used for on-screen menus. The mobile phone has a single central joystick-like device. This can be pushed left/right, up/down to navigate within the small 3×3 array of graphical icons as well as select from text menus.

2.4 DISPLAY DEVICES

The vast majority of interactive computer systems would be unthinkable without some sort of display screen, but many such systems do exist, though usually in specialized applications only. Thinking beyond the traditional, systems such as cars, hi-fis and security alarms all have different outputs from those expressible on a screen, but in the personal computer and workstation market, screens are pervasive.

Figure 2.8 Satellite TV remote control and mobile phone. Source: Photograph left by Alan Dix with permission from British Sky Broadcasting Limited, photograph right by Alan Dix (Ericsson phone)

In this section, we discuss the standard computer display in detail, looking at the properties of bitmap screens, at different screen technologies, at large and situated displays, and at a new technology, 'digital paper'.

2.4.1 Bitmap displays – resolution and color

Virtually all computer displays are based on some sort of bitmap. That is the display is made of vast numbers of colored dots or pixels in a rectangular grid. These pixels may be limited to black and white (for example, the small display on many TV remote controls), in grayscale, or full color.

The color or, for monochrome screens, the intensity at each pixel is held by the computer's video card. One bit per pixel can store on/off information, and hence only black and white (the term 'bitmap' dates from such displays). More bits per pixel give rise to more color or intensity possibilities. For example, 8 bits/pixel give rise to $2^8 = 256$ possible colors *at any one time*. The set of colors make up what is called the *colormap*, and the colormap can be altered at any time to produce a different set of colors. The system is therefore capable of actually displaying many more than the number of colors in the colormap, but not simultaneously. Most desktop computers now use 24 or 32 bits per pixel which allows virtually unlimited colors, but devices such as mobile phones and PDAs are often still monochrome or have limited color range.

As well as the number of colors that can be displayed at each pixel, the other measure that is important is the resolution of the screen. Actually the word 'resolution' is used in a confused (and confusing!) way for screens. There are two numbers to consider:

- the *total number* of pixels: in standard computer displays this is always in a 4:3 ratio, perhaps 1024 pixels across by 768 down, or 1600×1200; for PDAs this will be more in the order of a few hundred pixels in each direction.
- the *density* of pixels: this is measured in pixels per inch. Unlike printers (see Section 2.7 below) this density varies little between 72 and 96 pixels per inch.

To add to the confusion, a monitor, liquid crystal display (LCD) screen or other display device will quote its maximum resolution, but the computer may actually give it less than this. For example, the screen may be a 1200×900 resolution with 96 pixels per inch, but the computer only sends it 800×600. In the case of a cathode ray tube (CRT) this typically will mean that the image is stretched over the screen surface giving a lower density of 64 pixels per inch. An LCD screen cannot change its pixel size so it would keep 96 pixels per inch and simply not use all its screen space, adding a black border instead. Some LCD projectors will try to stretch or reduce what they are given, but this may mean that one pixel gets stretched to two, or two pixels get 'squashed' into one, giving rise to display 'artifacts' such as thin lines disappearing, or uniform lines becoming alternately thick or thin.

Although horizontal and vertical lines can be drawn perfectly on bitmap screens, and lines at 45 degrees reproduce reasonably well, lines at any other angle and curves have 'jaggies', rough edges caused by the attempt to approximate the line with pixels.

When using a single color jaggies are inevitable. Similar effects are seen in bitmap fonts. The problem of jaggies can be reduced by using high-resolution screens, or by a technique known as *anti-aliasing*. Anti-aliasing softens the edges of line segments, blurring the discontinuity and making the jaggies less obvious.

Look at the two images in Figure 2.9 with your eyes slightly screwed up. See how the second anti-aliased line looks better. Of course, screen resolution is much higher, but the same principle holds true. The reason this works is because our brains are constantly 'improving' what we see in the world: processing and manipulating the raw sensations of the rods and cones in our eyes and turning them into something meaningful. Often our vision is blurred because of poor light, things being out of focus, or defects in our vision. Our brain compensates and tidies up blurred images. By deliberately blurring the image, anti-aliasing triggers this processing in our brain and we appear to see a smooth line at an angle.

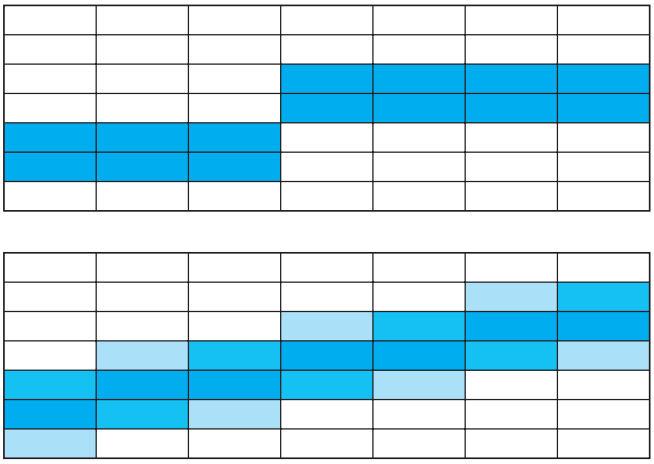


Figure 2.9 Magnified anti-aliased lines

2.4.2 Technologies

Cathode ray tube

The cathode ray tube is the television-like computer screen still most common as we write this, but rapidly being displaced by flat LCD screens. It works in a similar way to a standard television screen. A stream of electrons is emitted from an electron gun, which is then focussed and directed by magnetic fields. As the beam hits the phosphor-coated screen, the phosphor is excited by the electrons and glows (see Figure 2.10). The electron beam is scanned from left to right, and then flicked back to rescan the next line, from top to bottom. This is repeated, at about 30 Hz (that is, 30 times a second), per frame, although higher scan rates are sometimes used to reduce the flicker on the screen. Another way of reducing flicker is to use *interlacing*, in which the odd lines on the screen are all scanned first, followed by the even lines. Using a high-persistence phosphor, which glows for a longer time when excited, also reduces flicker, but causes image smearing especially if there is significant animation.

Black and white screens are able to display grayscale by varying the intensity of the electron beam; color is achieved using more complex means. Three electron guns are used, one each to hit red, green and blue phosphors. Combining these colors can

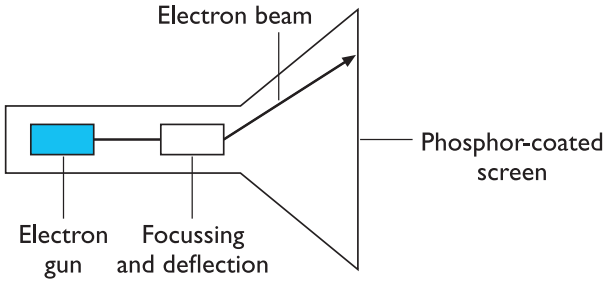


Figure 2.10 CRT screen

produce many others, including white, when they are all fully on. These three phosphor dots are focussed to make a single point using a *shadow mask*, which is imprecise and gives color screens a lower resolution than equivalent monochrome screens.

An alternative approach to producing color on the screen is to use *beam penetration*. A special phosphor glows a different color depending on the intensity of the beam hitting it.

The CRT is a cheap display device and has fast enough response times for rapid animation coupled with a high color capability. Note that animation does not necessarily mean little creatures and figures running about on the screen, but refers in a more general sense to the use of motion in displays: moving the cursor, opening windows, indicating processor-intensive calculations, or whatever. As screen resolution increases, however, the price rises. Because of the electron gun and focussing components behind the screen, CRTs are fairly bulky, though recent innovations have led to flatter displays in which the electron gun is not placed so that it fires directly at the screen, but fires parallel to the screen plane with the resulting beam bent through 90 degrees to hit the screen.

Health hazards of CRT displays

Most people who habitually use computers are aware that screens can often cause eyestrain and fatigue; this is usually due to flicker, poor legibility or low contrast. There have also been many concerns relating to the emission of radiation from screens. These can be categorized as follows:

- X-rays which are largely absorbed by the screen (but not at the rear!)
- ultraviolet and infrared radiation from phosphors in insignificant levels
- radio frequency emissions, plus ultrasound (approximately 16 kHz)
- electrostatic field which leaks out through the tube to the user. The intensity is dependent on distance and humidity. This can cause rashes in the user
- electromagnetic fields (50 Hz to 0.5 MHz) which create induction currents in conductive materials, including the human body. Two types of effects are attributed to this: in the visual system, a high incidence of cataracts in visual display unit (VDU) operators, and concern over reproductive disorders (miscarriages and birth defects).

Research into the potentially harmful effect of these emissions is generally inconclusive, in that it is difficult to determine precisely what the causes of illness are, and many health scares have been the result of misinformed media opinion rather than scientific fact. However, users who are pregnant ought to take especial care and observe simple precautions. Generally, there are a number of common-sense things that can be done to relieve strain and minimize any risk. These include

- not sitting too close to the screen
- not using very small fonts
- not looking at the screen for a long time without a break
- working in well-lit surroundings
- not placing the screen directly in front of a bright window.

Liquid crystal display

If you have used a personal organizer or notebook computer, you will have seen the light, flat plastic screens. These displays utilize liquid crystal technology and are smaller, lighter and consume far less power than traditional CRTs. These are also commonly referred to as flat-panel displays. They have no radiation problems associated with them, and are matrix addressable, which means that individual pixels can be accessed without the need for scanning.

Similar in principle to the digital watch, a thin layer of liquid crystal is sandwiched between two glass plates. The top plate is transparent and polarized, whilst the bottom plate is reflective. External light passes through the top plate and is polarized, which means that it only oscillates in one direction. This then passes through the crystal, reflects off the bottom plate and back to the eye, and so that cell looks white. When a voltage is applied to the crystal, via the conducting glass plates, the crystal twists. This causes it to turn the plane of polarization of the incoming light, rotating it so that it cannot return through the top plate, making the activated cell look black. The LCD requires refreshing at the usual rates, but the relatively slow response of the crystal means that flicker is not usually noticeable. The low intensity of the light emitted from the screen, coupled with the reduced flicker, means that the LCD is less tiring to use than standard CRT ones, with reduced eyestrain.

This different technology can be used to replace the standard screen on a desktop computer, and this is now common. However, the particular characteristics of compactness, light weight and low power consumption have meant that these screens have created a large niche in the computer market by monopolizing the notebook and portable computer systems side. The advent of these screens allowed small, light computers to be built, and created a large market that did not previously exist. Such computers, riding on the back of the technological wave, have opened up a different way of working for many people, who now have access to computers when away from the office, whether out on business or at home. Working in a different location on a smaller machine with different software obviously represents a different style of interaction and so once again we can see that differences in devices may alter the human–computer interaction considerably. The growing notebook computer market fed back into an investment in developing LCD screen technology, with supertwisted crystals increasing the viewing angle dramatically. Response times have also improved so that LCD screens are now used in personal DVD players and even in home television.

When the second edition of this book was being written the majority of LCD screens were black and white or grayscale, We wrote then ‘it will be interesting to see whether color LCD screens supersede grayscale by the time the third edition of this book is prepared’. Of course, this is precisely the case. Our expectation is that by the time we produce the next edition LCD monitors will have taken over from CRT monitors completely.

Special displays

There are a number of other display technologies used in niche markets. The one you are most likely to see is the gas plasma display, which is used in large screens (see Section 2.4.3 below).

The random scan display, also known as the *directed beam refresh*, or *vector display*, works differently from the bitmap display, also known as raster scan, that we discussed in Section 2.4.1. Instead of scanning the whole screen sequentially and horizontally, the random scan draws the lines to be displayed directly. By updating the screen at at least 30 Hz to reduce flicker, the direct drawing of lines at any angle means that jaggies are not created, and higher resolutions are possible, up to 4096 × 4096 pixels. Color on such displays is achieved using beam penetration technology, and is generally of a poorer quality. Eyestrain and fatigue are still a problem, and these displays are more expensive than raster scan ones, so they are now only used in niche applications.

The *direct view storage tube* is used extensively as the display for an analog storage oscilloscope, which is probably the only place that these displays are used in any great numbers. They are similar in operation to the random scan CRT but the image is maintained by flood guns which have the advantage of producing a stable display with no flicker. The screen image can be incrementally updated but not selectively erased; removing items has to be done by redrawing the new image on a completely erased screen. The screens have a high resolution, typically about 4096 × 3120 pixels, but suffer from low contrast, low brightness and a difficulty in displaying color.

2.4.3 Large displays and situated displays

Displays are no longer just things you have on your desktop or laptop. In Chapter 19 we will discuss meeting room environments that often depend on large shared screens. You may have attended lectures where the slides are projected from a computer onto a large screen. In shops and garages large screen adverts assault us from all sides.

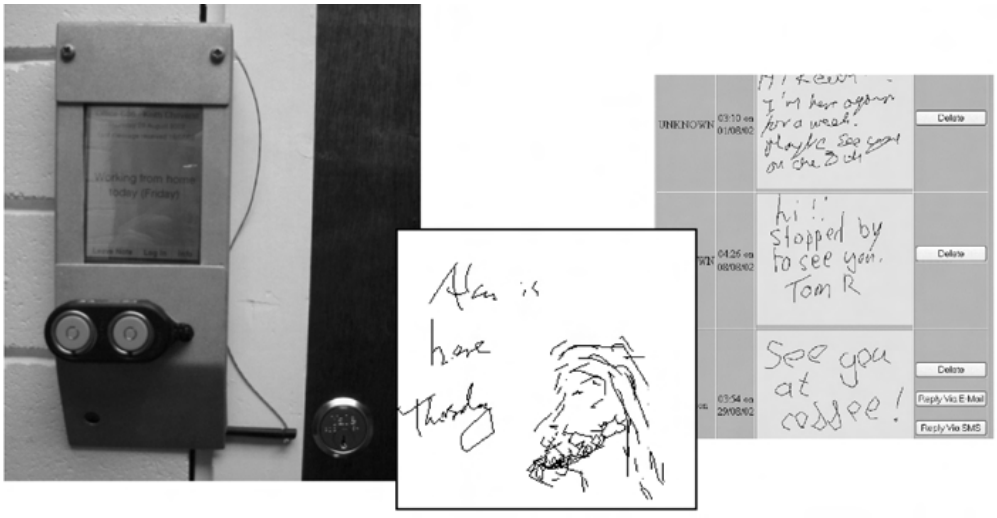
There are several types of large screen display. Some use gas plasma technology to create large flat bitmap displays. These behave just like a normal screen except they are big and usually have the HDTV (high definition television) wide screen format which has an aspect ratio of 16:9 instead of the 4:3 on traditional TV and monitors.

Where very large screen areas are required, several smaller screens, either LCD or CRT, can be placed together in a video wall. These can display separate images, or a single TV or computer image can be split up by software or hardware so that each screen displays a portion of the whole and the result is an enormous image. This is the technique often used in large concerts to display the artists or video images during the performance.

Possibly the large display you are most likely to have encountered is some sort of projector. There are two variants of these. In very large lecture theatres, especially older ones, you see projectors with large red, green and blue lenses. These each scan light across the screen to build a full color image. In smaller lecture theatres and in small meetings you are likely to see LCD projectors. Usually the size of a large book, these are like ordinary slide projectors except that where the slide would be there is a small LCD screen instead. The light from the projector passes through the tiny screen and is then focussed by the lens onto the screen.

The disadvantage of projected displays is that the presenter's shadow can often fall across the screen. Sometimes this is avoided in fixed lecture halls by using back projection. In a small room behind the screen of the lecture theatre there is a projector producing a right/left reversed image. The screen itself is a semi-frosted glass so that the image projected on the back can be seen in the lecture theatre. Because there are limits on how wide an angle the projector can manage without distortion, the size of the image is limited by the depth of the projection room behind, so these are less heavily used than front projection.

As well as for lectures and meetings, display screens can be used in various public places to offer information, link spaces or act as message areas. These are often called *situated displays* as they take their meaning from the location in which they are situated. These may be large screens where several people are expected to view or interact simultaneously, or they may be very small. Figure 2.11 shows an example of a small experimental situated display mounted by an office door to act as an electronic sticky note [70].



(i) Door display (ii) Occupant's note (iii) Web interface

Figure 2.11 Situated door display. Source: Courtesy of Keith Cheverst

DESIGN FOCUS

Hermes: a situated display

Office doors are often used as a noticeboard with messages from the occupant such as 'just gone out' or 'timetable for the week' and from visitors 'missed you, call when you get back'. The Hermes system is an electronic door display that offers some of the functions of sticky notes on a door [70]. Figure 2.11(i) shows an installed Hermes device fixed just beside the door, including the socket to use a Java iButton to authenticate the occupant. The occupant can leave messages that others can read (Figure 2.11(ii)) and people coming to the door can leave messages for the occupant. Electronic notes are smaller than paper ones, but because they are electronic they can be read remotely using a web interface (Figure 2.11(iii)), or added by SMS (see Chapter 19, Section 19.3.2).

The fact that it is situated – by a person's door – is very important. It establishes a context, 'Alan's door', and influences the way the system is used. For example, the idea of anonymous messages left on the door, where the visitor has had to be physically present, feels different from, say, anonymous emails.

See the book website for the full case study: /e3/casestudy/hermes/

2.4.4 Digital paper

A new form of 'display' that is still in its infancy is the various forms of digital paper. These are thin flexible materials that can be written to electronically, just like a computer screen, but which keep their contents even when removed from any electrical supply.

There are various technologies being investigated for this. One involves the whole surface being covered with tiny spheres, black one side, white the other. Electronics embedded into the material allow each tiny sphere to be rotated to make it black or white. When the electronic signal is removed the ball stays in its last orientation. A different technique has tiny tubes laid side by side. In each tube is light-absorbing liquid and a small reflective sphere. The sphere can be made to move to the top surface or away from it making the pixel white or black. Again the sphere stays in its last position once the electronic signal is removed.

Probably the first uses of these will be for large banners that can be reprogrammed or slowly animated. This is an ideal application, as it does not require very rapid updates and does not require the pixels to be small. As the technology matures, the aim is to have programmable sheets of paper that you attach to your computer to get a 'soft' printout that can later be changed. Perhaps one day you may be able to have a 'soft' book that appears just like a current book with soft pages that can be turned and skimmed, but where the contents and cover can be changed when you decide to download a new book from the net!

2.5 DEVICES FOR VIRTUAL REALITY AND 3D INTERACTION

Virtual reality (VR) systems and various forms of 3D visualization are discussed in detail in Chapter 20. These require you to navigate and interact in a three-dimensional space. Sometimes these use the ordinary controls and displays of a desktop computer system, but there are also special devices used both to move and interact with 3D objects and to enable you to see a 3D environment.

2.5.1 Positioning in 3D space

Virtual reality systems present a 3D virtual world. Users need to navigate through these spaces and manipulate the virtual objects they find there. Navigation is not simply a matter of moving to a particular location, but also of choosing a particular orientation. In addition, when you grab an object in real space, you don't simply move it around, but also twist and turn it, for example when opening a door. Thus the move from mice to 3D devices usually involves a change from two degrees of freedom to six degrees of freedom, not just three.

Cockpit and virtual controls

Helicopter and aircraft pilots already have to navigate in real space. Many arcade games and also more serious applications use controls modeled on an aircraft cockpit to 'fly' through virtual space. However, helicopter pilots are very skilled and it takes a lot of practice for users to be able to work easily in such environments.

In many PC games and *desktop virtual reality* (where the output is shown on an ordinary computer screen), the controls are themselves virtual. This may be a simulated form of the cockpit controls or more prosaic up/down left/right buttons. The user manipulates these virtual controls using an ordinary mouse (or other 2D device). Note that this means there are two levels of indirection. It is a tribute to the flexibility of the human mind that people can not only use such systems but also rapidly become proficient.

The 3D mouse

There are a variety of devices that act as 3D versions of a mouse. Rather than just moving the mouse on a tabletop, you can pick it up, move it in three dimensions, rotate the mouse and tip it forward and backward. The 3D mouse has a full six degrees of freedom as its position can be tracked (three degrees), and also its up/down angle (called *pitch*), its left/right orientation (called *yaw*) and the amount it is twisted about its own axis (called *roll*) (see Figure 2.12). Various sensors are used to track the mouse position and orientation: magnetic coils, ultrasound or even mechanical joints where the mouse is mounted rather like an angle-poise lamp.

With the 3D mouse, and indeed most 3D positioning devices, users may experience strain from having to hold the mouse in the air for a long period. Putting the

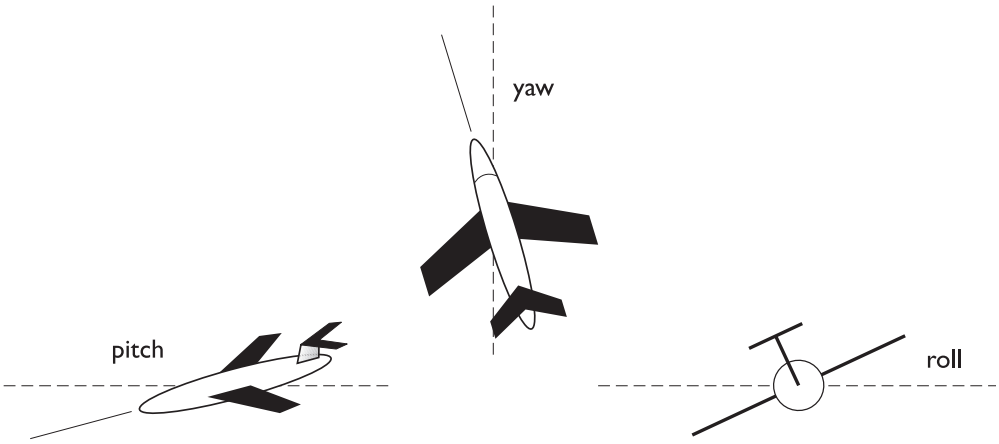


Figure 2.12 Pitch, yaw and roll

3D mouse down may even be treated as an action in the virtual environment, that is taking a nose dive.

Dataglove

One of the mainstays of high-end VR systems (see Chapter 20), the dataglove is a 3D input device. Consisting of a lycra glove with optical fibers laid along the fingers, it detects the joint angles of the fingers and thumb. As the fingers are bent, the fiber optic cable bends too; increasing bend causes more light to leak from the fiber, and the reduction in intensity is detected by the glove and related to the degree of bend in the joint. Attached to the top of the glove are two sensors that use ultrasound to determine 3D positional information as well as the angle of roll, that is the degree of wrist rotation. Such rich multi-dimensional input is currently a solution in search of a problem, in that most of the applications in use do not require such a comprehensive form of data input, whilst those that do cannot afford it. However, the availability of cheaper versions of the dataglove will encourage the development of more complex systems that are able to utilize the full power of the dataglove as an input device. There are a number of potential uses for this technology to assist disabled people, but cost remains the limiting factor at present.

The dataglove has the advantage that it is very easy to use, and is potentially very powerful and expressive (it can provide 10 joint angles, plus the 3D spatial information and degree of wrist rotation, 50 times a second). It suffers from extreme expense, and the fact that it is difficult to use in conjunction with a keyboard. However, such a limitation is shortsighted; one can imagine a keyboard drawn onto a desk, with software detecting hand positions and interpreting whether the virtual keys had been hit or not. The potential for the dataglove is vast; gesture recognition and sign language interpretation are two obvious areas that are the focus of active research, whilst less obvious applications are evolving all the time.

Virtual reality helmets

The helmets or goggles worn in some VR systems have two purposes: (i) they display the 3D world to each eye and (ii) they allow the user's head position to be tracked. We will discuss the former later when we consider output devices. The head tracking is used primarily to feed into the output side. As the user's head moves around the user ought to see different parts of the scene. However, some systems also use the user's head direction to determine the direction of movement within the space and even which objects to manipulate (rather like the eyegaze systems). You can think of this rather like leading a horse in reverse. If you want a horse to go in a particular direction, you use the reins to pull its head in the desired direction and the horse follows its head.

