

Open Research Online

The Open University's repository of research publications and other research outputs

Adapting user interfaces for visually disabled users

Thesis

How to cite:

Edwards, Alistair D. N. (1987). Adapting user interfaces for visually disabled users. PhD thesis The Open University.

For guidance on citations see [FAQs](#).

© 1987 The Author

Version: Version of Record

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online's data [policy](#) on reuse of materials please consult the policies page.

oro.open.ac.uk

DX 80409

UNRESTRICTED

Adapting user interfaces for visually disabled users.

Alistair D. N. Edwards, BSc., MSc.

This thesis is submitted in fulfilment of the requirements for
PhD. in Educational Technology, 15th July 1987.

Author's Number: HDN 66452
Date of Submission: July 1987
Date of Award: 15th July 1987

Adapting User Interfaces for Visually Disabled Users

Abstract

Recent developments in the design of human-machine interfaces have resulted in interfaces which make access to computer-based equipment more difficult for visually disabled people. The aim of this project was to explore whether it is possible to adapt such interfaces so as to make them usable by people who cannot see a screen.

The approach adopted was based upon two principles: the replacement of visual interface entities by auditory analogues and appropriately constraining the resultant interface. Two forms of sound were used to embody the auditory interface: musical tones and synthetic speech. The 'auditory screen' so produced consists of a grid of 'auditory objects', each of which is associated with a spatial location, a tone, a name and an action. In order to test the principles a word processing program was designed and implemented to demonstrate how a visual program might be adapted to be accessed through such an interface.

This program was evaluated with the assistance of a number of visually disabled testers. They were trained to use the word processor through completing a graded set of exercises. Data were obtained in a number of ways during and after the completion of the exercises. Since the auditory interface had some novel components, special evaluation methods were applied. The nature of the interaction was analyzed, using an approach based on work on keystroke models of visual interfaces. This yielded a simple model of the 'hand-ear' coordination employed, which forms a basis for prediction of user behaviour. It was also necessary to evaluate aspects of the program, such as ease-of-learning and these were investigated by interviewing the subjects. The results demonstrate that the approach is viable. The thesis also discusses a number of problems in using such an interface, most of which are related to the memory load on the user.

Acknowledgements

An enormous debt of gratitude is due to my supervisor, Tim O'Shea, for all his assistance in this project. He provided many ideas and much of the inspiration behind the research as well as invaluable guidance as to how it should be pursued. In addition I am grateful for his friendship and hospitality, not least during my stay in the United States. If my monster is big enough and hairy enough, it's thanks to him.

During the absence, on leave, of Dr O'Shea, Mark Elsom-Cook valiantly filled the breach acting as supervisor, during the vital phases of evaluation and writing up of this work. My thanks to him also.

This work would have been incomplete without the evaluation, which could not have been carried out without the assistance of a number of subjects who all gave up their time entirely voluntarily. It was agreed that they would remain anonymous. However, their contribution is most gratefully acknowledged, and I can mention - and thank - those people who recruited subjects for me. From the Royal National Institute for the Blind I thank Corrie Barrett and from the RNIB Technical College Geoff Jackman and Anne Stewart, while Karl Farrell organized volunteers from the Association for Visually Handicapped Office Workers. Also the RNIB's Map-Making Department were most generous and helpful in assisting to make tactile diagrams for use in this project.

Intermediate evaluation was carried out at the Royal National College for the Blind, Hereford, and I would like particularly to mention Ernest Bate and Jane Lilleystone.

Kathy McLaughlin performed an enormous load of proof-reading in an very short time and did it with incredible thoroughness. In addition she assisted with various experiments at different phases of the research. Thanks to her for her support and good humour.

The original inspiration behind this work was provided by Tom Vincent, who also provided valuable assistance while it was underway and took the trouble to read and comment on a draft of this thesis.

David Calderwood and Paul Blenkhorn provided very useful first-hand experience in the early days of this research.

Claire O'Malley kindly took the time and effort to read and provide useful comments on a draft of the thesis, which was dropped on her without warning.

Peter Whalley also played an invaluable role in assisting me far above and beyond the call of duty, in provision of equipment.

Jack Clegg of the University's Audio-Visual Department was extremely helpful in the making of an audio tape of a demonstration session of using the software.

The visit to the United States of America undertaken as part of this project proved to be incredibly valuable. Thanks is due to a number of people for making that so, and for being most generous with their time: Tom Moran, Stuart Card and Sara Bly of Xerox's Palo Alto Research Centre; David Ticchi of Kurzweil Applied Intelligence; Dr Bliss and his colleagues at Telesensory Incorporated; Dick Steele and Greg Goodrich at the Palo Alto Veterans' Administration Hospital; John Brabyn, Bill Gerrey and colleagues at the Smith Kettlewell Institute; Pat Damasco and his colleagues at Tufts University; Michael Rosen at Massachusetts Institute of Technology; Sharon Fleschute and Rick Blair of Apple Computers.

This research was supported by a studentship from the Science and Engineering Research Council, who also funded the visit to the United States.

Contents

Chapter 1: Introduction

1.1 Introduction to the thesis	1
1.2 The use of computers by people with disabilities	3
1.3 Introduction to the problem and proposed solution	7
1.4 A brief introduction to visual disability	10
1.5 Examples of the use of information technology	12
1.6 Alternative interfaces for sighted people	19
1.7 Chronology	19

Chapter 2: The Problem

2.1 Introduction	21
2.2 Modern interfaces	27
2.3 Some important wimp systems	36

Chapter 3: General Principles

3.1 Introduction	39
3.2 Visual-to-auditory translation	41
3.3 Constraining the interface	49
3.4 Implications of applying the design principles	51

Chapter 4: Design Principles for a Word Processor

4.1 Introduction	53
4.2 Principles	53
4.3 The design of the auditory screen	56
4.4 The design of menus	60
4.5 The design of dialogues	62
4.6 The design of alerts	63
4.7 The design of the document window	64
4.8 Summary of the design	69

Chapter 5: Implementation: Soundtrack

5.1 A case study: introduction	70
5.2 Hardware and software	70
5.3 Implementation of the screen	75
5.4 Document windows	80
5.5 Menu windows	86
5.6 Dialogues	91
5.7 Alerts	94
5.8 Diagonal alarms	94
5.9 Conclusions and review	95

Chapter 6: Evaluation methods

6.1 Introduction	96
6.2 The need for evaluation	96
6.3 Problems of evaluation	97
6.4 Design of the evaluation	103
6.5 Introduction to the subjects	105
6.6 Format of the evaluation	107
6.7 Conclusions	110

Chapter 7: Interviews

7.1 Introduction	112
7.2 Structured interviews	112
7.3 Individual subjects' responses	116
7.4 Unstructured interviews	126
7.5 Views of other researchers	128
7.6 Summary	130

Chapter 8: Timing data

8.1 Modelling the interaction	133
8.2 Relationship to models of sighted targetting	138
8.3 Measurements of location times	141
8.4 Discussion of the model	145
8.5 Errors	147
8.6 Individual data from Exercise 0.1	148
8.7 Locating smaller objects	152
8.8 Conclusions from the timing results	155

Chapter 9: Usage

9.1 Introduction to the traced exercises	159
9.2 General features	162
9.3 Individual subjects' results	167
9.4 Discussion of the traced exercises	185

Chapter 10: Evaluation outcomes

10.1 Introduction	187
10.2 Learning	187
10.3 Immediate conclusions	189
10.4 Adaptation	191
10.5 Providing a mouse reference frame	192
10.6 The text selection mechanism	194
10.7 Summary	198

Chapter 11: Conclusions

11.1 Introduction	199
11.2 Claims and achievements	199
11.3 Criticisms of the research	200
11.4 Enhancements to Soundtrack	200
11.5 Evaluation of pointing devices	209
11.6 Experiments with auditory modulations	210
11.7 Graphics programs	211
11.8 Visual programming languages	211
11.9 Integration with other senses	214
11.10 A generalized interface	215
11.11 Representations of text	219
11.12 Summary	221

References

Appendices

- A** Evaluation Session Record Sheet
- B** The Evaluation Exercises
- C** The Structured Interview Questionnaire
- D** Transcript of a sample session of using Soundtrack
- E** Useful Addresses

Table of Figures

1.1. The mechanism for specifying the section of text to be spoken on the Frank Audiodata.	14
2.1. The elements of the teletype terminal.	23
2.2. The concepts underlying display editors.	24
2.3. <i>Tiling</i> .	25
2.4. Overlapped multiple windows.	26
2.5. A typical icon representing a document file.	31
3.1. Four different possible configurations of overlapping windows.	50
4.1. Three ways of fitting a prime number (5) of objects into a grid.	56
4.2. A visual representation of an arbitrary auditory screen.	57
4.3. A screen with an active window.	59
4.4. The layout of an auditory menu.	60
4.5. An alternative, grid format for menus.	60
4.6. A single-column menu with eight entries.	61
4.7. An eight-entry menu with a two-column layout.	61
5.1. The layout of the windows within the auditory screen.	77
5.2. The Document window.	80
5.3. Use of the Step controls.	83
5.4. Use of the Jump controls.	84
5.5. The Speech menu.	87
5.6. The File menu.	88
5.7. The Edit menu.	89
5.8. The Interface menu.	90
5.9. The Select file dialogue.	91
5.10. The Save file dialogue.	92

5.11. The Get file name dialogue.	93
5.12. The Get target string dialogue.	93
7.1. A diagonal mouse movement, resolved into vertical and horizontal components.	129
8.1. A grid, illustrating the method used to measure distances.	134
8.2. The method used to locate objects on the auditory screen.	136
8.3. The layout of the auditory screen.	142
8.4. Plot of time to locate auditory windows against distance to the window.	144
8.5. Time taken to move across each of the windows in movements in which $d \geq 2$.	146
8.6. Individual times for locating windows.	149
8.7. Number of errors made by subject S3 in each session.	151
8.8. The Document window.	153
8.9. Mean time for subjects S1 and S2 to locate objects within a Document window.	153
11.1. A visual menu which has control-key equivalents for the entries New and Open .	203
11.2. A typical Ark screen, showing a physics simulation.	214

Chapter 1

Introduction

1.1 Introduction to the thesis

A major aim in the development of information technology has been to make computers easier to use. Much of this effort has been concentrated in the area of design of the interface between the computer and its user - the *human-machine* interface. Although developments have led to different styles of interaction, one aspect has remained constant: output from computers has always been almost entirely visual. Programs have been made easier to use with the advent of displays which are very complex visually. Any form of visual interface is obviously difficult for a person with a visual impairment to use. Nevertheless, simple visual interfaces *have* been adapted so that they can be used by people who are visually disabled, even to the point of being blind. Such adaptations become more difficult as interfaces become more complex visually, so that as interfaces have evolved which do make computers easier to use for sighted people, they have become less accessible to users who are impaired visually. The work described in this thesis addressed that more difficult problem.

Modern information technology is often presented as a major contribution to the emancipation of disabled people, giving new opportunities for communication and independence. In many specific cases this is true and is a very significant development. Yet at the same time information technology can exclude people, exacerbating the handicapping effect of their disability. The increased dependence on visual presentation of information mentioned above may become an example of this kind of negative effect.

The dominance of visual presentation of information is more to do with technological factors than human ones. In most situations people are accustomed to communicating complex information aurally. So it is that the work described herein may have a wider significance than just in the area of rehabilitation of visually disabled people. Auditory presentation of information by computers may be preferable to visual in many situations.

The thesis describes the formulation of an approach to solving the problem outlined above. The principles evolved were tested by being incorporated in a piece of software. That was a word processor, which was named *Soundtrack*. Soundtrack was evaluated by a number of subjects who had visual disabilities. Conclusions were drawn from that evaluation, both with reference to future work on the design of interfaces for visually disabled users and in a wider context.

This chapter sets out the background. It briefly describes the nature of visual disabilities, current devices which alleviate the problems of visually disabled computer users and why the approaches embodied by those devices are inadequate for modern interfaces. It is a basic premise that modern interfaces are very visual. It is therefore difficult to describe them verbally. Anyone who has used such an interface will be aware of how visual they are, but for the benefit of any readers who are not familiar with them, Chapter 2 gives a description.

Chapter 3 describes the foundations of the work carried out in this project, the principles on which it is based. The opportunity is taken to discuss those principles in general, before Chapter 4, which is more specific, in describing how they were applied in the design of a piece of software. The design is to some extent idealized, it describes how the software would appear but for the constraints of real computers and users who are human. So, separated in Chapter 5 is a description of the software which was actually implemented, *based on that design*.

That implementation was carried out and the resulting product was required to be evaluated. Chapters 6 to 9 cover the evaluation. Chapter 6 covers both the more philosophical

questions of what the evaluation should be, particularly within the context of a product designed as an aid for disabled people, as well as the practical problems of setting up such an exercise. The evaluation method which was devised is then described, with reference to other related work.

Chapter 7 presents the reactions of the test subjects to using the software, as revealed in interviews. One aspect of the evaluation concerned the nature of the interaction between users and the auditory interface to the word processor. This was investigated partly by building a model of the interaction, based on timings made of subjects carrying out word processing exercises using Soundtrack. These results are reported in Chapter 8, and have a general relevance to human-computer interfaces, and not necessarily interfaces specifically designed for disabled users. On the other hand, Chapter 9 is concerned more specifically with the performance of realistic editing tasks by visually disabled subjects. This is effectively a higher-level view of the interaction. Chapter 10 summarizes the conclusions from all aspects of the evaluation. Finally, Chapter 11 sets out the conclusions which can be drawn from this project as a whole.

There are three appendices which contain material used in the evaluation. A fourth appendix is a transcript of a typical word processing exercise. The final appendix contains relevant names and addresses. Most of these relate to specific products and organizations mentioned in the text.

1.2 The use of computers by people with disabilities

It is accepted that people have wide-ranging levels of achievement and potential in their mental and physical faculties. By definition, the level at which the majority function is considered to be average - or normal. Of course, many people are exceptional; they have an ability which is not average. Often such exceptions are evident only in one or two faculties

in an individual. So, for instance, a successful athlete is abnormal in her¹ physical powers (i.e abnormally well endowed). A painter has better artistic abilities - but probably could not run any faster than average. Indeed, the painter may be a slower runner than her contemporaries, but since running is not a vital skill in most people's lives, the painter's lack is not a serious one. However, if a person has an impairment which seriously affects their capacity to take a full part in society, they are said to be disabled.² (See Sutherland, 1981, for a much more full discussion of the definition of disability).

The line which divides those who are said to be disabled from those who are not is a blurred one. Someone who cannot hear at all, due to nerve damage, is clearly at a disadvantage in an aural world, and can be said to be disabled. However, among the population who can hear, the sense of hearing is not at all uniform. In any group of people, some will hear better than others. A few will have hearing significantly less acute than the majority, but will still be able to lead a similar life. Yet at some point, a person's hearing - while it does exist - is so dull that they are said to be disabled. In fact the great majority of people who have a hearing disability are not deaf, but *partially hearing*. This lack of clear definition is a feature common to all forms of disability. It would be very convenient if it was possible to divide the population into *the hearing* and *the deaf* or *the normal* and *the disabled* (and indeed some people try to do that), but to so do is to commit a gross and damaging over-simplification.

According to the loose definition of disability given above, people who are disabled need assistance in order to carry on their daily lives. Most people who have a disability require assistance in some form of special education. Computers can be used as part of that education. In particular, they can be used to deliver *Computer-Aided Learning*. There are a

¹ Throughout this thesis the reader can assume that feminine pronouns imply male or female people when used to refer to anonymous examples.

² The terms *disability* and *handicap* are often confused. Handicap refers to the effect that a person's disability has on her life.

number of arguments proposed as to why people with particular disabilities can benefit from this form of learning (see Edwards, 1984).

Disabled people may use special devices and software as communication aids and in Computer-Aided Learning lessons, but also they need to use computers for the same purposes as everyone else, in work, education and leisure. For example, many jobs entail the use of word processors, and access to databases is not only vital in some work, but it is also an option in many homes, through teletext systems. There may be a problem, however, of access for disabled people. Adaptations of one form or another may be needed to enable a person who has a disability to use existing devices and software.

The use of the term *adaptation* needs to be clarified. To simplify the discussion, it will be concentrated on the specific area of information technology for visually disabled users, but it could be broadened into more general areas.

Most information technology devices cannot be used by visually disabled people. There are three main approaches to overcoming that limitation. One is to build a special device, designing it from the outset with the needs and abilities of visually disabled users in mind. This should yield a device which is as near as possible to ideal, in terms of its ease-of-use for a visually disabled person. The drawback of this approach is its expense. For any one particular device which is especially useful to visually disabled people it may be worthwhile implementing such a custom-built device. However to duplicate every available information technology product in this way would clearly be impractical; the potential market of visually disabled people is just too small.

A second approach is to take a specific product and to modify it. This will be feasible if the effort (and hence the cost) involved is significantly less than it would be to reimplement the product. The disadvantage of this method is that the resulting adapted product is likely to involve compromises between the design of the original version and the needs of visually disabled users. This means it will probably be more difficult to use.

The third approach is to develop a means of changing the interface to a set of devices, a generalized adaptation. In this case more effort and cost can be put into the adaptation as that cost will effectively be spread over the number of devices to which it can be applied. Again though, it is likely that the combination of product plus generalized interface is less easy to use than a custom-built product would be. In fact, what is produced in this case is a device in its own right. It is something which would not be of use to a sighted person, but which makes other devices which are used by sighted people accessible to visually disabled ones.

To summarize, there are three possible approaches to what broadly might be called adaptation:

1. custom-build a device;
2. carry out relatively minor modifications to an existing device;
3. develop a device which can be used to make a range of existing devices accessible.

The distinction between the three above-mentioned approaches can be made clear by giving examples. The Versabrilie is an example of the first case (manufactured by TeleSensory Inc.). It is described more fully later in this chapter, but is essentially a microcomputer designed specifically for blind users. The Smith-Kettlewell Institute in San Francisco specializes in adaptations which mostly fall into the second category. Examples are an oscilloscope modified to give auditory output and a micrometer with a tactile scale (Smith-Kettlewell, 1983). The Optacon (also manufactured by TeleSensory Inc.) is an example of the third kind of adaptation. Again, this is described fully below, but it is a device which transforms written text into a form which can be read through the tactile sense of a finger. It is a device in its own right, which gives blind people access to written text in a wide variety of forms.

The simplest form of adaptation is to bring together two existing pieces of equipment. For instance, given a text-based computer terminal one can add an Optacon and one has

effectively adapted the terminal. This form of adaptation does not fall into any one of the above three categories because it is effectively a combination of 2 and 3. The effort - and cost - involved is low because the difficult part is developing the interface device (i.e. approach 3) and that has been done by someone else. However, if the adaptation device does not already exist then adaptation implies the creation or modification of existing equipment. Computer-based equipment has two major components - the hardware and the software and adaptations can be applied to either or both aspects. Hardware modification at this level (i.e. where new devices must be created) generally is more difficult and expensive than software modification. Both require input of skill resources, but software production needs no manufacturing capability.

As so often happens with categorizations, the dividing lines between the three forms of adaptation are not as sharp in practice as is implied above. For example, adding a speech synthesizer to a computer terminal is *not* in itself a viable adaptation. What is also required is some modification of the terminal's software to match it to the new form of 'display'.

1.3 Introduction to the problem and proposed solution

There is a pressing need to ensure that disabled people can have access to computers and software. The principal means of output from computers have always been visual which clearly presents a problem for people who cannot see. Currently that output is nearly always in the form of printed text, either on paper or on a computer screen. This has meant that the problem of ensuring access to computer information for visually disabled people has been that of giving them appropriate means to enable them to read text. This is, in itself, a difficult problem. It is not sufficient just to be able to perform a one-to-one translation of visual text into a form which can be detected by one of the other senses - hearing or touch. It is also necessary to restructure the information to make it understandable despite the change of medium. Nevertheless, a number of devices have been developed which achieve a high level of success at performing both the translation and the restructuring. Some examples are discussed below.

However, more recently computer interfaces have become more visual, with information displayed in forms other than text. Specifically there has been the development of the so-called window, icon, menu and pointer - or *wimp*³ - style of interface. In addition to the keyboard which was previously used as the means of communicating to the computer, wimp systems have a pointing device which can be used to point at, and interact with, visual images on the computer screen. Windows, icons and menus are categories of the images used in such interfaces.

In using such systems, visual properties of the screen contents become significant in communicating to the user. Such properties include: shape, colour, size and position (sometimes in three dimensions). Printed text is still used extensively also. Remembering that it is also necessary for the user to be able to point at these images, it should be clear that a visually disabled person is seriously at a disadvantage in using such systems.

This style of interface is becoming rapidly more common. Currently most microcomputers, except the very inexpensive ones, offer a *mouse* pointing device as an option and have software available which exploits the mouse. One major manufacturer, Apple Computers, has gone so far as to make all software of this type on its most recent models: the Macintosh and the Apple II GS.

The history of the development of wimp interfaces is presented in Chapter 2, along with a more complete description of them. If wimp systems are to become widely used then there is a real need to address the problem of ensuring access for visually disabled users, since there is already a large community of such people who use computers in work and

³ The word *wimp* is an acronym - a word formed from the initial letters of other words. Where such words can sensibly be pronounced phonetically they are usually treated like any other word - except that they are conventionally written in capital letters. There seems no logical reason why a word should SHOUT at the reader in this way because of its etymology. Therefore, in this thesis *wimp* and any other pronounceable acronyms are written using the normal rules for capitalization, while acronyms which are pronounced as initials are written in uppercase letters (e.g. HMI - human-machine interface).

education. This is a good example of the two-sided nature of information technology; it can be used to improve life for people with disabilities, but it can exclude them also.

There are two possible approaches to the provision of tools for use by people who have disabilities. One is to adapt existing devices and the other is to design devices specifically for use by people with a particular disability. The advantage of the latter approach is that it should yield a product which is better suited to the prospective users, but it is often not practical. For example, consider specifically computer software and its use by visually disabled people. A great deal of effort goes into the development of a variety of programs. There are programs of different kinds - word processors, database managers, statistics processors etc. and within each type there are alternative products with particular features and strengths. To give visually disabled users equal access to computing facilities would imply re-writing every one of the existing programs. That would be an enormous task and a duplication of effort. In fact to re-design even a small proportion of the set of programs would be impractical because the potential market - visually disabled people - is so small. The approach of adaptation is attractive because it may be possible to produce *one* adaptation which could make *all* the existing software accessible. That represents a much lower level of effort, which is much more feasible. Of course, such a universal adaptation is an ideal. It is not possible that one adaptation will work with all software but it is still worth pursuing adaptations that go some way towards that.

This project was orientated towards the development of adaptations. There is already a large selection of wimp-style software on the market and it seems likely that in future more and more programs will have that kind of interface. It would be impossible to re-implement all that software for use by visually disabled people, but it may be possible to develop an interface which can be added to those programs (or at least a good number of them) which will make them accessible.

As stated above, there is a problem of making visual interfaces accessible to people who cannot see a computer screen. There are a variety of approaches which might be made to solving that problem. In this project a set of design principles was devised which provided the framework for one possible solution. This was by no means the only potentially viable approach, it was just one possibility. Working within that framework it was possible to implement a particular piece of software. That implementation involved making further specific decisions, but it meant that it was possible to test the design principles in a practical manner. So, this thesis describes one solution to the problem. By no means is it the only solution possible.

1.4 A brief introduction to visual disability

It is beyond the scope of this thesis to go into a full discussion of the nature of visual impairment. It is a topic of which there are often misconceptions and it is worthwhile ensuring that the reader has a reasonable context within which to place the work which is described herein. Chapman (1978) presents a quick survey of the nature and causes of visual disability, and Jernigan (1965) gives an excellent discussion of some of the social handicaps imposed on visually disabled people. Lowenfeld (1975) presents a very full survey of the status of visually disabled people, although this is in more of an American context. Lowenfeld (1980) provides a very full survey of psychology relating to people with visual disabilities.

Blindness is not the only form of visual disability. Many people have a visual impairment which does not amount to total lack of sight, but which is sufficiently incapacitating to amount to a disability. In fact the vast majority of visually disabled people are not blind, they are classed as being *partially sighted*. The term *visual disability* is used to cover a continuum, described thus in Chapman (1978, page 16): "...at one end the situation of total blindness, followed by mere perception of light through to perception of objects in the near and distant environment, finally embracing the level of impaired sight which demands a minimum adaptation in the presentation of learning materials."

There are three main ways in which pathological conditions in the eye may result in impaired vision. The visual acuity may be reduced, the field of vision may be limited or defective, and colour vision may be imperfect.

Visual acuity This is measured by the use of the Snellen Chart. That is the chart seen in optician's surgeries, consisting of lines of letters of diminishing size. The acuity measured is usually expressed as a pair of numbers, resembling a fraction. The 'denominator' of this pair is usually 60 (in the UK). The upper number is the distance at which the subject would have to stand in order to recognize what the normally-sighted person could recognize from a distance of 6 metres. That is to say that a person with average sight would have a visual acuity of $60/60$.

Field of vision The field of vision may be affected in two ways: an eye may have central vision with the peripheral field restricted to a certain angle (often referred to as *tunnel vision*), or the eye may have one or more 'blind' spots (scotoma) - which may cause loss of central vision. Restrictions in the field of vision are mapped out with the *perimeter*, an instrument that indicates the field limitations in the various directions on a chart.

Colour vision Perception of colour is determined by the discrimination of the three qualities of colour: hue, saturation, and brightness. In the rare case of total colour blindness, all colours are seen as shades of black, grey and white. Most colour blindness is partial, wherein the person has difficulty in distinguishing between certain colours, usually reds and greens. Colour blindness by itself, though a visual impairment, is generally not regarded as amounting to a visual disability. However, some eye conditions that reduce vision also result in colour blindness. Problems of colour discrimination may also have a serious affect on a person's ability to use computers. Increasingly computers use displays in which colour is used to convey important information. Also, monochrome displays need not necessarily be black and white, but may display yellow text on a black background, or black text on green, for instance.