Whole-body tracking

Some VR systems aim to be immersive, that is to make the users feel as if they are really in the virtual world. In the real world it is possible (although not usually wise) to walk without looking in the direction you are going. If you are driving down the road and glance at something on the roadside you do not want the car to do a sudden 90-degree turn! Some VR systems therefore attempt to track different kinds of body movement. Some arcade games have a motorbike body on which you can lean into curves. More strangely, small trampolines have been wired up so that the user can control movement in virtual space by putting weight on different parts of the trampoline. The user can literally surf through virtual space. In the extreme the movement of the whole body may be tracked using devices similar to the dataglove, or using image-processing techniques. In the latter, white spots are stuck at various points of the user's body and the position of these tracked using two or more cameras, allowing the location of every joint to be mapped. Although the last of these sounds a little constraining for the fashion conscious it does point the way to less intrusive tracking techniques.

2.5.2 3D displays

Just as the 3D images used in VR have led to new forms of input device, they also require more sophisticated outputs. Desktop VR is delivered using a standard computer screen and a 3D impression is produced by using effects such as shadows, occlusion (where one object covers another) and perspective. This can be very effective and you can even view 3D images over the world wide web using a VRML (virtual reality markup language) enabled browser.

Seeing in 3D

Our eyes use many cues to perceive depth in the real world (see also Chapter 1). It is in fact quite remarkable as each eye sees only a flattened form of the world, like a photograph. One important effect is *stereoscopic vision* (or simply *stereo vision*).

Because each eye is looking at an object from a slightly different angle each sees a different image and our brain is able to use this to assess the relative distance of different objects. In desktop VR this stereoscopic effect is absent. However, various devices exist to deliver true stereoscopic images.

The start point of any stereoscopic device is the generation of images from different perspectives. As the computer is generating images for the virtual world anyway, this just means working out the right positions and angles corresponding to the typical distance between eyes on a human face. If this distance is too far from the natural one, the user will be presented with a giant's or gnat's eye view of the world!

Different techniques are then used to ensure that each eye sees the appropriate image. One method is to have two small screens fitted to a pair of goggles. A different image is then shown to each eye. These devices are currently still quite cumbersome and the popular image of VR is of a user with head encased in a helmet with something like a pair of inverted binoculars sticking out in front. However, smaller and lighter LCDs are now making it possible to reduce the devices towards the size and weight of ordinary spectacles.

An alternative method is to have a pair of special spectacles connected so that each eye can be blanked out by timed electrical signals. If this is synchronized with the frame rate of a computer monitor, each eye sees alternate images. Similar techniques use polarized filters in front of the monitor and spectacles with different polarized lenses. These techniques are both effectively using similar methods to the red–green 3D spectacles given away in some breakfast cereals. Indeed, these red–green spectacles have been used in experiments in wide-scale 3D television broadcasts. However, the quality of the 3D image from the polarized and blanked eye spectacles is substantially better.

The ideal would be to be able to look at a special 3D screen and see 3D images just as one does with a hologram – 3D television just like in all the best sci-fi movies! But there is no good solution to this yet. One method is to inscribe the screen with small vertical grooves forming hundreds of prisms. Each eye then sees only alternate dots on the screen allowing a stereo image at half the normal horizontal resolution. However, these screens have very narrow *viewing angles*, and are not ready yet for family viewing.

In fact, getting stereo images is not the whole story. Not only do our eyes see different things, but each eye also focusses on the current object of interest (small muscles change the shape of the lens in the pupil of the eye). The images presented to the eye are generated at some fixed focus, often with effectively infinite *depth of field*. This can be confusing and tiring. There has been some progress recently on using lasers to detect the focal depth of each eye and adjust the images correspondingly, similar to the technology used for eye tracking. However, this is not currently used extensively.

VR motion sickness

We all get annoyed when computers take a long time to change the screen, pop up a window, or play a digital movie. However, with VR the effects of poor display

performance can be more serious. In real life when we move our head the image our eyes see changes accordingly. VR systems produce the same effect by using sensors in the goggles or helmet and then using the position of the head to determine the right image to show. If the system is slow in producing these images a lag develops between the user moving his head and the scene changing. If this delay is more than a hundred milliseconds or so the feeling becomes disorienting. The effect is very similar to that of being at sea. You stand on the deck looking out to sea, the boat gently rocking below you. Tiny channels in your ears detect the movement telling your brain that you are moving; your eyes see the horizon moving in one direction and the boat in another. Your brain gets confused and you get sick. Users of VR can experience similar nausea and few can stand it for more than a short while. In fact, keeping laboratories sanitary has been a major push in improving VR technology.

Simulators and VR caves

Because of the problems of delivering a full 3D environment via head-mounted displays, some virtual reality systems work by putting the user within an environment where the virtual world is displayed upon it. The most obvious examples of this are large flight simulators – you go inside a mock-up of an aircraft cockpit and the scenes you would see through the windows are projected onto the virtual windows. In motorbike or skiing simulators in video arcades large screens are positioned to fill the main part of your visual field. You can still look over your shoulder and see your friends, but while you are engaged in the game it surrounds you.

More general-purpose rooms called caves have large displays positioned all around the user, or several back projectors. In these systems the user can look all around and see the virtual world surrounding them.

2.6 PHYSICAL CONTROLS, SENSORS AND SPECIAL DEVICES

As we have discussed, computers are coming out of the box. The mouse keyboard and screen of the traditional computer system are not relevant or possible in applications that now employ computers such as interactive TV, in-car navigation systems or personal entertainment. These devices may have special displays, may use sound, touch and smell as well as visual displays, may have dedicated controls and may sense the environment or your own bio-signs.

2.6.1 Special displays

Apart from the CRT screen there are a number of visual outputs utilized in complex systems, especially in embedded systems. These can take the form of analog representations of numerical values, such as dials, gauges or lights to signify a certain system state. Flashing light-emitting diodes (LEDs) are used on the back of some

computers to signify the processor state, whilst gauges and dials are found in process control systems. Once you start in this mode of thinking, you can contemplate numerous visual outputs that are unrelated to the screen. One visual display that has found a specialized niche is the head-up display that is used in aircraft. The pilot is fully occupied looking forward and finds it difficult to look around the cockpit to get information. There are many different things that need to be known, ranging from data from tactical systems to navigational information and aircraft status indicators. The head-up display projects a subset of this information into the pilot's line of vision so that the information is directly in front of her eyes. This obviates the need for large banks of information to be scanned with the corresponding lack of attention to what is happening outside, and makes the pilot's job easier. Less important information is usually presented on a smaller number of dials and gauges in the cockpit to avoid cluttering the head-up display, and these can be monitored less often, during times of low stress.

2.6.2 Sound output

Another mode of output that we should consider is that of auditory signals. Often designed to be used in conjunction with screen displays, auditory outputs are poorly understood: we do not yet know how to utilize sound in a sensible way to achieve maximum effect and information transference. We have discussed speech previously, but other sounds such as beeps, bongs, clanks, whistles and whirrs are all used to varying effect. As well as conveying system output, sounds offer an important level of feedback in interactive systems. Keyboards can be set to emit a click each time a key is pressed, and this appears to speed up interactive performance. Telephone keypads often sound different tones when the keys are pressed; a noise occurring signifies that the key has been successfully pressed, whilst the actual tone provides some information about the particular key that was pressed. The advantage of auditory feedback is evident when we consider a simple device such as a doorbell. If we press it and hear nothing, we are left undecided. Should we press it again, in case we did not do it right the first time, or did it ring but we did not hear it? And if we press it again but it actually did ring, will the people in the house think we are very rude, ringing insistently? We feel awkward and a little stressed. If we were using a computer system instead of a doorbell and were faced with a similar problem, we would not enjoy the interaction and would not perform as well. Yet it is a simple problem that could be easily rectified by a better initial design, using sound. Chapter 10 will discuss the use of the auditory channel in more detail.

2.6.3 Touch, feel and smell

Our other senses are used less in normal computer applications, but you may have played computer games where the joystick or artificial steering wheel vibrated, perhaps when a car was about to go off the track. In some VR applications, such as the use in medical domains to 'practice' surgical procedures, the *feel* of an instrument

moving through different tissue types is very important. The devices used to emulate these procedures have *force feedback*, giving different amounts of resistance depending on the state of the virtual operation. These various forms of force, resistance and texture that influence our physical senses are called *haptic* devices.

Haptic devices are not limited to virtual environments, but are used in specialist interfaces in the real world too. Electronic braille displays either have pins that rise or fall to give different patterns, or may involve small vibration pins. Force feedback has been used in the design of in-car controls.

In fact, the car gives a very good example of the power of tactile feedback. If you drive over a small bump in the road the car is sent slightly off course; however, the chances are that you will correct yourself before you are consciously aware of the bump. Within your body you have reactions that push back slightly against pressure to keep your limbs where you 'want' them, or move your limbs out of the way when you brush against something unexpected. These responses occur in your lower brain and are very fast, not involving any conscious effort. So, haptic devices can access very fast responses, but these responses are not fully controlled. This can be used effectively in design, but of course also with caution.

Texture is more difficult as it depends on small changes between neighboring points on the skin. Also, most of our senses notice change rather than fixed stimuli, so we usually feel textures when we move our fingers over a surface, not just when resting on it. Technology for this is just beginning to become available

There is evidence that smell is one of the strongest cues to memory. Various historical recreations such as the Jorvik Centre in York, England, use smells to create a feeling of immersion in their static displays of past life. Some arcade games also generate smells, for example, burning rubber as your racing car skids on the track. These examples both use a fixed smell in a particular location. There have been several attempts to produce devices to allow smells to be recreated dynamically in response to games or even internet sites. The technical difficulty is that our noses do not have a small set of basic smells that are mixed (like salt/sweet/sour/bitter/savoury on our tongue), but instead there are thousands of different types of receptor responding to different chemicals in the air. The general pattern of devices to generate smells is to have a large repertoire of tiny scent-containing capsules that are released in varying amounts on demand – rather like a printer cartridge with hundreds of ink colors! So far there appears to be no mass market for these devices, but they may eventually develop from niche markets.

Smell is a complex multi-dimensional sense and has a peculiar ability to trigger memory, but cannot be changed rapidly. These qualities may prove valuable in areas where a general sense of location and awareness is desirable. For example, a project at the Massachusetts Institute of Technology explored the use of a small battery of scent generators which may be particularly valuable for *ambient displays* and background awareness [198, 161].

DESIGN FOCUS

Feeling the road

In the BMW 7 Series you will find a single haptic feedback control for many of the functions that would normally have dedicated controls. It uses technology developed by Immersion Corporation who are also behind the force feedback found in many medical and entertainment haptic devices. The iDrive control slides backwards and forwards and rotates to give access to various menus and lists of options. The haptic feedback allows the user to feel 'clicks' appropriate to the number of items in the various menu lists.

See: www.immersion.com/ and www.bmw.com/ Picture courtesy of BMW AG

2.6.4 Physical controls

Look at Figure 2.13. In it you can see the controls for a microwave, a washing machine and a personal MiniDisc player. See how they each use very different physical devices: the microwave has a flat plastic sheet with soft buttons, the washing machine large switches and knobs, and the MiniDisc has small buttons and an interesting multi-function end.

A desktop computer system has to serve many functions and so has generic keys and controls that can be used for a variety of purposes. In contrast, these dedicated control panels have been designed for a particular device and for a single use. This is why they differ so much.

Looking first at the microwave, it has a flat plastic control panel. The buttons on the panel are pressed and 'give' slightly. The choice of the smooth panel is probably partly for visual design – it looks streamlined! However, there are also good practical reasons. The microwave is used in the kitchen whilst cooking, with hands that may be greasy or have food on them. The smooth controls have no gaps where food can accumulate and clog buttons, so it can easily be kept clean and hygienic.

When using the washing machine you are handling dirty clothes, which may be grubby, but not to the same extent, so the smooth easy-clean panel is less important (although some washing machines do have smooth panels). It has several major

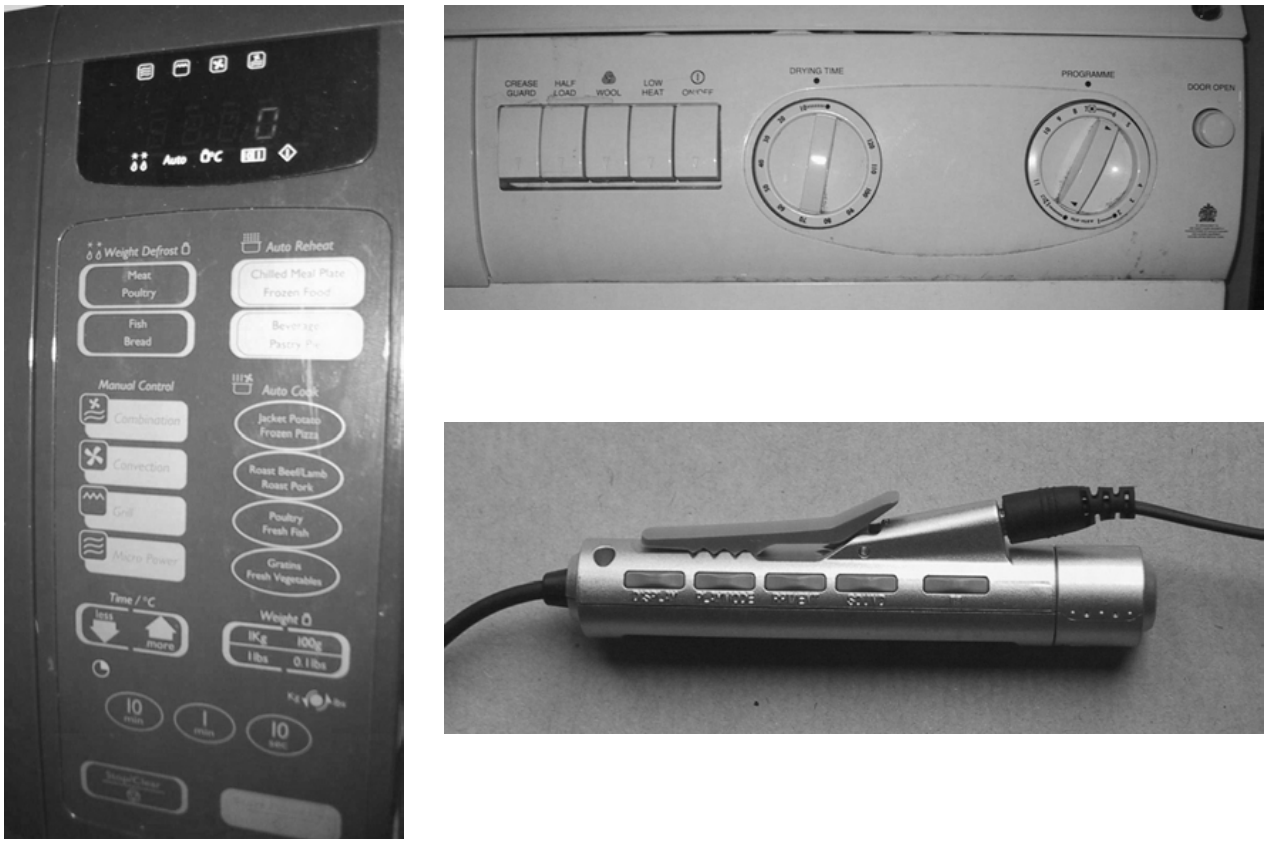


Figure 2.13 Physical controls on microwave, washing machine and MiniDisc. Source: Photograph bottom right by Alan Dix with permission from Sony (UK)

settings and the large buttons act both as control and display. Also the dials for dryer timer and the washing program act both as a means to set the desired time or program and to display the current state whilst the wash is in progress.

Finally, the MiniDisc controller needs to be small and unobtrusive. It has tiny buttons, but the end control is most interesting. It twists from side to side and also can be pulled and twisted. This means the same control can be used for two different purposes. This form of multi-function control is common in small devices.

We discussed the immediacy of haptic feedback and these lessons are also important at the level of creating physical devices; do keys, dials, etc., feel as if they have been pressed or turned? Getting the right level of resistance can make the device work naturally, give you feedback that you have done something, or let you know that you are controlling something. Where for some reason this is not possible, something has to be done to prevent the user getting confused, perhaps pressing buttons twice; for example, the smooth control panel of the microwave in Figure 2.13 offers no tactile feedback, but beeps for each keypress. We will discuss these design issues further when we look at user experience in Chapter 3 (Section 3.9).

Whereas texture is difficult to generate, it is easy to build into materials. This can make a difference to the ease of use of a device. For example, a touchpad is smooth, but a keyboard nipple is usually rubbery. If they were the other way round it would be hard to drag your finger across the touchpad or to operate the nipple without slipping. Texture can also be used to disambiguate. For example, most keyboards have a small raised dot on the 'home' keys for touch typists and some calculators and phones do the same on the '5' key. This is especially useful in applications when the eyes are elsewhere.

2.6.5 Environment and bio-sensing

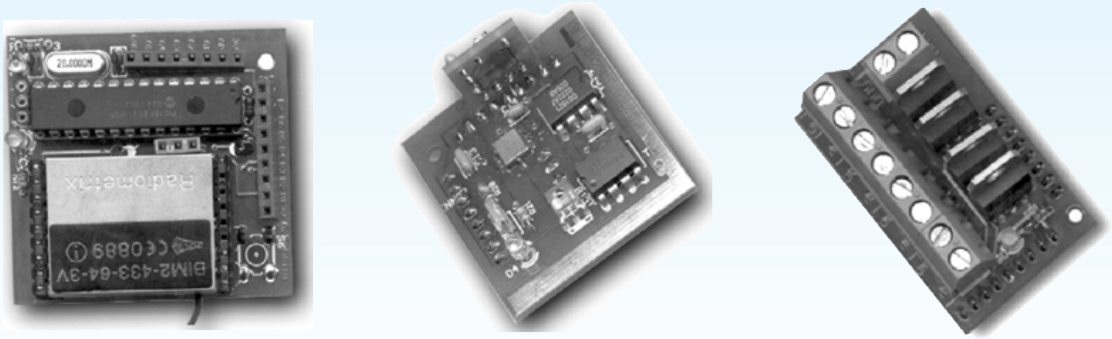
In a public washroom there are often no controls for the wash basins, you simply put your hands underneath and (hope that) the water flows. Similarly when you open the door of a car, the courtesy light turns on. The washbasin is controlled by a small infrared sensor that is triggered when your hands are in the basin (although it is

DESIGN FOCUS

Smart-Its – making using sensors easy

Building systems with physical sensors is no easy task. You need a soldering iron, plenty of experience in electronics, and even more patience. Although some issues are unique to each sensor or project, many of the basic building blocks are similar – connecting simple microprocessors to memory and networks, connecting various standard sensors such as temperature, tilt, etc.

The Smart-Its project has made this job easier by creating a collection of components and an architecture for adding new sensors. There are a number of basic Smart-It boards – the photo on the left shows a microprocessor with wireless connectivity. Onto these boards are plugged a variety of modules – in the center is a sensor board including temperature and light, and on the right is a power controller.



See: www.smart-its.org/ Source: Courtesy of Hans Gellersen

sometimes hard to find the 'sweet spot' where this happens!). The courtesy lights are triggered by a small switch in the car door.

Although we are not always conscious of them, there are many sensors in our environment – controlling automatic doors, energy saving lights, etc. and devices monitoring our behavior such as security tags in shops. The vision of ubiquitous computing (see Chapters 4 and 20) suggests that our world will be filled with such devices. Certainly the gap between science fiction and day-to-day life is narrow; for example, in the film *Minority Report* (20th Century Fox) iris scanners identify each passer-by to feed them dedicated advertisements, but you can buy just such an iris scanner as a security add-on for your home computer.

There are many different sensors available to measure virtually anything: temperature, movement (ultrasound, infrared, etc.), location (GPS, global positioning, in mobile devices), weight (pressure sensors). In addition audio and video information can be analyzed to identify individuals and to detect what they are doing. This all sounds big brother like, but is also used in ordinary applications, such as the washbasin.

Sensors can also be used to capture physiological signs such as body temperature, unconscious reactions such as blink rate, or unconscious aspects of activities such as typing rate, vocabulary shifts (e.g. modal verbs). For example, in a speech-based game, Tsukahara and Ward use gaps in speech and prosody (patterns of rhythm, pitch and loudness in speech) to infer the user's emotional state and thus the nature of acceptable responses [350] and Allanson discusses a variety of physiological sensors to create 'electrophysiological interactive computer systems' [12].

2.7 PAPER: PRINTING AND SCANNING

Some years ago, a recurrent theme of information technology was the *paperless office*. In the paperless office, documents would be produced, dispatched, read and filed online. The only time electronic information would be committed to paper would be when it went out of the office to ordinary customers, or to other firms who were laggards in this technological race. This vision was fuelled by rocketing property prices, and the realization that the floor space for a wastepaper basket could cost thousands in rent each year. Some years on, many traditional paper files are now online, but the desire for the completely paperless office has faded. Offices still have wastepaper baskets, and extra floor space is needed for the special computer tables to house 14-inch color monitors.

In this section, we will look at some of the available technology that exists to get information to and from paper. We will look first at printing, the basic technology, and issues raised by it. We will then go on to discuss the movement from paper back into electronic media. Although the paperless office is no longer seen as the goal, the less-paper office is perhaps closer, now that the technologies for moving between media are better.

2.7.1 Printing

If anything, computer systems have made it easier to produce paper documents. It is so easy to run off many copies of a letter (or book), in order to get it looking 'just right'. Older printers had a fixed set of characters available on a printhead. These varied from the traditional line printer to golf-ball and daisy-wheel printers. To change a typeface or the size of type meant changing the printhead, and was an awkward, and frequently messy, job, but for many years the daisy-wheel printer was the only means of producing high-quality output at an affordable price. However, the drop in the price of laser printers coupled with the availability of other cheap high-quality printers means that daisy-wheels are fast becoming a rarity.

All of the popular printing technologies, like screens, build the image on the paper as a series of dots. This enables, in theory, any character set or graphic to be printed,

Common types of dot-based printers

Dot-matrix printers

These use an inked ribbon, like a typewriter, but instead of a single character-shaped head striking the paper, a line of *pins* is used, each of which can strike the ribbon and hence dot the paper. Horizontal resolution can be varied by altering the speed of the head across the paper, and vertical resolution can be improved by sending the head twice across the paper at a slightly different position. So, dot-matrix printers can produce fast draft-quality output or slower 'letter'-quality output. They are cheap to run, but could not compete with the quality of jet and laser printers for general office and home printing. They are now only used for bulk printing, or where carbon paper is required for payslips, check printing, etc.)

Ink-jet and bubble-jet printers

These operate by sending tiny blobs of ink from the printhead to the paper. The ink is squirted at pressure from an ink-jet, whereas bubble-jets use heat to create a bubble. Both are quite quiet in operation. The ink from the bubble-jet (being a bubble rather than a droplet) dries more quickly than the ink-jet and so is less likely to smear. Both approach laser quality, but the bubble-jet dots tend to be more accurately positioned and of a less broken shape.

Laser printer

This uses similar technology to a photocopier: 'dots' of electrostatic charge are deposited on a drum, which then picks up toner (black powder). This is then rolled onto the paper and cured by heat. The curing is why laser printed documents come out warm, and the electrostatic charge is why they smell of ozone! In addition, some toner can be highly toxic if inhaled, but this is more a problem for full-time maintenance workers than end-users changing the occasional toner cartridge.

Laser printers give nearly typeset-quality output, with top-end printers used by desktop publishing firms. Indeed, many books are nowadays produced using laser printers. The authors of this book have produced camera-ready copy for other books on 300 and 600 dpi laser printers, although this book required higher quality and the first edition was typeset at 1200 dpi onto special bromide paper.

limited only by the resolution of the dots. This resolution is measured in *dots per inch* (dpi). Imagine a sheet of graph paper, and building up an image by putting dots at the intersection of each line. The number of lines per inch in each direction is the resolution in dpi. For some mechanical printers this is slightly confused: the dots printed may be bigger than the gaps, neighboring printheads may not be able to print simultaneously and may be offset relative to one another (a diamond-shaped rather than rectangular grid). These differences do not make too much difference to the user, but mean that, given two printers at the same nominal resolution, the output of one looks better than that of the other, because it has managed the physical constraints better.

The most common types of dot-based printers are dot-matrix printers, ink-jet printers and laser printers. These are listed roughly in order of increasing resolution and quality, where dot-matrix printers typically have a resolution of 80–120 dpi rising to about 300–600 dpi for ink-jet printers and 600–2400 dpi for laser printers. By varying the quantity of ink and quality of paper, ink-jet printers can be used to print photo-quality prints from digital photographs.

Printing in the workplace

Although ink-jet and laser printers have the lion's share of the office and home printer market, there are many more specialist applications that require different technology.

Most shop tills use dot-matrix printing where the arrangement is often very clever, with one printhead serving several purposes. The till will usually print one till roll which stays within the machine, recording all transactions for audit purposes. An identical receipt is printed for the customer. In addition, many will print onto the customer's own check or produce a credit card slip for the customer to sign. Sometimes the multiple copies are produced using two or more layers of paper where the top layer receives the ink and the lower layers use pressure-sensitive paper – not possible using ink-jet or laser technology. Alternatively, a single printhead may move back and forth over several small paper rolls within the same machine, as well as moving over the slot for the customer's own check.

As any printer owner will tell you, office printers are troublesome, especially as they age. Different printing technology is therefore needed in harsh environments or where a low level of supervision is required. Thermal printers use special heat-sensitive paper that changes color when heated. The printhead simply heats the paper where it wants a dot. Often only one line of dots is produced per pass, in contrast to dot-matrix and ink-jet printers, which have several pins or jets in parallel. The image is then produced using several passes per line, achieving a resolution similar to a dot-matrix. Thermal paper is relatively expensive and not particularly nice to look at, but thermal printers are mechanically simple and require little maintenance (no ink or toner splashing about). Thermal printers are used in niche applications, for example industrial equipment, some portable printers, and fax machines (although many now use plain paper).

As well as resolution, printers vary in speed and cost. Typically, office-quality ink-jet or laser printers produce between four and eight pages per minute. Dot-matrix printers are more often rated in *characters per second* (cps), and typical speeds may be 200 cps for draft and 50 cps for letter-quality print. In practice, this means no more than a page or so per minute. These are maximum speeds for simple text, and printers may operate much more slowly for graphics.

Color ink-jet printers are substantially cheaper than even monochrome laser printers. However, the recurrent costs of consumables may easily dominate this initial cost. Both jet and laser printers have special-purpose parts (print cartridges, toner, print drums), which need to be replaced every few thousand sheets; and they must also use high-grade paper. It may be more difficult to find suitable grades of recycled paper for laser printers.

2.7.2 Fonts and page description languages

Some printers can act in a mode whereby any characters sent to them (encoded in ASCII, see Section 2.8.5) are printed, typewriter style, in a single font. Another case, simple in theory, is when you have a bitmap picture and want to print it. The dots to print are sent to the printer, and no further interpretation is needed. However, in practice, it is rarely so simple.

Many printed documents are far more complex – they incorporate text in many different fonts and many sizes, often italicized, emboldened and underlined. Within the text you will find line drawings, digitized photographs and pictures generated from 'paint' packages, including the ubiquitous 'clip art'. Sometimes the computer does all the work, converting the page image into a bitmap of the right size to be sent to the printer. Alternatively, a description of the page may be sent to the printer. At the simplest level, this will include commands to set the print position on the page, and change the font size.

More sophisticated printers can accept a *page description language*, the most common of which is PostScript. This is a form of programming language for printing. It includes some standard programming constructs, but also some special ones: paths for drawing lines and curves, sophisticated character and font handling and scaled bitmaps. The idea is that the description of a page is far smaller than the associated bitmap, reducing the time taken to send the page to the printer. A bitmap version of an A4 laser printer page at 300 dpi takes 8 Mbytes; to send this down a standard serial printer cable would take 10 minutes! However, a computer in the printer has to interpret the PostScript program to print the page; this is typically faster than 10 minutes, but is still the limiting factor for many print jobs.

Text is printed in a font with a particular size and shape. The size of a font is measured in points (pt). The point is a printer's measure and is about 1/72 of an inch. The *point size* of the font is related to its height: a 12 point font has about six lines per inch. The shape of a font is determined by its *font name*, for example Times Roman, Courier or Helvetica. Times Roman font is similar to the type of many newspapers, such as *The Times*, whereas Courier has a typewritten shape.

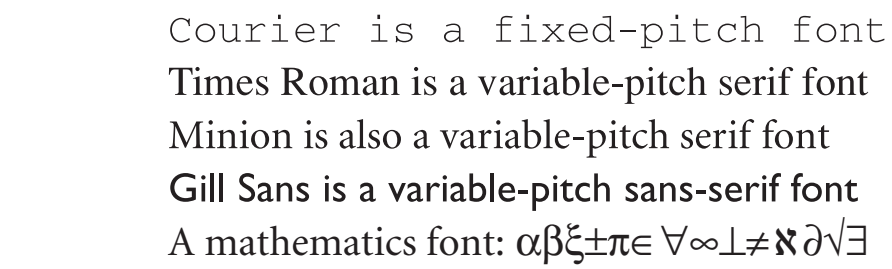


Figure 2.14 Examples of different fonts

Some fonts, such as Courier, are *fixed pitch*, that is each character has the same width. The alternative is a variable-pitched font, such as Times Roman or Gill Sans, where some characters, such as the 'm', are wider than others, such as the 'i'. Another characteristic of fonts is whether they are *serif* or *sans-serif*. A serif font has fine, short cross-lines at the ends of the strokes, imitating those found on cut stone lettering. A sans-serif font has square-ended strokes. In addition, there are special fonts looking like Gothic lettering or cursive script, and fonts of Greek letters and special mathematical symbols.

This book is set in 10 point Minion font using PostScript. Minion is a variable-pitched serif font. Figure 2.14 shows examples of different fonts.

DESIGN FOCUS

Readability of text

There is a substantial body of knowledge about the readability of text, both on screen and on paper. An MSc student visited a local software company and, on being shown some of their systems, remarked on the fact that they were using upper case throughout their displays. At that stage she had only completed part of an HCI course but she had read Chapter 1 of this book and already knew that WORDS WRITTEN IN BLOCK CAPITALS take longer to read than those in lower case. Recall that this is largely because of the clues given by word shapes and is the principle behind 'look and say' methods of teaching children to read. The company immediately recognized the value of the advice and she instantly rose in their esteem!

However, as with many interface design guidelines there are caveats. Although lower-case words are easier to read, individual letters and nonsense words are clearer in upper case. For example, one writes flight numbers as 'BA793' rather than 'ba793'. This is particularly important when naming keys to press (for example, 'Press Q to quit') as keyboards have upper-case legends.

Font shapes can also make a difference; for printed text, serif fonts make it easier to run one's eye along a line of text. However, they usually reproduce less well on screen where the resolution is poorer.

2.7.3 Screen and page

A common requirement of word processors and desktop publishing software is that *what you see is what you get* (see also Chapters 4 and 17), which is often called by its acronym *WYSIWYG* (pronounced whizz-ee-wig). This means that the appearance of the document on the screen should be the same as its eventual appearance on the printed page. In so far as this means that, for example, centered text is displayed centered on the screen, this is reasonable. However, this should not cloud the fact that screen and paper are very different media.

A typical screen resolution is about 72 dpi compared with a laser printer at over 600 dpi. Some packages can show magnified versions of the document in order to help in this. Most screens use an additive color model using red, green and blue light, whereas printers use a subtractive color model with cyan, magenta, yellow and black inks, so conversions have to be made. In addition, the sizes and aspect ratios are very different. An A4 page is about 11 inches tall by 8 wide (297×210 mm), whereas a screen is often of similar dimensions, but wider than it is tall.

These differences cause problems when designing software. Should you try to make the screen image as close to the paper as possible, or should you try to make the best of each? One approach to this would be to print only what could be displayed, but that would waste the extra resolution of the printer. On the other hand, one can try to make the screen as much like paper as possible, which is the intention behind the standard use of black text on a white background, rotatable A4 displays, and tablet PCs. This is a laudable aim, but cannot get rid of all the problems.

A particular problem lies with fonts. Imagine we have a line of 'm's, each having a width of 0.15 inch (4 mm). If we print them on a 72 dpi screen, then we can make the screen character 10 or 11 dots wide, in which case the screen version will be narrower or wider than the printed version. Alternatively, we can print the screen version as near as possible to where the printed characters would lie, in which case the 'm's on the screen would have different spaces between them: 'mm mm mm mm m'. The latter looks horrible on the screen, so most software chooses the former approach. This means that text that aligns on screen may not do so on printing. Some systems use a uniform representation for screen and printer, using the same font descriptions and even, in the case of the Next operating system, PostScript for screen display as well as printer output (also PDF with MacOS X). However, this simply exports the problem from the application program to the operating system.

The differences between screen and printer mean that different forms of graphic design are needed for each. For example, headings and changes in emphasis are made using font style and size on paper, but using color, brightness and line boxes on screen. This is not usually a problem for the display of the user's own documents as the aim is to give the user as good an impression of the printed page as possible, given the limitations. However, if one is designing parallel paper and screen forms, then one has to trade off consistency between the two representations with clarity in each.

An overall similar layout, but with different forms of presentation for details, may be appropriate.

2.7.4 Scanners and optical character recognition

Printers take electronic documents and put them on paper – *scanners* reverse this process. They start by turning the image into a bitmap, but with the aid of *optical character recognition* can convert the page right back into text. The image to be converted may be printed, but may also be a photograph or hand-drawn picture.

There are two main kinds of scanner: flat-bed and hand-held. With a flat-bed scanner, the page is placed on a flat glass plate and the whole page is converted into a bitmap. A variant of the flat-bed is where sheets to be scanned are pulled through the machine, common in multi-function devices (printer/fax/copier). Many flat-bed scanners allow a small pile of sheets to be placed in a feed tray so that they can all be scanned without user intervention. Hand-held scanners are pulled over the image by hand. As the head passes over an area it is read in, yielding a bitmap strip. A roller at the ends ensures that the scanner knows how fast it is being pulled and thus how big the image is. The scanner is typically only 3 or 4 inches (80 or 100 mm) wide and may even be the size of a large pen (mainly used for scanning individual lines of text). This means at least two or three strips must be 'glued' together by software to make a whole page image, quite a difficult process as the strips will overlap and may not be completely parallel to one another, as well as suffering from problems of different brightness and contrast. However, for desktop publishing small images such as photographs are quite common, and as long as one direction is less than the width of the scanner, they can be read in one pass.

Scanners work by shining a beam of light at the page and then recording the intensity and color of the reflection. Like photocopiers, the color of the light that is shone means that some colors may appear darker than others on a monochrome scanner. For example, if the light is pure red, then a red image will reflect the light completely and thus not appear on the scanned image.

Like printers, scanners differ in resolution, commonly between 600 and 2400 dpi, and like printers the quoted resolution needs careful interpretation. Many have a lower resolution scanhead but digitally interpolate additional pixels – the same is true for some digital cameras. Monochrome scanners are typically only found in multi-function devices, but color scanners usually have monochrome modes for black and white or grayscale copying. Scanners will usually return up to 256 levels of gray or RGB (red, green, blue) color. If a pure monochrome image is required (for instance, from a printed page), then it can *threshold* the grayscale image; that is, turn all pixels darker than some particular value black, and the rest white.

Scanners are used extensively in *desktop publishing* (*DTP*) for reading in hand-drawn pictures and photographs. This means that cut and paste can be performed electronically rather than with real glue. In addition, the images can be rotated,

scaled and otherwise transformed, using a variety of image manipulation software tools. Such tools are becoming increasingly powerful, allowing complex image transformations to be easily achieved; these range from color correction, through the merging of multiple images to the application of edge-detection and special effects filters. The use of multiple layers allows photomontage effects that would be impossible with traditional photographic or paper techniques. Even where a scanned image is simply going to be printed back out as part of a larger publication, some processing typically has to be performed to match the scanned colors with those produced during printing. For film photographs there are also special film scanners that can scan photographic negatives or color slides. Of course, if the photographs are digital no scanning is necessary.

Another application area is in document storage and retrieval systems, where paper documents are scanned and stored on computer rather than (or sometimes as well as) in a filing cabinet. The costs of maintaining paper records are enormous, and electronic storage can be cheaper, more reliable and more flexible. Storing a bitmap image is neither most useful (in terms of access methods), nor space efficient (as we will see later), so scanning may be combined with optical character recognition to obtain the text rather than the page image of the document.

Optical character recognition (OCR) is the process whereby the computer can 'read' the characters on the page. It is only comparatively recently that print could be reliably read, since the wide variety of typefaces and print sizes makes this more difficult than one would imagine – it is *not* simply a matter of matching a character shape to the image on the page. In fact, OCR is rather a misnomer nowadays as, although the document is optically scanned, the OCR software itself operates on the bitmap image. Current software can recognize 'unseen' fonts and can even produce output in word-processing formats, preserving super- and subscripts, centering, italics and so on.

Another important area is electronic publishing for multimedia and the world wide web. Whereas in desktop publishing the scanned image usually ends up (after editing) back on paper, in electronic publishing the scanned image is destined to be viewed on screen. These images may be used simply as digital photographs or may be made active, whereby clicking on some portion of the image causes pertinent information to be displayed (see Chapter 3 for more on the *point-and-click* style of interaction). One big problem when using electronic images is the plethora of formats for storing graphics (see Section 2.8.5). Another problem is the fact that different computers can display different numbers of colors and that the appearance of the same image on different monitors can be very different.

The importance of electronic publishing and also the ease of electronically manipulating images for printing have made the *digital camera* increasingly popular. Rather than capturing an image on film, a digital camera has a small light-sensitive chip that can directly record an image into memory.

Paper-based interaction

Paper is principally seen as an output medium. You type in some text, format it, print it and read it. The idea of the paperless office was to remove the paper from the write–read loop entirely, but it didn't fundamentally challenge its place in the cycle as an output medium. However, this view of paper as output has changed as OCR technology has improved and scanners become commonplace.

Workers at Xerox Palo Alto Research Center (also known as Xerox PARC) capitalized on this by using paper as a medium of interaction with computer systems [195]. A special identifying mark is printed onto forms and similar output. The printed forms may have check boxes or areas for writing numbers or (in block capitals!) words. The form can then be scanned back in. The system reads the identifying mark and thereby knows what sort of paper form it is dealing with. It doesn't have to use OCR on the printed text of the form as it printed it, but can detect the check boxes that have been filled in and even recognize the text that has been written. The identifying mark the researchers used is composed of backward and forward slashes, '\' and '/', and is called a *glyph*. An alternative would have been to use bar codes, but the slashes were found to fax and scan more reliably. The research version of this system was known as XAX, but it is now marketed as Xerox PaperWorks.

One application of this technology is mail order catalogs. The order form is printed with a glyph. When completed, forms can simply be collected into bundles and scanned in batches, generating orders automatically. If the customer faxes an order the fax-receiving software recognizes the glyph and the order is processed without ever being handled at the company end. Such a *paper user interface* may involve no screens or keyboards whatsoever.

Some types of paper now have identifying marks micro-printed like a form of textured watermark. This can be used both to identify the piece of paper (as the glyph does), and to identify the location on the paper. If this book were printed on such paper it would be possible to point at a word or diagram with a special pen-like device and have it work out what page you are on and where you are pointing and thus take you to appropriate web materials . . . perhaps the fourth edition . . .

It is paradoxical that Xerox PARC, where much of the driving work behind current 'mouse and window' computer interfaces began, has also developed this totally non-screen and non-mouse paradigm. However, the common principle behind each is the novel and appropriate use of different media for graceful interaction.

Worked exercise

What input and output devices would you use for the following systems? For each, compare and contrast alternatives, and if appropriate indicate why the conventional keyboard, mouse and CRT screen may be less suitable.

(a) portable word processor
(b) tourist information system
(c) tractor-mounted crop-spraying controller

(d) air traffic control system
(e) worldwide personal communications system
(f) digital cartographic system.

Answer In the later exercise on basic architecture (see Section 2.8.6), we focus on 'typical' systems, whereas here the emphasis is on the diversity of different devices needed for specialized purposes. You can 'collect' devices – watch out for shop tills, bank tellers, taxi meters, lift buttons, domestic appliances, etc.

(a) Portable word processor
The determining factors are size, weight and battery power. However, remember the purpose: this is a word processor not an address book or even a data entry device.
(i) LCD screen – low-power requirement
(ii) trackball or stylus for pointing
(iii) real keyboard – you can't word process without a reasonable keyboard and stylus handwriting recognition is not good enough
(iv) small, low-power bubble-jet printer – although not always necessary, this makes the package stand alone. It is probably not so necessary that the printer has a large battery capacity as printing can probably wait until a power point is found.

(b) Tourist information system
This is likely to be in a public place. Most users will only visit the system once, so the information and mode of interaction must be immediately obvious.
(i) touchscreen only – easy and direct interaction for first-time users (see also Chapter 3)
(ii) NO mice or styluses – in a public place they wouldn't stay long!

(c) Tractor-mounted crop-spraying controller
A hostile environment with plenty of mud and chemicals. Requires numerical input for flow rates, etc., but probably no text
(i) touch-sensitive keypad – ordinary keypads would get blocked up
(ii) small dedicated LED display (LCDs often can't be read in sunlight and large screens are fragile)
(iii) again no mice or styluses – they would get lost.

(d) Air traffic control system
The emphasis is on immediately available information and rapid interaction. The controller cannot afford to spend time searching for information; all frequently used information must be readily available.
(i) several specialized displays – including overlays of electronic information on radar
(ii) light pen or stylus – high-precision direct interaction
(iii) keyboard – for occasional text input, but consider making it fold out of the way.

(e) Worldwide personal communications system
Basically a super mobile phone! If it is to be kept on hand all the time it must be very light and pocket sized. However, to be a 'communications' system one would imagine that it should also act as a personal address/telephone book, etc.

(i) standard telephone keypad – the most frequent use
(ii) small dedicated LCD display – low power, specialized functions
(iii) possibly stylus for interaction – it allows relatively rich interaction with the address book software, but little space
(iv) a 'docking' facility – the system itself will be too small for a full-sized keyboard(!), but you won't want to enter in all your addresses and telephone numbers by stylus!

(f) Digital cartographic system
This calls for very high-precision input and output facilities. It is similar to CAD in terms of the screen facilities and printing, but in addition will require specialized data capture.
(i) large high-resolution color VDU (20 inch or bigger) – these tend to be enormously big (from back to front). LCD screens, although promising far thinner displays in the long term, cannot at present be made large enough
(ii) digitizing tablet – for tracing data on existing paper maps. It could also double up as a pointing device for some interaction
(iii) possibly thumbwheels – for detailed pointing and positioning tasks
(iv) large-format printer – indeed very large: an A2 or A1 plotter at minimum.

2.8 MEMORY

Like human memory, we can think of the computer's memory as operating at different levels, with those that have the faster access typically having less capacity. By analogy with the human memory, we can group these into short-term and long-term memories (STM and LTM), but the analogy is rather weak – the capacity of the computer's STM is a lot more than seven items! The different levels of computer memory are more commonly called primary and secondary storage.

The details of computer memory are not in themselves of direct interest to the user interface designer. However, the limitations in capacity and access methods are important constraints on the sort of interface that can be designed. After some fairly basic information, we will put the raw memory capacity into perspective with the sort of information which can be stored, as well as again seeing how advances in technology offer more scope for the designer to produce more effective interfaces. In particular, we will see how the capacity of typical memory copes with video images as these are becoming important as part of multimedia applications (see Chapter 21).

2.8.1 RAM and short-term memory (STM)

At the lowest level of computer memory are the registers on the computer chip, but these have little impact on the user except in so far as they affect the general speed of

the computer. Most currently active information is held in silicon-chip *random access memory* (*RAM*). Different forms of RAM differ as to their precise access times, power consumption and characteristics. Typical access times are of the order of 10 nanoseconds, that is a hundred-millionth of a second, and information can be accessed at a rate of around 100 Mbytes (million bytes) per second. Typical storage in modern personal computers is between 64 and 256 Mbytes.

Most RAM is *volatile*, that is its contents are lost when the power is turned off. However, many computers have small amount of *non-volatile RAM*, which retains its contents, perhaps with the aid of a small battery. This may be used to store setup information in a large computer, but in a pocket organizer will be the whole memory. Non-volatile RAM is more expensive so is only used where necessary, but with many notebook computers using very low-power static RAM, the divide is shrinking. By strict analogy, non-volatile RAM ought to be classed as LTM, but the important thing we want to emphasize is the gulf between STM and LTM in a traditional computer system.

In PDAs the distinctions become more confused as the battery power means that the system is never completely off, so RAM memory effectively lasts for ever. Some also use flash memory, which is a form of silicon memory that sits between fixed content ROM (read-only memory) chips and normal RAM. Flash memory is relatively slow to write, but once written retains its content even with no power whatsoever. These are sometimes called silicon disks on PDAs. Digital cameras typically store photographs in some form of flash media and small flash-based devices are used to plug into a laptop or desktop's USB port to transfer data.

2.8.2 Disks and long-term memory (LTM)

For most computer users the LTM consists of *disks*, possibly with small tapes for *backup*. The existence of backups, and appropriate software to generate and retrieve them, is an important area for user security. However, we will deal mainly with those forms of storage that impact the interactive computer user.

There are two main kinds of technology used in disks: *magnetic disks* and *optical disks*. The most common storage media, floppy disks and hard (or fixed) disks, are coated with magnetic material, like that found on an audio tape, on which the information is stored. Typical capacities of floppy disks lie between 300 kbytes and 1.4 Mbytes, but as they are removable, you can have as many as you have room for on your desk. Hard disks may store from under 40 Mbytes to several gigabytes (Gbytes), that is several thousand million bytes. With disks there are two access times to consider, the time taken to find the right track on the disk, and the time to read the track. The former dominates random reads, and is typically of the order of 10 ms for hard disks. The transfer rate once the track is found is then very high, perhaps several hundred kilobytes per second. Various forms of large removable media are also available, fitting somewhere between floppy disks and removable hard disks, and are especially important for multimedia storage.

Optical disks use laser light to read and (sometimes) write the information on the disk. There are various high capacity specialist optical devices, but the most common is the *CD-ROM*, using the same technology as audio compact discs. CD-ROMs have a capacity of around 650 megabytes, but cannot be written to at all. They are useful for published material such as online reference books, multimedia and software distribution. Recordable CDs are a form of *WORM* device (write-once read-many) and are more flexible in that information can be written, but (as the name suggests) only once at any location – more like a piece of paper than a blackboard. They are obviously very useful for backups and for producing very secure audit information. Finally, there are fully rewritable optical disks, but the rewrite time is typically much slower than the read time, so they are still primarily for archival not dynamic storage. Many CD-ROM reader/writers can also read DVD format, originally developed for storing movies. Optical media are more robust than magnetic disks and so it is easier to use a *jukebox* arrangement, whereby many optical disks can be brought online automatically as required. This can give an online capacity of many hundreds of gigabytes. However, as magnetic disk capacities have grown faster than the fixed standard of CD-ROMs, some massive capacity stores are moving to large disk arrays.