One assumption which is sometimes made, incorrectly, is that the reading problems of partially sighted people can always be alleviated by magnification. As the above description of visual disabilities should make clear, this is not the case. In fact magnification could make matters worse for a person with tunnel vision, since the larger the text is, the less will be in their (restricted) field of vision.

Another remedy to the problem of reading, which is less applicable than is often assumed, is the use of braille. The vast majority of visually disabled people cannot read braille. There are many reasons for this, and each individual has his or her own reasons for deciding whether or not to make the effort to learn braille. However, one important factor is that the major causes of visual disability are often associated with advancing age, and tactile sensitivity also tends to diminish with age. Even for those who decide to learn braille, and are successful, its benefits are limited.

Another misconception, which has really attained mythical status, is that the senses are balanced such that if a person is lacking one sense the others become more acute. For example, visually disabled people are often assumed to have better hearing. There is no evidence to support this idea, in fact, "Any higher efficiency of the blind in interpreting the sensory data perceived must be the result of attention, practice, adaptation, and increased use of the remaining faculties." (Lowenfeld, 1975, page 221, also see Lowenfeld, 1980).

1.5 Examples of the use of Information Technology

This section presents a brief description of a number of existing computers and micro-electronic aids which have been designed or adapted for use by visually disabled people. It is not a complete list of all that is available, but includes devices which are particularly interesting. It is significant that all the examples given relate to the presentation of *text*, which is a reflection of the current stage of the technology. An extensive, if slightly dated, review of speech-output devices for blind users appears in McGillivray, 1983.

The Frank Audiodata In adapting computer equipment there are two basic approaches: either to modify the hardware or the software. The Frank Audiodata is an example of the former. It is a hardware modification which is most commonly found on IBM PC computers, but is also available for the Osborne. It has facilities for both partially sighted and blind users. It provides magnification of the screen contents and a screen reader, which will translate text displayed on the screen into synthetic speech. The reading of the screen operates entirely independently of the application software, so that any text-based program can be run using this adaptation. Such adaptations are often referred to as *transparent*, in that the application runs normally, unaffected by the presence of the interface modification.

The unique feature of the Audiodata is the way in which the user controls the selection of text to be spoken. This is performed using two sliders, known as *tasos* (*tasos* is derived from *tactile acoustic screen orientation*) which are located on a modified keyboard. One slider moves vertically and the other horizontally, and they control the corresponding location of the area of the screen to be read.

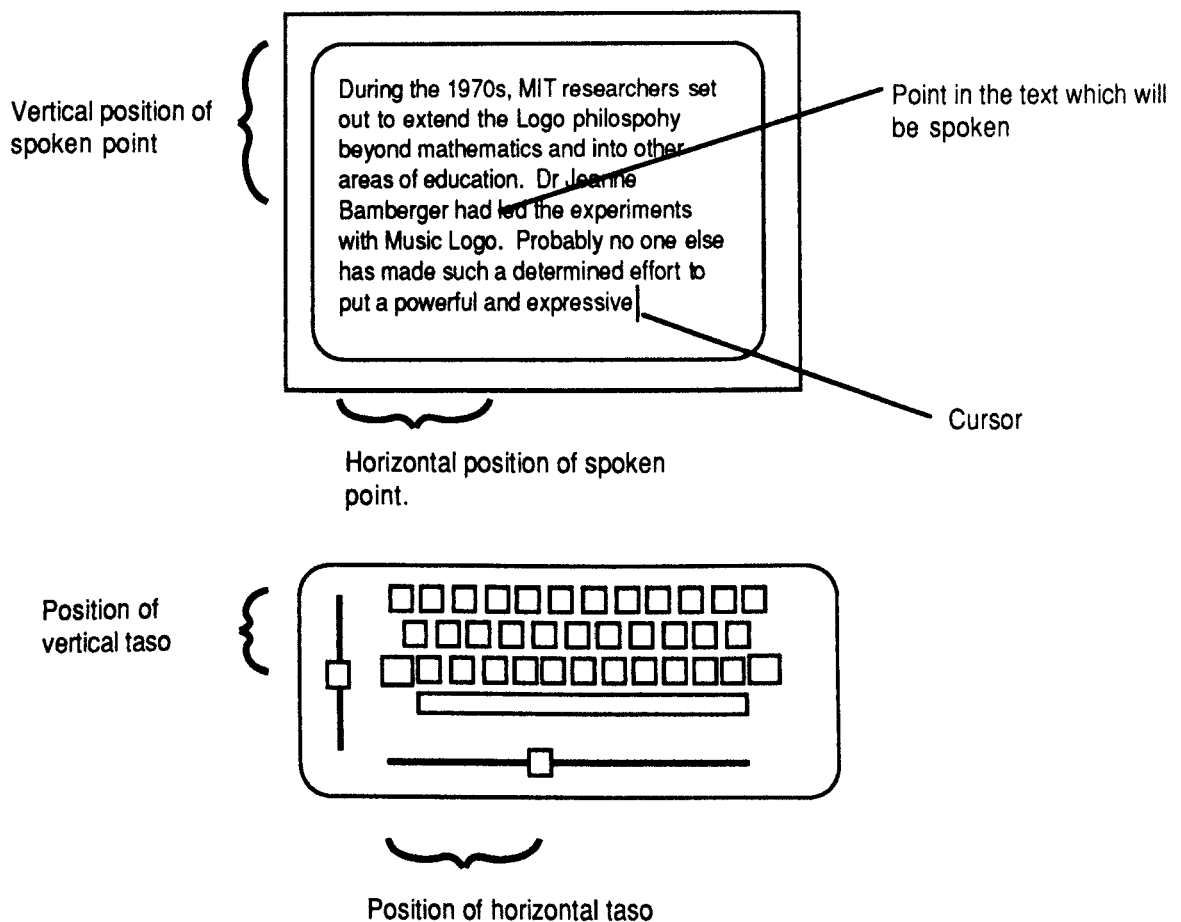


Figure 1.1. The mechanism for specifying the section of text to be spoken on the Frank Audiodata.

Figure 1.1 illustrates how the position of the section of text to be spoken corresponds to the positions of the tasos. With the tasos in the positions shown the character *l* (in "led") might be spoken, or the user might specify that the whole of the current line ("Bamberger had led the experiments"). Notice that this mechanism means that the specification of the text to be spoken is essentially independent of the input state of the program running. This separate mechanism means that the focus of the screen reader is independent of the position at which typed text appears - the so-called *cursor* (see Section 2.1) - although they can be made to coincide. The tasos can be repositioned with the guidance of auditory feedback in the form of short tones, or bleeps. Different pitches convey information to the user about the current

position of the active speech point - such as the number of lines moved through, whether the current line is blank etc.

The interface is controlled by a set of commands. These are entered as special key sequences. The commands are single letters, prefixed by the *Audio* command, which is a two-key combination (the *alt* key combined with function key *F10*). These commands can be used to control the output. For instance, *audio M* (that is, *audio* followed by pressing the *M*) will enable the Audiodata to pronounce mathematical terms correctly, *audio O* switches on the speaking of each keystroke, and *audio U* enables the user to distinguish lower- and upper-case letters, by speaking of the latter being preceded by a short tone.

As mentioned above, this adaptation can be used with almost any application software. That may be a word processor or a database, or almost any program which displays text on the screen. However, in this country at least, it seems to be most popularly used with the *Wordstar* (MicroPro International, 1981) word processor. This combination is the one used to teach word processing at both the Royal National Institute for the Blind (RNIB) Commercial Training College and the Royal National College for the Blind.

Vert Vert is a slightly different approach to the same problem tackled by the Frank Audiodata - that of adding a synthetic speech screen reader to the IBM PC, manufactured by TeleSensory Incorporated. It was stated above that either software or hardware can be adapted. In fact, Vert is available in different versions - which differ in the degree of modification there is to each of these components. At one end of the spectrum, there is *Soft Vert* which is entirely a software modification - except for the speech synthesizer. At the other end is the *3278 Vert* which includes an additional circuit board to be added to the PC computer. The important differences in the various versions is the performance - and the price paid for it; *Soft Vert* is the cheapest version.

All versions of Vert differ from the Audiodata approach in that the speech is controlled via commands typed on the keyboard; there are no additional hardware controls. These

commands are usually typed by using the letter keys in combination with the *control* and *alt* key - which are standard on the PC keyboard. The Vert does also support the principle of working independently of the application program so that it also works with any text-based software. In the case of Soft Vert this is achieved through multi-programming, whereby the Vert program and the application run in separate communicating processes.

Vincent Workstation The Vincent Workstation (Vincent 1986) is based upon a very sensible, pragmatic philosophy. Its design is based as far as possible upon the use of inexpensive, standard hardware; the adaptation is embodied almost entirely in the software. The system is based on an Acorn BBC microcomputer to which can be attached a selection of peripherals. Synthetic speech is used as the output medium, so that a speech synthesizer must be added but a number of different proprietary devices are compatible. The computer incorporates a conventional *qwerty* keyboard, but alternative inputs may be attached: a modified Perkins braille or a Concept Keyboard (manufactured by Star Microterminals). There is usually a need for sighted people to read the output from the system, so that a monitor and a printer may be added. A number of programs have been developed to run on this workstation, including a word processor and a Basic programming language interpreter.

Audiocalc Another program which uses synthetic speech on the BBC microcomputer is a spreadsheet program, called *Audiocalc* (Jennings, 1985). Spreadsheets are programs which are generally used in accounting applications. They allow the user to manipulate figures arranged in rows and columns of a grid. Since the spatial arrangement is an important aspect of such programs, its adaptation to auditory operation is of interest. In fact, the spatial information is given entirely in a spoken form, by the naming of cells; *A1* is spoken for row A, column 1 and so on. It does use some other auditory cues in the form of 'beeps', but these usually signal errors and do not give any spatial information.

Tactile screen One apparently obvious approach to the problem of presenting screen-based information to blind people is to provide them with a tactile representation of the screen. So

far however, this has proved impossible because of the practical problems involved. Most attempts to provide such a device have been based on arrays of pins which can be raised and lowered and so felt by the user. The patterns formed can be either braille characters or other shapes - perhaps the outlines of the printed letters. There are a number of sources of difficulty which arise. Firstly, any device which is electro-mechanical is likely to be very slow in operation. Similarly it will be prone to problems of mechanical unreliability. This can be very serious, since just one pin jamming in an up or down position can make the display illegible, particularly if braille is being used. There are also power dissipation problems. Most of these devices use a solenoid to hold each of the pins in either its up or down position. That means that on average each solenoid is active for half the time and the power needed - as well as the heat generated - can be significant.

The most advanced attempt to solve this problem has been developed into a prototype by the American Foundation for the Blind (Stevens, 1984, Plastics Design Forum, 1983 and Goodrich et al., 1986). So far this has been built as large as one quarter of a page. However, it has suffered from the above-mentioned reliability problems, and probably its most serious defect is its slowness, having a refresh time of *ninety* seconds.

Versabraille Tactile output can be provided successfully if the designer is less ambitious about its form. Thus, a number of computer terminals can display a single line of *refreshable* braille. Probably the most successful example is the Versabraille (Doorlag and Doorlag, 1983). Apart from the problem mentioned earlier of the small number of braille readers, there are a number of disadvantages to conventional embossed paper braille; it is very bulky and it is very difficult to 'skim' through reading it. This makes it difficult to organize texts through indexes. The Versabraille gets around these problems by storing the text on a cassette tape or floppy disc, which can be read by the machine and displayed on a line of braille cells. The device also has a braille keyboard which can be used as a word processor to create texts to be stored in the same way, on tapes or discs. As essentially a

text input and output device, the Versabrilie can also be used as a computer terminal and linked to a number of devices.

All the same, the problems of designing tactile displays has affected the success of this product. It is considered to be a rather expensive aid, and the most costly component is the braille display.

Magnified displays As mentioned above, not all visually disabled people are totally blind and some of those who are partially-sighted can access computers using their residual vision if the computer display is suitably enhanced. Using screens of different colours (e.g. green on black) may be of assistance to some individuals. Others may be helped by screen magnifiers of one form or another. These may be traditional, 'low technology' optical magnifiers. Alternatively, a software modification approach may be applied to display output on a standard screen, but in a magnified form. Or a third possibility is to use a specially designed display. *Vista* (manufactured by TeleSensory Inc.), is an example of the software approach, which runs on IBM PC's.

For the Apple Macintosh, Inlarge is a software screen magnifier, manufactured by Berkeley System Design. Its great power is that it is transparent; it runs in conjunction with any application program. It provides magnification of between two and sixteen times. It has a number of useful features to make it easier to use, such as a mechanism to scan automatically across the screen.

Brailink Brailink (Adams, 1982) is a microcomputer and terminal which displays its output in braille instead of text. It has a 48-cell wide refreshable braille display. Input can be typed in either on a qwerty keyboard, or by using the keyboard in a conventional seven-key braille key set. It has local editing facilities and data can be saved on cassette tapes. Alternatively it can be linked to another computer and used as a terminal. Brailink is the device most commonly used currently by programmers with visual disabilities. A serious question,

relating to this project, is whether it will become redundant in the future as more computers incorporate a non-text-based interface.

Some other approaches to presenting information to blind people in an auditory form are discussed later, in Section 3.2.

1.6 Alternative interfaces for sighted people

Although this thesis is mainly concerned with the development of interfaces for use by people who are visually disabled, it may also have relevance to the improvement of interfaces for those who can see. The trend towards incorporating more information in a visual display has already been mentioned. This not only causes problems for the would-be visually disabled user, but also increases the visual processing load on a sighted user. It may be that, if this trend continues, users will reach a point of 'visual overload'. Also, as will be seen later in this thesis, there is evidence to suggest that some forms of data are more readily received audibly. There are also concerns about the health of computer users using visual displays which can cause problems, such as eye strain and which do give out low levels of radiation (Pearce, 1984). All these considerations suggest that it is worthwhile considering the use of communication channels other than vision.

1.7 Chronology

Early research towards this thesis, looking quite broadly at the use of computers by people with disabilities, and particularly in Special Education was carried out in 1984. Some of the background covered in that investigation was published, in Edwards (1984), Salvadori Paleotti, Kilroy and Edwards (1985) and Edwards and O'Shea (1985). The existence of the problem of increasing difficulty in adapting interfaces for visually disabled users, addressed by this project, was discussed by the author, Tom Vincent and Tim O'Shea towards the end of 1984. Ideas for solutions began to emerge, and an early description of the design of an auditory word processor appears in Edwards and O'Shea (1986). By the summer of 1985 work had begun on translating them into reality, in the form of a piece of software. Its

development proceeded, subject to the external delays, such as late delivery of software tools, which seem an inevitable part of any programming project. It was tested and informally demonstrated during late 1985. A description of that implementation is given in Edwards (1986). The formal evaluation (described in Chapters 6 to 9) was carried out between January and March of 1986. The data obtained was analyzed and this thesis written during the subsequent months. Discussions were held with researchers in the United States during a one-month visit in September. A number of practical suggestions for enhancements to the interface were made during these discussions, which are reported in Section 7.5.

The basic problem addressed by this project - that of adapting highly visual interfaces - has been outlined in this chapter. In the following chapter that problem is described more fully through a brief history of the development of interfaces.

Chapter 2

The Problem

2.1 Introduction

Chapter 3 will describe how visually disabled people have learned to use the older generation of computer equipment with the help of adaptations and communication aids. Generally these approaches have involved translating the output from the computer into a form which can be received by one of the senses other than sight - that is, either hearing or touch. All the methods have relied on the fact that the output from computers is in a written form. Although writing is a visual communication medium, there already exist well-defined methods for translating it into a non-visual medium, particularly synthetic speech and braille. That has meant that as long as the output from computers has been written text, visually disabled people have been able to get access to it, by means which are often extensions of methods of transforming general forms of text. However, computer interfaces are becoming increasingly visual and so more difficult to adapt for access by visually disabled users. The first part of this chapter includes a brief history of the development of human-computer interface, which shows how they have increased in visual complexity. The latter part of the chapter then represents the position which has been reached in the development by describing the important elements of contemporary wimp-based interfaces.

The development of computer interfaces has been a series of overlapping or leapfrogging advances of hardware and software. Historically existing hardware would be put to use then software would be designed which made the most of its features, which inspired improvements in the hardware - and so on. To a large extent the progress in the design of

interfaces can be traced by the development of one form of interactive program - the text editor. A comprehensive history and description of text editors is given by Meyrowitz and van Dam (1982a and 1982b), but relevant highlights are given here.

With the advent of microcomputers the term *personal workstation* has been coined, to describe a computer which is used exclusively by one person. Although the nomenclature may be new, the concept is not. Early computers could be accessed by just one user at a time. Huge mainframes they may have been, but until the advent of multi-tasking (whereby more than one user can interact simultaneously with one computer) they were essentially personal workstations.

On most early computers, data - and particularly programs - were usually held on punched cards or paper tape. To run a program these cards or tape would be read into the computer's memory and then executed. The cards or tapes themselves were the medium on which the program was stored between runs. To modify a program on punched cards the user would locate the appropriate card, re-punch it and replace the new card in the card deck. The paper tape user would go through a corresponding process, involving duplicating the correct parts of the tape and re-punching the incorrect sections. An extension of this form of working which became available on some systems with the advent of mass storage devices (discs or drums) was that when cards were read in they would be stored as card images in a file. In this way a program could be re-run without the cards having to be re-input, but it also was possible to modify the program by editing the copied information which was stored in the file. Editing instructions could be input as a batch job on cards which contained the commands which acted on the card images in the file, so that this was still a batch-oriented way of working.

Early computers often borrowed existing technology for its input and output devices. These were the electromechanical typewriter/printers usually referred to by the trade name, the *teletype*. The essential features of the teletype terminal are illustrated schematically in Figure

2.1. The paper roll effectively represents a *history* of the state of the file; the listing of a line represents its contents at the time of writing and only the most recent appearance (that which is furthest down the paper) shows the current state of the line in the file. In other words, the vertical dimension of the paper really represents a temporal dimension.

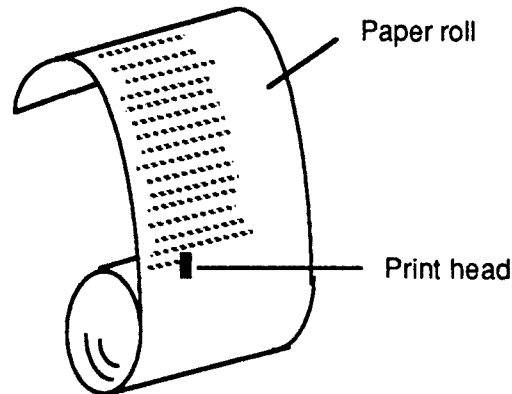


Figure 2.1. The elements of the teletype terminal.

By modern standards these interfaces were very limited. However, at the time they introduced a new and powerful concept into computing - the ability to modify programs (or other textual data) and to *see* the changes as they occurred. Because of the limitations of the terminals, the editing programs to do this were generally *line-oriented*. It was indeed a step to start thinking of lines and no longer of card images. The user could type a command to get a particular line printed on the terminal. She could then enter further commands specifying changes to be made to that line and get the modified line printed for checking. Examples of such editors are *CMS* (Meyrowitz and van Dam, 1982b), *ed* (Kernighan, 1978) and *Zed* (University of Cambridge, 1984).

In the early 1970s the paper roll of the teletype began to be replaced by the television-like cathode ray tube. The immediate benefit of this move was the rapidity with which text could be displayed - at the speed of a scanning electron beam instead of the momentum of an electromechanical print head. Initially these terminals were used in the same way, using the same software, as those which they had replaced. In fact they are often referred to as 'glass teletypes'. However, many editors began to appear which were based on the concepts

proposed by Irons and Djorup (1972) which extended the power of the software. These concepts are represented in Figure 2.2.

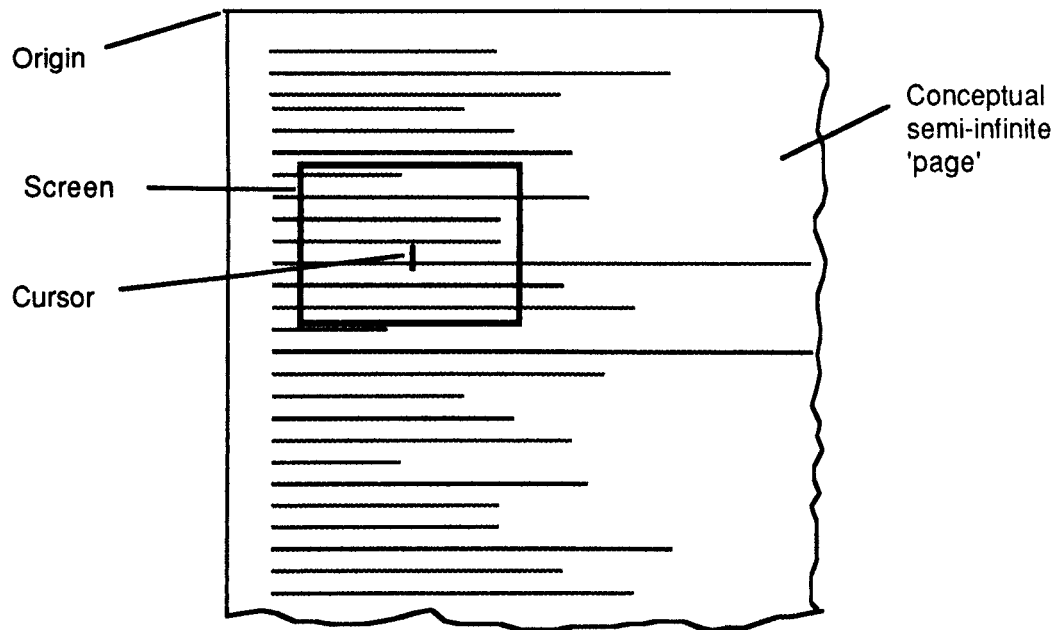


Figure 2.2. The concepts underlying display editors. The file is represented as a quarter plane with an origin corresponding to the beginning of the file. A portion of the plane can be viewed through the screen and actions occur at a point on the screen called the cursor.

A file is conceived as being a quarter plane which extends infinitely to the right and downward. The contents of the file are displayed on this plane, but unlike the teletype's paper roll the text displayed is constantly updated so that it always reflects the exact contents of the file. The dimensions of the plane therefore accurately reflect the spatial layout of the file's contents. The terminal screen acts as a frame through which the user can view an area of the plane and can be moved around the plane under the user's control. This is analogous to the operation of a microfiche reader. The file is like the microfiche (except it is infinite in two dimensions) and the reader moves over it, displaying one frame at a time.

On the screen is a point marker known as the *cursor*. Like the teletype print head this is the place at which displayed characters appear. However, unlike the print head, the cursor can also be moved by the user to the appropriate spatial position in the file at which the next

action should occur. Movements of the cursor are controlled by keys on the keyboard. A large number of editors have been developed based on these ideas. They are usually referred to as *display* editors, and examples are *Vi* (Joy and Horton, 1980), *Emacs* (Stallman, 1981) and *Ned* (Bilofsky 1977).

The last two examples of display editors mentioned above have a feature which to some extent suggests the transition to the next generation of editors. There is no logical reason why there should be only one 'frame' into a file. More than one frame can be fitted into one screen, each displaying different parts of a file - or indeed parts of separate files. Such frames are actually known as *windows*. Emacs allows the screen to be split into two windows, while Ned allows up to sixteen windows. Both Ned and Emacs use a technique known as *tiling* whereby windows are *not* allowed to overlap. Creation of a new window simply causes the current one to be sub-divided into two, one displaying the original window's contents and the second displaying whatever should be in the new one. This is illustrated in Figure 2.3.

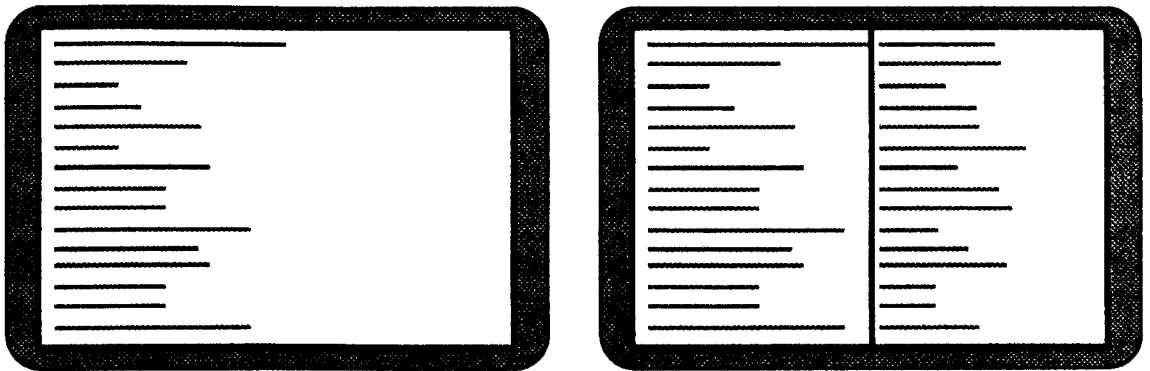


Figure 2.3. *Tiling*, a new window is created by sub-dividing the existing one. They do not overlap.

A disadvantage of tiling is that the more windows which are created the smaller each becomes, so that the information in each one is limited. Other systems allow windows to overlap. This means that each one can be of any size, up to the dimensions of the screen, but it also implies that information can be hidden, obscured behind other windows. This is illustrated in Figure 2.4.