2.8.3 Understanding speed and capacity

So what effect do the various capacities and speeds have on the user? Thinking of our typical personal computer system, we can summarize some typical capacities as in Table 2.1.

We think first of documents. This book is about 320,000 words, or about 2 Mbytes, so it would hardly make a dent in 256 Mbytes of RAM. (This size – 2 Mbytes – is unformatted and without illustrations; the actual size of the full data files is an order of magnitude bigger, but still well within the capacity of main memory.) To take a more popular work, the Bible would use about 4.5 Mbytes. This would still consume only 2% of main memory, and disappear on a hard disk. However, it might look tight on a smaller PDA. This makes the memory look not too bad, so long as you do not intend to put your entire library online. However, many word processors come with a dictionary and thesaurus, and there is no standard way to use the same one with several products. Together with help files and the program itself, it is not

Table 2.1 Typical capacities of different storage media

	STM small/fast	LTM large/slower
Media:	RAM	Hard disk
Capacity:	256 Mbytes	100 Gbytes
Access time:	10 ns	7 ms
Transfer rate:	100 Mbyte/s	30 Mbyte/s

unusual to find each application consuming tens or even hundreds of megabytes of disk space – it is not difficult to fill a few gigabytes of disk at all!

Similarly, although 256 Mbytes of RAM are enough to hold most (but not all) single programs, windowed systems will run several applications simultaneously, soon using up many megabytes. Operating systems handle this by *paging* unused bits of programs out of RAM onto disk, or even *swapping* the entire program onto disk. This makes little difference to the logical functioning of the program, but has a significant effect on interaction. If you select a window, and the relevant application happens to be currently swapped out onto the disk, it has to be swapped back in. The delay this causes can be considerable, and is both noticeable and annoying on many systems.

Technological change and storage capacity

Most of the changes in this book since the first and second editions have been additions where new developments have come along. However, this portion has had to be scrutinized line by line as the storage capacities of high-end machines when this book was first published in 1993 looked ridiculous as we revised it in 1997 and then again in 2003. One of our aims in this chapter was to give readers a concrete feel for the capacities and computational possibilities in standard computers. However, the pace of advances in this area means that it becomes out of date almost as fast as it is written! This is also a problem for design; it is easy to build a system that is sensible given a particular level of technology, but becomes meaningless later. The solution is either to issue ever more frequent updates and new versions, or to exercise a bit of foresight . . .

The delays due to swapping are a symptom of the *von Neumann bottleneck* between disk and main memory. There is plenty of information in the memory, but it is not where it is wanted, in the machine's RAM. The path between them is limited by the transfer rate of the disk and is too slow. Swapping due to the operating system may be difficult to avoid, but for an interactive system designer some of these problems can be avoided by thinking carefully about where information is stored and when it is transferred. For example, the program can be *lazy* about information transfer. Imagine the user wants to look at a document. Rather than reading in the whole thing before letting the user continue, just enough is read in for the first page to be displayed, and the rest is read during idle moments.

Returning to documents, if they are scanned as bitmaps (and not read using OCR), then the capacity of our system looks even less impressive. Say an 11×8 inch (297×210 mm) page is scanned with an 8 bit grayscale (256 levels) setting at 1200 dpi. The image contains about one billion bits, that is about 128 Mbyte. So, our 100 Gbyte disk could store 800 pages – just OK for this book, but not for the Bible.

If we turn to video, things are even worse. Imagine we want to store moving video using 12 bits for each pixel (4 bits for each primary color giving 16 levels of brightness), each frame is 512×512 pixels, and we store at 25 frames per second.

This is by no means a high-quality image, but each frame requires approximately 400 kbytes giving 10 Mbytes per second. Our disk will manage about three hours of video – one good movie. Lowering our sights to still photographs, good digital cameras usually take 2 to 4 mega pixels at 24 bit color; that is 10 Mbytes of raw uncompressed image – you'd not get all your holiday snaps into main memory!

2.8.4 Compression

In fact, things are not quite so bad, since *compression* techniques can be used to reduce the amount of storage required for text, bitmaps and video. All of these things are highly redundant. Consider text for a moment. In English, we know that if we use the letter 'q' then 'u' is almost bound to follow. At the level of words, some words like 'the' and 'and' appear frequently in text in general, and for any particular work one can find other common terms (this book mentions 'user' and 'computer' rather frequently). Similarly, in a bitmap, if one bit is white, there is a good chance the next will be as well. Compression algorithms take advantage of this redundancy. For example, *Huffman encoding* gives short codes to frequent words [182], and *run-length encoding* represents long runs of the same value by length value pairs. Text can easily be reduced by a factor of five and bitmaps often compress to 1% of their original size.

For video, in addition to compressing each frame, we can take advantage of the fact that successive frames are often similar. We can compute the *difference* between successive frames and then store only this – compressed, of course. More sophisticated algorithms detect when the camera pans and use this information also. These differencing methods fail when the scene changes, and so the process periodically has to restart and send a new, complete (but compressed) image. For storage purposes this is not a problem, but when used for transmission over telephone lines or networks it can mean glitches in the video as the system catches up.

With these reductions it is certainly possible to store low-quality video at 64 kbyte/s; that is, we can store five hours of highly compressed video on our 1 Gbyte hard disk. However, it still makes the humble video cassette look very good value.

Probably the leading edge of video still and photographic compression is *fractal compression*. Fractals have been popularized by the images of the *Mandelbrot set* (that swirling pattern of computer-generated colors seen on many T-shirts and posters). Fractals refer to any image that contains parts which, when suitably scaled, are similar to the whole. If we look at an image, it is possible to find parts which are approximately self-similar, and these parts can be stored as a fractal with only a few numeric parameters. Fractal compression is especially good for textured features, which cause problems for other compression techniques. The *decompression* of the image can be performed to any degree of accuracy, from a very rough soft-focus image, to one *more* detailed than the original. The former is very useful as one can produce poor-quality output quickly, and better quality given more time. The latter is rather remarkable – the fractal compression actually fills in details that are not in the original. These details are not accurate, but look convincing!

2.8.5 Storage format and standards

The most common data types stored by interactive programs are text and bitmap images, with increasing use of video and audio, and this subsection looks at the ridiculous range of file storage standards. We will consider database retrieval in the next subsection.

The basic standard for text storage is the *ASCII* (American standard code for information interchange) character codes, which assign to each standard printable character and several control characters an internationally recognized 7 bit code (decimal values 0–127), which can therefore be stored in an 8 bit byte, or be transmitted as 8 bits including parity. Many systems extend the codes to the values 128–255, including line-drawing characters, mathematical symbols and international letters such as 'æ'. There is a 16 bit extension, the UNICODE standard, which has enough room for a much larger range of characters including the Japanese Kanji character set.

As we have already discussed, modern documents consist of more than just characters. The text is in different fonts and includes formatting information such as centering, page headers and footers. On the whole, the storage of formatted text is vendor specific, since virtually every application has its own file format. This is not helped by the fact that many suppliers attempt to keep their file formats secret, or update them frequently to stop others' products being compatible. With the exception of bare ASCII, the most common shared format is *rich text format* (*RTF*), which encodes formatting information including style sheets. However, even where an application will import or export RTF, it may represent a cut-down version of the full document style.

RTF regards the document as formatted text, that is it concentrates on the appearance. Documents can also be regarded as structured objects: this book has chapters containing sections, subsections . . . paragraphs, sentences, words and characters. There are *ISO standards* for document structure and interchange, which in theory could be used for transfer between packages and sites, but these are rarely used in practice. Just as the PostScript language is used to describe the printed page, *SGML* (*standard generalized markup language*) can be used to store structured text in a reasonably extensible way. You can define your own structures (the definition itself in SGML), and produce documents according to them. XML (extensible markup language), a lightweight version of SGML, is now used extensively for web-based applications.

For bitmap storage the range of formats is seemingly unending. The stored image needs to record the size of the image, the number of bits per pixel, possibly a color map, as well as the bits of the image itself. In addition, an icon may have a 'hot-spot' for use as a cursor. If you think of all the ways of encoding these features, or leaving them implicit, and then consider all the combinations of these different encodings, you can see why there are problems. And all this before we have even considered the effects of compression! There is, in fact, a whole software industry producing packages that convert from one format to another.

Given the range of storage standards (or rather lack of standards), there is no easy advice as to which is best, but if you are writing a new word processor and are about to decide how to store the document on disk, think, just for a moment, before defining yet another format.

2.8.6 Methods of access

Standard database access is by special key fields with an associated index. The user has to know the key before the system can find the information. A telephone directory is a good example of this. You can find out someone's telephone number if you know their name (the key), but you cannot find the name given the number. This is evident in the interface of many computer systems. So often, when you contact an organization, they can only help you if you give your customer number, or last order number. The usability of the system is seriously impaired by a shortsighted reliance on a single key and index. In fact, most database systems will allow multiple keys and indices, allowing you to find a record given partial information. So these problems are avoidable with only slight foresight.

There are valid reasons for not indexing on too many items. Adding extra indices adds to the size of the database, so one has to balance ease of use against storage cost. However, with ever-increasing disk sizes, this is not a good excuse for all but extreme examples. Unfortunately, brought up on lectures about algorithmic efficiency, it is easy for computer scientists to be stingy with storage. Another, more valid, reason for restricting the fields you index is privacy and security. For example, telephone companies will typically hold an online index that, given a telephone number, would return the name and address of the subscriber, but to protect the privacy of their customers, this information is not divulged to the general public.

It is often said that dictionaries are only useful for people who can spell. Bad spellers do not know what a word looks like so cannot look it up to find out. Not only in spelling packages, but in general, an application can help the user by matching badly spelt versions of keywords. One example of this is *do what I mean* (*DWIM*) used in several of Xerox PARC's experimental programming environments. If a command name is misspelt the system prompts the user with a close correct name. Menu-based systems make this less of an issue, but one can easily imagine doing the same with, say, file selection. Another important instance of this principle is *Soundex*, a way of indexing words, especially names. Given a key, Soundex finds those words which sound similar. For example, given McCloud, it would find MacCleod. These are all examples of *forgiving systems*, and in general one should aim to accommodate the user's mistakes. Again, there are exceptions to this: you do not want a bank's automated teller machine (ATM) to give money when the PIN number is *almost* correct!

Not all databases allow long passages of text to be stored in records, perhaps setting a maximum length for text strings, or demanding the length be fixed in advance. Where this is the case, the database seriously restricts interface applications where text forms an important part. At the other extreme, *free text retrieval* systems are centered on unformatted, unstructured text. These systems work by keeping an index of every word in every document, and so you can ask 'give me all documents with the words "human" and "computer" in them'. Programs, such as versions of the UNIX 'grep' command, give some of the same facilities by quickly scanning a list of files for a certain word, but are much slower. On the web, free text search is of course the standard way to find things using search engines.

Worked exercise *What is the basic architecture of a computer system?*

Answer In an HCI context, you should be assessing the architecture from the point of view of the user. The material for this question is scattered throughout the chapter. Look too at personal computer magazines, where adverts and articles will give you some idea of typical capabilities . . . and costs. They may also raise some questions: just what is the difference to the user between an 8 ms and a 10 ms disk drive?

The example answer below gives the general style, although more detail would be expected of a full answer. In particular, you need to develop a feel for capacities either as ball-park figures or in terms of typical capabilities (seconds of video, pages of text).

Example

The basic architecture of a computer system consists of the computer itself (with associated memory), input and output devices for user interaction and various forms of hard-copy devices. (Note, the 'computer science' answer regards output to the user and output to a printer as essentially equivalent. This is not an acceptable user-centered view.)

A typical configuration of user input–output devices would be a screen with a keyboard for typing text and a mouse for pointing and positioning. Depending on circumstance, different pointing devices may be used such as a stylus (for more direct interaction) or a touchpad (especially on portable computers).

The computer itself can be considered as composed of some processing element and memory. The memory is itself divided into short-term memory which is lost when the machine is turned off and permanent memory which persists.

2.9 PROCESSING AND NETWORKS

Computers that run interactive programs will process in the order of 100 million instructions per second. It sounds a lot and yet, like memory, it can soon be used up. Indeed, the first program written by one of the authors (some while ago) 'hung' and all attempts to debug it failed. Later calculation showed that the program would have taken more than the known age of the universe to complete! Failures need not be as spectacular as that to render a system unusable. Consider, for example, one drawing system known to the authors. To draw a line you press down the mouse button at one end, drag the mouse and then release the mouse button at the other end of the line – but not too quickly. You have to press down the button and then actually hold your hand steady for a moment, otherwise the line starts half way! For activities involving the user's hand–eye coordination, delays of even a fraction of a second can be disastrous.

Moore's law

Everyone knows that computers just get faster and faster. However, in 1965 Gordon Moore, co-founder of Intel, noticed a regularity. It seemed that the speed of processors, related closely to the number of transistors that could be squashed on a silicon wafer, was doubling every 18 months – exponential growth. One of the authors bought his first 'proper' computer in 1987; it was a blindingly fast 1.47 MHz IBM compatible (Macs were too expensive). By 2002 a system costing the same in real terms would have had a 1.5 GHz processor – 1000 times faster or 2^{10} in 15 years, that is 10×18 months.

There is a similar pattern for computer memory, except that the doubling time for magnetic storage seems to be closer to one year. For example, when the first edition of this book was written one of the authors had a 20 Mbyte hard disk; now, 11 years later, his disk is 30 Gbytes – around 2^{10} times more storage in just 10 years.

The effects of this are dramatic. If you took a young baby today and started recording a full audio video diary of every moment, day and night, of that child's life, by the time she was an old lady her whole life experience would fit into memory the size of a small grain of dust.

For more on Moore's law and life recording see: /e3/online/moores-law/

2.9.1 Effects of finite processor speed

As we can see, speed of processing can seriously affect the user interface. These effects must be taken into account when designing an interactive system. There are two sorts of faults due to processing speed: those when it is too slow, and those when it is too fast!

We saw one example of the former above. This was a *functional fault*, in that the program did the wrong thing. The system is supposed to draw lines from where the mouse button is depressed to where it is released. However, the program gets it wrong – after realizing the button is down, it does not check the position of the mouse fast enough, and so the user may have moved the mouse before the start position is registered. This is a fault at the implementation stage of the system rather than of the design. But to be fair, the programmer may not be given the right sort of information from lower levels of system software.

A second fault due to slow processing is where, in a sense, the program does the right thing, but the feedback is too slow, leading to strange effects at the interface. In order to avoid faults of the first kind, the system *buffers* the user input; that is, it remembers keypresses and mouse buttons and movement. Unfortunately, this leads to problems of its own. One example of this sort of problem is *cursor tracking*, which happens in character-based text editors. The user is trying to move backwards on the same line to correct an error, and so presses the cursor-left key. The cursor moves and when it is over the correct position, the user releases the key. Unfortunately, the system is behind in responding to the user, and so has a few more cursor-left keys

to process – the cursor then overshoots. The user tries to correct this by pressing the cursor-right key, and again overshoots. There is typically no way for the user to tell whether the buffer is empty or not, except by interacting very slowly with the system and observing that the cursor has moved after every keypress.

A similar problem, *icon wars*, occurs on window systems. The user clicks the mouse on a menu or icon, and nothing happens; for some reason the machine is busy or slow. So the user clicks again, tries something else – then, suddenly, all the buffered mouse clicks are interpreted and the screen becomes a blur of flashing windows and menus. This time, it is not so much that the response is too slow – it is fast enough when it happens – but that the response is variable. The delays due to swapping programs in and out of main memory typically cause these problems.

Furthermore, a style of interaction that is optimal on one machine may not be so on a slower machine. In particular, mouse-based interfaces cannot tolerate delays between actions and feedback of more than a fraction of a second, otherwise the immediacy required for successful interaction is lost. If these responses cannot be met then a more old-fashioned, command-based interface may be required.

Whereas it is immediately obvious that slow responses can cause problems for the user, it is not so obvious why one should not always aim for a system to be as fast as possible. However, there are exceptions to this – the user must be able to read and understand the output of the system. For example, one of the authors was once given a demonstration disk for a spreadsheet. Unfortunately, the machine the demo was written on was clearly slower than the author's machine, not much, at worst half the speed, but different enough. The demo passed in a blur over the screen with nothing remaining on the screen long enough to read. Many high-resolution monitors suffer from a similar problem when they display text. Whereas older character-based terminals scrolled new text from the bottom of the screen or redrew from the top, bitmap screens often 'flash' up the new page, giving no indication of direction of movement. A final example is the rate of cursor flashing: the rate is often at a fixed

DESIGN FOCUS

The myth of the infinitely fast machine

The adverse effects of slow processing are made worse because the designers labor under the *myth of the infinitely fast machine* [93]. That is, they design and document their systems as if response will be immediate. Rather than blithely hoping that the eventual machine will be 'fast enough', the designer ought to plan explicitly for slow responses where these are possible. A good example, where buffering is clear and audible (if not visible) to the user, is telephones. Even if the user gets ahead of the telephone when entering a number, the tones can be heard as they are sent over the line. Now this is probably an accident of the design rather than deliberate policy, as there are so many other problems with telephones as interfaces. However, this type of serendipitous feedback should be emulated in other areas.

frequency, so varying the speed of the processor does not change the screen display. But a rate which is acceptable for a CRT screen is too fast for an LCD screen, which is more persistent, and the cursor may become invisible or a slight gray color.

In some ways the solution to these problems is easier: the designer can demand fixed delays (dependent on media and user preference) rather than just going as fast as the machine allows. To plan for the first problem, that of insufficient speed, the designer needs to understand the limitations of the computer system and take account of these at all stages in the design process.

2.9.2 Limitations on interactive performance

There are several factors that can limit the speed of an interactive system:

Computation bound This is rare for an interactive program, but possible, for example when using find/replace in a large document. The system should be designed so that long delays are not in the middle of interaction and so that the user gets some idea of how the job is progressing. For a very long process try to give an indication of duration *before* it starts; and during processing an indication of the stage that the process has reached is helpful. This can be achieved by having a counter or slowly filling bar on the screen that indicates the amount done, or by changing the cursor to indicate that processing is occurring. Many systems notice after they have been computing for some time and then say 'this may take some time: continue (Y/N)?'. Of course, by the time it says this the process may be nearly finished anyway!

Storage channel bound As we discussed in the previous section, the speed of memory access can interfere with interactive performance. We discussed one technique, laziness, for reducing this effect. In addition, if there is plenty of raw computation power and the system is held up solely by memory, it is possible to trade off memory against processing speed. For example, compressed data take less space to store, and is faster to read in and out, but must be compressed before storage and decompressed when retrieved. Thus faster memory access leads to increased processing time. If data is written more often than it is read, one can choose a technique that is expensive to compress but fairly simple to decompress. For many interactive systems the ability to browse quickly is very important, but users will accept delays when saving updated information.

Graphics bound For many modern interfaces, this is the most common bottleneck. It is easy to underestimate the time taken to perform what appear to be simple interface operations. Sometimes clever coding can reduce the time taken by common graphics operations, and there is tremendous variability in performance between programs running on the same hardware. Most computers include a special-purpose *graphics card* to handle many of the most common graphics operations. This is optimized for graphics operations and allows the main processor to do other work such as manipulating documents and other user data.

Network capacity Most computers are linked by networks. At the simplest this can mean using shared files on a remote machine. When accessing such files it can be the speed of the network rather than that of the memory which limits performance. This is discussed in greater detail below.

2.9.3 Networked computing

Computer systems in use today are much more powerful than they were a few years ago, which means that the standard computer on the desktop is quite capable of high-performance interaction without recourse to outside help. However, it is often the case that we use computers not in their standalone mode of operation, but linked together in networks. This brings added benefits in allowing communication between different parties, provided they are connected into the same network, as well as allowing the desktop computer to access resources remote from itself. Such networks are inherently much more powerful than the individual computers that make up the network: increased computing power and memory are only part of the story, since the effects of allowing people much more extensive, faster and easier access to information are highly significant to individuals, groups and institutions.

One of the biggest changes since the first edition of this book has been the explosive growth of the internet and global connectivity. As well as fixed networks it is now normal to use a high bandwidth modem or wireless local area network (LAN) to connect into the internet and world wide web from home or hotel room anywhere in the world. The effects of this on society at large can only be speculated upon at present, but there are already major effects on computer purchases and perhaps the whole face of personal computation. As more and more people buy computers principally to connect to the internet the idea of the *network computer* has arisen – a small computer with no disks whose sole purpose is to connect up to networks.

The internet

The internet has its roots back in 1969 as DARPANET when the US Government's Department of Defense commissioned research into networking. The initial four mainframe computers grew to 23 in 1971 and the system had been renamed ARPANET. Growth has accelerated ever since: in 1984 there were over a thousand machines connected, in 1989 the 100,000 mark had been reached, and the latest estimates are in the millions. All the computers on the system, now known as the internet, speak a set of common languages (protocols); the two most important of these are *Transmission Control Protocol* (*TCP*) which moves data from A to B, and the *Internet Protocol* (*IP*) which specifies which B is being referred to so that the data goes to the correct place. Together these protocols are known as *TCP/IP*. Thus, at its most basic level, the internet is simply millions of computers connected together and talking to each other. Other protocols then build on these low-level capabilities to provide services such as electronic mail, in which participants send messages to each other; news, where articles of interest are posted to a special interest group and can be read by anyone subscribing to that group; and of course the world wide web.

Such networked systems have an effect on interactivity, over and above any additional access to distant peripherals or information sources. Networks sometimes operate over large distances, and the transmission of information may take some appreciable time, which affects the response time of the system and hence the nature of the interactivity. There may be a noticeable delay in response, and if the user is not informed of what is going on, he may assume that his command has been ignored, or lost, and may then repeat it. This lack of feedback is an important factor in the poor performance and frustration users feel when using such systems, and can be alleviated by more sensible use of the capabilities of the desktop machine to inform users of what is happening over the network.

Another effect is that the interaction between human and machine becomes an open loop, rather than a closed one. Many people may be interacting with the machine at once, and their actions may affect the response to your own. Many users accessing a single central machine will slow its response; database updates carried out by one user may mean that the same query by another user at slightly different times may produce different results. The networked computer system, by the very nature of its dispersal, distribution and multi-user access, has been transformed from a fully predictable, deterministic system, under the total control of the user, into a non-deterministic one, with an individual user being unaware of many important things that are happening to the system as a whole. Such systems pose a particular problem since ideals of consistency, informative feedback and predictable response are violated (see Chapter 7 for more on these principles). However, the additional power and flexibility offered by networked systems means that they are likely to be with us for a long time, and these issues need to be carefully addressed in their design.

Worked exercise

How do you think new, fast, high-density memory devices and quick processors have influenced recent developments in HCI? Do they make systems any easier to use? Do they expand the range of applications of computer systems?

Answer

Arguably it is not so much the increase in computer power as the decrease in the cost of that power which has had the most profound effect. Because 'ordinary' users have powerful machines on their desktops it has become possible to view that power as available for the interface rather than hoarded for number-crunching applications.

Modern graphical interaction consumes vast amounts of processing power and would have been completely impossible only a few years ago. There is an extent to which systems have to run faster to stay still, in that as screen size, resolution and color range increase, so does the necessary processing power to maintain the 'same' interaction. However, this extra processing is not really producing the same effect; screen quality is still a major block on effective interaction.

The increase in RAM means that larger programs can be written, effectively allowing the programmer 'elbow room'. This is used in two ways: to allow extra functionality and to support easier interaction. Whether the former really improves usability is debatable – unused functionality is a good marketing point, but is of no benefit to the user. The ease of use of a system is often determined by a host of small features, such as the

appropriate choice of default options. These features make the interface seem 'simple', but make the program very complex . . . and large. Certainly the availability of elbow room, both in terms of memory and processing power, has made such features possible.

The increase in both short-term (RAM) and long-term (disks and optical storage) memory has also removed many of the arbitrary limits in systems: it is possible to edit documents of virtually unlimited size and to treat the computer (suitably backed up) as one's primary information repository.

Some whole new application areas have become possible because of advances in memory and processing. Most applications of multimedia including voice recognition and online storage and capture of video and audio, require enormous amounts of processing and/or memory. In particular, large magnetic and optical storage devices have been the key to electronic document storage whereby all paper documents are scanned and stored within a computer system. In some contexts such systems have completely replaced paper-based filing cabinets.

2.10 SUMMARY

In Sections 2.2 and 2.3, we described a range of input devices. These performed two main functions: text entry and pointing. The principal text entry device is the QWERTY keyboard, but we also discussed alternative keyboards, chord keyboards, the telephone keypad and speech input. Pointing devices included the mouse, touchpad, trackball and joystick, as well as a large array of less common alternatives including eyegaze systems.

Section 2.4 dealt mainly with the screen as a direct output device. We discussed several different technologies, in particular CRT and LCD screens and the common properties of all bitmap display devices. We considered some more recent display methods including large displays, situated displays and digital paper.

Section 2.5 looked at the devices used for manipulating and seeing virtual reality and 3D spaces. This included the dataglove, body tracking, head-mounted displays and cave environments.

In Section 2.6 we moved outside the computer entirely and looked at physical devices such as the special displays, knobs and switches of electronic appliances. We also briefly considered sound, touch and smell as outputs from computer systems and environmental and bio-sensing as inputs. These are topics that will be revisited later in the book.

Section 2.7 discussed various forms of printer and scanner. Typical office printers include ink-jet, bubble-jet and laser printers. In addition, dot-matrix and thermal printers are used in specialized equipment. We also discussed font styles and page description languages. Scanners are used to convert printed images and documents into electronic form. They are particularly valuable in desktop publishing and for electronic document storage systems.

In Section 2.8, we considered the typical capacities of computer memory, both of main RAM, likened to human short-term memory, and long-term memory stored on magnetic and optical disks. The storage capacities were compared with document sizes and video images. We saw that a typical hard disk could only hold about two minutes of moving video, but that compression techniques can increase the capacity dramatically. We also discussed storage standards – or rather the lack of them – including the ASCII character set and markup languages. The user ought to be able to access information in ways that are natural and tolerant of small slips. Techniques which can help this included multiple indices, free text databases, DWIM (do what I mean) and Soundex.

Section 2.9 showed how processing speed, whether too slow or too fast, can affect the user interface. In particular, we discussed the effects of buffering: cursor tracking and icon wars. Processing speed is limited by various factors: computation, memory access, graphics and network delays.

The lesson from this chapter is that the interface designer needs to be aware of the properties of the devices with which a system is built. This includes not only input and output devices, but all the factors that influence the behavior of the interface, since all of these influence the nature and style of the interaction.

EXERCISES

2.1 Individually or in a group find as many different examples as you can of physical controls and displays.

(a) List them.
(b) Try to group them, or classify them.
(c) Discuss whether you believe the control or display is suitable for its purpose (Section 3.9.3 may also help).

Exercises 2.2 and 2.3 involve you examining a range of input and output devices in order to understand how they influence interaction.

2.2 A typical computer system comprises a QWERTY keyboard, a mouse and a color screen. There is usually some form of loudspeaker as well. You should know how the keyboard, mouse and screen work – if not, read up on it.

What sort of input does the keyboard support? What sort of input does the mouse support? Are these adequate for all possible applications? If not, to which areas are they most suited? Do these areas map well onto the typical requirements for users of computer systems?

If you were designing a keyboard for a modern computer, and you wanted to produce a faster, easier-to-use layout, what information would you need to know and how would that influence the design?

2.3 Pick a couple of computer input devices that you are aware of (joystick, light pen, touchscreen, trackball, eyegaze, dataglove, etc.) and note down how each has different attributes that support certain forms of interaction. You ought to know a little about all of these devices – if you don't, research them.

2.4 What is the myth of the infinitely fast machine?

2.5 Pick one of the following scenarios, and choose a suitable combination of input and output devices to best support the intended interaction. It may help to identify typical users or classes of user, and identify how the devices chosen support these people in their tasks. Explain the major problems that the input and output devices solve.

(a) *Environmental database*
A computer database is under development that will hold environmental information. This ranges from meteorological measurements through fish catches to descriptions of pollution, and will include topographical details and sketches and photographs. The data has to be accessed only by experts, but they want to be able to describe and retrieve any piece of data within a few seconds.

(b) *Word processor for blind people*
A word processor for blind users is needed, which can also be operated by sighted people. It has to support the standard set of word-processing tasks.

2.6 Describe Fitts' law (see Chapter 1). How does Fitts' law change for different physical selection devices, such as a three-button mouse, a touchpad, or a pen/stylus? (You'll need to do some research for this.)

RECOMMENDED READING

W. Buxton, There's more to interaction than meets the eye: some issues in manual input. In R. Baecker and W. Buxton, editors, *Readings in Human–Computer Interaction: A Multidisciplinary Approach*, Morgan Kaufmann, 1987.

D. J. Mayhew, *Principles and Guidelines in Software User Interface Design*, Chapter 12, Prentice Hall, 1992.
A look at input and output devices, complete with guidelines for using different devices.

A. Dix, Network-based interaction. In J. Jacko and A. Sears, editors, *Human–Computer Interaction Handbook*, Chapter 16, pp. 331–57, Lawrence Erlbaum, 2003.
Includes different kinds of network application and the effects of networks on interaction including rich media. Also look at all of Part II of Jacko and Sears, which includes chapters on input technology and on haptic interfaces.

3

The interaction

OVERVIEW

- Interaction models help us to understand what is going on in the interaction between user and system. They address the translations between what the user wants and what the system does.
- Ergonomics looks at the physical characteristics of the interaction and how these influence its effectiveness.
- The dialog between user and system is influenced by the style of the interface.
- The interaction takes place within a social and organizational context that affects both user and system.

3.1 INTRODUCTION

In the previous two chapters we have looked at the human and the computer respectively. However, in the context of this book, we are not concerned with them in isolation. We are interested in how the human user uses the computer as a tool to perform, simplify or support a task. In order to do this the user must communicate his requirements to the computer.

There are a number of ways in which the user can communicate with the system. At one extreme is batch input, in which the user provides all the information to the computer at once and leaves the machine to perform the task. This approach does involve an interaction between the user and computer but does not support many tasks well. At the other extreme are highly interactive input devices and paradigms, such as *direct manipulation* (see Chapter 4) and the applications of *virtual reality* (Chapter 20). Here the user is constantly providing instruction and receiving feedback. These are the types of interactive system we are considering.

In this chapter, we consider the communication between user and system: the *interaction*. We will look at some models of interaction that enable us to identify and evaluate components of the interaction, and at the physical, social and organizational issues that provide the context for it. We will also survey some of the different styles of interaction that are used and consider how well they support the user.

3.2 MODELS OF INTERACTION

In previous chapters we have seen the usefulness of models to help us to understand complex behavior and complex systems. Interaction involves at least two participants: the user and the system. Both are complex, as we have seen, and are very different from each other in the way that they communicate and view the domain and the task. The interface must therefore effectively translate between them to allow the interaction to be successful. This translation can fail at a number of points and for a number of reasons. The use of models of interaction can help us to understand exactly what is going on in the interaction and identify the likely root of difficulties. They also provide us with a framework to compare different interaction styles and to consider interaction problems.

We begin by considering the most influential model of interaction, Norman's *execution–evaluation cycle*; then we look at another model which extends the ideas of Norman's cycle. Both of these models describe the interaction in terms of the goals and actions of the user. We will therefore briefly discuss the terminology used and the assumptions inherent in the models, before describing the models themselves.

3.2.1 The terms of interaction

Traditionally, the purpose of an interactive system is to aid a user in accomplishing *goals* from some application *domain*. (Later in this book we will look at alternative interactions but this model holds for many work-oriented applications.) A domain defines an area of expertise and knowledge in some real-world activity. Some examples of domains are graphic design, authoring and process control in a factory. A domain consists of concepts that highlight its important aspects. In a graphic design domain, some of the important concepts are geometric shapes, a drawing surface and a drawing utensil. *Tasks* are operations to manipulate the concepts of a domain. A *goal* is the desired output from a performed task. For example, one task within the graphic design domain is the construction of a specific geometric shape with particular attributes on the drawing surface. A related goal would be to produce a solid red triangle centered on the canvas. An *intention* is a specific action required to meet the goal.

Task analysis involves the identification of the problem space (which we discussed in Chapter 1) for the user of an interactive system in terms of the domain, goals, intentions and tasks. We can use our knowledge of tasks and goals to assess the interactive system that is designed to support them. We discuss task analysis in detail in Chapter 15. The concepts used in the design of the system and the description of the user are separate, and so we can refer to them as distinct components, called the *System* and the *User*, respectively. The *System* and *User* are each described by means of a language that can express concepts relevant in the domain of the application. The *System*'s language we will refer to as the *core language* and the *User*'s language we will refer to as the *task language*. The core language describes computational attributes of the domain relevant to the *System* state, whereas the task language describes psychological attributes of the domain relevant to the *User* state.

The system is assumed to be some computerized application, in the context of this book, but the models apply equally to non-computer applications. It is also a common assumption that by distinguishing between user and system we are restricted to single-user applications. This is not the case. However, the emphasis is on the view of the interaction from a single user's perspective. From this point of view, other users, such as those in a multi-party conferencing system, form part of the system.

3.2.2 The execution–evaluation cycle

Norman's model of interaction is perhaps the most influential in Human–Computer Interaction, possibly because of its closeness to our intuitive understanding of the interaction between human user and computer [265]. The user formulates a plan of action, which is then executed at the computer interface. When the plan, or part of the plan, has been executed, the user observes the computer interface to evaluate the result of the executed plan, and to determine further actions.

The interactive cycle can be divided into two major phases: execution and evaluation. These can then be subdivided into further stages, seven in all. The stages in Norman's model of interaction are as follows:

1. Establishing the goal.
2. Forming the intention.
3. Specifying the action sequence.
4. Executing the action.
5. Perceiving the system state.
6. Interpreting the system state.
7. Evaluating the system state with respect to the goals and intentions.

Each stage is, of course, an activity of the user. First the user forms a goal. This is the user's notion of what needs to be done and is framed in terms of the domain, in the task language. It is liable to be imprecise and therefore needs to be translated into the more specific intention, and the actual actions that will reach the goal, before it can be executed by the user. The user perceives the new state of the system, after execution of the action sequence, and interprets it in terms of his expectations. If the system state reflects the user's goal then the computer has done what he wanted and the interaction has been successful; otherwise the user must formulate a new goal and repeat the cycle.

Norman uses a simple example of switching on a light to illustrate this cycle. Imagine you are sitting reading as evening falls. You decide you need more light; that is you establish the goal to get more light. From there you form an intention to switch on the desk lamp, and you specify the actions required, to reach over and press the lamp switch. If someone else is closer the intention may be different – you may ask them to switch on the light for you. Your goal is the same but the intention and actions are different. When you have executed the action you perceive the result, either the light is on or it isn't and you interpret this, based on your knowledge of the world. For example, if the light does not come on you may interpret this as indicating the bulb has blown or the lamp is not plugged into the mains, and you will formulate new goals to deal with this. If the light does come on, you will evaluate the new state according to the original goals – is there now enough light? If so, the cycle is complete. If not, you may formulate a new intention to switch on the main ceiling light as well.

Norman uses this model of interaction to demonstrate why some interfaces cause problems to their users. He describes these in terms of the *gulfs of execution* and the *gulfs of evaluation*. As we noted earlier, the user and the system do not use the same terms to describe the domain and goals – remember that we called the language of the system the *core language* and the language of the user the *task language*. The gulf of execution is the difference between the user's formulation of the actions to reach the goal and the actions allowed by the system. If the actions allowed by the system correspond to those intended by the user, the interaction will be effective. The interface should therefore aim to reduce this gulf.

The gulf of evaluation is the distance between the physical presentation of the system state and the expectation of the user. If the user can readily evaluate the presentation in terms of his goal, the gulf of evaluation is small. The more effort that is required on the part of the user to interpret the presentation, the less effective the interaction.

Human error – slips and mistakes

Human errors are often classified into *slips* and *mistakes*. We can distinguish these using Norman's gulf of execution.

If you understand a system well you may know exactly what to do to satisfy your goals – you have formulated the correct action. However, perhaps you mistype or you accidentally press the mouse button at the wrong time. These are called *slips*; you have formulated the right action, but fail to execute that action correctly.

However, if you don't know the system well you may not even formulate the right goal. For example, you may think that the magnifying glass icon is the 'find' function, but in fact it is to magnify the text. This is called a *mistake*.

If we discover that an interface is leading to errors it is important to understand whether they are slips or mistakes. Slips may be corrected by, for instance, better screen design, perhaps putting more space between buttons. However, mistakes need users to have a better understanding of the systems, so will require far more radical redesign or improved training, perhaps a totally different metaphor for use.

Norman's model is a useful means of understanding the interaction, in a way that is clear and intuitive. It allows other, more detailed, empirical and analytic work to be placed within a common framework. However, it only considers the system as far as the interface. It concentrates wholly on the user's view of the interaction. It does not attempt to deal with the system's communication through the interface. An extension of Norman's model, proposed by Abowd and Beale, addresses this problem [3]. This is described in the next section.

3.2.3 The interaction framework

The interaction framework attempts a more realistic description of interaction by including the system explicitly, and breaks it into four main components, as shown in Figure 3.1. The nodes represent the four major components in an interactive system – the *System*, the *User*, the *Input* and the *Output*. Each component has its own language. In addition to the *User*'s task language and the *System*'s core language, which we have already introduced, there are languages for both the *Input* and *Output* components. *Input* and *Output* together form the *Interface*.

As the interface sits between the *User* and the *System*, there are four steps in the interactive cycle, each corresponding to a translation from one component to another, as shown by the labeled arcs in Figure 3.2. The *User* begins the interactive cycle with the formulation of a goal and a task to achieve that goal. The only way the user can manipulate the machine is through the *Input*, and so the task must be articulated within the input language. The input language is translated into the core

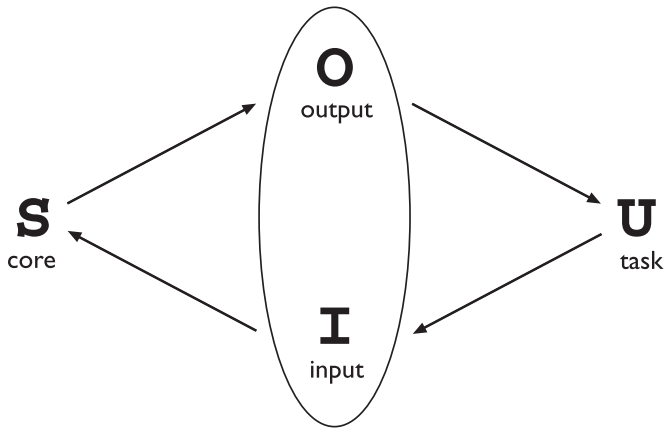


Figure 3.1 The general interaction framework

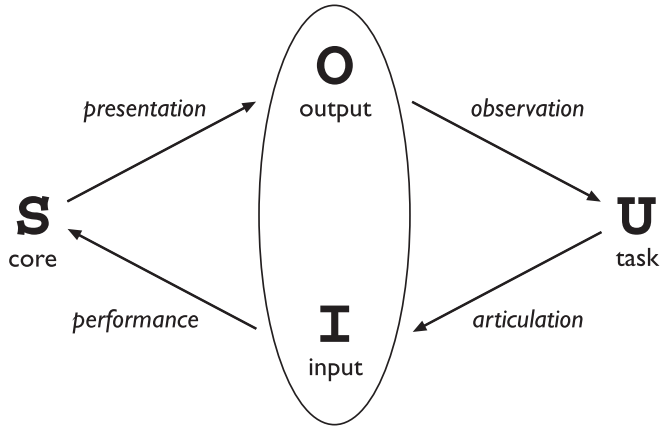


Figure 3.2 Translations between components

language as operations to be performed by the *System*. The *System* then transforms itself as described by the operations; the execution phase of the cycle is complete and the evaluation phase now begins. The *System* is in a new state, which must now be communicated to the *User*. The current values of system attributes are rendered as concepts or features of the *Output*. It is then up to the *User* to observe the *Output* and assess the results of the interaction relative to the original goal, ending the evaluation phase and, hence, the interactive cycle. There are four main translations involved in the interaction: articulation, performance, presentation and observation.

The *User*'s formulation of the desired task to achieve some goal needs to be *articulated* in the input language. The tasks are responses of the *User* and they need to be translated to stimuli for the *Input*. As pointed out above, this articulation is judged in terms of the coverage from tasks to input and the relative ease with which the translation can be accomplished. The task is phrased in terms of certain psychological attributes that highlight the important features of the domain for the *User*. If these psychological attributes map clearly onto the input language, then articulation of the task will be made much simpler. An example of a poor mapping, as pointed

out by Norman, is a large room with overhead lighting controlled by a bank of switches. It is often desirable to control the lighting so that only one section of the room is lit. We are then faced with the puzzle of determining which switch controls which lights. The result is usually repeated trials and frustration. This arises from the difficulty of articulating a goal (for example, 'Turn on the lights in the front of the room') in an input language that consists of a linear row of switches, which may or may not be oriented to reflect the room layout.

Conversely, an example of a good mapping is in virtual reality systems, where input devices such as datagloves are specifically geared towards easing articulation by making the user's psychological notion of gesturing an act that can be directly realized at the interface. Direct manipulation interfaces, such as those found on common desktop operating systems like the Macintosh and Windows, make the articulation of some file handling commands easier. On the other hand, some tasks, such as repetitive file renaming or launching a program whose icon is not visible, are not at all easy to articulate with such an interface.

At the next stage, the responses of the *Input* are translated to stimuli for the *System*. Of interest in assessing this translation is whether the translated input language can reach as many states of the *System* as is possible using the *System* stimuli directly. For example, the remote control units for some compact disc players do not allow the user to turn the power off on the player unit; hence the off state of the player cannot be reached using the remote control's input language. On the panel of the compact disc player, however, there is usually a button that controls the power. The ease with which this translation from *Input* to *System* takes place is of less importance because the effort is not expended by the user. However, there can be a real effort expended by the designer and programmer. In this case, the ease of the translation is viewed in terms of the cost of implementation.

Once a state transition has occurred within the *System*, the execution phase of the interaction is complete and the evaluation phase begins. The new state of the *System* must be communicated to the *User*, and this begins by translating the *System* responses to the transition into stimuli for the *Output* component. This presentation translation must preserve the relevant system attributes from the domain in the limited expressiveness of the output devices. The ability to capture the domain concepts of the *System* within the *Output* is a question of expressiveness for this translation.

For example, while writing a paper with some word-processing package, it is necessary at times to see both the immediate surrounding text where one is currently composing, say, the current paragraph, and a wider context within the whole paper that cannot be easily displayed on one screen (for example, the current chapter).

Ultimately, the user must interpret the output to evaluate what has happened. The response from the *Output* is translated to stimuli for the *User* which trigger assessment. The observation translation will address the ease and coverage of this final translation. For example, it is difficult to tell the time accurately on an unmarked analog clock, especially if it is not oriented properly. It is difficult in a command line interface to determine the result of copying and moving files in a hierarchical file system. Developing a website using a markup language like HTML would be virtually impossible without being able to preview the output through a browser.

Assessing overall interaction

The interaction framework is presented as a means to judge the overall usability of an entire interactive system. In reality, all of the analysis that is suggested by the framework is dependent on the current task (or set of tasks) in which the *User* is engaged. This is not surprising since it is only in attempting to perform a particular task within some domain that we are able to determine if the tools we use are adequate. For example, different text editors are better at different things. For a particular editing task, one can choose the text editor best suited for interaction relative to the task. The best editor, if we are forced to choose only one, is the one that best suits the tasks most frequently performed. Therefore, it is not too disappointing that we cannot extend the interaction analysis beyond the scope of a particular task.

DESIGN FOCUS

Video recorder

A simple example of programming a VCR from a remote control shows that all four translations in the interaction cycle can affect the overall interaction. Ineffective interaction is indicated by the user not being sure the VCR is set to record properly. This could be because the user has pressed the keys on the remote control unit in the wrong order; this can be classified as an articulatory problem. Or maybe the VCR is able to record on any channel but the remote control lacks the ability to select channels, indicating a coverage problem for the performance translation. It may be the case that the VCR display panel does not indicate that the program has been set, a presentation problem. Or maybe the user does not interpret the feedback properly, an observational error. Any one or more of these deficiencies would give rise to ineffective interaction.

3.3 FRAMEWORKS AND HCI

As well as providing a means of discussing the details of a particular interaction, frameworks provide a basis for discussing other issues that relate to the interaction. The ACM SIGCHI Curriculum Development Group presents a framework similar to that presented here, and uses it to place different areas that relate to HCI [9].

In Figure 3.3 these aspects are shown as they relate to the interaction framework. In particular, the field of *ergonomics* addresses issues on the user side of the interface, covering both input and output, as well as the user's immediate context. Dialog design and interface styles can be placed particularly along the input branch of the framework, addressing both articulation and performance. However, dialog is most usually associated with the computer and so is biased to that side of the framework.

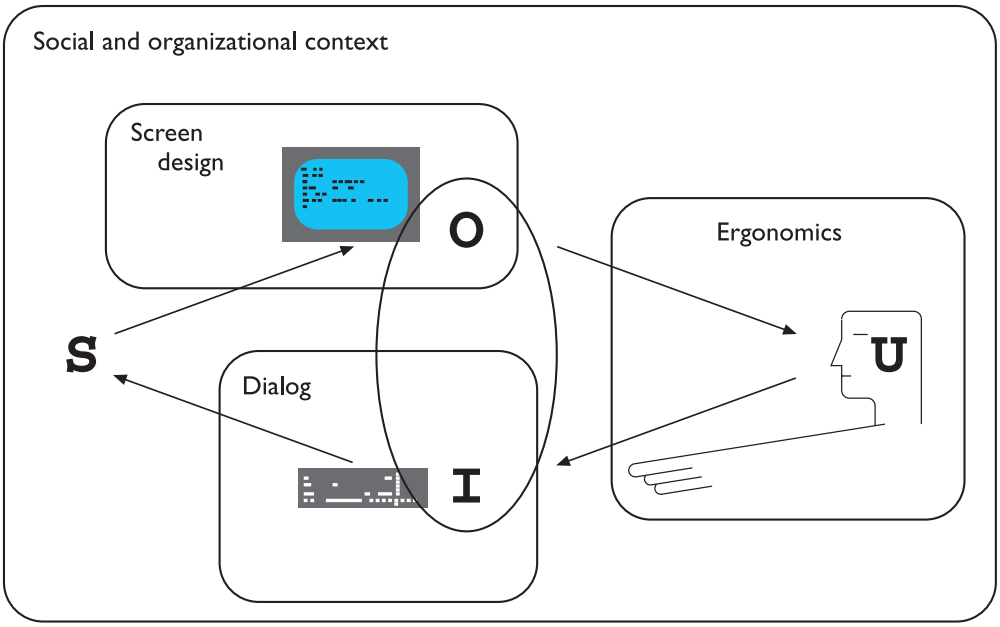


Figure 3.3 A framework for human–computer interaction. Adapted from ACM SIGCHI Curriculum Development Group [9]

Presentation and screen design relates to the output branch of the framework. The entire framework can be placed within a social and organizational context that also affects the interaction. Each of these areas has important implications for the design of interactive systems and the performance of the user. We will discuss these in brief in the following sections, with the exception of screen design which we will save until Chapter 5.