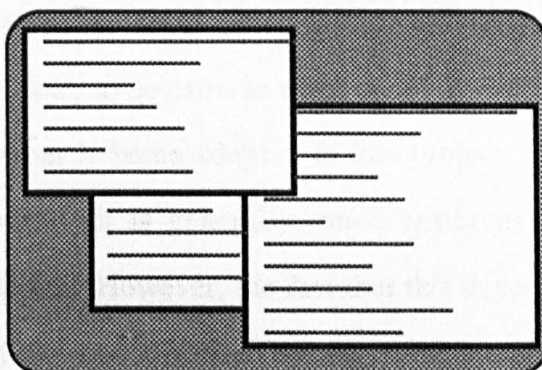


Figure 2.4. Overlapped multiple windows. Different views into one or more files can be displayed on the screen.

Manipulating overlapping windows becomes analogous to shuffling pieces of paper in three dimensions, but of course the contents of any window are not fixed like text on paper. A large conceptual leap has thus been made from the single, physical roll of paper. In fact the advent of multiple windows represents just one new feature which is common in modern editors. Again development has been led by improvements in display technology. High-resolution graphics screens enable editors to display not only the written content of a file but also typesetting information. This gives rise to the concept of *what-you-see-is-what-you-get*, or *wysiwyg*. In addition, more attention has been given to the interface between the user and the program. The nature of this interface is the topic of the next section.

The point of this short history is to show how developments in human-machine interfaces have been accompanied by an increasing degree of visual complexity. All the forms of interface described relied on the user's ability to see, but until recent developments the need was only to see *text* (and sometimes also control characters). This has meant, to some extent, that the handicap of visually disabled computer users is the inability to read. Of course that is a problem in the wider world outside computing and techniques have been devised to get around it, such as the use of tactile and auditory alternatives to written text. The remainder of this chapter describes modern visual interfaces as a way of explaining the extent of the problem of adapting them for use by people who cannot see.

2.2 Modern interfaces

The previous section described the development of human-computer interfaces up to the present time. It is now necessary to describe in some detail modern, wimp-based interfaces in order to make it clear what is being adapted in this project. Their operation is very difficult to describe in words; it is generally much easier to explain their use by a demonstration on the computer. However, the fact that this description is so difficult is a powerful demonstration of the problem faced by any blind person who would use such a system, without the possibility of *seeing* a demonstration. The design of such interfaces is far from being standardized. The description given here is intended to cover the common features. It does not describe any one particular implementation, and is not comprehensive. This section introduces a certain amount of vocabulary which is used to describe interactions in subsequent chapters of this thesis.

One aspect of terminals which remained constant throughout all the developments mentioned above was that all input from the user was typed on a keyboard. Another effect of using high-resolution graphics screens is that it is possible to represent information in a form other than printed text. Graphical objects can be used as a medium for communication by way of a pointer on the screen which can be made to point at other screen images. Unfortunately this pointer has also become known as a cursor, which causes confusion with the active point in a piece of text. In this thesis it will be referred to as the *pointer* (or *mouse pointer* - see below). By way of clarification of the distinction, a screen must have exactly one mouse pointer, which can be used anywhere on that screen, while each window will have its own cursor, which operates only within that window. The shape of the mouse pointer need not be fixed; it may change according to its role and location on the screen. Often it will be an arrow, but may, for example, change into cross-wires in a graphics program which requires more precise pointing.

The mouse pointer is usually controlled by an input device which the user controls by hand. A number of devices are available for this purpose, such as joysticks, lightpens, bitpads and

mice. The most common is the *mouse*, which is a box which sits on a flat horizontal surface, such as a desk, and is connected to the computer. On its underside is a mechanism (usually a ball) which detects movements over the surface. Movements of the mouse are reflected by similar motions of a pointer on the screen; moving the mouse to the right will move the pointer to the right, while pushing it forwards moves the pointer upward on the screen. This close connection between the mouse and the pointer can cause some apparent inconsistency. People will often talk of pointing the mouse at an object. What they are really referring to is pointing the mouse pointer - under the control of the mouse. However, it is the mouse which seems more immediate¹. Within this text this colloquialism will be used since it should not cause any serious confusion.

Card, English and Burr (1978) conducted experiments in which they compared mice with joysticks and key-based selection mechanisms. They concluded (op. cit, page 612):

"Of the four devices tested the mouse is clearly the superior device for text selection on a CRT [cathode ray tube]:

1. The positioning time of the mouse is significantly faster than that of the other devices. This is true overall and at every distance and size combination save for single character targets.
2. The error rate of the mouse is significantly lower than that of the other devices.
3. The rate of movement of the mouse is nearly maximal with respect to the information processing capabilities of the eye-hand guidance system."

Mice generally have one, two or three buttons on them which are used to communicate with the program and most actions of the mouse consist of the user pointing at an object via the mouse and then pressing one of the buttons. There are different ways in which the buttons

¹ It will be noted later in this thesis that this concept is perhaps even stronger in an auditory program in which there is no visible cursor.

can be pressed. The simple press and release of a button is usually referred to as a *click* or *single-click*. Holding the mouse down for longer than some pre-set time may be treated as repeated single-clicks, in a manner similar to the way most electronic keyboards have an 'auto-repeat' facility whereby holding a key down produces multiple copies of the corresponding character. A *double-click* consists of two presses of the button which occur within a short, pre-defined time period (e.g. half a second) during which the mouse has not moved. Sometimes a mouse button is pressed but then held down while the mouse is moved. Such an action is known as *dragging*. Pointing at an item and clicking on it is often used as a mechanism to make the item active and is referred to as *selecting* it.

There are a number of different images which can be displayed on a screen, to be pointed to by the mouse. The principal ones, however are named in the acronym *wimp*: the *window*, *icon* and *menu*. Taking these in reverse order, menus generally come in one of two varieties: *pull-down* and *pop-up*. A pull-down menu has a fixed location on the screen, which is marked by a menu heading. Pointing the mouse at that heading and pressing one of the buttons will cause a list of menu entries to appear below the heading. One of those entries can then be selected. The effect of so doing depends on its identity.

Pop-up menus, however, do not have a fixed location, but appear wherever the mouse is when a button is pressed. That is to say, the user presses a mouse button and a menu appears, positioned so that the mouse pointer is pointing at an entry within it. Once again, one of those entries can then be selected the effect of which depends on its identity.

Sometimes menu entries do not control commands as such, but the setting of a state within a program. For example, a word processor may allow the text to be right-justified or to have a ragged right margin. Input will be formatted according to the selected state of the program. There are two ways of displaying the current state. One is for the program to change the text of the menu entry. For example, when the state is such that the text is justified the menu entry might be:

Turn justify off

Selecting that entry would switch the justification off, and cause the menu entry to be altered to:

Turn justify on

An alternative mechanism is to mark a menu entry with a special symbol, such as a 'tick' (✓), to signify that it is active. In this case, if the text is being justified, the menu entry might appear as:

✓ Justified

Selecting that entry would toggle the state and change the entry to:

Justified

Selecting commands from menus is generally a slow form of interaction. It is therefore sometimes desirable to give users an alternative, quicker way of communicating commands. Control-key commands, similar to those found in older generation programs, can be allowed as alternatives to the menu entry. For example, a program may be written such that it has a menu of file commands which include an *Open* command which the user can select in the normal manner, but in which typing the control code, **Control-O** will have the same effect as the *Open* command. Such *control-key alternatives* are not necessarily provided for all the menu entries, but just for frequently used commands. Generally it is observed that novice users use the menu entries while they are learning to use the software, but as they become more expert they will more frequently use the control-key alternatives as they memorize their identities - so becoming more efficient at using the program.

An icon is a small picture on the screen which represents a real object associated with the program.² That object may be, for instance, a file, a directory of files, a printer or a disc. The design of the icon will give the user pictorial information as to its identity. So, for example, an icon representing a physical object such as a printer will conventionally

² Such an object is 'real' in the sense that it has an existence independent of the program. It may be a physical entity, such as a disc drive, but it may be something less tangible, such as a program.

resemble a printing machine. Icons representing more abstract objects such as files are more stylized but they can still communicate significant information. For example, icons representing text files on a system might have identical outlines, while the inner detail will be different according to which application program created the file. Icons may be selected. Usually they can be moved around the screen and a command may be implied by the place to which an icon is moved. For instance, in the Star system, a document will be printed if its icon is placed over the icon for the printer (Smith, Irby, Kimball, Verplank and Harslem, 1982).

Figure 2.5 shows an icon which could represent a document file on a Star or a Macintosh.



Figure 2.5. A typical icon representing a document file.

That icon comprises the following information:

- that the file is a document - indicated by its resemblance to a sheet of paper;
- which program created the file - designated by the quill picture;
- the name of the file - *letter*;
- the fact that the file is accessible - since the icon is not shaded;
- that the file is not currently selected - it is not highlighted in inverse video.

Windows are used in many ways and in different roles. Usually a large number of windows may be opened. They may be of different sizes and often the user is allowed to alter the size of a window. Similarly, the user may usually move windows around the screen. Furthermore they are often allowed to overlap, so that manipulating them effectively becomes a problem in three dimensions. As described above, windows may be used to

display the contents of files. Because of the graphics capability of the screens, the files need not necessarily be text files, but may contain pictures. The program controlling the window may format data in a particular way, so that, for example, the contents of a database may be displayed as text in an appropriate layout.

Windows may contain a number of different types of objects with which the mouse can interact. The simplest object is the *button*. This is an area of the screen with a label. Clicking in that area communicates a message to the program. Within the window there may be objects which control the window itself. The mouse can be used to move the window or alter its size through interaction with these objects.

The contents of document files may be displayed in a window. The user can usually designate a section of the text as *selected*, which means that all editing commands will operate on that portion of the document. The selection may be defined by two markers at its beginning and end and is usually identified by the text between the markers being displayed in a highlighted manner. For instance, in the example below the word *dog* is selected:

The quick brown fox jumps over the lazy **dog**.

It may be that none of the text is selected, in which case the cursor marks the point at which editing operations will occur. So, in the following example, the cursor is a vertical bar, and is positioned between the *g* in *dog* and the fullstop:

The quick brown fox jumps over the lazy dog|.

Another way of thinking about this is that the two markers have converged to a point. The method by which the user defines the selection varies between systems. Moving the mouse pointer to a point in the text and pressing the button will usually cause the cursor to be moved to that point. To select a range of text, the pointer might be moved to the start of the intended section the button pressed and held down while the mouse is *dragged* until the pointer reaches the end of the intended selection.

One object found in windows on most systems is the *scroll bar*, although the details of its implementation varies between systems. Screen objects have a finite size and so are limited in the amount of material they can display. A good example is a file window which can display a portion of the file, as described above. To move to another area of the file, the window can be (conceptually) slid over the file. This is known as *scrolling*, which may be controlled by a scroll bar. Commonly a scroll bar incorporates a marker which moves within it, called the *thumb bar*. The location of the marker is an indication of the the current relative position of the window within the file, but it is not only an indicator, it can also be used to change the portion of the file displayed. The position of the marker within the scroll bar - and hence the location of the window within the file - can be altered in a number of ways. Usually there are buttons which will move it at different speeds in either direction. As the button is pressed the material *scrolls* through the window, and the thumb bar moves to reflect the change. Alternatively the user can drag the thumb bar to a new position and the display will be adjusted to show the corresponding part of the file. Scroll bars are used not only within windows. For instance, it may sometimes not be possible to display all the entries in a menu simultaneously, in which case the menu may act as a frame displaying a number of the entries and other entries may be scrolled into the menu.

Although there may be more than one window on a screen, a user can usually interact with only one at a time. There is, therefore, generally a mechanism for *activating* windows. Activating a window usually brings it to the 'front' of the screen, so that it is not obscured by any other windows. Thereafter the user can interact with the window's contents.

Some windows are opened by the program to enable the user to communicate with it. For example, if the user executes a command to open a file from disc (usually through a menu) the user must communicate the name of the file to be opened. For this purpose a window known as a *dialogue* may be opened³. Through this the user may specify the name -

³ The terms *dialogue* and *alert* (see below) are used in Apple documentation. Other manufacturers use different terms for similar entities, such as *notification windows* and *interaction windows* respectively.

possibly from a menu or by typing into it. There are two forms of dialogue: *modal* and *modeless*. Modal dialogues are so-called because opening one puts the user into a special state or *mode*, whereby she can interact only with the dialogue. As soon as a modal dialogue is opened it is automatically activated and clicking the mouse anywhere else on the screen will have no effect - except probably the sounding of an error tone. The dialogue can only be closed again through interaction with its contents. Normally the user will communicate the necessary information (e.g. the appropriate file name) and then signal that the program should continue, using that information. However, one button in any dialogue should always allow the user to cancel the command which opened it in the first place, in case it was opened in error. Modal dialogues are used when the program cannot continue without further information from the user. So, in the above example of opening a file the dialogue would be a modal one since the program could not complete the open command without the file name. The user would specify the chosen name and then signal that the program should continue, in which case the appropriate file would be opened. If the open command had been selected in error, then the user would cancel the dialogue and no file would be opened.

Other windows are modeless, and modeless dialogues are a particular example. On opening, it is not automatically activated and the user is not forced to interact with it. It will remain open until the user explicitly closes it and at any time the user can activate it and interact with it. A modeless dialogue might be used, for instance, to specify the target when the user has executed a 'find string' command in a word processor. Generally the user would specify the target string as soon as the dialogue opens by typing into the dialogue, and then signal that the search should begin, by clicking on a button in it. When the search is complete the user may wish to return to the window in which the document is displayed and interact with it. If, however, at some later stage she decides to search again - for the same or some other string - then the dialogue will still be available for her to activate and interact with as appropriate.

Another special type of window, similar to the dialogue, is the *alert*. An alert is opened when a serious error occurs. It will give a description of the error and some alternative actions which the user user may take in response. Alerts are *always* modal, so that the user is forced to be aware that something has gone wrong. In this way, it is often the case that there is only one action allowed by an alert - that of continuing, but in specifying that action the user is effectively acknowledging that she has read the message.

Dialogues and alerts may contain the following items:

- text,
- buttons,
- editable text,
- lists.

Text is used to communicate messages to the user. It may also be used to label any of the other items. Buttons are labelled areas within the dialogue or alert. They are activated by the user pressing a mouse button while the cursor is pointing within that area. The effect of so-doing depends on the type and identity of the button. Editable text provides a means for the user to communicate more complex information to the program. For example, the user may type in a string which is to be searched for within a document. Lists may be provided when the user has to make a choice from a number of names. For example, if she has entered an **open file** command, a dialogue may be opened which includes a list of all the file names on the disc, from which she can select the one she wants to open. Only a fixed, small number of items may be displayed in a frame within the dialogue, but the number of items in a list may vary, so facilities are provided so that the list can be scrolled through the frame. Once the required name appears in the frame it can be selected by pointing at it and pressing a mouse button. Some dialogue implementations, as an alternative, allow the user to type in the name - or a suitable abbreviation of it. This may be a more efficient interaction, especially if the list of names is long.

Many errors can be avoided by constraining the user's interaction. There are situations in which particular commands cannot be executed and to preclude such errors it is often possible for the program to *disable* the corresponding screen object. The fact that an object is disabled will usually be made visible in some way - by its being shaded, for example. To attempt to use a disabled object will usually evoke an error tone - but not cause any action to be carried out. While inactive objects are usually labelled visually it is also often useful for a program to mark out currently active objects visually. This is known as *highlighting*, and usually is signalled by the pertinent object being displayed in inverse video.

2.3 Some important wimp systems

Most of the concepts described above, as part of wimp-based programs, were developed as part of the Smalltalk project, started by the Learning Research Group at Xerox's Palo Alto Research Centre (PARC) in the 1970s (Goldberg and Robson, 1983, and Krasner, 1983). Their work was based upon Alan Kay's concept of a portable, powerful and easy-to-use general-purpose computer, known as the *Dynabook*. The goal of the Learning Research Group was "to create a powerful information system in which the user can store, access and manipulate information so that the system can grow as the user's ideas grow." To that end the group has concentrated on two principal areas of research: "a language of description (a programming language) which serves as an interface between the models in the human mind and those in computing hardware, and a language of interaction (a user interface) which matches the human communication system to that of the computer." (Both the above quotes are taken from Goldberg and Robson, 1983, page vii). It is the latter area of research which has had the greater influence on computer systems development, and that influence has been substantial. This is probably due to the fact that the interface is a visual, attractive product which can be seen to make interaction with computers easier, while on the other hand, the programming language may be seen as rather esoteric and somewhat different from conventional languages.

The development of Smalltalk proceeded in parallel with the development of a computer on which it could run, namely the Xerox Alto (Thacker et al., 1982). This was a 16-bit mini-computer which was accessed through terminals which had high-resolution, bit-mapped graphics displays and mouse pointing devices. Also developed on the Altos at Xerox PARC was the *Bravo* editor (Meyrowitz and van Dam, 1982b). These two systems display between them most of the interface concepts mentioned in Section 2.2.

Bravo was succeeded by the *Star*, which was released as a commercial product (Smith, Irby, Kimball, Verplank and Harslem, 1982, and Smith, Irby, Kimball and Harslem, 1982). An important feature of the development of the *Star* was that the first consideration in its design was the *user*. Problems of hardware and software were treated as secondary. As explained in Smith, Irby, Kimball, Verplank and Harslem (1982), "We have learned from *Star* the importance of formulating the fundamental concepts (the user's conceptual model) *before* software is written, rather than tacking on a user interface *afterward* [sic]." (page 246). Having a conceptual model enables the user to understand and interact with the system. The designers of the *Star* chose to provide the users with a model of the electronic 'world' in which they would be operating by way of an analogy with a real, concrete world with which they are already familiar. The world they chose was that of the office. At the centre of an office is usually a desk, and the screen display of the computer interface is described as the *desktop*. On this desktop are displayed icons representing objects such as documents, folders and in- and out-trays. Document icons may be opened to reveal their contents, which can be read from the screen. No analogy is exact, and in some cases it is where the analogy breaks down that it gives most information (Lakoff and Johnson, 1980). On the other hand weaknesses in the analogy can cause confusion. For instance, the paper by Smith et al. (op cit.) includes an interesting discussion on the impossibility of making some aspects of the analogy totally consistent, in this case with regard to the mechanism whereby a document is printed.

In the early 1980s a number of personal workstations appeared on the market, which incorporated some form of wimp interface. They included the Apollo, the Perq and the Sun. These are all powerful - and expensive - computers. It was with the release of the Apple Macintosh in 1984 that wimps began to reach the mass market. All software available for the Macintosh provides a standard wimp-based interface. Currently most other new micro - computers provide a mouse at least as an option and more software is being written on that expectation. There is no reason to suppose that this trend will not continue.

From the above description of wimp interfaces it should be clear that they are intensely visual and that adapting them for use by people who cannot see is a significant undertaking. The following chapter describes principles which were used as the starting point of such an adaptation.

Chapter 3

General Principles

3.1 Introduction

In Chapter 1 it was explained that there is a problem of making modern, wimp interfaces accessible to visually disabled people. Recall the proposition that it is preferable to adapt visual software whenever possible because this maximizes the amount of software made accessible while minimizing the development effort. So, in Chapter 2 the mechanisms used in visual interfaces were described, and now we are now in a position to discuss approaches to that adaptation problem.

The senses are what a computer scientist might describe as the body's input devices; they receive information from the environment which they pass on to the brain. There are two essential approaches to using technology to overcome a sensory disability: either one tries to compensate for the weakness in the affected sense so that it becomes powerful enough to be useful, or one by-passes that sense and translates those inputs which it would normally receive into a form which can be apprehended by one of the other senses. The latter approach must be used when the disability is so severe that the sense is non-existent, whereas the former approach will nearly always be applied if any of the sense's power remains at all.

In the case of visual disabilities this distinction is clear. A large proportion of the population have what amounts to a mild visual disability, but their weakness of vision can easily be alleviated by the use of spectacles so that they would not generally be considered to be disabled at all. Even people who are officially classed as being disabled in that they are

partially-sighted usually employ technology which enables them to make the most of whatever residual vision they have. Blind people must, however, use their other senses as a replacement for their sight.

Developing aids for partially sighted people is a very different enterprise from developing them for those who are totally blind. In the latter case the developer must exclusively exploit senses other than sight whereas in the former she must effectively consider *only* sight. This is because of the phenomenon which is observed among partially sighted people that they generally insist on using their residual vision as much as possible and resist using devices which do not expect them to use sight at all. (This phenomenon is mentioned again later in Chapter 6, in which the problems of evaluating a device are discussed). Since the majority of visually disabled people do have partial sight, it may be more useful practically to develop aids for their use. However, it is also more difficult, in that the variety of visual impairments means that the developer has a less well defined target population. For instance, it was mentioned earlier that magnifying the contents of a computer screen will help some partially sighted people, but actually make using it more difficult for others.

Within this project it was, therefore, decided to develop a device which could be used by someone with no useful vision. It was assumed that it would be necessary to communicate using their other senses and that these were of average power. This is a major decision since it significantly constrains the set of possible solutions.

Examination of the computer adaptations mentioned in Chapter 1 shows that in all cases of adaptation for use by totally blind people, it is the aural and haptic senses which are used instead of sight. In other words, visual information is translated into either an auditory or tactile form. It is the auditory approach which has been adopted in this project. This decision imposes a second major constraint on the form of the eventual product.

The first principle of adapting graphical interfaces is that visible objects should be replaced by audible ones. This alone is not, however, sufficient. The degree of complexity in

graphical interfaces is such that a second principle must be applied: that of constraining the auditory interface. In this chapter the two principles outlined above are firstly discussed in general terms, including quite a wide view of their relationship to work in other fields. Then the way these were applied within the case study word processor is described.

3.2 Visual-to-auditory translation

Sounds can be modulated to communicate information in a number of ways. A tone has a given frequency (or pitch), amplitude (or volume) and timbre. In addition, such a tone can only be sounded for a finite time and so it must have a duration. Using just these qualities, a single tone can be used to carry simple information. For instance, in a single tone a car horn delivers a very simple message. More complicated information can be transmitted by combining tones in different ways, using all of the above properties. In fact, the most complex medium of communication we use is speech - which uses all of those properties. Whereas, simple information can be transmitted in the form of single tones, complicated information can be transmitted in the complex form which we recognize as speech.

There is not, however, a direct relationship between complexity of sounds and information content. For example, morse code has a very simple format: two tones of the same pitch but different duration, and yet morse code can be used to communicate any information which can be written in natural language. On the other hand, an orchestra generates sounds whose complexity is comparable with that of speech. Certainly music is a form of communication, and yet it is a very imprecise medium. Music communicates more directly with the emotions than with the conscious mind. The important difference between speech and music is not to do with their complexity but rather with their function. Speech is our most important communication medium.

The rate of auditory communication is another interesting consideration. In the above example of the car horn, an important factor is that its message is urgent. The information content is small - merely that there is a potential danger of some sort - but content is

sacrificed in favour of speed of transmission. It would be possible to replace the horn by a loud hailer, in which case the information content of the message could be much increased, "Man in the blue jacket, I am about to run you over!", but the dangers of the loss of speed are obvious. Indeed, speech does appear to be a fairly slow medium. The brain can accept auditory information more quickly than normal speech. For one thing, most people read much more quickly than they talk. However, the limitation is not associated with hearing rather than reading information. This is proved by those visually disabled people who listen to 'talking books' on tape recorders which play back at a faster speed than the recording was made (usually with frequency adjustment so that the voice is at a normal pitch despite the speeding up).

One form of translation of visual to auditory information is very common: the reading out of text. There are a number of interesting features of this. There is a well-defined set of rules for this translation, but it is not without ambiguity. This is easily illustrated by observing any two productions of a play. The limitation is, of course, in the medium of written language which has not sufficient power to record all the subtleties of the playwright's intentions. Again, it is a slow method of transmission, slower than silent reading. However, with modern technology, it is a form of translation which can be successfully mechanized. So it is that several of the word processors which were mentioned in Chapter 2 were adapted by the addition of a facility to translate computer-generated text into synthetic speech. Some visually disabled people use machines which goes one step further than that - the translation of *printed* text into speech. This has been achieved very successfully by the Kurzweil Reading Machine (SurrIDGE, 1986, Kurzweil, 1986).

The limitations of text-to-speech translation should be clear. Firstly it is a slow form of transmission, and secondly it is difficult to translate if the original information is not in a textual form. Outside text-to-speech, the only existing research into the translation of visual information into an auditory form has been carried out in the context of the control of aircraft. Recent work has been carried out in two related areas. Firstly, auditory warnings

have long been used on aircraft as alarms but only latterly has anyone looked analytically at the design of such alarms. Secondly, the flying of an aircraft is a complex human task which relies on the pilot receiving and analysing large amounts of information. Up until now, nearly all that information has been transmitted visually - by the pilot *looking* at instruments. A question addressed by some researchers is whether a pilot might be able to function better if this one input channel were less loaded and the load shared with the auditory input (Hofmann and Heimstra, 1972, Vinje and Pitkin, 1972).

The use of alarm horns in aircraft really comes into the same category as the car horn. The intention is to transmit a simple but urgent message. Unfortunately, the information in conventional warning systems might be said to have been badly designed. There has been an assumption that because alarms are used to signal potentially dangerous error situations, they must be loud (large amplitude). At the same time little imagination has been applied to their temporal patterns. Most warnings have no temporal pattern, they are continuous. Those which do have a pattern are usually simple, interrupted horns. Modern commercial aircraft may have as many as sixteen different auditory warnings on their flight deck. However, there is evidence (Patterson, 1982) to suggest that it is very difficult to maintain perfect identification of such a large set.

Patterson has proposed a much more rational design for such auditory warning systems, which exploits the information capacity of the sounds used. Not all warnings are of equal seriousness and it is possible to convey this in the design of the auditory signals. Different temporal patterns have been found to evoke different levels of urgency (Edworthy and Patterson, 1985) and this can be exploited. Notice that here we are almost talking of music - short tunes - and of the emotional quality of urgency. Similarly, increased amplitude or volume will cause a signal to seem more urgent. Indeed, that is one reason conventional warnings are criticized, that their uniformly loud sounds encourage a startle reaction. The level of urgency of a warning can change - and its signal can be adjusted accordingly. For instance, a low-priority situation may become more dangerous if it is not remedied within a

certain time. In such a case the alarm should be sounded with increasing urgency until it has been correctly handled. To get over the problem of identifying different warnings, synthetic speech is added; a tone quickly gives warning that an error has occurred, followed by speech giving more information - but less quickly.

A few researchers have looked at the possibility of improving the human-machine interface by adding auditory output. Bly (1982) established that it is possible to receive quite complex information audibly. She encoded multi-dimensional data into sounds by controlling the following characteristics: frequency (pitch), amplitude (volume), duration, fundamental waveshape, attack envelope, fifth harmonic and ninth harmonic. It was concluded that "sound does convey information accurately and...can enhance graphic presentations." (op.cit., page ii).

Two groups of researchers have recently investigated the idea of creating auditory counterparts of visual icons as components of the user interface, to enhance its use by sighted people. However they suggest different approaches. The approach adopted by Sumikawa and her colleagues is related to that of Bly, in that they use sounds, encoded in a systematic - but artificial - way (Sumikawa, Blattner, Joy and Greenberg, 1986, and Sumikawa, Blattner and Greenberg, 1986). So-called *earcons* are constructed of *motives*, where "a motive is a brief succession of pitches arranged in such a way as to produce a tonal pattern sufficiently distinct to allow it to function as an individual recognizable entity," (Sumikawa, Blattner, Joy and Greenberg, 1986, page 5). In addition to varying the pitch and rhythm of motives, further information can be conveyed by varying their timbre, register (gross changes of pitch) and dynamics. Simple information is conveyed in simple motives: the fact that a system is closing down might be represented by a single note, the amplitude of which diminishes. More complex information can be communicated by combining motives. Families of earcons can be constructed. For example, there might be a family of error messages, which all share the same initial motive. Operating system errors would be signalled by compound motives which start with the error motive, followed by

another different one. That second motive would be quite distinct from the second component of a motive which would signal an execution error. Further levels can be added to achieve the required complexity. This work is still in an early stage and there has yet to be an interface implemented to test these ideas.

The form of encoding described above can be said to be a symbolic mapping between data and its representation. A less symbolic approach is to use sounds which depend on the physics of the items they represent, as suggested by Gaver (1987). For instance, the arrival of a message in an electronic mail system might be signalled by a noise resembling a package falling on a mat. This would seem to be an attractive approach in that it has some of the power of using visual icons which resemble the entity which they represent and which a user can comprehend with a minimum of learning. Furthermore, although Gaver's work is concerned with the enhancement of visual interfaces, it may be that such direct associations between operations and sounds will be very powerful for visually disabled people, who are accustomed to relying heavily on auditory cues. Once again, however, Gaver's work is somewhat speculative since it has yet to be applied in the implementation of a computer interface.

Another project developed methods of presenting data audibly to blind people. This was intended as a means of describing chemical spectra to blind students (Lunney and Morrison, 1981). Such spectra are normally represented visually, and by inspecting them it is possible to recognize patterns which indicate the presence of particular chemicals. In this way it is possible to analyze the components of a sample. For blind students, however, the spectra are encoded into 96 units which are assigned musical notes. By listening to the resulting notes - both separately and as chords - it is possible to recognize corresponding patterns, and so analyze the sample.

Among the senses, sight has a unique ability to receive information from multiple sources simultaneously. Even when our central vision is focussed on one point, we are still aware

of other areas - as we say, "out of the corner of our eyes". Furthermore, the eyes have an ability to shift that central focus very quickly and accurately. To all intents and purposes, therefore, sight can be said to receive information from multiple sources in parallel. In this way it is possible to distribute a large amount of information over a computer screen and rely on the user to be able to visually select the appropriate part of that information at any time.

By contrast, the other senses are much more attuned to dealing with information in a serial manner. It is true that hearing has some of the parallel capabilities of sight, but these are less powerful. For instance, the "corner of the eye" phenomenon has an analogue, illustrated by the situation in which a person can be engaged in a conversation in a noisy party, but hear their own name spoken across the room. Also, in listening to music, a person is undoubtedly receiving information in parallel. What is lacking, however, is the ability to focus quickly and accurately on any one area of that input.

While hearing undoubtedly does have some capacity for receiving information from more than one source simultaneously, it does have its limitations and it is not clear what the nature of these limitations is. For instance, it may be possible for a listener to follow two different conversations simultaneously, but that may be confined to listening to speech, in which there is a very high degree of redundancy. Music embodies a different form of auditory input. In listening to say an opera, a person is receiving sounds from a vast number of sources, and most people could distinguish the singing of the soprano from the sounds of the orchestra. There may be scope then for exploiting such gross differences in sounds, but more subtle differences may be less useful. For example, most people will perceive the difference between two single notes played separately and between each of those notes and the two played together as a chord. However, many people could not hear a two-note chord and identify identify the two components (Egger & Ivinshis, 1969). Clearly there is the possibility that sounds could be used in some way to communicate information in a parallel form, analogously to sight, but there is a need for further investigation into the mechanisms which might be applied and their limitations.

Hearing does not have the precise spatial acuity which sight has. People often make errors in locating sounds, to the extent that they confuse a sound source which is in front of them as being located behind them (Mills, 1958). Excluding such gross errors, it has been shown by Stevens and Newman (1936) that people make errors of between 4° and 16°, depending on the angle from the median plane. Clearly much greater errors than would be made by someone locating an object visually.

The selectivity of sight makes it possible to simultaneously display a very large number of objects on a computer screen. It is left to the user to choose on which objects to concentrate. If visual objects were replaced by auditory ones which emitted sounds as long as they were on the screen, the user might become lost, swamped in an excess of noise.

Most of the devices mentioned in Chapter 2, to aid visually disabled people rely on the fact that written text, although visual, is not essentially parallel in nature. This means it can be converted into (synthetic) speech or tactile sensations to be 'read' by a single finger. The vital difference which wimp interfaces introduce is that the parallel nature of visual information is exploited. For instance, multiple menus may be available on screen as readily accessible cues to prompt the user, clicking the mouse in one area of the screen may have an effect which is visible on another part (e.g. deleting text from a document).

Another problem of translating non-textual visual information into an auditory form is that symbols can convey a lot of information. Whereas this amount of information can be taken in 'at a glance', there is no way of transmitting it quickly in an auditory form. In general, complex information can only be expressed as speech - which is slow. Within wimp programs the most complex example is the icon. Recall Figure 2.5, which illustrated a typical icon, which contained five items of information which can be assimilated by a sighted person literally 'at a glance'. Furthermore, it is easy for a person to extract as much information as they need from an icon, and no more. For instance, in one situation the user may need to know only that a disc is not empty, as designated by the mere presence of any

icon. In another case she may be looking for documents, which will entail a slightly more careful look at the icon, but one which would take a negligible time. In this way the user can take in as much of the information as she needs, but always very quickly. If an icon is to be represented audibly, to somehow replace it with the speech, "Document *letter*, created by *Qwriter*, accessible." would be somewhat long-winded, but what simple mechanism could there be to 'filter out' the unwanted information?

Another problem arises in trying to point at objects on a screen with a mouse. To do so visually is a question of application of visual-motor coordination. The person receives continuous feedback as to the progress of the cursor towards the target and can control their movements of the mouse accordingly. It would be possible to implement an auditory analogue of this interaction, whereby a tone would be sounded which would vary in some way according to proximity to a target. It might be, for instance, that its volume (amplitude) would be decreased in proportion to the closeness to the target, such that the tone would be silenced completely once the target is 'hit'. This kind of signal is, again, used in aircraft, to lead pilots into guiding radio beams. The obstacle to applying such an approach to adaptation of wimp screens is an extension of the selection problem. At any time the user may want to point at any one of the many objects on a screen. If the screen is to be represented in the sort of auditory fashion outlined, how is the user to specify which object is to be the target at any time?

In aiming the mouse on a screen, visual feedback is most important, and users tend to ignore kinaesthetic information from the hand and arm muscles. However, this can become much more important to a person who is not receiving the visual information. It is significant to know that the mouse is being moved to the right, or forwards away from the body, for instance.

If the user has a mental map of the relative positions of objects and a means of detecting the boundaries between objects, then she can navigate around them with the aid of kinaesthetic

input. She will need to also know their starting position, and in fact ought to have a mechanism whereby she can check her current location at any time, in case she makes any errors. Ensuring that the objects are always arranged in a consistent, regular pattern will aid the user to build the necessary mental map. This is the basis of the approach adopted in this project.

3.3 Constraining the interface

As mentioned above, two important design decisions were made at an early stage of this project: that the resultant product would be designed to be usable by people who are totally blind and that auditory information would be used to replace visual information. These decisions imposed constraints on the design of the product. However, these implied constraints are not sufficient to specify a usable interface; just any form of visual-to-auditory translation will not do. Before a design was formulated it was therefore necessary to consider other limitations which should be imposed as principles in designing the interface.

Broadly speaking, vision is better able to process information from different sources which are in complex relationships spatially and temporally. This implies that it is not sufficient to simply transform the behaviour of a visual program into sounds; restrictions must be applied to the interface so that the user will be able to process the information received aurally.

The screen of a wimp-based interface represents a three-dimensional space. Windows usually can overlap each other. They may even totally hide each other. Menus will generally pull down or pop up on top of other screen contents. The relationships may be complex and dynamic. For example, clicking the mouse within a window generally causes it to be brought to the 'front' of the screen, which may involve a major re-arrangement of all the windows on the screen. However, this third dimension generally does not cause the sighted user significant problems. Visually, it is generally quite straight-forward to see the relationships on the screen. Perceptually, the sighted user has a well-developed spatial ability. Indeed, wimp interfaces are mostly designed on the basis of the desktop metaphor

which exploits the fact that the user will have had 'practice' at shuffling pieces of paper in piles on a desk.

The level of complexity involved is illustrated by Figure 3.1. The four parts of Figure 3.1 illustrate two windows on a screen in four different possible configurations. In 3.1A, the windows do not overlap, their contents are entirely visible. In 3.1B window 1 partially overlaps window 2. That means that all of the contents of window 1 are still visible, but some of window 2 is obscured. Similarly in 3.1C all the contents of window 1 are visible but it is hiding some of window 2, but in this case the spatial relationship is different since window 1 is entirely within the area of window 2. In 3.1D the assumption is that window 2 now entirely obscures window 1, although it is not possible to know by visual means alone that window 1 exists at all. Note that it is possible to distinguish these four very different situations in Figure 3.1 literally 'at a glance'.

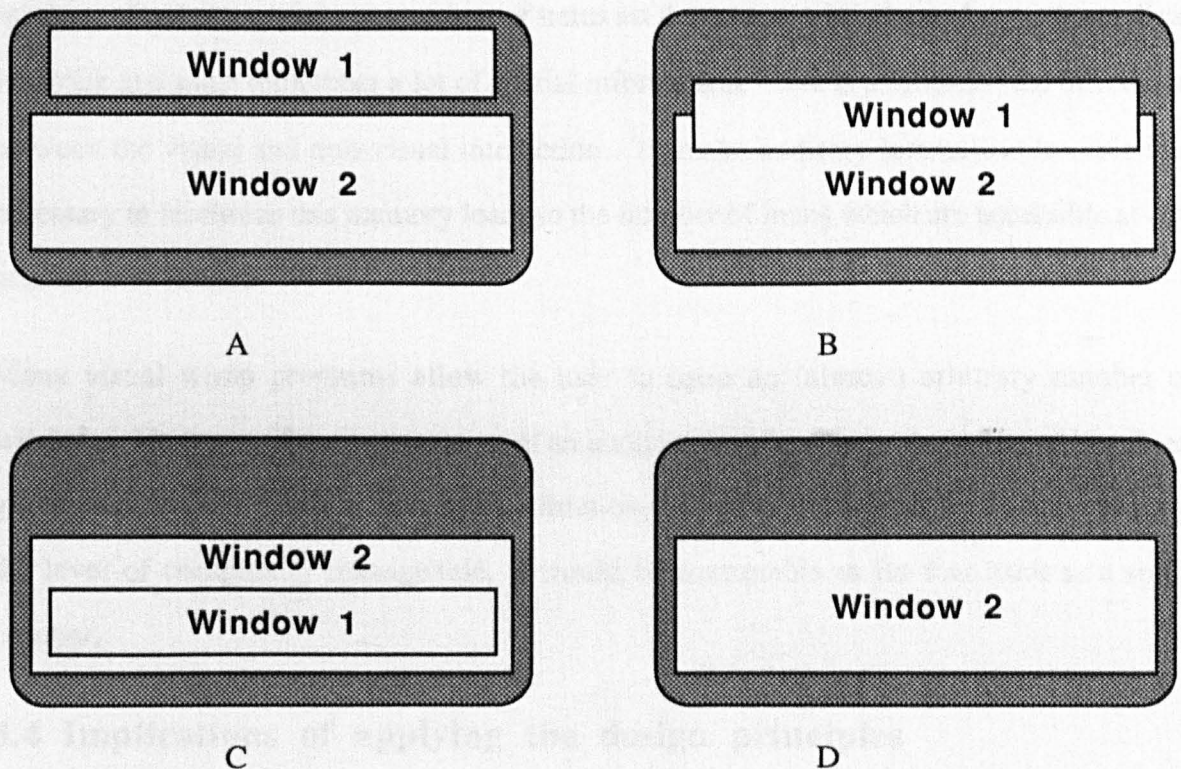


Figure 3.1 Four different possible configurations of overlapping windows.

There is no simple way that such complex information could be communicated quickly aurally. Different sounds might be used to represent windows 1 and 2, but how could those sounds be combined to represent the complex and changing relationships described above? The principle of visual-to-auditory translation effectively creates an auditory space, the constraint must be added that this will be a two-dimensional space. In other words, objects in this space cannot overlap.

The situation in Figure 3.1D represents an interesting case. As mentioned above, it is not possible to know from the visual information presented that window 1 exists at all. In fact it is up to the user to remember that window 1 is in existence. In most other cases there is no necessity for the user to remember the relationships between objects since they are immediately obvious visually. In fact those relationships may change as a result of interactions between objects. However, whereas the sighted user has no need to remember anything about the spatial arrangement of items on the screen, a blind user has no immediate feedback and must remember a lot of spatial information. This is a fundamental difference between the visual and non-visual interaction. To make auditory interaction feasible it is necessary to minimize this memory load, so the number of items which are accessible at any time must be limited.

Many visual wimp programs allow the user to open an (almost) arbitrary number of windows. However, if the components of an auditory interface are to be arranged in a fixed grid layout, there is going to have to be a limit on the number of windows. In order to keep the level of complexity manageable, it would be reasonable to fix that limit at a small number.

3.4 Implications of applying the design principles

The description of visual wimp interfaces in Chapter 2 showed that the problem tackled by this project is a large one. In order to manage the complexity of the solution it is necessary

to impose some constraints on it at an early stage of its development. To some extent these constraints will always be arbitrary, since their full effects cannot be appreciated until a later time when the development has begun to mature. This chapter has described the major principles which were adopted in this project: the use of visual-to-auditory translation and the constraining of the interface. The implications of these decisions will be seen in subsequent chapters, but before that it is worth considering alternative approaches which might have been adopted.

Different forms of auditory output might have been chosen. For a word processing program it would be impossible to avoid using speech in a purely auditory interface, but it is possible to use *only* speech. Programs have been developed for blind users which rely entirely on synthetic speech for their output. The Vincent Workstation (Vincent, 1986) and Audiocalc (Jennings, 1985) are examples.

Auditory output need not have been used, but replaced by a tactile form. An obvious approach would be to incorporate a device similar to the Optacon into the interface, whereby the user receives a tactile outline of the same shape as the visual form. This might be linked to the mouse, so that the user would be able to examine the area of the screen surrounding the mouse pointer.

Both auditory and tactile output might be linked. Another form of tactile output would be to translate text into braille. (The braille would be displayed on a refreshable 'soft' display, rather than permanently embossed paper). This could be used in the place of synthetic speech in the above proposal, with tones being used as above. In other words, spatial information would be encoded as tones, but more complex information would be presented in braille.

Chapter 4

Design Principles for a Word Processor

4.1 Introduction

Chapter 3 outlined principles which might be applied to the adaptation of visual interfaces for visually disabled users. In order to test the applicability of those principles there was a need to find a vehicle by which they could be tested. It was decided that this should be done by developing a piece of application software which could be evaluated by visually disabled subjects. In this way it would be possible to get a fairly clear idea whether the approach was likely to be useful in realistic use. This chapter describes the design of that software. Chapter 5 describes the implementation of the word processor, and to what extent it deviated from the ideas of the design described here.

4.2 Principles

A word processor was chosen as the particular application to be designed and implemented. It was named *Soundtrack*. That particular kind of program was chosen because it would be the easiest type of software to evaluate. As mentioned earlier, there are already a large number of visually disabled people who use word processors. Such people already have a level of knowledge and understanding about computers and word processing so that only the aspects of *Soundtrack* which were under evaluation would be new to them. So, for instance, they would already have a clear idea as to what it means to delete a word from a computer document, but would have to learn how to do that with a mouse. Testing a complete piece of application software was also more motivating for subjects; they could

clearly see the point of the work, which they might not if they had been asked to carry out more abstract exercises.

There are some disadvantages to testing the principles in this way. One is that some specific questions will not be answered. For example, described below is the pattern of sound tones which were used in Soundtrack to designate the objects pointed at by the mouse. It might be that there are better patterns which could be used. Abstract tests in which the performance of a number of subjects was measured when different patterns were presented could be used to investigate this. However, such variations could not be included in a program with which a person was expected to learn to perform complex operations.

Another disadvantage is that in evaluating the word processor, the subjects would naturally think in terms of that particular application. However, the real objective of this project is *not* to develop a better word processor, but to test some more general principles about the adaptation of software. The way in which any such bias was reduced is described in Chapter 6, on the evaluation.

Since this project aimed to investigate the *adaptation* of visual software, it was important that the auditory interface should have all the functionality of a similar visual program. The design therefore attempted as far as possible to provide *direct* analogues of the visual interaction.

As mentioned above, the approach to adaptation chosen was based on two principles: the replacement of visual information by auditory information and the constraining of the resultant interface. All interactions which a sighted person would see between the mouse cursor and objects on the screen would be replaced by sounds. Thus, *auditory objects* are defined as an interaction between the mouse and program which can be *heard*. The sounds heard take two forms: musical tones and synthetic speech. However, some of the interactions which a sighted person would normally undertake are too complex to translate directly into an auditory analogue, usually because of the serial nature of auditory input as

opposed to the parallel capacity of sight. This means that it is necessary to constrain the auditory interface. This section describes how these principles are expanded into the design of the user interface.

An auditory object is defined by the following properties:

a spatial location,

a tone,

an action,

a name.

It was decided that their spatial arrangement should be based on a grid pattern. This satisfies the constraint that objects should not overlap. It also implies limitations on the design of objects. The number of objects contained in a grid is the product of the number of rows and columns. Any number of items can be arranged in a grid which consists of a single row (or column). However, if the designer decides that there should be more than one row (or column) then the number of objects must not be prime, in fact it must be a multiple of the rows (or columns). This means that if the application dictates that an object should contain a non-prime number of subsidiary objects the designer has three options: introduce dummy objects to make the total number non-prime, arrange them in a single column or use a layout which does not strictly conform to a grid. These three options are illustrated in Figure 4.1. Later on it will be shown that both of the latter two of these approaches were applied in this project. This is discussed further below.

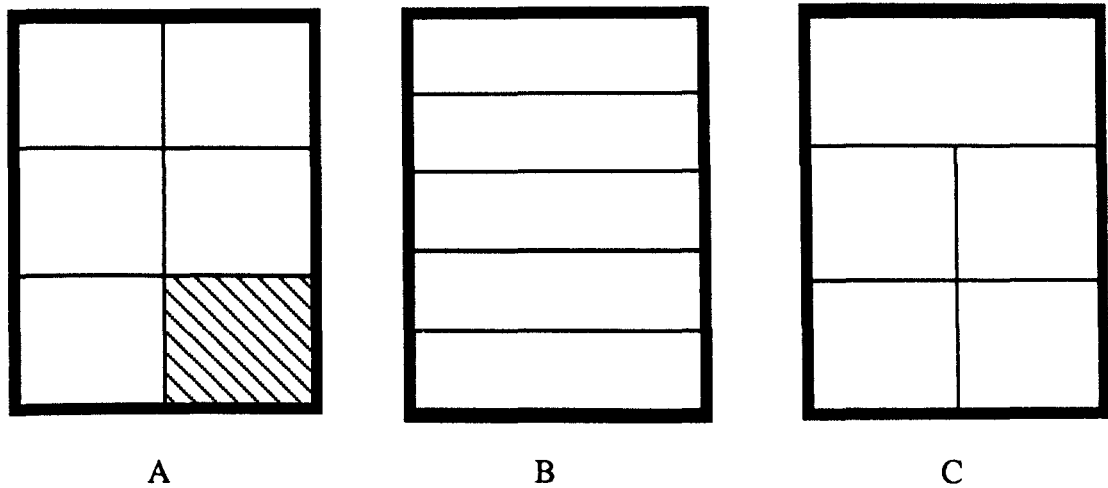


Figure 4.1. Three ways of fitting a prime number (5) of objects into a grid. In version A the shaded box represents a dummy object. Version B uses a single-column grid and version C is not a strict grid.

Interactions are defined between the mouse and the above-mentioned properties of objects. An object's *tone* is sounded whenever the mouse moves to point at it. This is a means of communicating the mouse's position quickly to the user. The *action* is the role of the object within the program. By pressing a mouse button or buttons, the user can signal that the action of the current object is to be *executed*. Generally speaking, the object's tone should also be sounded when it is executed, as a means of confirming to the user that this has occurred. The object's *name* describes its action. The user must be able to elicit synthetic speech to tell her the name of the object currently being pointed at. The way the user specifies that the name should be spoken or action executed is by way of the mouse buttons. The mouse button protocol used will depend on the hardware. For instance, on a mouse with more than one button, one of them can be used as a *speak* button, while a second would be the *execute* one.

4.3 The design of the auditory screen

By analogy with the visual interface, the area in which the mouse moves is known as the *auditory screen*. This screen is divided into a grid of (primary) objects which are similarly

named *auditory windows*. Auditory windows differ from their visual counterparts in a number of ways. For instance, most visual wimp programs allow the size of windows to be altered and for them to be moved around and made to overlap each other. The auditory windows are arranged in a rigid grid; they cannot be moved or have their size altered. A second constraint of the design is that there is a fixed number of these auditory windows. This implies that there is a limit on the number of documents which a user can open - depending on how many of the windows are allocated as document windows. This contrasts with visual programs which usually allow the user to open a large number of windows. The *role* of any one window never changes - although its contents may do.

The auditory screen and its contents can be represented entirely as sounds. The screen is delimited by a tone, such that as long as the mouse is off the screen this tone will be generated. Within the screen the structure is that of the grid of windows. Figure 4.2 is a visual representation of an arbitrary auditory screen.

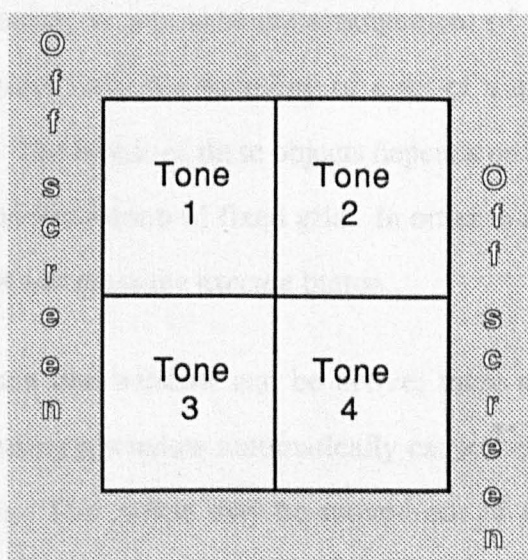


Figure 4.2. A visual representation of an arbitrary auditory screen, which consists of four windows, each of which has an associated tone.

If the mouse is in the off-screen area a particular tone will sound. Within the screen the tones labelled in Figure 4.2 will be sounded as the mouse crosses the border into a tone's area - in other words, into a window.

Although it has been stated that the auditory screen can be represented entirely as sounds, visual feedback may be given also. For example, the windows may be drawn on the screen and text in documents may be displayed. This information may be of some use to partially sighted users. Also, for blind users it may be useful to be able to get assistance from sighted colleagues who can see what is happening on the screen.

Auditory windows are used to represent composite screen objects, such as menus and (visual) windows. An auditory window is treated as an object in itself when it is in an *inactive* state. That is to say that if all the windows are inactive and the user moves the mouse around the screen, then the single tones corresponding to each of the windows will be sounded. In order to interact with the objects contained within a window it must be in an *active* state. Once a window is activated the arrangement of objects within it becomes apparent, as can be detected from the sounding of a set of tones as the mouse is moved within that window area. The layout of these objects depends on the identity of the window, but will again be based on some form of fixed grid. In order to activate a window the user must point the mouse at it and press the *execute* button.

At any time no more than one window can be active; there are situations in which no window is active. Activating a window automatically causes the previously active one (if any) to become inactive. The mouse may be moved out of an active window and the interaction with the remaining, inactive windows is unaltered. Indeed, this is the mechanism by which the identity of the active window is altered: the mouse is moved from the active window to another one and the appropriate button is then pressed. However, as the mouse crosses out of the active window an additional, distinctive tone is sounded to signal the fact. Conversely, should the mouse be moved out of the active window and then returned to it

while it is still active (that is to say that no other window was activated) another distinctive sound is heard.