3.4 ERGONOMICS

Ergonomics (or human factors) is traditionally the study of the physical characteristics of the interaction: how the controls are designed, the physical environment in which the interaction takes place, and the layout and physical qualities of the screen. A primary focus is on user performance and how the interface enhances or detracts from this. In seeking to evaluate these aspects of the interaction, ergonomics will certainly also touch upon human psychology and system constraints. It is a large and established field, which is closely related to but distinct from HCI, and full coverage would demand a book in its own right. Here we consider a few of the issues addressed by ergonomics as an introduction to the field. We will briefly look at the arrangement of controls and displays, the physical environment, health issues and the use of color. These are by no means exhaustive and are intended only to give an

indication of the types of issues and problems addressed by ergonomics. For more information on ergonomic issues the reader is referred to the recommended reading list at the end of the chapter.

3.4.1 Arrangement of controls and displays

In Chapter 1 we considered perceptual and cognitive issues that affect the way we present information on a screen and provide control mechanisms to the user. In addition to these cognitive aspects of design, physical aspects are also important. Sets of controls and parts of the display should be grouped logically to allow rapid access by the user (more on this in Chapter 5). This may not seem so important when we are considering a single user of a spreadsheet on a PC, but it becomes vital when we turn to safety-critical applications such as plant control, aviation and air traffic control. In each of these contexts, users are under pressure and are faced with a huge range of displays and controls. Here it is crucial that the physical layout of these be appropriate. Indeed, returning to the less critical PC application, inappropriate placement of controls and displays can lead to inefficiency and frustration. For example, on one particular electronic newsreader, used by one of the authors, the command key to read articles from a newsgroup (y) is directly beside the command key to unsubscribe from a newsgroup (u) on the keyboard. This poor design frequently leads to inadvertent removal of newsgroups. Although this is recoverable it wastes time and is annoying to the user. We saw similar examples in the Introduction to this book including the MacOS X dock. We can therefore see that appropriate layout is important in all applications.

We have already touched on the importance of grouping controls together logically (and keeping opposing controls separate). The exact organization that this will suggest will depend on the domain and the application, but possible organizations include the following:

functional controls and displays are organized so that those that are functionally related are placed together;

sequential controls and displays are organized to reflect the order of their use in a typical interaction (this may be especially appropriate in domains where a particular task sequence is enforced, such as aviation);

frequency controls and displays are organized according to how frequently they are used, with the most commonly used controls being the most easily accessible.

In addition to the organization of the controls and displays in relation to each other, the entire system interface must be arranged appropriately in relation to the user's position. So, for example, the user should be able to reach all controls necessary and view all displays without excessive body movement. Critical displays should be at eye level. Lighting should be arranged to avoid glare and reflection distorting displays. Controls should be spaced to provide adequate room for the user to manoeuvre.

DESIGN FOCUS

Industrial interfaces

The interfaces to office systems have changed dramatically since the 1980s. However, some care is needed in transferring the idioms of office-based systems into the industrial domain. Office information is primarily textual and slow varying, whereas industrial interfaces may require the rapid assimilation of multiple numeric displays, each of which is varying in response to the environment. Furthermore, the environmental conditions may rule out certain interaction styles (for example, the oil-soaked mouse). Consequently, industrial interfaces raise some additional design issues rarely encountered in the office.

Glass interfaces vs. dials and knobs

The traditional machine interface consists of dials and knobs directly wired or piped to the equipment. Increasingly, some or all of the controls are replaced with a glass interface, a computer screen through which the equipment is monitored and controlled. Many of the issues are similar for the two kinds of interface, but glass interfaces do have some special advantages and problems. For a complex system, a glass interface can be both cheaper and more flexible, and it is easy to show the same information in multiple forms (Figure 3.4). For example, a data value might be given both in a precise numeric field and also in a quick to assimilate graphical form. In addition, the same information can be shown on several screens. However, the information is not located in physical space and so vital clues to context are missing – it is easy to get lost navigating complex menu systems. Also, limited display resolution often means that an electronic representation of a dial is harder to read than its physical counterpart; in some circumstances both may be necessary, as is the case on the flight deck of a modern aircraft.

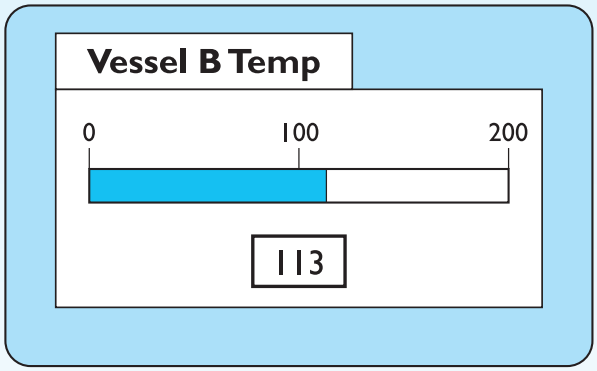


Figure 3.4 Multiple representations of the same information

Indirect manipulation

The phrase 'direct manipulation' dominates office system design (Figure 3.5). There are arguments about its meaning and appropriateness even there, but it is certainly dependent on the user being in primary control of the changes in the interface. The autonomous nature of industrial processes makes this an inappropriate model. In a direct manipulation system, the user interacts with an artificial world inside the computer (for example, the electronic desktop).

In contrast, an industrial interface is merely an intermediary between the operator and the real world. One implication of this indirectness is that the interface must provide feedback at two levels

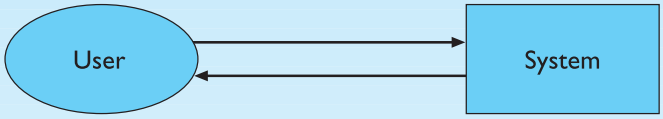


Figure 3.5 Office system – direct manipulation

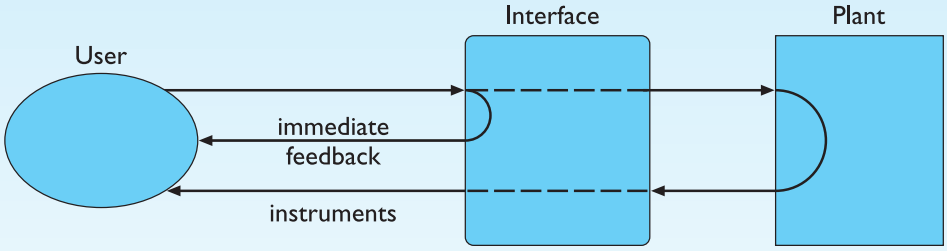


Figure 3.6 Indirect manipulation – two kinds of feedback

(Figure 3.6). At one level, the user must receive immediate feedback, generated by the interface, that keystrokes and other actions have been received. In addition, the user's actions will have some effect on the equipment controlled by the interface and adequate monitoring must be provided for this.

The indirectness also causes problems with simple monitoring tasks. Delays due to periodic sampling, slow communication and digital processing often mean that the data displayed are somewhat out of date. If the operator is not aware of these delays, diagnoses of system state may be wrong. These problems are compounded if the interface produces summary information displays. If the data comprising such a display are of different timeliness the result may be misleading.

3.4.2 The physical environment of the interaction

As well as addressing physical issues in the layout and arrangement of the machine interface, ergonomics is concerned with the design of the work environment itself. Where will the system be used? By whom will it be used? Will users be sitting, standing or moving about? Again, this will depend largely on the domain and will be more critical in specific control and operational settings than in general computer use. However, the physical environment in which the system is used may influence how well it is accepted and even the health and safety of its users. It should therefore be considered in all design.

The first consideration here is the size of the users. Obviously this is going to vary considerably. However, in any system the smallest user should be able to reach all the controls (this may include a user in a wheelchair), and the largest user should not be cramped in the environment.

In particular, all users should be comfortably able to see critical displays. For long periods of use, the user should be seated for comfort and stability. Seating should provide back support. If required to stand, the user should have room to move around in order to reach all the controls.

3.4.3 Health issues

Perhaps we do not immediately think of computer use as a hazardous activity but we should bear in mind possible consequences of our designs on the health and safety of users. Leaving aside the obvious safety risks of poorly designed safety-critical systems (aircraft crashing, nuclear plant leaks and worse), there are a number of factors that may affect the use of more general computers. Again these are factors in the physical environment that directly affect the quality of the interaction and the user's performance:

Physical position As we noted in the previous section, users should be able to reach all controls comfortably and see all displays. Users should not be expected to stand for long periods and, if sitting, should be provided with back support. If a particular position for a part of the body is to be adopted for long periods (for example, in typing) support should be provided to allow rest.

Temperature Although most users can adapt to slight changes in temperature without adverse effect, extremes of hot or cold will affect performance and, in excessive cases, health. Experimental studies show that performance deteriorates at high or low temperatures, with users being unable to concentrate efficiently.

Lighting The lighting level will again depend on the work environment. However, adequate lighting should be provided to allow users to see the computer screen without discomfort or eyestrain. The light source should also be positioned to avoid glare affecting the display.

Noise Excessive noise can be harmful to health, causing the user pain, and in acute cases, loss of hearing. Noise levels should be maintained at a comfortable level in the work environment. This does not necessarily mean no noise at all. Noise can be a stimulus to users and can provide needed confirmation of system activity.

Time The time users spend using the system should also be controlled. As we saw in the previous chapter, it has been suggested that excessive use of CRT displays can be harmful to users, particularly pregnant women.

3.4.4 The use of color

In this section we have concentrated on the ergonomics of physical characteristics of systems, including the physical environment in which they are used. However, ergonomics has a close relationship to human psychology in that it is also concerned with the perceptual limitations of humans. For example, the use of color in displays is an ergonomics issue. As we saw in Chapter 1, the visual system has some limitations with regard to color, including the number of colors that are distinguishable and the relatively low blue acuity. We also saw that a relatively high proportion of the population suffers from a deficiency in color vision. Each of these psychological phenomena leads to ergonomic guidelines; some examples are discussed below.

Colors used in the display should be as distinct as possible and the distinction should not be affected by changes in contrast. Blue should not be used to display critical information. If color is used as an indicator it should not be the only cue: additional coding information should be included.

The colors used should also correspond to common conventions and user expectations. Red, green and yellow are colors frequently associated with stop, go and standby respectively. Therefore, red may be used to indicate emergency and alarms; green, normal activity; and yellow, standby and auxiliary function. These conventions should not be violated without very good cause.

However, we should remember that color conventions are culturally determined. For example, red is associated with danger and warnings in most western cultures, but in China it symbolizes happiness and good fortune. The color of mourning is black in some cultures and white in others. Awareness of the cultural associations of color is particularly important in designing systems and websites for a global market. We will return to these issues in more detail in Chapter 10.

3.4.5 Ergonomics and HCI

Ergonomics is a huge area, which is distinct from HCI but sits alongside it. Its contribution to HCI is in determining constraints on the way we design systems and suggesting detailed and specific guidelines and standards. Ergonomic factors are in general well established and understood and are therefore used as the basis for standardizing hardware designs. This issue is discussed further in Chapter 7.

3.5 INTERACTION STYLES

Interaction can be seen as a dialog between the computer and the user. The choice of interface style can have a profound effect on the nature of this dialog. Dialog design is discussed in detail in Chapter 16. Here we introduce the most common interface styles and note the different effects these have on the interaction. There are a number of common interface styles including

- command line interface
- menus
- natural language
- question/answer and query dialog
- form-fills and spreadsheets
- WIMP
- point and click
- three-dimensional interfaces.

As the WIMP interface is the most common and complex, we will discuss each of its elements in greater detail in Section 3.6.

```
sable.soc.staffs.ac.uk> javac HelloWorldApp
javac: invalid argument: HelloWorldApp
use: javac [-g][-O][-classpath path][-d dir] file.java…
sable.soc.staffs.ac.uk> javac HelloWorldApp.java
sable.soc.staffs.ac.uk> java HelloWorldApp
Hello world!!
sable.soc.staffs.ac.uk>
```

Figure 3.7 Command line interface

3.5.1 Command line interface

The command line interface (Figure 3.7) was the first interactive dialog style to be commonly used and, in spite of the availability of menu-driven interfaces, it is still widely used. It provides a means of expressing instructions to the computer directly, using function keys, single characters, abbreviations or whole-word commands. In some systems the command line is the only way of communicating with the system, especially for remote access using *telnet*. More commonly today it is supplementary to menu-based interfaces, providing accelerated access to the system's functionality for experienced users.

Command line interfaces are powerful in that they offer direct access to system functionality (as opposed to the hierarchical nature of menus), and can be combined to apply a number of tools to the same data. They are also flexible: the command often has a number of options or parameters that will vary its behavior in some way, and it can be applied to many objects at once, making it useful for repetitive tasks. However, this flexibility and power brings with it difficulty in use and learning. Commands must be remembered, as no cue is provided in the command line to indicate which command is needed. They are therefore better for expert users than for novices. This problem can be alleviated a little by using consistent and meaningful commands and abbreviations. The commands used should be terms within the vocabulary of the user rather than the technician. Unfortunately, commands are often obscure and vary across systems, causing confusion to the user and increasing the overhead of learning.

3.5.2 Menus

In a menu-driven interface, the set of options available to the user is displayed on the screen, and selected using the mouse, or numeric or alphabetic keys. Since the options are visible they are less demanding of the user, relying on recognition rather than recall. However, menu options still need to be meaningful and logically grouped to aid recognition. Often menus are hierarchically ordered and the option required is not available at the top layer of the hierarchy. The grouping

```
PAYMENT DETAILS          P3-7

please select payment method:
  1. cash
  2. check
  3. credit card
  4. invoice

  9. abort transaction
```

Figure 3.8 Menu-driven interface

and naming of menu options then provides the only cue for the user to find the required option. Such systems either can be purely text based, with the menu options being presented as numbered choices (see Figure 3.8), or may have a graphical component in which the menu appears within a rectangular box and choices are made, perhaps by typing the initial letter of the desired selection, or by entering the associated number, or by moving around the menu with the arrow keys. This is a restricted form of a full WIMP system, described in more detail shortly.

3.5.3 Natural language

Perhaps the most attractive means of communicating with computers, at least at first glance, is by natural language. Users, unable to remember a command or lost in a hierarchy of menus, may long for the computer that is able to understand instructions expressed in everyday words! Natural language understanding, both of speech and written input, is the subject of much interest and research. Unfortunately, however, the ambiguity of natural language makes it very difficult for a machine to understand. Language is ambiguous at a number of levels. First, the syntax, or structure, of a phrase may not be clear. If we are given the sentence

```
The boy hit the dog with the stick
```

we cannot be sure whether the boy is using the stick to hit the dog or whether the dog is holding the stick when it is hit.

Even if a sentence's structure is clear, we may find ambiguity in the meaning of the words used. For example, the word 'pitch' may refer to a sports field, a throw, a waterproofing substance or even, colloquially, a territory. We often rely on the context and our general knowledge to sort out these ambiguities. This information is difficult to provide to the machine. To complicate matters more, the use of pronouns and relative terms adds further ambiguity.

Given these problems, it seems unlikely that a general natural language interface will be available for some time. However, systems can be built to understand restricted subsets of a language. For a known and constrained domain, the system can be provided with sufficient information to disambiguate terms. It is important in interfaces which use natural language in this restricted form that the user is aware of the limitations of the system and does not expect too much understanding.

The use of natural language in restricted domains is relatively successful, but it is debatable whether this can really be called natural language. The user still has to learn which phrases the computer understands and may become frustrated if too much is expected. However, it is also not clear how useful a general natural language interface would be. Language is by nature vague and imprecise: this gives it its flexibility and allows creativity in expression. Computers, on the other hand, require precise instructions. Given a free rein, would we be able to describe our requirements precisely enough to guarantee a particular response? And, if we could, would the language we used turn out to be a restricted subset of natural language anyway?

3.5.4 Question/answer and query dialog

Question and answer dialog is a simple mechanism for providing input to an application in a specific domain. The user is asked a series of questions (mainly with yes/no responses, multiple choice, or codes) and so is led through the interaction step by step. An example of this would be web questionnaires.

These interfaces are easy to learn and use, but are limited in functionality and power. As such, they are appropriate for restricted domains (particularly information systems) and for novice or casual users.

Query languages, on the other hand, are used to construct queries to retrieve information from a database. They use natural-language-style phrases, but in fact require specific syntax, as well as knowledge of the database structure. Queries usually require the user to specify an attribute or attributes for which to search the database, as well as the attributes of interest to be displayed. This is straightforward where there is a single attribute, but becomes complex when multiple attributes are involved, particularly if the user is interested in attribute A or attribute B, or attribute A and not attribute B, or where values of attributes are to be compared. Most query languages do not provide direct confirmation of what was requested, so that the only validation the user has is the result of the search. The effective use of query languages therefore requires some experience. A specialized example is the web search engine.

3.5.5 Form-fills and spreadsheets

Form-filling interfaces are used primarily for data entry but can also be useful in data retrieval applications. The user is presented with a display resembling a paper

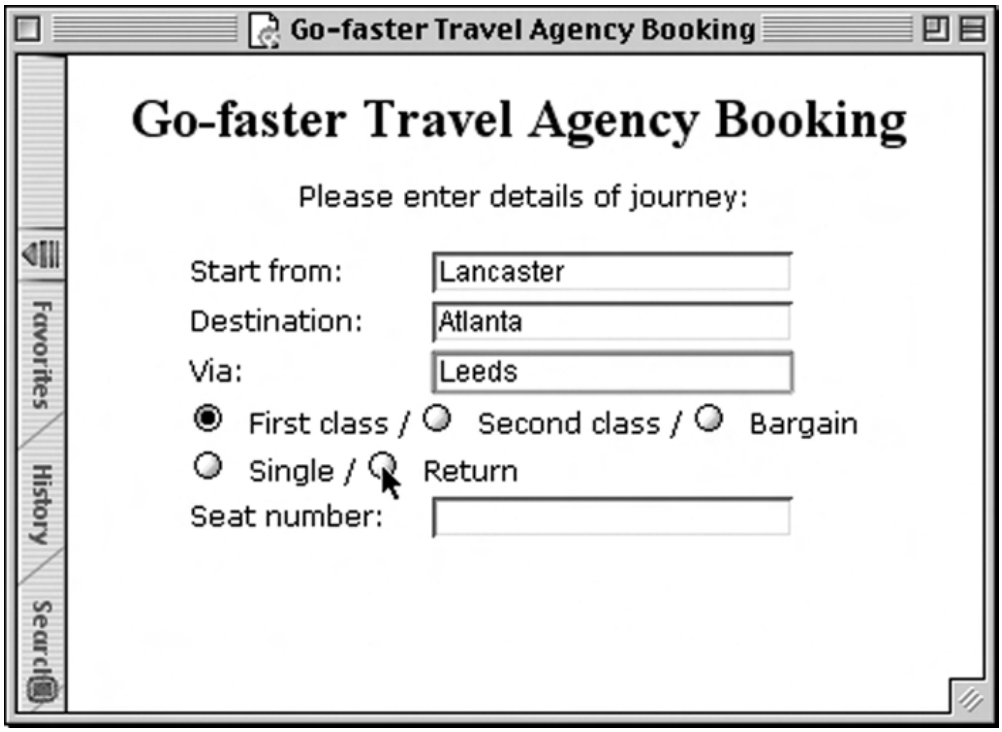


Figure 3.9 A typical form-filling interface. Screen shot frame reprinted by permission from Microsoft Corporation

form, with slots to fill in (see Figure 3.9). Often the form display is based upon an actual form with which the user is familiar, which makes the interface easier to use. The user works through the form, filling in appropriate values. The data are then entered into the application in the correct place. Most form-filling interfaces allow easy movement around the form and allow some fields to be left blank. They also require correction facilities, as users may change their minds or make a mistake about the value that belongs in each field. The dialog style is useful primarily for data entry applications and, as it is easy to learn and use, for novice users. However, assuming a design that allows flexible entry, form filling is also appropriate for expert users.

Spreadsheets are a sophisticated variation of form filling. The spreadsheet comprises a grid of cells, each of which can contain a value or a formula (see Figure 3.10). The formula can involve the values of other cells (for example, the total of all cells in this column). The user can enter and alter values and formulae in any order and the system will maintain consistency amongst the values displayed, ensuring that all formulae are obeyed. The user can therefore manipulate values to see the effects of changing different parameters. Spreadsheets are an attractive medium for interaction: the user is free to manipulate values at will and the distinction between input and output is blurred, making the interface more flexible and natural.

	Pooches Pet Emporium				
Date	*Description*	*Dog*	*Income*	*Outgoings*	*Balance*
9/2/02	Fees – Mr C. Brown	Snoopy	96.37		96.37
10/2/02	Rubber bones			36.26	60.11
10/2/02	Fees – Mrs E. R. Windsor	7 corgis	1006.45		1066.56
12/2/02	Special order: 7 red carpets			47.28	992.28
16/2/02	Fees – Master T. Tin	Snowy	32.98		1025.26
17/2/02	Beefy Bruno's Bonemeal			243.47	781.79
21/2/02	Fees – Mr F. Flintstone	Dino	21.95		803.74
21/2/02	Special order: 1 Brontosaurus bone			6.47	797.27
28/2/02	Wages – Mr S. H. Ovelit			489.46	307.81

Figure 3.10 A typical spreadsheet

3.5.6 The WIMP interface

Currently many common environments for interactive computing are examples of the *WIMP* interface style, often simply called windowing systems. WIMP stands for windows, icons, menus and pointers (sometimes windows, icons, mice and pull-down menus), and is the default interface style for the majority of interactive computer systems in use today, especially in the PC and desktop workstation arena. Examples of WIMP interfaces include Microsoft Windows for IBM PC compatibles, MacOS for Apple Macintosh compatibles and various X Windows-based systems for UNIX.

Mixing styles

The UNIX windowing environments are interesting as the contents of many of the windows are often themselves simply command line or character-based programs (see Figure 3.11). In fact, this mixing of interface styles in the same system is quite common, especially where older *legacy systems* are used at the same time as more modern applications. It can be a problem if users attempt to use commands and methods suitable for one environment in another. On the Apple Macintosh, HyperCard uses a point-and-click style. However, HyperCard stack buttons look very like Macintosh folders. If you double click on them, as you would to open a folder, your two mouse clicks are treated as separate actions. The first click opens the stack (as you wanted), but the second is then interpreted in the context of the newly opened stack, behaving in an apparently arbitrary fashion! This is an example of the importance of *consistency* in the interface, an issue we shall return to in Chapter 7.