Activation is illustrated in Figure 4.3. The window previously associated with tone 4 has been activated. The structure of that window is now apparent as two tones and should the mouse be moved out of the active window, across one of the thicker lines, another tone will sound.

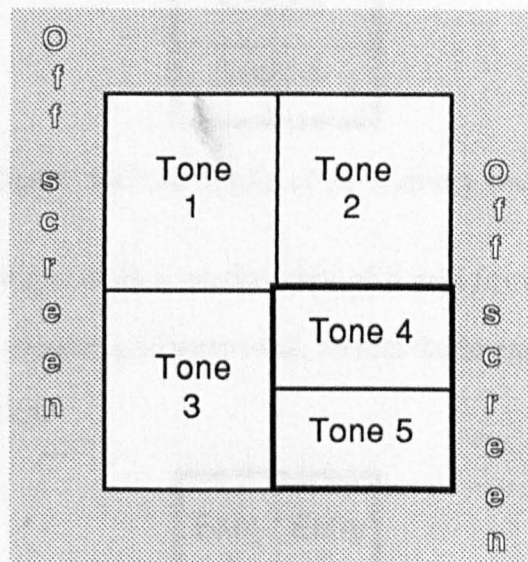


Figure 4.3. A screen with an active window.

The activation concept facilitates the chunking of items in memory. Related screen objects are grouped together so that they can be treated as a chunk. Psychological tests suggest that the capacity of short-term memory is no more than nine chunks (Miller, 1956), which would suggest that the maximum components in a screen object (windows in a screen or objects in a window) should be of that order. It is worth emphasizing that such memory limitations are much more significant than in a visual program because the user has immediate cues available on the screen.

4.4 The design of menus

The layout of auditory menus is closely based on that of their visual counterpart. This simplifies the adaptation of visual software. Objects are arranged in a single column, with the vertical ordering of entries maintained as in the visual menu, as in Figure 4.4.

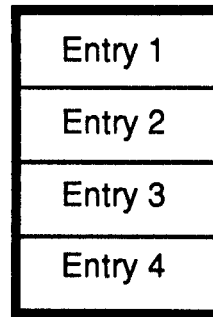


Figure 4.4 The layout of an auditory menu.

This layout might be thought of as a special case of a grid format. From some aspects it might be better if a multi-column grid were used, so that the menu illustrated above might be represented as in Figure 4.5.

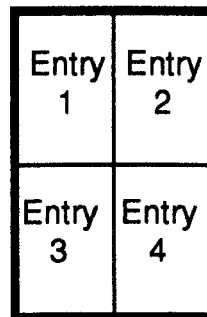


Figure 4.5. An alternative, grid format for menus.

This would maintain consistency with other windows. It would also make more efficient use of the area of the window. For instance, if a menu has a large number of entries the height per entry in the single column format would become very small, as in Figure 4.6, making precise selection difficult.

Entry 1
Entry 2
Entry 3
Entry 4
Entry 5
Entry 6
Entry 7
Entry 8

Figure 4.6. A single-column menu with eight entries.

In such a case, a multi-column layout, as in Figure 4.7, might be easier to use since the entries would not be as narrow.

Entry 1	Entry 2
Entry 3	Entry 4
Entry 5	Entry 6
Entry 7	Entry 8

Figure 4.7. An eight-entry menu with a two-column layout.

However, multi-column formats were not used for menus for a number of reasons. Firstly, as mentioned above, maintaining congruency with the visual menu format should make adaptation more straight-forward. Secondly, in an adapted program the number of entries within a menu would be dictated by the original, unadapted program and there would be problems if the number of entries was prime, as discussed earlier. Finally, the consistency of all menus having the same, single-column format makes it easier for users to remember the layout of different menus.

Control-key alternatives (as described in Section 2.2) can be provided for frequently-used menu commands.

4.5 The design of dialogues

Auditory dialogues must have the same functionality as their visual counterparts. For the most part this presents no problem, because most dialogues consist mainly of buttons. Thus, designing an auditory dialogue generally involves arranging the necessary buttons into a grid layout. In addition to buttons, however, dialogues include a message describing the role of the dialogue (*Save file as...?*, *Open which file?* etc). The auditory version must include an object which corresponds to this message. The user should be able to get the message repeated as required, by pressing the *speak* button on that object.

In addition to simple buttons, some dialogues contain lists - out of which file names may be selected, for instance. The number of entries in such a list will not be fixed, depending for example on the number of files on a disc. It was decided that such lists should be embodied by three objects. One of the objects is effectively a frame in which *one* of the list entries can be 'displayed'. This differs from the list frame in a visual dialogue in which several names are displayed simultaneously. The current name in the frame can be heard by pressing the *speak* button within that object. The other two list objects are buttons controlling movement of the frame up and down the list. Scrolling through list entries one at a time can be a slow process, therefore a facility is provided whereby the user can type the appropriate name whenever the dialogue is active. This means that if the user remembers the name of the file she can just type it, otherwise she must scroll through the list until she finds the name she wants.

As mentioned earlier, there are two forms of visual dialogues: *modal* and *modeless*. Both forms can be realized in an auditory form. Once a modal dialogue has been opened the user is forced to interact with it; the mouse cannot be used with any other objects on the screen. Of course, the dialogue should always contain a *Cancel* button through which the user can close the dialogue and return to using other objects which can be used in case of an error. That means in the auditory version that the mouse can be moved to other windows and their identity checked by tone or name, but only the dialogue window can be activated. Modeless

dialogues behave more like windows. The user only interacts with the dialogue after having activated it and it remains open generally until the user explicitly closes it.

4.6 The design of alerts

Visual programs have a further category of object - the *alert*. The operation of the alert is quite similar to that of the dialogue. The difference is that an alert is not opened by a user action, but by the program in the event of an error occurring. Such errors usually have causes external to the application program, such as the computer running out of disc space during a file saving operation. All alerts are modal in interaction, to ensure that the user is aware that an error has occurred before she continues to use the program. Some alerts offer the user options as to what action to take in response to the error; others have a single 'continue' button which the user should press having read the message in the alert.

There is another type of error, which is less serious in nature, and internal to the application program, caused by the user. These are usually signalled by the sounding of a tone. It is then up to the user to interpret what they have done wrong - often by inspection of the screen. An example of such an error would be to attempt to type when there is no document window open. One feature of wimp programs is that the potential for such errors can be reduced by the program managing commands such that they are enabled only when appropriate. For instance, if there is a limit on the number of document windows which can be opened, the *Open* command should be disabled when that limit has been reached.

Alerts work much in the same way as dialogues in visual interfaces. However, since interpretation of a mistake can be more difficult for someone who cannot see what is happening on a screen, the (auditory) alert can be extended to allow a modeless form of interaction. Two categories of error can be distinguished. There are those which are due to faults in the system and those which the user makes. The former group would include events such as a disc becoming full or an attempt to open a non-existent file. These are not caused directly by the user, but are serious and cannot be ignored. They would cause the

opening of a modal alert in a visual program. The second type of error includes actions such as the user typing when a document window is not active. The user will normally be able to diagnose these errors quickly without help and in most cases ignoring them will not have catastrophic effects. An error tone would still be used as a means of warning of errors from either category. At the same time an alert would be opened. If the error is a serious one, from the first category, then the alert will be a modal one, forcing the user to activate the alert and to execute a 'continue' object before they could carry on. For errors from the second category the alert would be a modeless one, giving the user the option of activating the alert and obtaining a spoken description of the error, or ignoring it. Should the user choose to ignore the error, she need not activate the alert, but do some other action and the alert will be automatically closed.

4.7 The Design of the document window

As has been described above, synthetic speech is used as a cue to aid users in navigating with the mouse. In addition, since the program is a word processor, speech is used to communicate the contents of documents being edited. Text is 'displayed' in auditory document windows although, for reasons explained below, this is one window which is *not* a direct analogue of its visual counterpart.

A piece of written text has two forms of structure: spatial and grammatical. In narrative, non-artistic texts the two are largely independent. Exceptions are that paragraphs (which are grammatical units) are delimited by spatial cues (the inclusion of a blank line and/or the indenting of the beginning of a line), and that headings are separated spatially from text. The ease of reading and understanding of this kind of text can be aided by its spatial design, but it does not have any effect on its fundamental information content. To a large extent the layout depends on factors such as the size of paper used and the length of the words chosen by the writer - which are irrelevant to the semantics of the piece. To some extent the design of a lot of word processors implicitly acknowledge this fact in that the user does not have to

manually break lines at the edge of the page. The program will do this automatically, but in such a way that the format is not fixed and can be readjusted if the writer makes changes to the text. The writer is obliged to explicitly insert new lines only to mark the end of paragraphs and headings. A writer may change the semantic structure of a piece of writing, but the computer can take care independently of the spatial structure. Such programs, therefore, may treat any text enclosed by blank lines as a paragraph. The fact that such a definition will in fact encompass headings and treat them as paragraphs is unlikely to cause the user serious confusion.

None of the designers of word processors seems to have paid particular attention to what internal representation the user will have of the text on which they are working. In fact, it seems likely that different representations will be used, depending on the role of the user in relation to the text. For instance, a writer who is composing on the word processor will be intimately concerned with its semantic content and will envision the text in these terms. On the other hand, a secretary who is given a copy of a document, marked up with corrections, is more likely to think of it in spatial terms. She might, for example, look for the first word, on the third line of the second page, and over-type it with a substitute, without even thinking about what the word means, far less its role in its containing sentence.

Although interaction in most word processors is based upon the spatial form of the text - as characters arranged in lines, there are programs now available which are based on different structures. Programs such as *More* (manufactured by Living Videotext Inc.) could be described as *idea processors* which allow the user to structure text in topics and sub-topics. Within computer program writing the syntax of the text is strictly defined, such that editors can be based upon the syntactic structure of the text - see Teitelbaum (1981) for an example.

However, the majority of current word processors are largely spatially oriented. Within such programs, the cursor is moved along and up and down lines - regardless of how it is controlled, be it by keys or mouse or other device. The only facility provided by most word

processors which might be said to relate to semantic features of the text is string-searching. The emphasis on spatial properties is not generally a problem for writers, presumably because they do think partly of text in spatial terms and partly because writing was always thus, and they have learned to cope - even when they are thinking of the text in semantic terms. If a writer has sight, the spatial structure of a piece of writing is easy to observe and appreciate. At the same time visually disabled writers (braillists, typists, word processor operators etc) seem to have also learned to cope, despite the fact that it may be harder for them to appreciate the spatial structure of a text. The display of a piece of text on a word processor may be extremely dynamic and yet visually disabled people do use adapted visual software which retains these potentially confusing features. An example of how confusing this can be is that the typing of a single letter may cause the contents of a screen to alter completely - because it happened to overflow the last line on the screen, so causing a 'scroll' to the next screen.

All word processors have some facility for moving markers within a piece of text and the position of the markers determine where actions will occur. Conventional, key-driven word processors generally have keys which will move the markers in units of characters, lines, screens and words (words being one concession to the syntactic structure). Visual wimp programs allow the user to move the markers more directly, via the mouse. Editing commands act on the *selected* portion of the text, and the selection is made by pointing at the text on the screen. An informal experiment with implementing a direct analogue of such a form of pointing and selection was carried out. Tones were sounded as the cursor was moved horizontally across characters and vertically over lines. If the appropriate options were active, characters or words could be spoken as the cursor pointed at them. The trial suggested that this was not a viable approach; the feedback on the mouse movements was too slow and too difficult to interpret.

It was decided, therefore, that the document window would have to be implemented such that it was not a direct analogue of the visual window. Instead it was based on giving the

user the facility to manipulate the text in terms of its syntactic structure. The selection concept is retained, but text is selected in syntactic units. The document window consists of a set of buttons and controls which manipulate the selection. Whereas visual programs specify the current selection by displaying in inverse video, in an auditory program the current selection should be *spoken* whenever it is altered (or if the user specifically so requests). The syntactic units used are:

- the whole document,
- the paragraph,
- the sentence,
- the word,
- the character,
- the point between two characters.

Each document has a selection *level* associated with it, corresponding to the six syntactic units. Manipulations of the selection are based upon the current level. For instance, if the current level is **word** the user can move the selection forward - to the next word - and so on. The document window consists of controls which can be used to alter the current selection level and to move it forwards and backwards through the document.

All editing commands act on the current selection. In addition, any typing on the keyboard will replace the selection. For instance, in the sample below the letter *a* is selected:

The cat **sa**t on the mat.

If an *i* is typed it will replace the *a*, changing the word to *sit* :

The cat **si**t on the mat.

Similarly, a whole word may be selected:

The **sa**t sit on the mat.

Then as soon as the user presses a key the word will be deleted (in this case the *d* is pressed):

The d sit on the mat.

The user can then continue to type letters which will follow the *d* and in this way replace the whole word:

The dogs sit on the mat.

There will often be situations in which the user will want to insert text - without deleting anything. This is when the **point** selection level is used. The user sets the level to **point** and moves the selection in between the two characters which will surround the insertion. So, to change *mat* to *mats* the selection is moved to the point between the *t* and the fullstop:

The dogs sit on the mat|.

and the *s* is typed:

The dogs sit on the mats.

The one form of visual interaction which was not tackled in this project was the use of icons. There are a number of problems in devising a suitable auditory representation for icon objects. Firstly, icons generally can be moved freely around the screen, although some implementations do restrict their movement to some extent (see Smith, Irby, Kimball, Verplank and Harslem, 1982, on the Star interface). Such flexibility does not confuse sighted users because they will generally have information immediately available as to the current position of objects - simply by scanning the screen visually. However, it is likely to be very difficult for a visually disabled person to cope with objects whose position is not fixed. The memory load on the person would be increased in that they would have to attempt to remember where they 'left' objects. In case they should not succeed in so doing always there would have to be some facility for relocating 'lost' objects, corresponding to a sighted user scanning the screen with their eyes. Any facility based directly on the auditory protocol of tones and spoken object names, described above, would be likely to be tedious and difficult to use. A second complication of icon objects arises from the problem discussed in Section 2.2, that an icon can convey a lot of information, which cannot be easily translated into sounds. Recall that, for instance, an icon representing a file can convey:

the file's name;

the type of the file (program, data etc.);

which program created it;

whether it is currently accessible;

whether it is currently selected.

For these reasons no attempt was made in this project to implement an auditory icon object. However, see Section 3.2 for a discussion of possible designs of auditory icons. A simpler moveable object was included as an experiment as part of the Scroll bar, and possible designs which might incorporate icons are discussed in Section 11.6.

4.8 Summary of the design

This chapter has described the design of an auditory word processor. However, to some extent that design is idealized and could not be implemented within the constraints of a real program on a microcomputer. Chapter 5 therefore describes the program as it was implemented and the extent to which it deviates from the design described herein.

Chapter 5

Implementation: Soundtrack

5.1 A case study: introduction

The design of an auditory word processor described in Chapter 4 is to some extent idealized. Within this project there were constraints imposed on what could be realistically implemented. Some of the restrictions were imposed on the software by its role within this research project, while others were practical considerations such as the hardware available and programming limitations. It must be made clear that Soundtrack was designed as a test-bed to try out certain ideas; it was not intended to be a production word processor. It was a means of testing how well people could use particular forms of interaction; the novel feature of the computer used is the mouse, and the design was such that the user was obliged to use the mouse. As will be illustrated below, there are interactions in which it might be easier for the user to type information and commands, but the design of Soundtrack did not allow this.

5.2 Hardware and software

Soundtrack was implemented on an Apple Macintosh microcomputer. The configuration used included 512 Kbytes of ram and two 400-Kbyte, single-sided floppy discs. This computer includes, as standard, a mouse pointing device. Indeed, the machine was chosen since it was the first such computer available within a similar price range to non-wimp microcomputers. The mouse includes a single button. The details of the Macintosh hardware are given in Apple Computer Inc. (1986). The only (minor) modification to the hardware was that braille labels were affixed to the 'home' keys on the keyboard (*F*, *J* and

6) to assist users with placing their hands. (More recent versions of the Macintosh keyboard have the home keys marked by tactile dots as standard).

One piece of non-standard hardware which was used during part of the testing of Soundtrack was the Summagraphics MacTablet. This is a bitpad which can be used in place of the mouse. It consists of a rectangular tablet and a stylus pointer. Movements of the stylus over the surface of the tablet are followed by the cursor in the same manner as it follows the mouse. The stylus has a button corresponding to that on the mouse, which can be operated by the index finger. Additionally, the stylus nib is spring-loaded and pressing it has the same effect as pressing the mouse button.

There is an important difference in the way that the bitpad operates compared to the mouse, which is that all movements of the stylus are *absolute* whereas mouse movements are *relative*. That is to say that a point on the tablet corresponds to one point on the screen; if the stylus is moved to the top lefthand corner, for instance, the cursor moves to the corresponding area of the screen. Even if the stylus is lifted from the surface of the tablet and moved to another point the cursor will jump to the new position on the screen; the tablet constitutes a global frame of reference for all motions. In contrast, the mouse detects movements only as long as the mouse is in contact with a surface. In this way any movement of the mouse is detected relative only to its previous position and its 'origin' can be reset arbitrarily. In fact, on the Macintosh the cursor is constrained within the (visual) screen and once it has reached an edge it remains in the same position - even if the mouse is moved further in the same direction. However, if the mouse is then reversed the cursor will move immediately. This implies that the apparent correspondence between a point on the surface beneath the mouse and the screen can easily - and inadvertently - be altered.

One added advantage of using the Macintosh is that it was possible to generate synthetic speech from it without the need for additional hardware. Built into the Macintosh are free-form sound wave generators, and *Smoothtalker* software (manufactured by First Byte) is

available which drives the sound generators as a speech synthesizer. Smoothtalker can be called from user programs. It accepts input either as strings of ascii characters which would be printable as text or as strings containing special phonetic symbols. Both forms of input were used by Soundtrack. The Macintosh also has a separate square-wave sound generator, and this was used to produce the tones associated with the auditory objects.

The choice of the Macintosh inevitably had some influence on the implementation of Soundtrack. One hardware detail is that the Macintosh's mouse has a single button. This influences the choice of mouse protocols to be implemented. With a single button there can practically be just three forms of button press recognized: the single-click, the double-click and the drag. In this implementation of Soundtrack a single-click is used as the *speech* button and a double-click for *execute*. Dragging is used with just one object, the **Thumb bar**, which is described below, in Section 5.4.

The Macintosh was by no means the only possible vehicle on which Soundtrack could have been implemented. Most other machines would have needed hardware enhancements however. Many microcomputers do not have a mouse as standard, though they are usually available as an option. Similarly it would be necessary to add a hardware speech synthesizer to most other computers. The critical requirement is for a machine with sufficient memory, since Soundtrack is a fairly complex - and hence large - program. It would have been possible to use a more powerful machine, such as one of the current personal workstations (Sun, Apollo or Perq, for example). However these are specialized computers which are unlikely to be used by a significant number of visually disabled people. In this way the Macintosh represented a more realistic target.

Soundtrack was implemented in the *MacAdvantage* version (SofTech, 1984) of Pascal (Jensen and Wirth, 1975). In fact, *MacAdvantage* is based upon the UCSD implementation of Pascal (Clark and Koehler, 1982). This language and implementation were chosen simply because it was the only one available at the time which had complete access to the

Macintosh's system routines (known as the *Toolbox* - see Apple, 1986). In fact, the language implementation was far from being an ideal one. Its limitations are detailed in Edwards and Woodman (1985), but arise principally from the fact that the UCSD compiler generates a pseudo code (*p-code*) which is interpreted at runtime by another program. The interpreter (also known as the *pseudo machine* or *p-machine*), was designed to have a strictly bounded 64-Kbyte user address space. The Macintosh has very much more real memory than that. Most significantly there are areas of the Toolbox which must be accessed which are outside the user's address space. MacAdvantage includes extensions to the language to give the necessary facilities, but these are somewhat cumbersome and inelegant.

There is a wider, more philosophical question as to whether a language such as Pascal is the appropriate medium for expressing programs of the type which run on wimp-based computers. Such programs must manipulate the sort of screen images mentioned repeatedly in this thesis, such as menus, icons and windows. Such concepts did not exist when that language was invented. As has been mentioned above, many of the concepts were developed within the Smalltalk project, but a major component of that project was the development of a new style of programming language (Goldberg and Robson, 1983). It would appear that manufacturers and developers have caught on to the attractive user interface features which were evolved but have not paid as much attention to the language lessons. At the time Soundtrack was implemented these programming techniques had not had any influence on programming tools on computers like the Macintosh. Subsequently, however, a number of object-oriented languages have been developed for the Macintosh (see Schmucker, 1986).

The design of software is also influenced by the system software on which it runs. The Macintosh Toolbox contains procedures which manipulate wimp entities at a high level. For instance, there is a set of calls which manipulate windows, obviating the need for the user to define them in terms of lower-level graphical objects. Using these Toolbox calls can greatly reduce the programming effort, but does mean that, at the level of presentation, one

Macintosh application usually looks much like any other. This is to some extent deliberate and Apple does try to enforce standards of interface design. The objective is to ease the learning of how to use software. Ideally once a person has learned to use one Macintosh program they will be able to use any others with minimal further help.

Furthermore, one of the areas of investigation within this project is feasibility of adapting visual software and therefore the auditory interface to Soundtrack was built onto a visual word processor. The design of that visual program was deliberately similar to that of a commercial program, *MacWrite* (Apple Computer Inc., 1984). However, Soundtrack was intended as a prototype, not as a product. That implied that some features of *MacWrite* which were less likely to be useful, and which would add little or nothing to the information available on the usefulness of the word processor could be omitted. Soundtrack does *not* provide the following facilities, which are provided by *Macwrite*:

- multiple typefaces, sizes and styles;
- selectable justification of text (i.e. the choice of left, right or full justification);
- setting of margins;
- tabs;
- printing facilities;
- search and replace strings;
- undoing of commands;
- the ability to rename a file (*Save as...*).

One facility provided by Soundtrack which *MacWrite* does not have is the ability to have two documents open at a time. This is an important capability of modern multi-window interfaces which ought to be tested as part of any adaptation. It was a feature omitted from *MacWrite* because of the memory limitations of early versions of the Macintosh.

In implementing Soundtrack some care was taken not to make it too specific to the Macintosh. In principle it ought to be possible to transfer the ideas embodied in the program

to other machines. For instance, the Macintosh Toolbox includes facilities whereby mouse interactions with standard screen objects (menus, dialogues etc.) can be 'filtered' through routines supplied by the programmer (i.e. not part of the Toolbox) and so handled in a non-standard way. (See Apple Computer Inc., 1986, for a full explanation of the technicalities). This means that it would be possible to adapt a visual program by the addition of appropriate filter routines. These would be invoked if the program was being used in an auditory mode, while the standard routines would be called when it was in a visual mode. However, this technique was not used in the implementation of Soundtrack since this would have made the implementation too specific to the Macintosh. Instead auditory objects are essentially defined 'from scratch' in a way which could be transferred to other systems.

Similarly, Macintosh programs use the concept of *resources*, which are items of data associated with a code file which can be altered without re-compiling the program. Again, because this is a technique specific to the Macintosh its use was avoided as far as possible in the implementation of Soundtrack.

5.3 Implementation of the screen

In order to demonstrate that the principles embodied by Soundtrack could be applied to the adaptation of visual wimp software, its implementation was built on top of an equivalent visual program. It is possible to switch at any time between visual and auditory interfaces to the same underlying word processing program. The visual interface is not described in any detail herein since it is not a significant or novel part of the project. The auditory interface to Soundtrack is embodied entirely in sounds with nothing visible on the screen. In the description of the design it was pointed out that some visual feedback might be useful in an auditory interface. However, for the purposes of evaluation within this project, it was thought that it should be that sighted and partially sighted people should not be at any advantage in using the program. Therefore the screen remains blank when the program is in the auditory mode.

In this implementation the size of the auditory screen was fixed by that of the visual screen. Movements of the Macintosh mouse are not reflected by equal distance movements of the cursor. The ratio of mouse distance to pointer movement is actually approximately 1.3:1. That is to say that the mouse is moved through an area approximately one-third larger than that of the screen, but when the cursor reaches the edge of the visual screen the auditory screen edge tone is sounded (see Figure 4.2). That tone is a low frequency buzz (33 Hz or C three octaves below middle C), which sounds continuously as long as the cursor is off screen and ceases only when the cursor is moved back onto the screen.

At an early stage in the design process it is necessary to decide on the degree of functionality which is to be built in to the program. In other words, the number of word processing features which were to be available had to be selected. A number of factors had to be taken into consideration. One was the selection of features to include, as discussed in the previous section. There was also the question of how many windows to include in the auditory screen. As discussed in the previous chapter, adopting a grid layout of screen items poses problems of fitting the appropriate number of items into the grid. The factors to be considered are:

- the number of items should not be so large as to cause the user working memory problems;
- there must be sufficient items to provide the number of features required;
- the items must fit some way into a grid layout.

As mentioned in Chapter 4, it was decided that the maximum number of items should be around nine. That number is too small to provide all the features of a word processor, which is the reason why a two-level style of interaction was chosen. At this upper level there is a degree of flexibility available to the designer in deciding how many windows will be provided, and then arranging the items within that number of windows. In Soundtrack it

was decided that there should be *eight* windows, as shown in Figure 5.1. This is a non-prime number, so allowing a true grid to be used, and is less than the memory limit of nine.

Document 1	Speech Menu	Interface Menu	Dialogues
Document 2	Edit Menu	File Menu	Alerts

Figure 5.1. The layout of the windows within the auditory screen (also shown in Figure 8.3).

As was mentioned in Chapter 3, tones can be modulated using their frequency, amplitude, timbre or duration. It was decided that the tones denoting screen objects would be differentiated by their frequency - or pitch. Even people who do not have a trained musical ear can usually hear that there is a difference between two notes of different pitch (Ward, 1963). Those with more training can extract more information, such as which note is higher, or the tone interval. A pragmatic reason for using pitch was that on the Macintosh it was easy to generate pure tones of different pitch, using the built-in square-wave generator. Using tones of different duration would be impractical, for two reasons. Firstly, it would be difficult to differentiate the duration of notes unless there was a wide variation. Secondly, this would be a very slow method; it would take a significant time just to hear the longest of (say) eight tones. Judging the volume (amplitude) of different tones is also very imprecise and subjective. Timbre is a quality which has such a wide variability that it could be an interesting way of communicating the necessary information. It should be possible to generate a very wide range of easily distinguished sounds. These could be produced on a

Macintosh also, through its free-form wave generators, but to develop suitable sounds would be time-consuming. Also, there seems no obvious way of systematically assigning different sounds to screen objects.

Windows were allocated tones in a simple pattern, increasing in pitch from left to right and from top to bottom. The top lefthand window has the tone of pitch C below middle C (132 Hz). Tones increase in intervals of four semitones. The tone allocation is shown in Table 5.1.

Window	Note	Frequency (Hz)
Document 1	C	132
Speech menu	E _b	158.4
Interface Menu	G	198
Dialogues	B	247.5
Document 2	E _b	316.8
Edit menu	G	396
File menu	B	495
Alerts	E _b	633.6

Table 5.1. The pitch of the tones associated with each of the windows.

This pattern of allocation means that the interval between two windows which are adjacent horizontally is four semitones, and between two vertically adjacent windows is twenty semitones.

Moving the mouse into a window causes its tone to be sounded - as the cursor crosses the edge of the window. If the mouse is clicked once, the window's name is spoken. To activate a window the mouse must be double-clicked within it. The fact that it has been

activated is signalled by its tone being sounded twice. If the mouse is moved out of the active window a short buzz is sounded as it crosses the boundary to the next window. Moving back into an already active window causes its tone to be sounded twice.

As mentioned earlier, the role of a window never changes. The contents of the menu windows do remain the same. However the contents of the other windows may change during the use of the program. These windows may be empty, in which case they cannot be activated. Certain commands will cause them to become *filled*. For instance, when a file is to be opened the **Dialogue** window will be filled with the **Select file** dialogue. This can then be activated so that the user can interact with it.

Once a window has been activated the objects therein become accessible to the user. Objects within a window are assigned tones in the same pattern as the windows within the screen. The top lefthand object has the note with the lowest pitch and that note is the same as that for the enclosing window. The other objects are assigned tones of pitch rising in the same four-semitone intervals as the windows. The same mouse protocol also applies; a single-click will cause the name of the current object to be spoken and a double-click will cause it to be executed. The effect of executing an object depends on its identity.

The user has several cues that she is working within the activated window. The first is that she should normally remember that she has activated the window. She will also find that tones are sounded as she moves the mouse through small distances. In other words, the auditory objects are smaller. If she should move out of the active window she will hear a special tone as she crosses its border. If she moves into the active window she will hear three tones. Finally, as always, she can press the mouse button to get spoken information as to its position.

As soon as a window is activated the cursor will be pointing at one of its constituent objects and its tone will be sounded. However, recall that on activating a window its tone is

sounded twice. That means that, in fact, on activation three tones are heard: two from the window and one from a contained object.

5.4 Document windows

If a **Document** window is empty, clicking in the window will elicit the speech, "Empty document one", or "Empty document two", as appropriate. If it has been filled by a file copied from disc (by the **Open** command from the **File** menu - see below) then clicking within it will elicit the file's name. If it has been filled by a newly created document (via the **New** command in the **File** menu) then the file will not yet have a name, so clicking in the window will produce the speech, "Unnamed document".

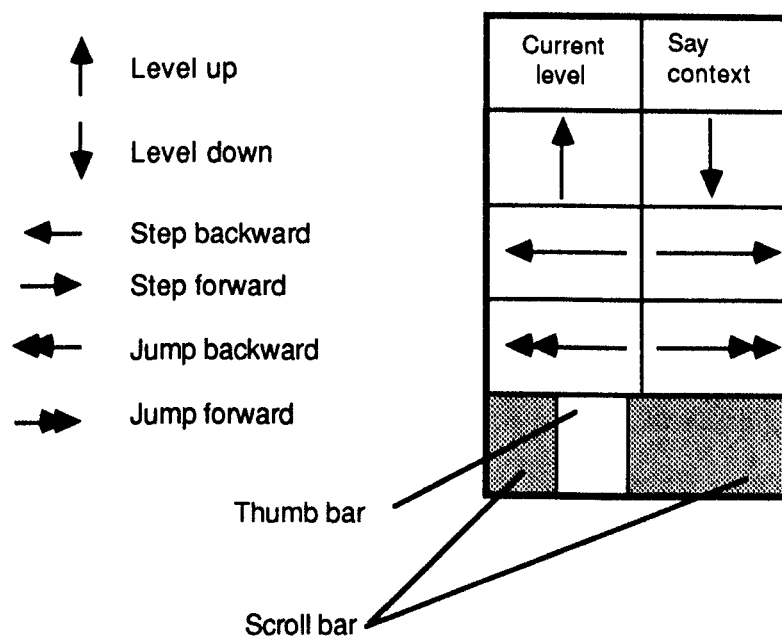


Figure 5.2. The **Document** window (also shown in Figure 8.8).

As described earlier, the auditory document window has two properties associated with it: a *selection* and a *level*, and the (activated) **Document** window consists of objects which control these, as shown in Figure 5.2. The **Level up** and **Level down** objects control the setting of the level. As mentioned earlier, the available levels are (in descending order):

whole text,
paragraph,

sentence,
word,
character,
point.

To attempt to execute **Level up** when the level is **whole text**, or **Level down** when it is **point** is an error.

As is shown in Figure 5.2, the **Document** window contains eight objects. This is the largest number of components of any window in Soundtrack, and is at the upper end of the suggested limit of objects (seven, plus or minus two).

As with a visual word processor, the selection is the portion of the document on which editing operations act. It is defined by a start and an end position. The start and end positions may be coincident, in which case the selection will be a point. The text selected is communicated to the user by its being spoken - unless it is a point. There are a variety of modes in which it can be spoken, which are explained below within the description of the **Speech** menu.

Levels and selections are defined purely in terms of the syntactic structure of documents. Within a paragraph no line structure is defined. Using the auditory interface there is no need to be concerned about lines. If a document is to be displayed visually, either through the visual interface to Soundtrack or when a document is printed, lines are broken automatically to fit the screen or paper.

Specific definitions must be given for the units which Soundtrack treats as paragraphs, words and sentences. Obviously the beginning of the document can be the left-most delimiter of any of these units, and the end of the document can be the right-most. Generally, though, a paragraph is defined as the text contained between two newline characters, not including the newlines. A word is defined as an unbroken string of letters

and/or digits. A word will thus be delimited by two characters which are not letters or digits (i.e. usually a space or a punctuation mark). The delimiting characters are not counted as part of the word. One unfortunate deficiency in this definition of a word is caused by the ambiguous use of the single quote symbol as both a quotation mark and the apostrophe. Without a certain amount of analysis of a piece of text it is not possible to decide which role the symbol is in at any one appearance. The definition used here treats it always as a quotation mark, which means that apostrophized words are treated as two separate words.

The end of a paragraph (a newline character) may mark the end of a sentence. Otherwise, a sentence is defined as being terminated by either a full-stop, an exclamation mark or a question mark which is followed by one or more spaces which in turn is followed by a capital letter. This definition aims to reduce possible confusion which might be caused by the use of full-stops in other roles. For instance it would mean that the sentence,

An envelope marked O.H.M.S. has been delivered.

would be treated as such, and not effectively truncated after any of the full-stops in *O.H.M.S.* Of course this definition is still not clever enough to recognize,

My name is A. Edwards.

as one sentence. A sentence is defined as starting with the first capital letter following the end of the previous sentence, as defined above.

Altering the level will cause the selection to be adjusted. Moving a level up will cause the end of the selection to be extended forward through the text to the end of the nearest unit of the new level. For example, if a character is selected and the level is moved up to **word** the end of the selection will be extended to the end of the current word. The beginning of the selection (i.e. the original character) will not be affected. Moving the level down correspondingly causes the selection to be contracted from the right. For instance, if a

sentence is selected and the level is moved down to **word** the selection will become the first word of the sentence.

The selection can also be altered by moving it through the document. The *step* objects cause the selection to move in units corresponding to the current level. Suppose, for example, that the current level is **sentence** and a complete sentence is selected. If **Step backward** is executed then the previous sentence will become the new selection, as illustrated in Figure 5.3.

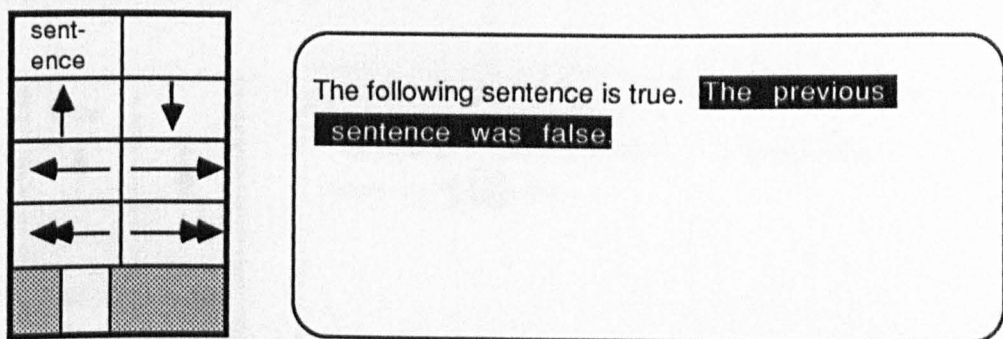


Figure 5.3a. The level is **sentence** and a sentence is selected - as shown by being displayed in inverse video.

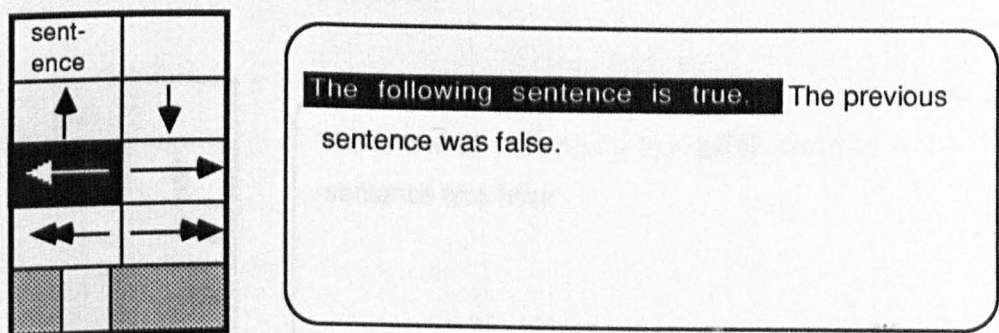


Figure 5.3b. The **Step backward** control has been executed and the first sentence is now selected.

It is possible for the level and the unit currently selected to be different. For example, even though the level is **word** the selection may be a string which is less than a whole word. This phenomenon and its implications is discussed more fully in Section 10.6. The movement of the selection is implemented in a consistent fashion. For instance, if the

selection is moved *backwards* then the start of the selection is moved to the left to the first starting position appropriate to the current level. The end of the selection is then adjusted - if necessary - to correspond to that new start.

The *jump* controls allow the selection to be moved in larger quanta. In fact the selection is moved through the distance corresponding to the unit one level up from the current level. That is to say that if the current level is **word**, for instance, and a **Jump backward** is executed, the selection will move to the first word of the sentence, and so on. Figure 5.4a gives an example of this. Jumps cannot be executed if the level is paragraph.

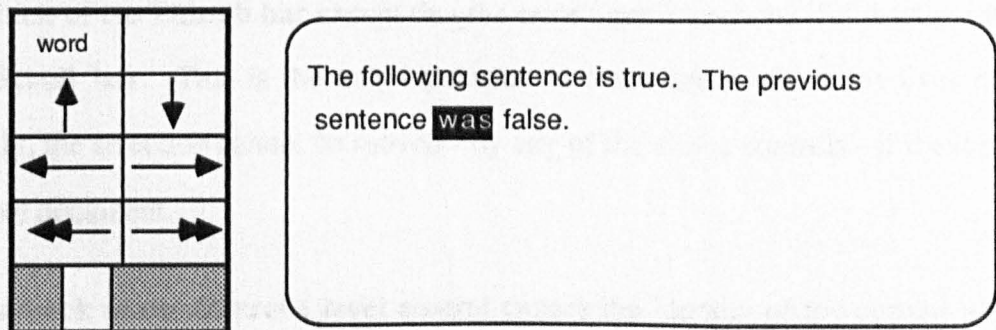


Figure 5.4a. The level is **word** and the word *was* in the second sentence is selected.

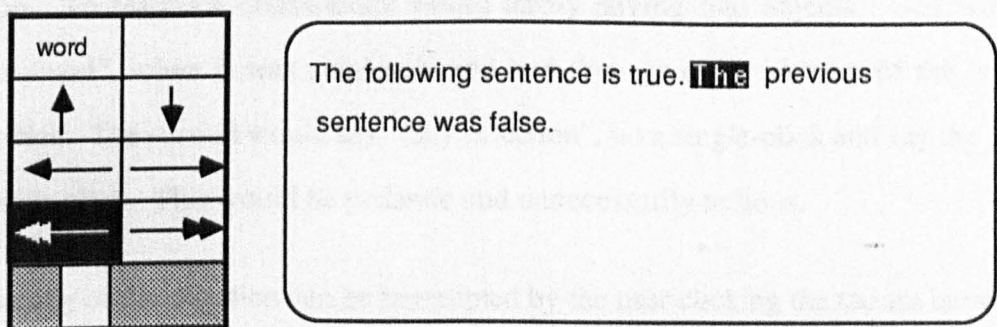


Figure 5.4b. The **Jump backward** control has been executed and the selection has moved to the first word of the sentence.

The **Thumb bar** and **Scroll bar** provide another means of moving the selection through large distances. They work according to the same principles as their visual counterparts, as described in Chapter 2. The position of the **Thumb bar** within the **Scroll bar** reflects the position of the selection within the document. So, if the selection is at the start of the

document the **Thumb bar** will be at the left-hand edge of the **Scroll bar**; if it is at the end of the document, the **Thumb bar** will be on the right and intermediate positions will be represented proportionately. However, not only do these objects act as indicators of the selection's position, but they can be used to alter it. The **Thumb bar** can be dragged within the **Scroll bar** and wherever it is 'put down' the selection will be adjusted to reflect the new position. In order to drag the **Thumb bar** the user must double-click within it, but hold the mouse button down on the second click. The mouse is then moved horizontally before the button is released. The **Thumb bar** will then be moved to the release position - and the selection adjusted correspondingly. During the dragging there is no auditory indication of the position of the **Thumb bar** except that the error buzz is sounded if it reaches either end of the **Scroll bar**. This is the only operation in Soundtrack which involves dragging. Naturally, the selection cannot be moved - by any of the above controls - if it extends over the whole document.

A single-click of the **Current level** control causes the identity of the current level to be spoken ("paragraph level" or whatever). A double-click will cause the current selection to be spoken. This is inconsistent with the way other objects operate, but is a pragmatic deviation. To maintain consistency would imply having two objects. One would say, "Current level", when it was single-clicked and then give the identity of the level on a double-click. The second would say, "Say selection", on a single-click and say the selection on a double-click. This would be pedantic and unnecessarily tedious.

The speaking of the selection can be interrupted by the user clicking the mouse button. If the level is currently **word** or lower this will cause the speech to cease as soon as it has completed saying the selection. If the level is higher it will stop once it has completed saying the next unit corresponding to one level down from the current level. That means, for example, that if the selection is being spoken at **paragraph** level speech will stop at the end of the current sentence. The interruption is not immediate because speech output is asynchronous. In order to allow interruption, it is therefore necessary to send strings to the

speech synthesizer a packet at a time so that the program can check 'between packets' whether the mouse button has been pressed. If one were to allow interruption after each word the packets would have to be words. However, the quality of speech is better if text is sent to the speech synthesizer in larger units; Smoothtalker does take account of punctuation and puts appropriate intonation onto sentences. Also there is a perceptible gap between the speaking of packets. To listen to, say, a whole paragraph of separated words spoken without intonation would be unnecessarily tedious. The method implemented is therefore a compromise. It does allow interruption, while maintaining the spoken quality of longer speeches. Sentences are, however, spoken word-by-word - without inflection. If the speaking of the selection is interrupted, the start of the selection is re-set to the *beginning* of the last unit spoken.

The **Current level** control can be used to get the selection spoken. The **Say context** object enables the user to hear the text which surrounds the current selection. If the level is **point**, executing **Say context** will cause the two characters on either side of the selection point to be spoken. At other levels **Say context** enables the user to hear the text one level up without altering the selection or the level. For example, at **character** level, the user can hear the word containing the currently selected character, and so on.

5.5 Menu windows

Menus were implemented as described in the design, except that control-key alternatives were not provided. Within this project it was necessary to assess how effective auditory menus are since they are one of the features which is new within this program. If control-key alternatives were available users might avoid using menus to a large extent. In describing the implementation of the screen above, it was pointed out that the designer had some flexibility to choose how many windows the program would have. There is less flexibility in designing the menus, in which the number of entries is dictated by overriding decisions which have been made about the facilities within a program. In fact, in adapting

an existing visual program the decision about the number of entries in each menu will have been taken by someone else - the designer of the original program. Because of the problem of accommodating non-prime numbers of menu entries it was decided in Soundtrack not to implement menus as multi-column grids, but as single columns.

The way in which the selection in a document will be spoken can be controlled via the **Speech** menu (Figure 5.5). All of the entries in this menu are switches which are said to be *ticked* when they are on. The fact that an entry is ticked is communicated to the user when the entry's name is spoken. For example referring to Figure 5.5, if the user was to single-click in the **Say typing** object, the program would say "Say typing, ticked". Switches are toggled by double-clicking.

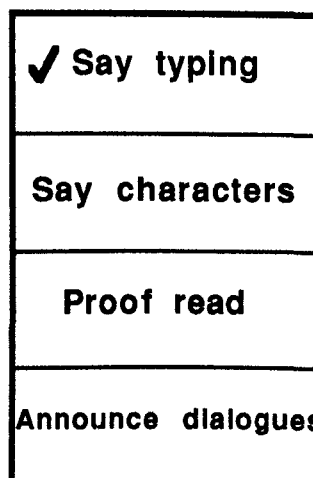


Figure 5.5. The **Speech** menu.

As described below, certain commands cause the **Dialogue** window to be filled. The fact that it has been filled will be signalled by its tone being sounded, but in addition it is possible to have the identity of the filling dialogue spoken. This will occur if the **Announce dialogues** entry of this menu is ticked.

If the **Say characters** entry is ticked then all selections will be spelled out letter-by-letter. This will occur regardless of the current level of the document. If the entry is not ticked, selections are spoken as complete words. Of course it is also possible to hear individual

letters within a document by using **character** level; the nature of the users current task will dictate which method is appropriate.

The **Proof read** entry enables the speaking of all the characters in a document. Ticking **Say characters** alone only allows letters and digits to be spoken, whereas ticking **Proof read** causes punctuation to be spoken also. If **Say characters** is not ticked and the level is higher than **character** the selection is spoken as words but with punctuation marks pronounced. In that case words are spoken separately, without intonation.

If the **Say typing** entry is ticked then input from the keyboard will be spoken. If **Say characters** is ticked then each key will be spoken, otherwise words will be spoken as they are completed.

The **File** menu is illustrated in Figure 5.6. Executing the **New** object causes one of the **Document** windows to be filled with a new, empty document. The **Open** object will also cause a document to be opened in a **Document** window. However, it firstly fills the **Dialogue** window with the **Select file** dialogue, through which the user can specify which file to open. A maximum of two documents can be opened. A new or opened document is put into the lower numbered available **Document** window.

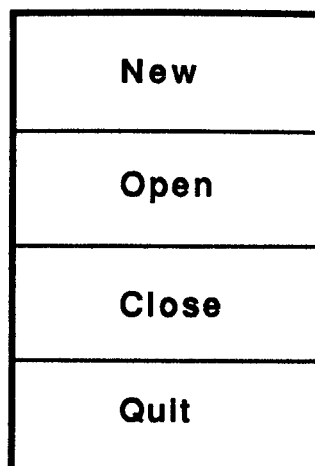


Figure 5.6. The **File** menu.

Executing the **Close** object causes whichever document was most recently active to be closed. This will entail further interaction with dialogues concerning the saving of the document on disc (see below). The **Quit** command causes the whole program to shut down, but will ensure that files are firstly closed and their contents saved if necessary, via dialogues.

The **Edit** menu (Figure 5.7) contains commands which act on the current selection in a document. Executing **Cut** causes the text currently selected to be deleted. The remaining text is readjusted to close any gaps and the selection becomes a point in the position of the deleted text - regardless of the current selection level. The text deleted is held in a buffer. It is possible to copy the contents of that buffer into a document, using the **Paste** command. Text can also be *copied* into the buffer, via the **Copy** command. Having executed **Copy** the current selection will also be put into the buffer but not deleted from the document. It is an error to attempt to execute **Cut** or **Copy** if there are no characters currently selected (that is, the selection is a point).

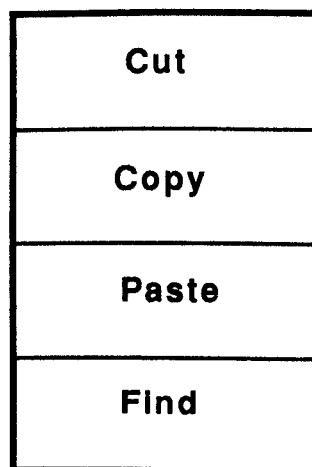


Figure 5.7. The **Edit** menu.

Executing **Paste** causes the current selection to be replaced by the buffer contents. Should the selection be a point, the text will be inserted at that point and no text is lost. The only way of changing the buffer contents is to execute a **Cut** or **Copy**; pasting does not cause the buffer contents to be altered, which means that its contents can be pasted in more than one

place. There is just one buffer, which is used for both **Document** windows. This implies that text can be transferred *between* documents in the manner described.

The **Find** command enables the user to do a string search in a document. Executing it causes a dialogue to be opened, through which the user can specify the target string, see below for details.

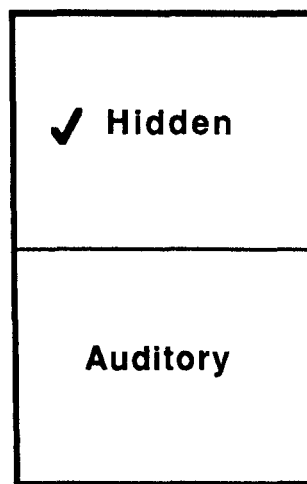


Figure 5.8. The **Interface** menu.

It has been stated that Soundtrack has two interfaces to the word processor, an auditory one and a visual one. It is possible to switch between the two modes by way of the **Auditory** switch in the **Interface** menu (Figure 5.8). There is also an intermediate mode of operation in which the user interacts with the auditory screen and windows, but any Document windows are displayed visually on the screen. This mode is of use to people who have sight who use the program, or who assist visually disabled people who use it. The switch to this mode is achieved via the **Hidden** entry in this menu. If Soundtrack is used with the **Hidden** entry ticked then nothing is shown on the screen relating to the operation of the program. This was done deliberately for testing purposes, so that sighted and partially sighted users of the program would have no obvious advantage over blind users.

5.6 Dialogues

Open in the **File** menu enables the user to open a file from the disc. The **Select file** dialogue (Figure 5.9) enables the user to specify which disc file is to be opened. This dialogue contains a list of file names through which the user can scroll to find the appropriate name. The list of file names is in an alphabetical order and when the dialogue is first opened the alphabetically earliest name is displayed in the **Name** object. The current name at any time can be heard by clicking in that object. The **Down button** can be used to move through the list, and the **Up button** to return to earlier entries. Once the correct name is displayed the user can open the file by double-clicking either on the **Name** or the **Open** object. Executing **Cancel** causes the open operation to be aborted.

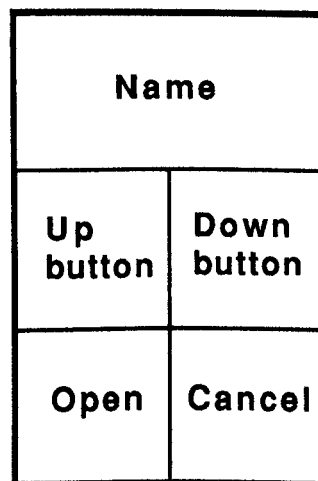


Figure 5.9. The **Select file** dialogue.

Typing the file's name as an alternative to scrolling it into the **Name** object, as described in the design in Section 4.5, was not implemented in Soundtrack. This was another example where a feature was omitted for the purposes of the evaluation; it was necessary to force the users to use the list mechanism to assess how effective it was.

Figure 5.9 shows another approach to the problem of fitting a non-prime number of items into a window. The **Select file** dialogue requires five items. These will not fit into a symmetrical grid. Therefore, it was decided to extend one (the **Name** object) so that it

effectively takes up two squares in the grid. The effect of this inconsistency on users' behaviour is discussed later, in Chapter 9.