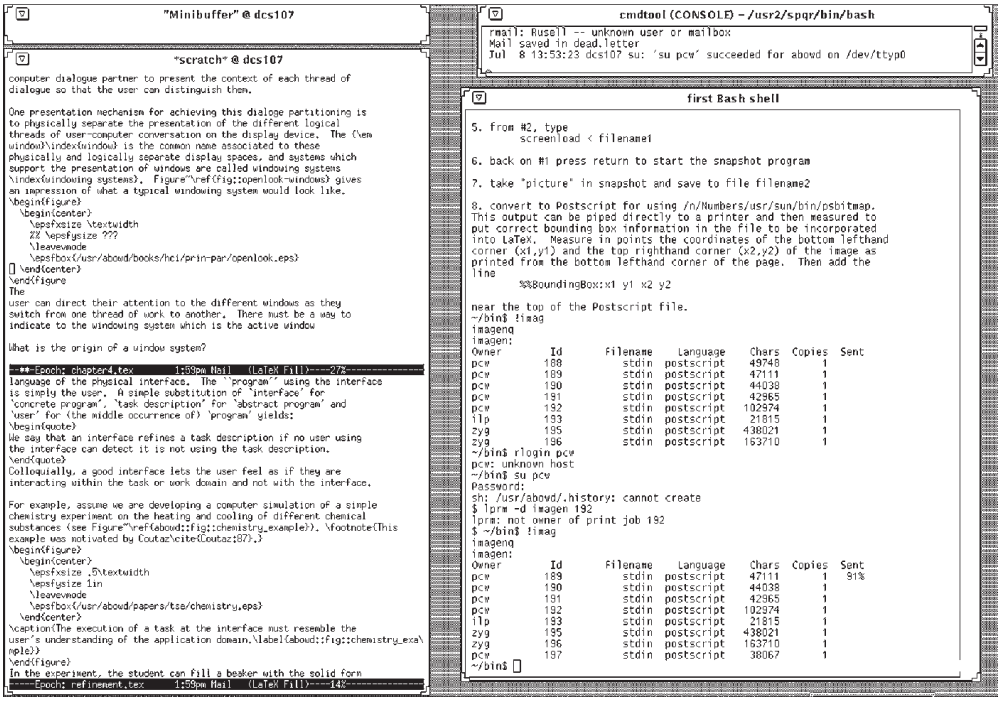


Figure 3.11 A typical UNIX windowing system – the OpenLook system. Source: Sun Microsystems, Inc.

3.5.7 Point-and-click interfaces

In most multimedia systems and in web browsers, virtually all actions take only a single click of the mouse button. You may point at a city on a map and when you click a window opens, showing you tourist information about the city. You may point at a word in some text and when you click you see a definition of the word. You may point at a recognizable iconic button and when you click some action is performed.

This point-and-click interface style is obviously closely related to the WIMP style. It clearly overlaps in the use of buttons, but may also include other WIMP elements. However, the philosophy is simpler and more closely tied to ideas of *hypertext*. In addition, the point-and-click style is not tied to mouse-based interfaces, and is also extensively used in touchscreen information systems. In this case, it is often combined with a menu-driven interface.

The point-and-click style has been popularized by world wide web pages, which incorporate all the above types of point-and-click navigation: highlighted words, maps and iconic buttons.

3.5.8 Three-dimensional interfaces

There is an increasing use of three-dimensional effects in user interfaces. The most obvious example is virtual reality, but VR is only part of a range of 3D techniques available to the interface designer.

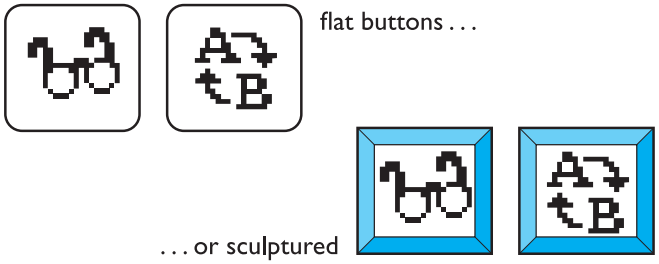


Figure 3.12 Buttons in 3D say 'press me'

The simplest technique is where ordinary WIMP elements, buttons, scroll bars, etc., are given a 3D appearance using shading, giving the appearance of being sculpted out of stone. By unstated convention, such interfaces have a light source at their top right. Where used judiciously, the raised areas are easily identifiable and can be used to highlight active areas (Figure 3.12). Unfortunately, some interfaces make indiscriminate use of sculptural effects, on every text area, border and menu, so all sense of differentiation is lost.

A more complex technique uses interfaces with 3D workspaces. The objects displayed in such systems are usually flat, but are displayed in perspective when at an angle to the viewer and shrink when they are 'further away'. Figure 3.13 shows one such system, WebBook [57]. Notice how size, light and occlusion provide a sense of

Figure 3.13 WebBook – using 3D to make more space (Card S.K., Robertson G.G. and York W. (1996). The WebBook and the Web Forager: An Information workspace for the World-Wide Web. *CHI96 Conference Proceedings*, 111–17. Copyright © 1996 ACM, Inc. Reprinted by permission)

distance. Notice also that as objects get further away they take up less screen space. Three-dimensional workspaces give you extra space, but in a more natural way than iconizing windows.

Finally, there are virtual reality and information visualization systems where the user can move about within a simulated 3D world. These are discussed in detail in Chapter 20.

These mechanisms overlap with other interaction styles, especially the use of sculptured elements in WIMP interfaces. However, there is a distinct interaction style for 3D interfaces in that they invite us to use our tacit abilities for the real world, and translate them into the electronic world. Novice users must learn that an oval area with a word or picture in it is a button to be pressed, but a 3D button says 'push me'. Further, more complete 3D environments invite one to move within the virtual environment, rather than watch as a spectator.

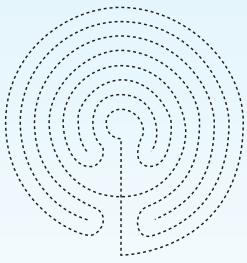
DESIGN FOCUS

Navigation in 3D and 2D

We live in a three-dimensional world. So clearly 3D interfaces are good . . . or are they? Actually, our 3D stereo vision only works well close to us and after that we rely on cruder measures such as 'this is in front of that'. We are good at moving obects around with our hands in three dimensions, rotating, turning them on their side. However, we walk around in two dimensions and do not fly. Not surprisingly, people find it hard to visualize and control movement in three dimensions.

Normally, we use gravity to give us a fixed direction in space. This is partly through the channels in the inner ear, but also largely through kinesthetic senses – feeling the weight of limbs. When we lose these senses it is easy to become disoriented and we can lose track of which direction is up: divers are trained to watch the direction their bubbles move and if buried in an avalanche you should spit and feel which direction the spittle flows.

Where humans have to navigate in three dimensions they need extra aids such as the artificial horizon in an airplane. Helicopters, where there are many degrees of freedom, are particularly difficult.



Even in the two-dimensional world of walking about we do not rely on neat Cartesian maps in our head. Instead we mostly use models of location such as 'down the road near the church' that rely on approximate topological understanding and landmarks. We also rely on properties of normal space, such as the ability to go backwards and the fact that things that are close can be reached quickly. When two-dimensional worlds are not like this, for example in a one-way traffic system or in a labyrinth, we have great difficulty [98].

When we design systems we should take into account how people navigate in the real world and use this to guide our navigation aids. For example, if we have a 3D interface or a virtual reality world we should normally show a ground plane and by default lock movement to be parallel to the ground. In information systems we can recruit our more network-based models of 2D space by giving landmarks and making it as easy to 'step back' as to go forwards (as with the web browser 'back' button).

See the book website for more about 3D vision: /e3/online/seeing-3D/

3.6 ELEMENTS OF THE WIMP INTERFACE

We have already noted the four key features of the WIMP interface that give it its name – windows, icons, pointers and menus – and we will now describe these in turn. There are also many additional interaction objects and techniques commonly used in WIMP interfaces, some designed for specific purposes and others more general. We will look at buttons, toolbars, palettes and dialog boxes. Most of these elements can be seen in Figure 3.14.

Together, these elements of the WIMP interfaces are called *widgets*, and they comprise the toolkit for interaction between user and system. In Chapter 8 we will describe windowing systems and interaction widgets more from the programmer's perspective. There we will discover that though most modern windowing systems provide the same set of basic widgets, the 'look and feel' – how widgets are physically displayed and how users can interact with them to access their functionality – of different windowing systems and toolkits can differ drastically.

3.6.1 Windows

Windows are areas of the screen that behave as if they were independent terminals in their own right. A window can usually contain text or graphics, and can be moved

Figure 3.14 Elements of the WIMP interface – Microsoft Word 5.1 on an Apple Macintosh. Screen shot reprinted by permission from Apple Computer, Inc.

or resized. More than one window can be on a screen at once, allowing separate tasks to be visible at the same time. Users can direct their attention to the different windows as they switch from one thread of work to another.

If one window overlaps the other, the back window is partially obscured, and then refreshed when exposed again. Overlapping windows can cause problems by obscuring vital information, so windows may also be *tiled*, when they adjoin but do not overlap each other. Alternatively, windows may be placed in a *cascading* fashion, where each new window is placed slightly to the left and below the previous window. In some systems this *layout policy* is fixed, in others it can be selected by the user.

Usually, windows have various things associated with them that increase their usefulness. *Scrollbars* are one such attachment, allowing the user to move the contents of the window up and down, or from side to side. This makes the window behave as if it were a real window onto a much larger world, where new information is brought into view by manipulating the scrollbars.

There is usually a title bar attached to the top of a window, identifying it to the user, and there may be special boxes in the corners of the window to aid resizing, closing, or making as large as possible. Each of these can be seen in Figure 3.15.

In addition, some systems allow windows within windows. For example, in Microsoft Office applications, such as Excel and Word, each application has its own window and then within this each document has a window. It is often possible to have different layout policies within the different application windows.

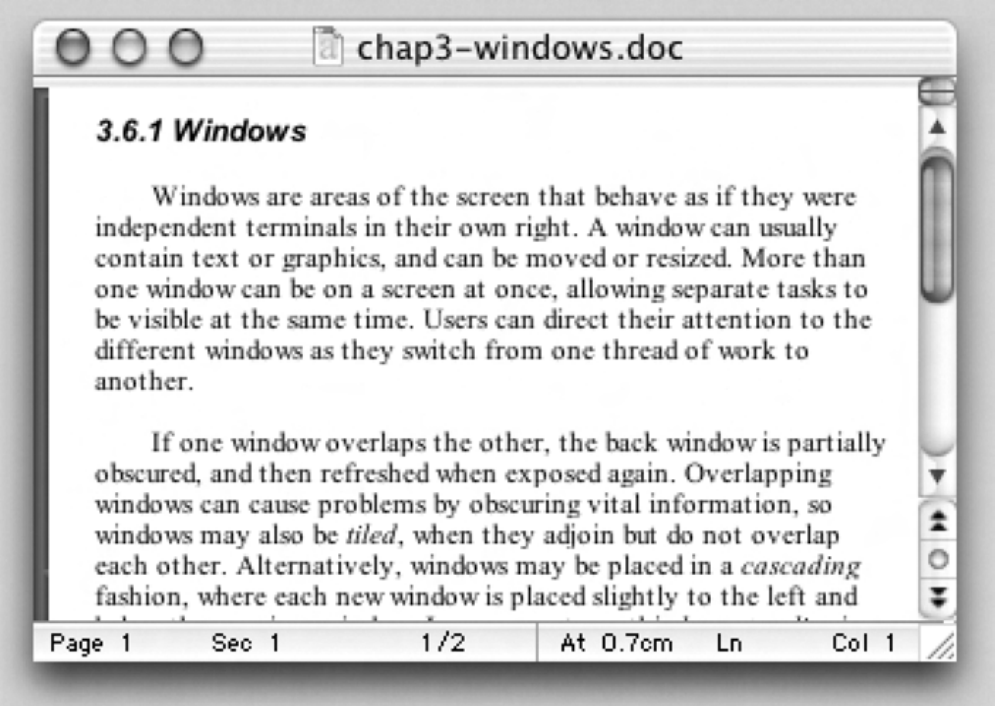


Figure 3.15 A typical window. Screen shot reprinted by permission from Apple Computer, Inc.

Figure 3.16 A variety of icons. Screen shot reprinted by permission from Apple Computer, Inc.

3.6.2 Icons

Windows can be closed and lost for ever, or they can be shrunk to some very reduced representation. A small picture is used to represent a closed window, and this representation is known as an *icon*. By allowing icons, many windows can be available on the screen at the same time, ready to be expanded to their full size by clicking on the icon. Shrinking a window to its icon is known as *iconifying* the window. When a user temporarily does not want to follow a particular thread of dialog, he can suspend that dialog by iconifying the window containing the dialog. The icon saves space on the screen and serves as a reminder to the user that he can subsequently resume the dialog by opening up the window. Figure 3.16 shows a few examples of some icons used in a typical windowing system (MacOS X).

Icons can also be used to represent other aspects of the system, such as a waste-basket for throwing unwanted files into, or various disks, programs or functions that are accessible to the user. Icons can take many forms: they can be realistic representations of the objects that they stand for, or they can be highly stylized. They can even be arbitrary symbols, but these can be difficult for users to interpret.

3.6.3 Pointers

The pointer is an important component of the WIMP interface, since the interaction style required by WIMP relies very much on pointing and selecting things such as icons. The mouse provides an input device capable of such tasks, although joysticks and trackballs are other alternatives, as we have previously seen in Chapter 2. The user is presented with a cursor on the screen that is controlled by the input device. A variety of pointer cursors are shown in Figure 3.17.

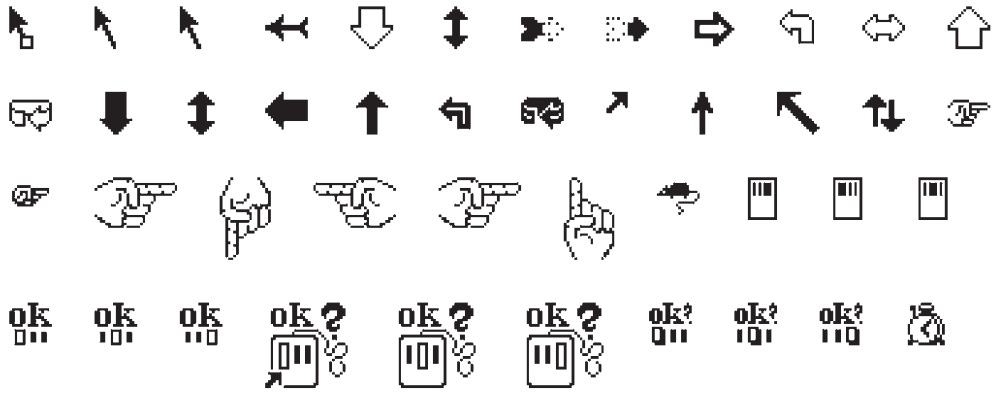


Figure 3.17 A variety of pointer cursors. Source: Sun Microsystems, Inc.

The different shapes of cursor are often used to distinguish *modes*, for example the normal pointer cursor may be an arrow, but change to cross-hairs when drawing a line. Cursors are also used to tell the user about system activity, for example a watch or hour-glass cursor may be displayed when the system is busy reading a file.

Pointer cursors are like icons, being small bitmap images, but in addition all cursors have a *hot-spot*, the location to which they point. For example, the three arrows at the start of Figure 3.17 each have a hot-spot at the top left, whereas the right-pointing hand on the second line has a hot-spot on its right. Sometimes the hot-spot is not clear from the appearance of the cursor, in which case users will find it hard to click on small targets. When designing your own cursors, make sure the image has an obvious hot-spot.

3.6.4 Menus

The last main feature of windowing systems is the *menu*, an interaction technique that is common across many non-windowing systems as well. A menu presents a choice of operations or services that can be performed by the system at a given time. In Chapter 1, we pointed out that our ability to recall information is inferior to our ability to recognize it from some visual cue. Menus provide information cues in the form of an ordered list of operations that can be scanned. This implies that the names used for the commands in the menu should be meaningful and informative.

The pointing device is used to indicate the desired option. As the pointer moves to the position of a menu item, the item is usually highlighted (by inverse video, or some similar strategy) to indicate that it is the potential candidate for selection. Selection usually requires some additional user action, such as pressing a button on the mouse that controls the pointer cursor on the screen or pressing some special key on the keyboard. Menus are inefficient when they have too many items, and so cascading menus are utilized, in which item selection opens up another menu adjacent to the item, allowing refinement of the selection. Several layers of cascading menus can be used.

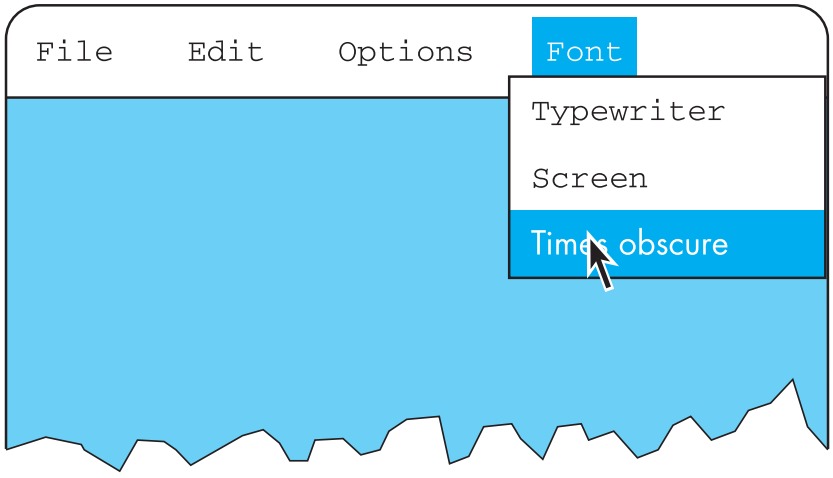


Figure 3.18 Pull-down menu

The main menu can be visible to the user all the time, as a *menu bar* and submenus can be pulled down or across from it upon request (Figure 3.18). Menu bars are often placed at the top of the screen (for example, MacOS) or at the top of each window (for example, Microsoft Windows). Alternatives include menu bars along one side of the screen, or even placed amongst the windows in the main 'desktop' area. Websites use a variety of menu bar locations, including top, bottom and either side of the screen. Alternatively, the main menu can be hidden and upon request it will pop up onto the screen. These *pop-up menus* are often used to present context-sensitive options, for example allowing one to examine properties of particular on-screen objects. In some systems they are also used to access more global actions when the mouse is depressed over the screen background.

Pull-down menus are dragged down from the title at the top of the screen, by moving the mouse pointer into the title bar area and pressing the button. Fall-down menus are similar, except that the menu automatically appears when the mouse pointer enters the title bar, without the user having to press the button. Some menus are pin-up menus, in that they can be 'pinned' to the screen, staying in place until explicitly asked to go away. Pop-up menus appear when a particular region of the screen, maybe designated by an icon, is selected, but they only stay as long as the mouse button is depressed.

Another approach to menu selection is to arrange the options in a circular fashion. The pointer appears in the center of the circle, and so there is the same distance to travel to any of the selections. This has the advantages that it is easier to select items, since they can each have a larger target area, and that the selection time for each item is the same, since the pointer is equidistant from them all. Compare this with a standard menu: remembering Fitts' law from Chapter 1, we can see that it will take longer to select items near the bottom of the menu than at the top. However, these *pie menus*, as they are known [54], take up more screen space and are therefore less common in interfaces.

Keyboard accelerators

Menus often offer *keyboard accelerators*, key combinations that have the same effect as selecting the menu item. This allows more expert users, familiar with the system, to manipulate things without moving off the keyboard, which is often faster. The accelerators are often displayed alongside the menu items so that frequent use makes them familiar. Unfortunately most systems do not allow you to use the accelerators while the menu is displayed. So, for example, the menu might say

Find	F3

However, when the user presses function key F3 nothing happens. F3 only works when the menu is *not* displayed – when the menu is there you must press 'F' instead! This is an example of an interface that is *dishonest* (see also Chapter 7).

The major problems with menus in general are deciding what items to include and how to group those items. Including too many items makes menus too long or creates too many of them, whereas grouping causes problems in that items that relate to the same topic need to come under the same heading, yet many items could be grouped under more than one heading. In pull-down menus the menu label should be chosen to reflect the function of the menu items, and items grouped within menus by function. These groupings should be consistent across applications so that the user can transfer learning to new applications. Menu items should be ordered in the menu according to importance and frequency of use, and opposite functionalities (such as 'save' and 'delete') should be kept apart to prevent accidental selection of the wrong function, with potentially disastrous consequences.

3.6.5 Buttons

Buttons are individual and isolated regions within a display that can be selected by the user to invoke specific operations. These regions are referred to as buttons because they are purposely made to resemble the push buttons you would find on a control panel. 'Pushing' the button invokes a command, the meaning of which is usually indicated by a textual label or a small icon. Buttons can also be used to toggle between two states, displaying status information such as whether the current font is italicized or not in a word processor, or selecting options on a web form. Such toggle buttons can be grouped together to allow a user to select one feature from a set of mutually exclusive options, such as the size in points of the current font. These are called *radio buttons*, since the collection functions much like the old-fashioned mechanical control buttons on car radios. If a set of options is not mutually exclusive, such as font characteristics like bold, italics and underlining, then a set of toggle buttons can be used to indicate the on/off status of the options. This type of collection of buttons is sometimes referred to as *check boxes*.

3.6.6 Toolbars

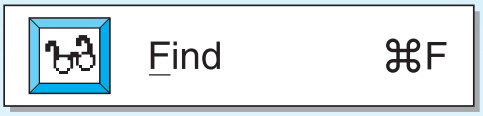
Many systems have a collection of small buttons, each with icons, placed at the top or side of the window and offering commonly used functions. The function of this *toolbar* is similar to a menu bar, but as the icons are smaller than the equivalent text more functions can be simultaneously displayed. Sometimes the content of the toolbar is fixed, but often users can *customize* it, either changing which functions are made available, or choosing which of several predefined toolbars is displayed.

DESIGN FOCUS

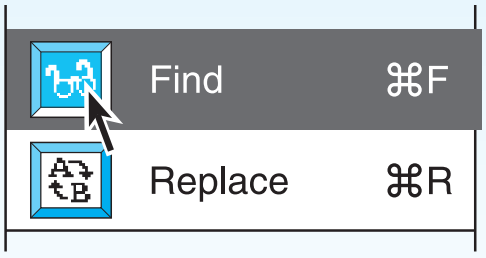
Learning toolbars

Although many applications now have toolbars, they are often underused because users simply do not know what the icons represent. Once learned the meaning is often relatively easy to remember, but most users do not want to spend time reading a manual, or even using online help to find out what each button does – they simply reach for the menu.

There is an obvious solution – put the icons on the menus in the same way that accelerator keys are written there. So in the 'Edit' menu one might find the option



Imagine now selecting this. As the mouse drags down through the menu selections, each highlights in turn. If the mouse is dragged down the extreme left, the effect will be very similar to selecting the icon from the toolbar, except that it will be incidental to selecting the menu item. In this way, the toolbar icon will be naturally learned from normal menu interaction.



Selecting the menu option = selecting the icon

This trivial fix is based on accepted and tested knowledge of learning and has been described in more detail by one of the authors elsewhere [95]. Given its simplicity, this technique should clearly be used everywhere, but until recently was rare. However, it has now been taken up in the Office 97 suite and later Microsoft Office products, so perhaps will soon become standard.

3.6.7 Palettes

In many application programs, interaction can enter one of several *modes*. The defining characteristic of modes is that the interpretation of actions, such as keystrokes or gestures with the mouse, changes as the mode changes. For example, using the standard UNIX text editor vi, keystrokes can be interpreted either as operations to insert characters in the document (insert mode) or as operations to perform file manipulation (command mode). Problems occur if the user is not aware of the current mode. Palettes are a mechanism for making the set of possible modes and the active mode visible to the user. A palette is usually a collection of icons that are reminiscent of the purpose of the various modes. An example in a drawing package would be a collection of icons to indicate the pixel color or pattern that is used to fill in objects, much like an artist's palette for paint.

Some systems allow the user to create palettes from menus or toolbars. In the case of pull-down menus, the user may be able 'tear off' the menu, turning it into a palette showing the menu items. In the case of toolbars, he may be able to drag the toolbar away from its normal position and place it anywhere on the screen. Tear-off menus are usually those that are heavily graphical anyway, for example line-style or color selection in a drawing package.