Documents may be closed if the user executes either the **Close** or the **Quit** entry in the **File** menu. In the former case the document which was more recently active is closed, while in the latter case any files which are open may be closed. Whenever a file is closed the user has the option of saving it to disc. To ascertain whether the user wants the document saved the **Save file** dialogue is opened. The **Update file** object in this dialogue (see Figure 5.10) behaves differently depending on whether the document in question is a new one or one which already exists on disc - that is to say, whether it was opened via **New** or **Open** in the **File** menu. If it is an existing file then clicking the **Update file** object will elicit a question whether that file should be saved. For example, if the document is called *test* the dialogue would say "Update test?" The user then has the option of executing any of the other three objects.

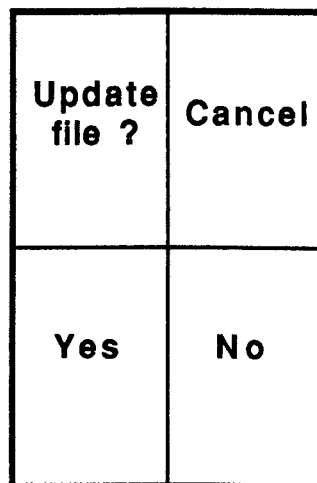


Figure 5.10. The **Save file** dialogue.

Double-clicking **Yes** will cause the **Document** window to be emptied and file to be written to disc without further interaction. So doing, it will over-write the old contents of the file on disc. Executing **No** will still cause the **Document** window to be emptied, but the disc file will not be modified. Double-clicking **Cancel** will nullify whichever command had evoked the **Update File** dialogue (i.e. **Close** or **Quit**), leaving the document open. If the

document is a new one it will not yet have a name, in which case clicking the **Update File** object will elicit the speech, "Create a new file?" In this case, if the user executes **Yes** then she must specify the name of the file in which it is to be saved through the **Get file name** dialogue (Figure 5.11).

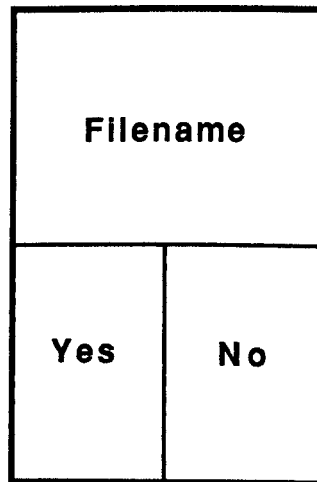


Figure 5.11. The **Get file name** dialogue

Whenever the **Get file name** dialogue is active the user can type in the name by which the file is to be saved on disc. Clicking in the **Filename** object will cause that name, as typed so far, to be spoken. Double-clicking on that object will cause the file to be saved, as will executing the **Yes** object. Executing the **No** object will cancel the closing of the file.

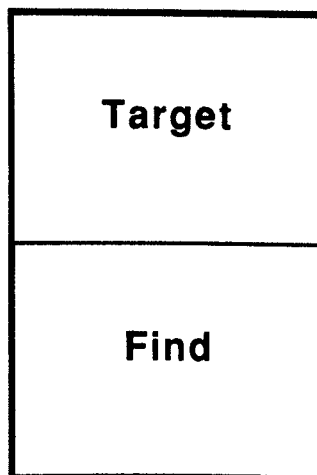


Figure 5.12. The **Get target string** dialogue.

The **Get target string** dialogue (Figure 5.12) is the only modeless dialogue in Soundtrack. It is opened if the user executes the **Find** entry in the **Edit** menu. When the dialogue has been activated the user can type in the string which is to be located in the document which was more recently active. Double clicking the **Find** object in the dialogue will cause the search to start. The document will be searched forwards from the current selection. If the target is found the selection will be set to that occurrence of the string. The dialogue remains open and active, so that executing the **Find** object again will cause the next occurrence of the string to be searched for. If the user wishes to search for a different string, double-clicking the **Target** object will clear the target string, allowing her to type in a new string. The dialogue will continue to fill the dialogue window until another dialogue is opened.

5.7 Alerts

Alerts were not implemented in Soundtrack. Program size limitations meant that it was not possible to implement the complete design. As was mentioned above, the version of Pascal used has a limitation of 64 Kbytes on the user data space and this limit was reached before the implementation was complete. For the purposes of evaluating this prototype alerts were a feature which were not vital. As explained earlier, alerts should be used to signal two types of error: serious external ones and less serious user errors. Within the testing of a program like Soundtrack it is desirable to avoid external errors which would distract the user from the working of the program, so that alerts should not be needed for such events. It is possible to achieve this within controlled testing, by being careful about managing disc space etc.

5.8 Diagonal alarms

During the evaluation of the program it became clear that a problem of using the mouse with Soundtrack was that it was easy to move it inadvertently in a diagonal direction. It was decided to experiment with possible solutions to this problem, although it was not a part of

the original design. The idea which was tried out was to give the user auditory feedback on the motion of the mouse.

Two forms of this were implemented. The first assigned two tones, associated with horizontal and vertical movements. If the user moved the mouse in a purely horizontal direction, bleeps of the appropriate tone were heard - in addition to the normal tones sounded as it moved from one object to another. Moving purely vertically generated other very distinctive bleeps. Of course, diagonal movements produced both kinds of bleep. The second form of feedback took a different approach in that a special tone was sounded only if a movement was in a diagonal direction; pure horizontal and vertical movements produced no sounds other than the normal object tones.

5.9 Conclusions and review

An impression of how Soundtrack operates is given in the transcript of a sample session of using it in Appendix D.

This chapter has described the word processor as it was implemented - based on the design outlined in Chapter 4. The implementation was intended to provide a vehicle for testing ideas about how programs might be adapted, a means of testing how well people could use particular forms of interaction. The novel feature was the mouse, and the implementation was such that the user was obliged to use the mouse. So, for example, control-key alternatives to menu commands were not provided, nor was the alternative mechanism whereby names could be typed into dialogues instead of being selected from a list.

Chapter 4 outlined the design of a word processor, this chapter described how it was actually implemented. That implementation had to be put to the test, and Chapter 6 discusses how the necessary evaluation was set up and carried out.

Chapter 6

Evaluation methods

6.1 Introduction

This chapter describes a study which evaluated the success of adapting wimp software in the way described earlier. The first section explains the need for evaluation, especially in the context of an aid for people with a disability. Following that, there is a discussion of some of the problems of carrying out such an evaluation - both in general and within this particular project. The results of the different phases of the evaluation are presented and discussed in the following four chapters.

6.2 The need for evaluation

The relationship between people with disabilities and those who would develop devices for their use is quite complex. To a person with average sight the idea of not having sight is devastatingly frightening. From that viewpoint it is easy to assume that a blind person would give almost *anything* for restoration of their sight - or for a device which compensates to some extent for the missing sense. In fact, however, the blind person will not give 'anything'. There is a limit to what effort they will *invest* in such a device. The fact that the majority of visually disabled people have rejected complex, high-technology mobility aids in favour of the simple white stick illustrates this proposition.

To any person - whether they are classed as disabled or not - training in a skill is an *investment*. An individual will be sufficiently motivated to work on their training if she believes that the benefits of having the skill will be sufficient pay-off for the effort she will

have to expend. A very good illustration concerns the use of braille. Most visually disabled people do not read braille. They do not believe that it is worth their while investing whatever amount of effort and time it would take for them to learn it for the return it would give them in terms of access to literature. Most sighted people - who read print frequently - assume that the learning of braille as a substitute would be of the highest priority to a blind person.

Two phases of using a device should be distinguished: learning and application. Applied use is the utilization of the device to perform routine tasks after the operator has acquired the necessary skills through practice - during the learning phase. Attention must be given to both these stages. Learning is important because if a person finds learning too difficult they may give up - and never even reach the application stage. Generally, if they do attain the application stage they can be expected to continue using the device. Thus, for example, if a person can get over the hurdle of learning braille, it is a facility they will continue to use.

So it is that it is not sufficient to develop a device which enables (say) blind people to do something they could not do before; it is necessary that it should not require an excessive amount of effort to do so. Soundtrack gives blind people access to a type of computer which was formerly not possible, but the question to be answered is whether it does so in a manner such that people will *want* to use it. The aim of the evaluation phase of this research project therefore is to gain insight into the *cost* of using an auditory wimp program.

6.3 Problems of evaluation

This project aims to assess the feasibility of visually disabled people using adapted wimp software by testing a particular program with such an interface. It is therefore necessary to evaluate the quality of that program. The characteristic of the software to be assessed concerns the cost of using it. Several components of the cost can be considered: the *time* taken, the *ease-of-use* and the *ease-of-learning*.

In evaluating a product which represents an innovation it is not possible to perform a comparative study, since there is nothing with which to make a comparison. The novelty

implies that there cannot be any applicable methods which have been established for evaluation. In the case of this project no other wimp-based programs exist for use by blind people. To present a blind subject with Soundtrack and a visual wimp-based word processor and to ask them which they preferred would hardly be a fair test! Of course, adapted non-wimp word processors do exist, but to run a straight comparison of one or more of these with Soundtrack would be to miss the true objective of this evaluation. The aim is not to evaluate Soundtrack as a word processor, but to judge the auditory interface as incorporated in Soundtrack. The objective is, after all, that the interface could be applied to programs other than word processors. Also, as was explained earlier, Soundtrack was a 'test bed', designed to test certain forms of interaction and not as a production word processor.

Bearing in mind what has been said above about the innovative nature of this project, it is still useful to look at related evaluations which have been carried out. Relevant work has been done which gives pointers to how this evaluation should or should not be carried out. None of the existing audibly-enhanced word processors has been evaluated in any systematic manner. However, one relevant area which has been studied quite extensively is the evaluation of text editors - for use by sighted people. Embley and Nagey (1981) identify five approaches to the study of editors and editing processes. These are each described briefly below.

1. Introspection By *introspection* Embley and Nagey mean the use of the designers' own intuition and experience. They assert (page 35), "It is surprising how many programs intended for use by others, including text editors, appear to be based on this slim foundation." Clearly this approach has little to contribute to the present evaluation, except to emphasize the problem of finding appropriate methods.

2. Field observations This approach involves the collection of data in an environment in which the software is in regular, practical use. It can be used as a means of post-

implementation comparison of existing editors. Such results could also influence future designs. Hammer and Rouse (1979) used this sort of procedure to compare performance in using two editors, Teco and SOS.

3. Formal analysis It is possible to use formal notations to build models of interactions such as those involved in text editing. Two notations have been applied in this way: formal grammars and state-transition models. Formal grammars were used by Reisner (1981) as a means of comparing two graphical editors and state transition models have been applied by Anandan (1979). This style of approach suffers from the fact that it is difficult to say what relationship there is between the results and the behaviour of people using the editors; the models can describe accurately the nature of a task, but not its cost.

4. Controlled experiments By their very nature, controlled experiments are generally confined to investigating very specific aspects of a problem; the number of variables must be minimized. This method has, therefore, been used to evaluate particular facets of editors, but not of such programs as a whole. Most of the studies have concerned the command language defined by the text editor, as in Freedman and Landauer (1966), Roberts (1979), Walther and O'Neil (1974) and Ledgard et al. (1980).

5. Psychological models Building an accurate psychological model of the editing process has the advantage that not only does it provide insight into the nature of the interaction but it can also be used to make predictions about users' behaviour. It can thus be used as a means of testing and selecting from hypothetical designs before they are actually implemented. Card, Moran and Newell (1983) propose such a model, based upon what they call the *Model Human Processor* and go on to develop the more sophisticated *Goms* (Goals, Operators, Methods and Selection rules) model.

Roberts and Moran (1982) use a selection of methods in their attempt to formulate a standardized procedure for evaluating text editors, based to a large extent on the research reported in Roberts (1979). The method is based upon a set of 212 editing tasks which can

be potentially performed by a text editor. These are used to make measurements of four facets (or dimensions) of editor usage: time, errors, learning and functionality. The first three of these are measured experimentally on the basis of a subset of *core* editing tasks. Essentially, the core editing tasks involve the application of the following operations:

- insert,
- delete,
- replace,
- move,
- copy,
- transpose,
- split,
- merge,

to each of the following objects:

- character,
- word,
- line,
- sentence,
- paragraph,
- section.

Timing data was obtained by getting expert users to perform a number of benchmark exercises, based on the core tasks. Stopwatch times were taken and a score calculated as the total error-free time divided by the number of tasks completed. In addition, the keystroke-level model of Card, Moran and Newell was used to calculate expected times to complete the same tasks.

The fact that users make errors using text editors is a complication which a lot of analyses choose to ignore (as Card and his colleagues chose to do, for instance). However, Roberts and Moran's procedure does attempt to make simple measurements of the effects of errors.

A score is calculated as the time spent correcting non-trivial errors, expressed as a percentage of the user's error-free time score.

Learning was measured by using a different set of subjects who were novices at using computers. They were given individual tuition, based on the core editing tasks, and periodically tested on their proficiency. A score was calculated as the total time spent on learning divided by the number of tasks learned (as identified by the tests).

The functionality of editors was measured on the basis of the full set of editing tasks. Experienced users - with the aid of any necessary documentation - assessed whether particular tasks were possible with the given editor. Half-marks could be given if a task was possible but awkward to perform.

The resulting evaluation methodology has a number of limitations - most of which are conceded by its proponents. It is unfortunate that they did not attempt to run a control - such as a questionnaire - to see whether the scores they obtained bore any relationship to users' perceptions of the editors.

Roberts and Moran applied their methodology to evaluate nine text editors, with a view to this being the start of a database for comparison of text editors. They suggest that a reduced version of their procedure could be applied to the evaluation of proposed text editor designs which have not (yet) been implemented and which thus have no experienced users. This would be achieved by using analytical data on two of the four dimensions considered: time and functionality.

Times could not be calculated in this way for Soundtrack, since the keystroke level model applies only to visual editors. It should be possible to extend this model to auditory interactions, and this is suggested as a topic for further research, in Chapter 10.

The functionality dimension could be measured for Soundtrack. There seems little point, however, since it would be bound to score low, for two reasons. Firstly Soundtrack was

never intended to be a fully functional word processor, but merely a 'test bed' prototype. Secondly, in applying the principle of constraining the interface some functionality is lost.

One study which attempted to directly measure the ease-of-use of a text processing program was carried out by Good (1981). This looked at four aspects of using the *Etude* text processing system, as follows:

1. Ease of learning. This was measured by timing how long it took computer-naive subjects to learn how to create and edit letters.
2. Ease of use once learned. This was measured as the time taken to type and edit a one-page business letter, by subjects who had just completed the above training.
3. Anxiety. A standard *State-Trait Anxiety Inventory* (Spielberger, 1972) was employed to assess levels of anxiety in the subjects.
4. User attitudes. A particular form of questionnaire, in which users assigned numerical scores to ranges of qualities, such as *Unhelpful...helpful* and *Powerless...powerful*, was used to measure such attitudes.

Twenty subjects took part in the evaluation and results of statistical significance were thus obtained. It was concluded that most of the subjects took less than two hours and twenty minutes to learn to use *Etude*. Most of them took longer to create and edit letters using *Etude* than with a typewriter. There was no systematic difference in subjects' anxiety levels in using *Etude* and using a typewriter. Overall subjects attitudes to *Etude* were positive. The need for a large number of subjects for this kind of study precluded using this approach in the evaluation of *Soundtrack*. However, it might be borne in mind if a more extensive study is to be carried out in future.

In the previous section the case was made that evaluation ought to be a major part of the development of a product for use by disabled people. However, in this section it has been

shown that there are many obstacles which make evaluation difficult, particularly when the product is the first of its kind. This is illustrated by a number of books which have been written covering the field of electronic aids for disabled people. Often such books present anecdotal examples of individuals who have benefitted from aids, rather than any form of objective testing. Examples of such books are Hawkrige et al. (1985), Goldenberg et al. (1984) and Rostron and Sewell (1984).

6.4 Design of the evaluation

The preceding sections set out the context in which this evaluation was carried out. It is now possible to discuss considerations which applied particularly to this study. The most serious practical problem of organizing the evaluation of Soundtrack was that of finding a sufficiently large and homogeneous group of subjects who could be asked to devote a significant amount of time to the work. It was desirable that all subjects should have had prior experience of using word processors or other computer equipment. The aim of the training in this evaluation was to teach them about Soundtrack in particular and not about computers in general. Also it was not appropriate to use subjects who might have serious fears about using computers.

Soundtrack was designed in such a way that it could be run with no visible feedback from the computer screen. Thus it was intended that sighted and partially sighted people could test it without having any significant advantage over blind subjects. The idea was that it would not be necessary for all the subjects to be totally blind. The proportion of visually disabled people who are completely blind is quite small and in this way it was hoped to increase the potential number of test subjects. However, this idea did not take account of the psychological attitude of visually disabled people. Most partially sighted people prefer to make as much use as possible of their residual sight. This means that they generally are resistant to using any devices - such as Soundtrack - which do not allow use of vision.

The spatial abilities of a person who is congenitally blind are likely to be different from those of a blind person who has had experience of sight (though such abilities do exist, see Hermelin and O'Connor, 1982 and Dickinson, 1977). The ability to work in a two-dimensional space is an important part of using Soundtrack. Thus, for simplicity, it would be desirable to use only subjects who have had sight, and whose spatial abilities will therefore be similar to those of sighted people - and each other. In fact in this evaluation one of the subjects had been blind since birth.

The question of using subjects' time was a difficult one, since they were all volunteers. To pay subjects for their contribution would, to some extent, alleviate this problem. However, all the subjects were offered and refused any payment. This may be partly due to the complex position of disabled people within society. Such people are accustomed to being objects of charity, although they may well not like this role. It would seem that to offer payment to such people for their participation would be a sign that they were *not* being exploited. However, to refuse payment is a chance for them to offer almost charitable assistance to a researcher on a limited budget and to the eventual benefit of other disabled people.

To some extent the problems of finding subjects to test aids for disabled people are caused by the fact that the proportion of the population with any particular disability is comparatively small. Thus the potential population for inclusion in an evaluation is small, compared to testing of a product for use by people without any particular disability.

So it is that many devices developed for use by people with disabilities are *not* commercial propositions. Development is often sponsored by grants which can be written-off and not recouped by sales. This approach does facilitate the production of largely tailor-made devices. However, it also helps to perpetuate the position of disabled people being dependent on charity.

6.5 Introduction to the subjects

A total of eight subjects were used and they fall into two groups. In order to preserve anonymity, they are referred to by numbers. Two of them (subjects S1 and S2) were students at the Royal National Institution for the Blind (RNIB) Commercial Training College, while the rest of them were members of the Association of Visually Handicapped Office Workers (Avhow). All but one of them (subject S8) were male. Results are not reported herein for subjects S5 and S6 because neither of them completed the evaluation. Subject S5 was Dr Mark Elsom-Cook, acting supervisor of this research, who undertook the earlier part of the evaluation exercises under real conditions. This was a way for him to familiarize himself with the software and a preparation for the interviews he carried out as part of the evaluation (see Section 7.2). Subject S6 attended but one session and did not return. She was undergoing training in the use of another word processor as part of her job at that time, and feared that being trained on two different systems at the same time would cause interference in her learning.

The students from the RNIB College were adults who were registered as blind within recent years. They had completed earlier rehabilitation training and were now on a one-year course on office skills with a view to obtaining employment. Part of their course involved the use of audio-enhanced word processors. As full-time students they were able and willing to devote quite a considerable amount of time to the evaluation. No fixed time limit was set in advance for these subjects; they were able to work through all of the planned evaluation exercises. The total time spent by each of them is given in Table 6.1, along with other background information. They had two sessions per day (one in the morning, one in the afternoon) twice per week (Wednesday and Thursday) and completed twelve such sessions each.

The Avhow subjects were all in employment in office jobs. Their evaluation sessions were held in the evening at a central location (an office of the RNIB). Because of the greater inconvenience involved in these people spending time on the testing, it was agreed that they

would devote a fixed amount of time (six half-hour sessions) to it. Their total times are also given in Table 6.1. Note that the last of the six sessions was taken up with answering the questionnaire, which explains why the total times are of the order of 2¹/₂ hours. They used a variety of equipment in their jobs. All were competent touch-typists and most had some experience of word processing.

Subject number	Time on evaluation (hours:mins)	'Origin'	Years registered as disabled	Musical ability	Relevant experience
S1	7:45	RNIB	4	Has played piano and accordian	Uses Wordstar daily
S2	6:27	RNIB	7	Plays guitar	Uses Audiodata daily
S3	2:48	Avhow	18	Listener	Uses typewriter daily. Has used unadapted word-processor.
S4	2:41	Avhow	Since birth	Perfect pitch.	Uses typewriter daily. No use of word processor.
S7	2:32	Avhow	Since birth	Plays some guitar. Listens frequently	Uses Brailink and typewriter daily
S8	2:30	Avhow	Since childhood	Has played piano and flute	Uses typewriter a few times per week. Occasionally used
S9	2:26	Avhow	13	None	Uses Audiodata daily

Table 6.1. Summary of background information on each of the evaluation subjects.

All the subjects worked through the exercises in sessions each of approximately one half-hour (the exact time was recorded on the session record sheet, see Appendix A). After some early experimentation this appeared to be optimal length - although some of the subjects questioned that, (see Section 7.2). It is interesting to note that although S2 spent less time on the exercises, as shown in Table 6.1, he completed the same number of exercises as S1. Because they spent less time on the evaluation, the Avhow volunteers did not complete all the exercises in the series. However, all the Avhow subjects reached approximately the same stage in the evaluation.

One method of gathering data in the evaluation was the completion of a questionnaire. The first section of this covered background information on the subjects. This information is summarized in Table 6.1. (The results from the rest of the questionnaire appear in Chapter 7). Further relevant information on individuals is given below.

Subject S1. This subject was the only one of the subjects who still had a useful degree of sight. He had been registered as visually disabled for four years. However, his sight was deteriorating quite rapidly and he was expected to become completely blind. He used Wordstar running on an IBM PC - *without* audio enhancements. Using a green screen, he was able to use that system utilizing his residual vision.

Subject S3. S3 had experimented with using an ICL word processor without any modifications for visually disabled users. This was possible with some teaching from a sighted colleague and by memorizing the layout of the fixed command menu used by the program. He read text with the aid of an Optacon.

Subject S8. This was the only subject to complete the evaluation who was female.

6.6 Format of the evaluation

The evaluation consisted of teaching visually disabled subjects to use Soundtrack, and collecting data in several forms on their use of the program. The teaching took the form of

the tester leading the subjects through a set of exercises using the word processor. These exercises were graded in such a way as to gradually introduce the subjects to Soundtrack. See Appendix B for a full description of the exercises presented. Using these prescribed exercises ensured that all subjects received essentially uniform training and progress could be monitored quite accurately. A record sheet was completed for each session of exercises - see Appendix A. In all cases the tester was the author of this thesis.

Tactile diagrams of the screen and the windows were available to which subjects could refer during the evaluation sessions - but these could not be taken away by them. The diagrams were versions of some of the figures in Chapter 5, on which the lines were converted into ridges on the paper. The subjects were not given any learning aids to supplement their training outside the sessions. Several asked for aids such as braille notes or tactile diagrams which they could take away to reinforce their learning of the layout of objects on the auditory screen. These were not supplied because it would have introduced a degree of heterogeneity into the group; different subjects would have been likely to devote different amounts of their time to such 'homework'. As it was, the amount of time they put into it was precisely logged.

A written record of each session was kept on a standardized form - as in Appendix A. Some of the entries on that sheet should be explained. The exact starting time of certain exercises was recorded, so that they could be correlated with times recorded by the computer. Any significant comments voiced by the subject were recorded, so that they might be followed-up at a later stage. Similarly any observations made by the tester could be recorded. One section of the form was designed to record data on the way in which the subject behaved in certain exercises (Exercises 0.1 and 0.2: "0 exercises") - see Chapter 7. This information was used to aid the interpretation of the trace. There are two questions relating to a "dribble file". These were included as a reminder to the tester to do some 'housekeeping' on the computer at the end of the session - and had no direct relevance to the data.

Data were gleaned from the exercises in several ways, as outlined below.

1. Tracing interactions

The mouse was the element which made Soundtrack novel and so it was important to collect data on the degree of efficiency with which the subjects used it. In order to do this Soundtrack was designed to keep a record of all interactions with the user. That is to say that events such as movements of the mouse and key presses - and their timings - were recorded and saved on a dribble file. This file was later analyzed with the aid of another program. Each of the subjects thus completed particular exercises with tracing switched on.

2. Audio recordings

The last exercise completed by each subject was a comparatively complex one; recall that the set of exercises was graded and this was the ultimate one in that set. None of the subjects attained the level of skill at which they could be given such a task to work on unaided. Thus, the instructions for the exercise were quite involved and it was necessary for the tester to give instructions while the subject was working. In order to record the amount of assistance each subject received a tape recording was taken of the spoken dialogue. At the same time tracing was enabled, so that these two forms of record are integrated in transcripts, as in Chapter 8. In fact, these transcripts give a very full picture of the subjects' behaviour.

3. Structured interviews

Other aspects of using Soundtrack were explored by way of interviews with the evaluation subjects, structured around a questionnaire. Using this technique it was possible to collect subjects' opinions on using of Soundtrack. Also, by depersonalizing some of the questions (i.e. asking for their opinion of how someone else might react to using it) it was possible to make more general judgements about the interface. The questionnaire forms were filled in by the tester - because of the obvious problems of getting a visually disabled person to do this.

4. Interviews

For two of the subjects it was also possible to collect further subjective data by conducting a less structured interview. This was done in order to see whether additional information would be obtained. Also, the interview was carried out by a different person to see whether there was any personal bias.