3.6.8 Dialog boxes

Dialog boxes are information windows used by the system to bring the user's attention to some important information, possibly an error or a warning used to prevent a possible error. Alternatively, they are used to invoke a subdialog between user and system for a very specific task that will normally be embedded within some larger task. For example, most interactive applications result in the user creating some file that will have to be named and stored within the filing system. When the user or system wants to save the file, a dialog box can be used to allow the user to name the file and indicate where it is to be located within the filing system. When the save subdialog is complete, the dialog box will disappear. Just as windows are used to separate the different threads of user–system dialog, so too are dialog boxes used to factor out auxiliary task threads from the main task dialog.

3.7 INTERACTIVITY

When looking at an interface, it is easy to focus on the visually distinct parts (the buttons, menus, text areas) but the dynamics, the way they react to a user's actions, are less obvious. Dialog design, discussed in Chapter 16, is focussed almost entirely on the choice and specification of appropriate sequences of actions and corresponding changes in the interface state. However, it is typically not used at a fine level of detail and deliberately ignores the 'semantic' level of an interface: for example, the validation of numeric information in a forms-based system.

It is worth remembering that *interactivity* is the defining feature of an *interactive* system. This can be seen in many areas of HCI. For example, the recognition rate for *speech recognition* is too low to allow transcription from tape, but in an airline reservation system, so long as the system can reliably recognize *yes* and *no* it can reflect back its understanding of what you said and seek confirmation. Speech-based *input* is difficult, speech-based *interaction* easier. Also, in the area of information visualization the most exciting developments are all where users can interact with a visualization in real time, changing parameters and seeing the effect.

Interactivity is also crucial in determining the 'feel' of a WIMP environment. All WIMP systems appear to have virtually the same elements: windows, icons, menus, pointers, dialog boxes, buttons, etc. However, the precise behavior of these elements differs both within a single environment and between environments. For example, we have already discussed the different behavior of pull-down and fall-down menus. These look the same, but fall-down menus are more easily invoked by accident (and not surprisingly the windowing environments that use them have largely fallen into disuse!). In fact, menus are a major difference between the MacOS and Microsoft Windows environments: in MacOS you have to keep the mouse depressed throughout menu selection; in Windows you can click on the menu bar and a pull-down menu appears and remains there until an item is selected or it is cancelled. Similarly the detailed behavior of buttons is quite complex, as we shall see in Chapter 17.

In older computer systems, the order of interaction was largely determined by the machine. You did things when the computer was ready. In WIMP environments, the user takes the initiative, with many options and often many applications simultaneously available. The exceptions to this are *pre-emptive* parts of the interface, where the system for various reasons wrests the initiative away from the user, perhaps because of a problem or because it needs information in order to continue.

The major example of this is *modal dialog boxes*. It is often the case that when a dialog box appears the application will not allow you to do anything else until the dialog box has been completed or cancelled. In some cases this may simply block the application, but you can perform tasks in other applications. In other cases you can do nothing at all until the dialog box has been completed. An especially annoying example is when the dialog box asks a question, perhaps simply for confirmation of an action, but the information you need to answer is hidden by the dialog box!

There are occasions when modal dialog boxes are necessary, for example when a major fault has been detected, or for certain kinds of instructional software. However, the general philosophy of modern systems suggests that one should minimize the use of pre-emptive elements, allowing the user maximum flexibility.

Interactivity is also critical in dealing with errors. We discussed slips and mistakes earlier in the chapter, and some ways to try to prevent these types of errors. The other way to deal with errors is to make sure that the user or the system is able to tell when errors have occurred. If users can *detect* errors then they can correct them. So, even if errors occur, the interaction as a whole succeeds. Several of the principles in Chapter 7 deal with issues that relate to this. This ability to detect and correct is important both at the small scale of button presses and keystrokes and also at the large scale. For example, if you have sent a client a letter and expect a reply, you can

put in your diary a note on the day you expect a reply. If the other person forgets to reply or the letter gets lost in the post you know to send a reminder or ring when the due day passes.

3.8 THE CONTEXT OF THE INTERACTION

We have been considering the interaction between a user and a system, and how this is affected by interface design. This interaction does not occur within a vacuum. We have already noted some of the physical factors in the environment that can directly affect the quality of the interaction. This is part of the context in which the interaction takes place. But this still assumes a single user operating a single, albeit complex, machine. In reality, users work within a wider social and organizational context. This provides the wider context for the interaction, and may influence the activity and motivation of the user. In Chapter 13, we discuss some methods that can be used to gain a fuller understanding of this context, and, in Chapter 14, we consider in more detail the issues involved when more than one user attempts to work together on a system. Here we will confine our discussion to the influence social and organizational factors may have on the user's interaction with the system. These may not be factors over which the designer has control. However, it is important to be aware of such influences to understand the user and the work domain fully.

Bank managers don't type . . .

The safe in most banks is operated by at least two keys, held by different employees of the bank. This makes it difficult for a bank robber to obtain both keys, and also protects the bank against light-fingered managers! ATMs contain a lot of cash and so need to be protected by similar measures. In one bank, which shall remain nameless, the ATM had an electronic locking device. The machine could not be opened to replenish or remove cash until a long key sequence had been entered. In order to preserve security, the bank gave half the sequence to one manager and half to another, so both managers had to be present in order to open the ATM. However, these were traditional bank managers who were not used to typing – that was a job for a secretary! So they each gave their part of the key sequence to a secretary to type in when they wanted to gain entry to the ATM. In fact, they both gave their respective parts of the key sequence to the *same* secretary. Happily the secretary was honest, but the moral is you cannot ignore social expectations and relationships when designing any sort of computer system, however simple it may be.

The presence of other people in a work environment affects the performance of the worker in any task. In the case of peers, competition increases performance, at least for known tasks. Similarly the desire to impress management and superiors improves performance on these tasks. However, when it comes to acquisition of

new skills, the presence of these groups can inhibit performance, owing to the fear of failure. Consequently, privacy is important to allow users the opportunity to experiment.

In order to perform well, users must be motivated. There are a number of possible sources of motivation, as well as those we have already mentioned, including fear, allegiance, ambition and self-satisfaction. The last of these is influenced by the user's perception of the quality of the work done, which leads to job satisfaction. If a system makes it difficult for the user to perform necessary tasks, or is frustrating to use, the user's job satisfaction, and consequently performance, will be reduced.

The user may also lose motivation if a system is introduced that does not match the actual requirements of the job to be done. Often systems are chosen and introduced by managers rather than the users themselves. In some cases the manager's perception of the job may be based upon observation of results and not on actual activity. The system introduced may therefore impose a way of working that is unsatisfactory to the users. If this happens there may be three results: the system will be rejected, the users will be resentful and unmotivated, or the user will adapt the intended interaction to his own requirements. This indicates the importance of involving actual users in the design process.

DESIGN FOCUS

Half the picture?

When systems are not designed to match the way people actually work, then users end up having to do 'work arounds'. Integrated student records systems are becoming popular in universities in the UK. They bring the benefits of integrating examination systems with enrolment and finance systems so all data can be maintained together and cross-checked. All very useful and time saving – in theory. However, one commonly used system only holds a single overall mark per module for each student, whereas many modules on UK courses have multiple elements of assessment. Knowing a student's mark on each part of the assessment is often useful to academics making decisions in examination boards as it provides a more detailed picture of performance. In many cases staff are therefore supplementing the official records system with their own unofficial spreadsheets to provide this information – making additional work for staff and increased opportunity for error.

On the other hand, the introduction of new technology may prove to be a motivation to users, particularly if it is well designed, integrated with the user's current work, and challenging. Providing adequate feedback is an important source of motivation for users. If no feedback is given during a session, the user may become bored, unmotivated or, worse, unsure of whether the actions performed have been successful. In general, an action should have an obvious effect to prevent this confusion and to allow early recovery in the case of error. Similarly, if system delays occur, feedback can be used to prevent frustration on the part of the user – the user is then aware of what is happening and is not left wondering if the system is still working.

3.9 EXPERIENCE, ENGAGEMENT AND FUN

Ask many in HCI about usability and they may use the words 'effective' and 'efficient'. Some may add 'satisfaction' as well. This view of usability seems to stem mainly from the Taylorist tradition of time and motion studies: if you can get the worker to pull the levers and turn the knobs in the right order then you can shave 10% off production costs.

However, users no longer see themselves as cogs in a machine. Increasingly, applications are focussed outside the closed work environment: on the home, leisure, entertainment, shopping. It is not sufficient that people can use a system, they must *want* to use it.

Even from a pure economic standpoint, your employees are likely to work better and more effectively if they enjoy what they are doing!

In this section we'll look at these more experiential aspects of interaction.

3.9.1 Understanding experience

Shopping is an interesting example to consider. Most internet stores allow you to buy things, but do you go shopping? Shopping is as much about going to the shops, feeling the clothes, being with friends. You can go shopping and never intend to spend money. Shopping is not about an efficient financial transaction, it is an experience.

But experience is a difficult thing to pin down; we understand the idea of a good experience, but how do we define it and even more difficult how do we design it?

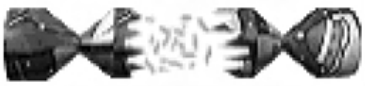
Csikszentimihalyi [82] looked at extreme experiences such as climbing a rock face in order to understand that feeling of total engagement that can sometimes happen. He calls this *flow* and it is perhaps related to what some sportspeople refer to as being 'in the zone'. This sense of flow occurs when there is a balance between anxiety and boredom. If you do something that you know you can do it is not engaging; you may do it automatically while thinking of something else, or you may simply become bored. Alternatively, if you do something completely outside your abilities you may become anxious and, if you are half way up a rock face, afraid. Flow comes when you are teetering at the edge of your abilities, stretching yourself to or a little beyond your limits.

In education there is a similar phenomenon. The *zone of proximal development* is those things that you cannot quite do yourself, but you can do with some support, whether from teachers, fellow pupils, or electronic or physical materials. Learning is at its best in this zone. Notice again this touching of limits.

Of course, this does not fully capture the sense of experience, and there is an active subfield of HCI researchers striving to make sense of this, building on the work of psychologists and philosophers on the one hand and literary analysis, film making and drama on the other.

3.9.2 Designing experience

Some of the authors were involved in the design of virtual Christmas crackers. These are rather like electronic greetings cards, but are based on crackers. For those who have not come across them, Christmas crackers are small tubes of paper between 8 and 12 inches long (20–30 cm). Inside there are a small toy, a joke or motto and a paper hat. A small strip of card is threaded through, partly coated with gunpowder. When two people at a party pull the cracker, it bursts apart with a small bang from the gunpowder and the contents spill out.



The virtual cracker does not attempt to fully replicate each aspect of the physical characteristics and process of pulling the cracker, but instead seeks to reproduce the experience. To do this the original crackers experience was deconstructed and each aspect of the experience produced in a similar, but sometimes different, way in the new media. Table 3.1 shows the aspects of the experience deconstructed and reconstructed in the virtual cracker.

For example, the cracker contents are hidden inside; no one knows what toy or joke will be inside. Similarly, when you create a virtual cracker you normally cannot see the contents until the recipient has opened it. Even the recipient initially sees a page with just an image of the cracker; it is only after the recipient has clicked on the 'open' icon that the cracker slowly opens and you get to see the joke, web toy and mask.

The mask is also worth looking at. The first potential design was to have a picture of a face with a hat on it – well, it wouldn't rank highly on excitement! The essential feature of the paper hat is that you can dress up. An iconic hat hardly does that.

Table 3.1 The crackers experience [101]

	Real cracker	Virtual cracker
Surface elements		
Design	Cheap and cheerful	Simple page/graphics
Play	Plastic toy and joke	Web toy and joke
Dressing up	Paper hat	Mask to cut out
Experienced effects		
Shared	Offered to another	Sent by email, message
Co-experience	Pulled together	Sender can't see content until opened by recipient
Excitement	Cultural connotations	Recruited expectation
Hiddenness	Contents inside	First page – no contents
Suspense	Pulling cracker	Slow . . . page change
Surprise	Bang (when it works)	WAV file (when it works)

Instead the cracker has a link to a web page with a picture of a mask that you can print, cut out and wear. Even if you don't actually print it out, the fact that you could changes the experience – it is some dressing up you just happen not to have done yet.

A full description of the virtual crackers case study is on the book website at: /e3/casestudy/crackers/

3.9.3 Physical design and engagement

In Chapter 2 we talked about physical controls. Figure 2.13 showed controllers for a microwave, washing machine and personal MiniDisc player. We saw then how certain physical interfaces were suited for different contexts: smooth plastic controls for an easy clean microwave, multi-function knob for the MiniDisc.

Designers are faced with many constraints:

Ergonomic You cannot physically push buttons if they are too small or too close.

Physical The size or nature of the device may force certain positions or styles of control, for example, a dial like the one on the washing machine would not fit on the MiniDisc controller; high-voltage switches cannot be as small as low-voltage ones.

Legal and safety Cooker controls must be far enough from the pans that you do not burn yourself, but also high enough to prevent small children turning them on.

Context and environment The microwave's controls are smooth to make them easy to clean in the kitchen.

Aesthetic The controls must look good.

Economic It must not cost too much!

These constraints are themselves often contradictory and require trade-offs to be made. For example, even within the safety category front-mounted controls are better in that they can be turned on or off without putting your hands over the pans and hot steam, but back-mounted controls are further from children's grasp. The MiniDisc player is another example; it physically needs to be small, but this means there is not room for all the controls you want given the minimum size that can be manipulated. In the case of the cooker there is no obvious best solution and so different designs favor one or the other. In the case of the MiniDisc player the end knob is multi-function. This means the knob is ergonomically big enough to turn and physically small enough to fit, but at the cost of a more complex interaction style.

To add to this list of constraints there is another that makes a major impact on the ease of use and also the ability of the user to become engaged with the device, for it to become natural to use:

Fluidity The extent to which the physical structure and manipulation of the device naturally relate to the logical functions it supports.

This is related closely to the idea of *affordances*, which we discuss in Section 5.7.2. The knob at the end of the MiniDisc controller affords turning – it is an obvious thing to do. However, this may not have mapped naturally onto the logical functions. Two of the press buttons are for cycling round the display options and for changing sound options. Imagine a design where turning the knob to clockwise cycled through the display options and turning it anti-clockwise cycled through the sound options. This would be a compact design satisfying all the ergonomic, physical and aesthetic constraints, but would not have led to as fluid an interaction. The physically opposite motions lead to logically distinct effects. However, the designers did a better job than this! The twist knob is used to move backwards and forwards through the tracks of the MiniDisc – that is, opposite physical movements produce opposite logical effects. Holding the knob out and twisting turns the volume up and down. Again, although the pull action is not a natural mapping, the twist maps very naturally onto controlling the sound level.

As well as being fluid in action, some controls portray by their physical appearance the underlying state they control. For example, the dial on the washing machine both sets the program and reflects the current stage in the washing cycle as it turns. A simple on/off switch also does this. However, it is also common to see the power on computers and hifi devices controlled by a push button – press for on, then press again for off. The button does not reflect the state at all. When the screen is on this is not a problem as the fact that there is something on the screen acts as a very immediate indicator of the state. But if the screen has a power save then you might accidentally turn the machine off thinking that you are turning it on! For this reason, this type of power button often has a light beside it to show you the power is on. A simple switch tells you that itself!

3.9.4 Managing value

If we want people to *want* to use a device or application we need to understand their personal values. Why should they want to use it? What value do they get from using it? Now when we say value here we don't mean monetary value, although that may be part of the story, but all the things that drive a person. For some people this may include being nice to colleagues, being ecologically friendly, being successful in their career. Whatever their personal values are, if we ask someone to do something or use something they are only likely to do it if the value to them exceeds the cost.

This is complicated by the fact that for many systems the costs such as purchase cost, download time of a free application, learning effort are incurred up front, whereas often the returns – faster work, enjoyment of use – are seen later. In economics, businesses use a measure called 'net present value' to calculate what a future gain is worth today; because money can be invested, £100 today is worth the same as perhaps £200 in five years' time. Future gain is discounted. For human decision making, future gains are typically discounted very highly; many of us are bad at saving for tomorrow or even keeping the best bits of our dinner until last. This means that not only must we understand people's value systems, but we must be able to offer

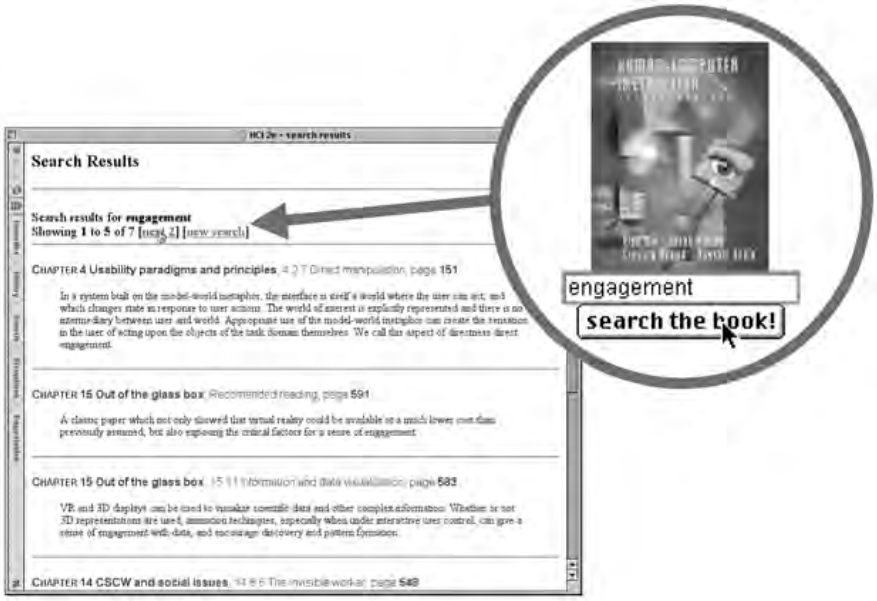


Figure 3.19 The web-based book search facility. Screen shot frame reprinted by permission from Microsoft Corporation

gains sooner as well as later, or at least produce a very good demonstration of potential future gains so that they have a *perceived* current value.

When we were preparing the website for the second edition of this book we thought very hard about how to give things that were of value to those who had the book, and also to those who hadn't. The latter is partly because we are all academics and researchers in the field and so want to contribute to the HCI community, but also of course we would like lots of people to buy the book. One option we thought of was to put the text online, which would be good for people without the book, but this would have less value to people who have the book (they might even be annoyed that those who hadn't paid should have access). The search mechanism was the result of this process (Figure 3.19). It gives value to those who have the book because it is a way of finding things. It is of value to those who don't because it acts as a sort of online encyclopedia of HCI. However, because it always gives the chapter and page number in the book it also says to those who haven't got the book: 'buy me'. See an extended case study about the design of the book search on the website at /e3/casestudy/search/

3.10 SUMMARY

In this chapter, we have looked at the interaction between human and computer, and, in particular, how we can ensure that the interaction is effective to allow the user to get the required job done. We have seen how we can use Norman's execution–evaluation model, and the interaction framework that extends it, to analyze the

interaction in terms of how easy or difficult it is for the user to express what he wants and determine whether it has been done.

We have also looked at the role of ergonomics in interface design, in analyzing the physical characteristics of the interaction, and we have discussed a number of interface styles. We have considered how each of these factors can influence the effectiveness of the interaction.

Interactivity is at the heart of all modern interfaces and is important at many levels. Interaction between user and computer does not take place in a vacuum, but is affected by numerous social and organizational factors. These may be beyond the designer's control, but awareness of them can help to limit any negative effects on the interaction.

EXERCISES

3.1 Choose two of the interface styles (described in Section 3.5) that you have experience of using. Use the interaction framework to analyze the interaction involved in using these interface styles for a database selection task. Which of the distances is greatest in each case?

3.2 Find out all you can about natural language interfaces. Are there any successful systems? For what applications are these most appropriate?

3.3 What influence does the social environment in which you work have on your interaction with the computer? What effect does the organization (commercial or academic) to which you belong have on the interaction?

3.4 (a) Group the following functions under appropriate headings, assuming that they are to form the basis for a menu-driven word-processing system – the headings you choose will become the menu titles, with the functions appearing under the appropriate one. You can choose as many or as few menu headings as you wish. You may also alter the wordings of the functions slightly if you wish.

save, save as, new, delete, open mail, send mail, quit, undo, table, glossary, preferences, character style, format paragraph, lay out document, position on page, plain text, bold text, italic text, underline, open file, close file, open copy of file, increase point size, decrease point size, change font, add footnote, cut, copy, paste, clear, repaginate, add page break, insert graphic, insert index entry, print, print preview, page setup, view page, find word, change word, go to, go back, check spelling, view index, see table of contents, count words, renumber pages, repeat edit, show alternative document, help

(b) If possible, show someone else your headings, and ask them to group the functions under your headings. Compare their groupings with yours. You should find that there are areas of great similarity, and some differences. Discuss the similarities and discrepancies.

Why do some functions always seem to be grouped together?
Why do some groups of functions always get categorized correctly?
Why are some less easy to place under the 'correct' heading?
Why is this important?

3.5 Using your function groupings from Exercise 3.4, count the number of items in your menus.

(a) What is the average?
What is the disadvantage of putting all the functions on the screen at once?
What is the problem with using lots of menu headings?
What is the problem of using very few menu headings?

Consider the following: I can group my functions either into three menus, with lots of functions in each one, or into eight menus with fewer in each. Which will be easier to use? Why?

(b) *Optional experiment*
Design an experiment to test your answers. Perform the experiment and report on your results.

3.6 Describe (in words as well as graphically) the interaction framework introduced in *Human–Computer Interaction*. Show how it can be used to explain problems in the dialog between a user and a computer.

3.7 Describe briefly four different interaction styles used to accommodate the dialog between user and computer.

3.8 The typical computer screen has a WIMP setup (what does WIMP stand for?). Most common WIMP arrangements work on the basis of a desktop metaphor, in which common actions are likened to similar actions in the real world. For example, moving a file is achieved by selecting it and dragging it into a relevant folder or filing cabinet. The advantage of using a metaphor is that the user can identify with the environment presented on the screen. Having a metaphor allows users to predict the outcome of their actions more easily.

Note that the metaphor can break down, however. What is the real-world equivalent of formatting a disk? Is there a direct analogy for the concept of 'undo'? Think of some more examples yourself.

RECOMMENDED READING

D. A. Norman, *The Psychology of Everyday Things*, Basic Books, 1988. (Republished as *The Design of Everyday Things* by Penguin, 1991.)
A classic text, which discusses psychological issues in designing everyday objects and addresses why such objects are often so difficult to use. Discusses the execution–evaluation cycle. Very readable and entertaining. See also his more recent books *Turn Signals are the Facial Expressions of Automobiles* [267], *Things That Make Us Smart* [268] and *The Invisible Computer* [269].

R. W. Bailey, *Human Performance Engineering: A Guide for System Designers*, Prentice Hall, 1982.
Detailed coverage of human factors and ergonomics issues, with plenty of examples.

G. Salvendy, *Handbook of Human Factors and Ergonomics*, John Wiley, 1997.
Comprehensive collection of contributions from experts in human factors and ergonomics.

M. Helander, editor, *Handbook of Human–Computer Interaction. Part II: User Interface Design*, North-Holland, 1988.
Comprehensive coverage of interface styles.

J. Raskin, *The Humane Interface: New Directions for Designing Interactive Systems*, Addison Wesley, 2000.
Jef Raskin was one of the central designers of the original Mac user interface. This book gives a personal, deep and practical examination of many issues of interaction and its application in user interface design.

M. Blythe, A. Monk, K. Overbeeke and P. Wright, editors, *Funology: From Usability to Enjoyment*, Kluwer, 2003.
This is an edited book with chapters covering many areas of user experience. It includes an extensive review of theory from many disciplines from psychology to literary theory and chapters giving design frameworks based on these. The theoretical and design base is grounded by many examples and case studies including a detailed analysis of virtual crackers.