6.7 Conclusions

Kemmis (1977) distinguishes between two approaches to evaluation: *nomothetic* and *idiographic*. Nomothetic approaches are those concerned with establishing laws, approximating natural science methods; whereas idiographic methods are concerned with the intensive study of individuals. The evaluation applied to Soundtrack borrows from both categories. The measurement of targetting times was a means whereby common features in the behaviour of a number of people could be identified and expressed in a model (admittedly this does stop short of something which could be said to be a *law*). The other aspects of the evaluation, however, were more concerned with discovering what individuals learned by using Soundtrack. Any attempt to perform a controlled experiment style evaluation would clearly have been ridiculous. Instead, it is justified to examine Soundtrack as something in its own right. It was not possible to say that it was being evaluated in terms of anything else, since there is nothing else sufficiently closely related. However, now this has been done once, there does exist a benchmark against which another auditory interfaces might be measured.

This chapter has described the format of the evaluation applied to Soundtrack. The following three chapters present and discuss the results in the different phases of the evaluation. Chapter 7 looks at subjects' views as gleaned in interviews. Chapter 8 presents data on the timing of interactions and Chapter 9 describes their performance in the more complex, but realistic editing exercises, illustrated by extracts from transcripts of those

exercise sessions. Finally, Chapter 10 contains a discussion of the overall results of the evaluation.

Chapter 7

Interviews

7.1 Introduction

An important aspect of evaluation of a program such as a word processor is the subjective opinion of the users. This relates particularly to its ease-of-use. To this end two forms of interviews were conducted with the evaluation subjects after they had completed the training exercises. The first interview was based upon the completion of a questionnaire. The second was more of a free-form interview but was not undertaken with the full set of subjects. The opinions of other researchers in related fields are also valuable. A number of such people have seen and commented on Soundtrack, and their comments are reported in this chapter. The last section of the chapter summarizes the conclusions of the interviews.

7.2 Structured interviews

These interviews were based upon a questionnaire. In order to overcome the obvious problems of written communication for the visually disabled subjects, the questions were read out by the tester and replies written down by him. In writing down responses, the tester endeavoured to get agreement from the subject as to the appropriate wording. There was, however, a problem in working in this manner. On the one hand, the tester ought to be a neutral recorder. On the other hand, clearer answers might be obtained if he was to pursue a question further, but then there was a danger of his introducing an element of bias. To have used a different person to administer the questionnaire would have reduced the possibility of bias, but at the price of possibly losing clarity in the replies.

The questionnaire form is included in Appendix C. It was divided into six sections. Their contents and the responses are summarized below.

1. Background.

The answers to this section were already summarized above, in Section 6.5. However, to recap briefly: All but one of the subjects were totally blind. The majority were familiar with using computer-based technology in their work or training. Two of them spent over six hours on the evaluation, while the remainder spent around 2¹/₂ hours.

2. Training

This section was included partly as a means of getting subjects' opinions on Soundtrack but in such a way as to depersonalize responses. In the guise of asking about how other people might be trained to use it, it was hoped to enable subjects to talk about difficult aspects of using the program without fear that they would be revealing their own inadequacies. These replies can be contrasted with those given later in the questionnaire which did ask for the subject's own opinion. Similarly, one question asked the subjects to estimate how long it would take a trainee to become proficient at using the program, where *proficiency* was defined as sufficiently skilled to be left to use it unaided. This estimate could be compared with the amount of training the respondent had themselves received.

Some of the questions in this section were genuinely intended to elicit information about training. Most of the subjects agreed that trainees should be given aids to learning, particularly with regard to memorizing the layout of auditory objects. These should take the form of braille notes, tactile diagrams or audio tapes depending on what is appropriate for individuals. It was interesting that all the subjects agreed that otherwise they would not arrange the training of other users differently from that which they had received.

The first question in this section concerned the minimum degree of previous experience the subject considered was necessary before a person could begin training on using Soundtrack. Two subjects considered that in most cases no previous experience was necessary but the

remainder all agreed that the ability to touch-type was necessary. There was a feeling, however, that there was a lot of scope for individual variation.

A problem with using the program which was mentioned frequently was that of remembering the layout of the objects within the auditory windows. Most subjects felt that the arrangement of the windows within the screen was not too difficult, but the sub-structure within the activated windows was. This problem will be discussed later.

3. Auditory objects

Many subjects agreed that the arrangement of the windows should be altered in such a way as to reflect their frequency of use, so minimizing the amount of mouse movement required. It was interesting that in remembering the layout as it was, nearly all the subjects used a verbal aide memoire, usually based upon the initial letters of the window names.

In question 3.5 subjects were asked about the pattern of the pitch of the tones. They were asked to describe it, so that it was possible to see whether they were aware of the pattern. They were also asked whether they found the pattern helpful in locating objects and whether they could suggest a more helpful pattern. Responses generally fell into one of two categories: those who thought that the pattern was helpful (though they were not always sure what the pattern was!) and those who did not use the pitch of the tones but merely counted the number of them. No one - from either group - could suggest a better pattern.

It must be pointed out that some of the subjects did not test the **Scroll bar** and **Thumb bar** objects in the **Document** window. (See Figure 5.2). As described earlier, the **Thumb bar** is actually the active object, which moves within the **Scroll Bar**. This was not a successful adaptation. One exercise (number 24) was included specifically to introduce the subjects to these objects and to test its operation. Subjects S1 and S2 attempted this exercise, but were unable to use it well. For this reason the exercise was not given to the other subjects. A proposal for a better design of a scroll bar mechanism is given in Section 11.4.

4. Overall view of the program

In this section some fairly general questions were asked about the subjects' impressions of the program. These are generally most informative when compared with answers to other questions. Opinions varied as to what aspects of the program the subjects liked. In response to a question about what they did *not* like it was interesting that most subjects did distinguish this from the aspect of the program they found most difficult. In order to roughly quantify subjects' opinions on the ease of use of Soundtrack they were asked what degree of encouragement they would need before they would use it in a job. Allowed responses ranged from "None. You would be happy to work with it." to "You would not be prepared to use it in a job." The majority chose the option that they would want to compare Soundtrack with alternative auditory word processors before choosing. For a number of the subjects, who already used another word processor, this response generally implied that they preferred their present system.

5. Added features

During the evaluation process a number of deficiencies of Soundtrack were identified by the subjects - and possible solutions suggested. Obviously the program as presented to different subjects could not be altered if their results were going to be compared. However, some software features were added which were optional, which could be switched on to be tested separately from the 'standard' version. These were tested informally. Additionally some of the subjects were able to try out a different piece of hardware - a bitpad and stylus used instead of the mouse (see Section 5.2). In this section of the questionnaire subjects were also asked what further modifications they would suggest making. One problem with using the mouse on an auditory grid is that it is easy to make inadvertent diagonal movements. It was suggested that some form of auditory warning of diagonal motion could be added. Informal testing of two forms of such additional feedback (as described in Section 5.8) was carried out by two of the subjects, S1 and S2. This suggested that using additional tones to signal diagonal movements was not practical, because it over-loaded the user's auditory

input. An alternative approach to solving the problem of inadvertent diagonal movements, using filtering of mouse movement signals, is discussed in Section 7.5.

6. Other comments

The final question was simply a prompting for any further comments. None of the subjects had anything of relevance to add, which suggests that the questionnaire had been sufficiently complete.

7.3 Individual subjects' responses

Subject S1 This subject thought that for would-be trainees some familiarity with computers and their terminology would help, and might avoid problems of "initial panic with new machinery". He thought that with sufficiently explicit instruction a trainee with little experience might cope, but that would depend on their level of intelligence. He estimated that training spread over one week would be sufficient to become "fairly proficient".

He seemed to think that the most difficult aspect to learn is the layout of the objects within windows. He said that the arrangement of the windows was not so difficult to learn, but the structure within activated windows was. This was complicated by the need to remember which window contains particular objects. His example was that if you want to open a document you must firstly recall that the **Open** control is part of the **File** menu. He largely confirmed this opinion in response to question 4.3 on what he found most difficult about the program. He said this was "memory". He hinted that this could be quite frustrating and could result in a loss of patience. It should be pointed out that he was answering the questionnaire shortly after (the morning following) having completed the most difficult exercise, during which he had shown signs of frustration. He suggested that a way of avoiding this would be to give people a lot of "play time" to freely familiarize themselves with the program. He considered that he had reached a stage where he could use it unaided

on a simple level. However, in answering this question he did once again mention the problems of becoming frustrated.

S1 had a fairly specific feature of the program which he disliked, which was the fact that the selection level of a document automatically becomes **Point** level as soon as anything is typed into the document. He recommended that the level should not be altered by typing. As for the question of what he did like, he replied, "It does work. You can do what you want to do - as you become more proficient". This seems to suggest that this subject thought it was a usable program. However, it has to be noted that he replied (in question 4.5) that he would not necessarily use it in a job if he might use a Frank Audiodata instead. Comments he made reinforced the suggestion that S1 suspected that Soundtrack would be more useful once the user had got over the initial learning 'hump'.

S1 used a verbal method to remember the arrangement of the windows. He did also have some kind of mental visual picture of a document window. This fact may be related to the fact that he was the subject who had most recently had normal sight. He had experienced some problems in using document windows because he felt that the **Level up** and **Level down** controls (see Figure 5.2) should logically be transposed. He said that he had found it easier once an explanation of their ordering had been given. As an extension of his aide memoire he suggested that the layout of objects might be easier to remember if they were arranged alphabetically according to their names - in menus at least. He did feel that some objects were too small, and suggested that windows could be expanded when activated. It is questionable whether this would be a viable suggestion since it implies violating the principle of using fixed-grid layouts.

Although he was able to correctly describe the tone pattern he said that he did not use the pitch of the tones to any great extent, but more the number of them. He suggested that a wider range of notes might be more useful. He also thought that the 'top' object in each window should have the same tone. It is not possible to say whether this would have been a

better arrangement of the tones. In fact, it seems that using simple tones of different pitches was not very helpful, so that their arrangement may be irrelevant.

S1 said that using Soundtrack made him feel tense and he would not like that if he had to use it in his job. He added that the type of work would be an important consideration and that for little jobs there was not much advantage using Soundtrack over using a typewriter. This reflects comments he made during Exercise 31. In that exercise he had made a large number of errors, due to his hands being displaced from the home keys and he commented that correcting this was so difficult that it might be quicker to re-type the whole document.

S1 was one of the subjects who tested the 'diagonal movement signal'. His opinion was that its usefulness would depend on its sensitivity. He thought that it "might" help. However, its usefulness was limited since there was another source of confusion in that it was possible to move between two (say) horizontally adjacent objects as intended, but still moving diagonally, causing the diagonal tone to be generated.

Subject S2 This man thought that getting used to using synthetic speech might be a problem for would-be trainees. He was used to hearing synthetic speech, being a regular user of a Frank Audiodata but in response to question 4.2 (on which aspects of the program he did not like) he again mentioned the quality of the speech. In particular, Smoothtalker does tend to 'swallow' the word "the". He considered that the ability to touch-type was sufficient prior experience, but also mentioned "Memory is the real problem". He also cited memory as the main problem he had experienced in learning to use the program, but he did not mention this in his answer to subsequent questions on training. Instead he suggested that trainee users would find it hardest to get used to the mouse. In particular he felt it was very sensitive to movement. This could cause odd, complex patterns of tones to be sounded, which were hard to interpret. Also there were problems of inadvertently moving diagonally. S2 tested the version of Soundtrack which incorporated diagonal signals, but he did not appear to find it an improvement. He said there were "too many sounds". Also he pointed

out that the way it was implemented it was not possible to hear along which diagonal he was moving. That is to say that he might, for instance, know from his kinæsthetic senses that he was moving the mouse to the right and hear that he was moving diagonally. He would not know whether the vertical component was upwards or downwards.

Having received 6¹/₂ hours of training himself, S2 considered that a trainee could become proficient at using the program within four half-hour lessons.

One aspect of the program which S2 liked was the fact that he did not have to rely on the voice. By this he presumably meant that the tones often provided sufficient information alone. This is backed up by his behaviour in the traced exercises, in which he was often sufficiently confident that he had located a target object that he did not bother to click the mouse for (spoken) confirmation. It was interesting that he was quite positive about the mouse in a way which seems more concerned with emotions than functionality. He stated that he "felt more *independent* using the mouse, rather than switches." He also felt it was "an extension of your arm", and that it did not feel part of the machine.

S2 used a mnemonic (*SIFE*) to remember the location of the menu windows (which were grouped in the middle of the screen). Other objects he "just remembered". He could not suggest any other arrangement which would be easier to remember.

S2 was able to correctly describe the tone pattern and was aware of the fact that the pattern was the same within windows as within the screen. He did think the pattern was helpful, especially when "whizzing along". He thought that the distinctive 'buzz' marking the screen edges was useful and said he used the edges to orientate himself.

S2 said that he would be happy to use Soundtrack in a job although he made two suggestions of improvements. The first was "When you activate a window could you change the sound, so you don't get so many beeps". Presumably this refers to the fact that when an window is activated, three tones are sounded. His second suggestion was "a

definite sound when something has been activated". This presumably means that he found the current protocol confusing, that although a sound is generated when any object is activated or executed, it is not sufficiently distinctive.

Subject S3. Interestingly, S3 thought that the major problem for a would-be trainee would not be with the completely new piece of hardware, the mouse, but with the keyboard, the layout of which was different from others with which he was familiar.¹ However, when asked what he had found most difficult he listed in order: "Remembering the layouts" and "Orientating around the screen". There is indeed no evidence from the other data that the layout of the keyboard gave S3 any particular difficulty.

S3 considered that training sessions should not be more than one hour long, and that after four such sessions a trainee would at least be in a position to ask questions to help their learning. He considered that he was in a position to use the program on his own, provided he had backup notes on the layout.

There were no aspects of the program which he said he liked. He disliked the "fiddliness" of having to take his hand away from the keyboard to the mouse. He also disliked having to find objects.

The tone interval between two windows which are vertically adjacent is much greater than between two horizontally adjacent ones. However, S3 was able to describe the tone pattern along a row but said he could not hear the difference between the two rows. He said that he did not use the tone pattern to help locate objects so it can be surmised that he was not actually paying a lot of attention to the pattern. He said that he did use the edge 'buzz' to orientate himself - as was observed in his exercise traces.

¹ It cannot be said that the keyboard configuration of this model of the Macintosh is non-standard since it conforms to an International Standards Organisation standard. It is rather that the majority of keyboards do not conform to the standard.

S3 said that he would be happy to use Soundtrack in a job. However, he did add later in the questionnaire that he could see "no advantage of this over other programs."

He said that he preferred the bitpad over the mouse - even without the overlay on the pad.

Subject S4. S4 considered that no one aspect of using the program was "particularly more difficult" for a would-be trainee to grasp. His own greatest problem, he said, was the size of the small objects and staying within them. This was probably related to problems of manual dexterity. These are apparent within some of the exercise traces. It was also observed by the tester and noted that it might have been partly due to the smallness of the subject's hands. He often used two hands to double-click: one holding the mouse still while the other pressed the button. Otherwise there was a tendency for the mouse to move between the two clicks so that they were treated by the program as two separate single clicks. His difficulties in double-clicking seemed at least as bad when using the bitpad and stylus. He commented that this appeared to be more temperamental, but that "It looks as if it *should* be better than the mouse."

S4's estimate of how long it would take to learn to use the program was "about the same time as to learn any other word processor with speech". Recall, however, that S4 was speaking as someone who was not himself so trained. He was certain that he could now use the program as long as he had notes to help him.

He liked the ability to work between two documents. He said there were no aspects of the program which he did not like.

S4 was the one subject who appeared to use the pitch of tones to locate windows. He said that he used the tones to remember the location of objects, combined with some spatial information, such as that the documents were on the left-hand side of the screen. He said he could not readily think of any better tone pattern. Furthermore, in response to the question

as to what additions or modifications he would make, he replied that he could not say that he found anything missing.

S4 said that he would want to compare other word processors before he would decide whether he would be willing to use Soundtrack in a job. He added at the end of the questionnaire that he was unsure how useful Soundtrack would be to a visually disabled person, compared to other, cheaper computers of which he had heard.

Subject S7. S7 considered that the hardest aspect for a new user to learn would be, "Understanding that there has to be an action point, normally represented by the cursor, which is where what is going to happen next will occur - particularly in a piece of text." He considered this concept to be "rather abstract". His own greatest difficulty was rather more specific. He found editing a document awkward because of the fact that a task often had to be tackled as a set of sub-goals. He mentioned in particular the frequent need to move up or down several selection levels. He contrasted this with the more direct action in Wordstar.

He estimated that with 2 hours per day instruction it would take about three days for a trainee to learn to use the program.

S7 liked the program's ability to work on two documents. He considered that this meant that Soundtrack was approaching the functionality of a corresponding visual program. His dislikes of the program were at a fairly philosophical level. He dislikes the apparent trend towards more pictorial information and away from verbal. Although he was speaking as a blind person, he considered that, even for a sighted person, to "resort" to using a diagram instead of words was an "admission of defeat".

Interestingly S7 claimed that he was aware of the different pitches of the tones, and said he did think they aided location of objects. However he was unsure about the description of the tone pattern, and in fact described it incorrectly. He (correctly) thought that tones were higher at the right-hand end, but (incorrectly) that the lower row was lower in pitch. He

strongly recommended that the notes for each object should be chosen to form musical arpeggios.

S7 said that he would need to look at other word processors - and assess Soundtrack further - before he would decide about using Soundtrack in a job. He stated that he did not prefer the bitpad to the mouse. He felt he had got used to the mouse and he did not like the idea of using non-standard hardware. He did add that the bitpad might be used as a teaching aid, presumably as a prelude to using the mouse.

Subject S8. S8 considered that the layout of the different objects was the most difficult aspect for a would-be user of the program. She said that if she were the trainer she would ensure that the student learned the layout of the windows within the screen first, before teaching the contents of the windows themselves. In describing her own difficulties she also mentioned the layout, but speculated that once she had learned the layout the "actual manipulations" might become the more significant problem. She added that she would need more practice yet before she would feel happy enough to use the program unaided, although she qualified that by saying that she does not have a lot of patience with technology, including devices such as Brailink and Optacon. Furthermore she was unwilling to make an estimate of how long it would take another person to learn to use Soundtrack. She merely stated that it "...depends on how quickly they learn".

Aspects of Soundtrack which S8 liked included the fact that it spoke. It should be remembered that in saying this she is someone who does not normally use devices with speech. However, she did note that doing word processing was quite slow. She also disliked the way the mouse moved freely on the table with nothing to which its position could be referenced.

S8 also used the initial letters of window names to remember their positions, but said she also had a picture of them in her mind. She had not extended this method to learning the layout within the windows and so had to use the mouse and speech more.

S8 could not describe the pattern of the tones and had not used the pitch to locate objects. This is perhaps a little surprising as she used to play the piano and flute and so might be expected to have a reasonably trained ear.

She has not had much experience of using word processors, and she replied that she would want to test other ones before she would decide about using Soundtrack in a job.

S8 preferred the bitpad to the mouse. She thought that the bitpad did give more of a frame of reference and found the stylus easier to hold and manipulate. She thought it should have an overlay, but one which included the details of the inner layouts of the windows. This is actually an impractical suggestion as the contents of windows varies during the running of the program, as discussed further in Chapter 10.

Subject S9. S9 said that the greatest difficulty a would-be trainee would have was "getting used to the program, learning what the different commands do." This was also the aspect of the program which he stated he found most difficult. It is interesting that this is a level deeper than most of the other subjects, who thought of the difficulty as being concerned with the screen layout. In Exercise 33, S9 appeared to be the most competent subject (see Section 9.3). It would seem therefore that he had mastered most of the mechanics of using the program and so was indeed able to think more about higher-level concepts. The layout of the screen was also the feature he liked least. He suggested that the layout ought to be rearranged to reflect the frequency of use of windows. In fact, he thought that the layout of the windows should be capable of being varied to suit individuals. Indeed, he suggested that less-used windows could be moved to a secondary screen. As it was, he did not use any particular method to remember the location of objects.

To train other users, S9 would not use half-hour sessions. He suggested that the length of sessions should in fact be tailored to individuals, possibly increasing in length as they progressed. Another strategy he suggested was to give the trainee an initial training session of "as much as they could take". They could then practise within their job and then have a

follow-up session to answer any questions which had arisen. He felt that he was not yet ready to use Soundtrack unaided. He felt that this was because the sessions had been too short and too far apart. The gap meant that he forgot a lot between sessions. It is interesting that S9 appeared to be more competent than the other subjects, yet all of them felt more ready to use Soundtrack unaided than he did. This presumably reflects a lack of self-confidence on the part of S9.

S9 could not describe the pattern of tones and did not use their pitch in locating objects but he thought that this would be useful for those who do have a musical ear.

A practical problem which S9 raised was that of using Soundtrack in a noisy office. In his own job he uses stethoscope headphones. He preferred these to big headphones which do reduce extraneous noise better, but which make him feel more cut off.

In replying to the question as to how willing he would be to use Soundtrack in a job S9 was definite that he would prefer to stick with the Frank Audiodata. He added that he was used to using that system and had invested a lot of effort on his own to reach his current level of competence.

S9 preferred the bitpad to the mouse. He pointed out that it was possible to pick the pen up and put it down anywhere - which "must be faster". He would like the addition of another button, so that one could be used for speech and one for activation, so avoiding the double-click problems.

He made a number of other suggestions for improvements. He would like there to be some means of finding out where you were in a document, in terms of page and line number. Also he would like an indication of the position within a line and suggested there should be a margin bell. He commented that the method of using sentence structure was contrary to that found in his work where "layout is everything to some people". Although the suggestion about giving page and line numbers would be of practical use, it might be difficult to

incorporate in an adaptation of a visual program in which such information is may not be available. The idea of adding a margin bell is also interesting. They are normally found on typewriters on which the typist must manually start a new line as their typing approaches the end of the current one. In a word processor, in which new lines are started automatically, such a warning about the imminent end of the line would seem unnecessary. However, this experienced word processor user considers they would be useful. Presumably he sees this as an aid to getting layout as correct as he is obviously expected to do in his job.

S9 remarked that the mechanism for selecting a file to be opened, via the **Get File Name** dialogue (as described in Section 5.6) was quite tedious, especially if the target file name came low in the alphabet. He suggested that it would be quicker to be able to type in file names. This is a sensible option, which is discussed further below, in Section 11.4. The dialogue was implemented in the way it was merely to test the feasibility of auditory lists which scroll.

7.4 Unstructured interviews

The two RNIB students, subjects S1 and S2, were again able to give more time to the study and so took part in a further form of an unstructured interview. This was intended to give them an open opportunity to give their opinions on Soundtrack. So that the subjects would feel less inhibited in expressing their criticisms, the interview was carried out by someone other than the system's developer - Dr Mark Elsom-Cook, who was acting supervisor of the project. The interviews took place several weeks after the subjects had completed their final practical exercises on Soundtrack. They did not reveal any significant new information which had not already come up in the other phases of the evaluation. To a large extent they confirmed the earlier results, although there was some change in emphasis, perhaps due to the time the subjects had had to reflect on the use of the program.

Subject S1

In the interview, S1 emphasized the comparison of Soundtrack with Wordstar on a Frank Audiodata. He said that he found Wordstar easier to use, although he qualified his opinion by saying that he thought it was partly a question of what one is used to. He said that his main obstacle in using Soundtrack was the mechanism for selecting text. As in the structured interview, S1 mentioned the word "frustration", this time identifying selection level changing as the major cause. He pointed out, with some perception, that editing involved large numbers of subsidiary problems and goals - like the changing of levels.

He remembered the layout of the screen well and said that he found using the mouse "okay" but he added that he preferred the *taso* sliders on the Frank Audiodata. He said that the speech quality was "not too bad", but suggested that a pronunciation dictionary could be added so that the intelligibility of exceptional words could be improved.

One problem which S1 mentioned, but which he had not brought up in the structured interview, was with the keyboard. This was partly due to its layout being different from that to which he was accustomed. That is a problem in moving to any new keyboard. However, there was also a problem more specific to Soundtrack - that of reorientation when moving the hand between the keyboard and the mouse.

Subject S2

Subject S2 also compared Soundtrack with Wordstar in the interview. He agreed that the latter had a wider range of commands and was better developed, but added that he thought Soundtrack was fine within its limitations. In particular he said that he liked the fact that Soundtrack gives feedback for every operation performed, whereas with audibly-enhanced Wordstar he did not always know if a command had been successfully executed. However he did prefer the *taso* sliders on the Frank Audiodata to the mouse.

He said that most of his problems were caused by problems moving around menus. He found it tedious to have to move frequently between a document and the menus. He resisted

any idea that windows should be enlarged, since he felt that this would slow down the operation. He also disliked the selection level concept, although he could not offer any suggestions of alternative approaches. He said that he used the tones to orientate himself, but relied on their number rather than their pitch.

He said that he noticed that when talking about the screen he found himself pointing at the desk on which the mouse had been. He said that he found this a bit disturbing.

7.5 Views of other researchers

In addition to the more formal evaluation, Soundtrack has been examined by a number of interested researchers - particularly during a visit to the United States. Comments fall into two categories: practical suggestions about improvements to Soundtrack and more general points. These discussions took place after the evaluation had been completed, so that the problems of using Soundtrack had been identified already.

Perhaps the most interesting comment came from Bill Gerrey, a researcher with a great deal of experience in the developments of aids for visually disabled people, who is himself blind. He described Soundtrack as "The darndest little bit of software I've ever seen."

One of the practical suggestions, which was made by two people, was that stereophonic sound might be used. Although hearing is less acute directionally, as discussed earlier, even such an imprecise hint to an object's location could be a valuable cue.

Another suggestion related to the problem of inadvertent diagonal movements of the mouse. Movement information from the mouse could be 'filtered' so that only vertical and horizontal movements are treated as significant. Figure 7.1 represents a possible diagonal mouse movement. The mouse has been moved four units horizontally, but there is also a vertical component of two units. A filtering program would treat this as a purely horizontal movement of four units.

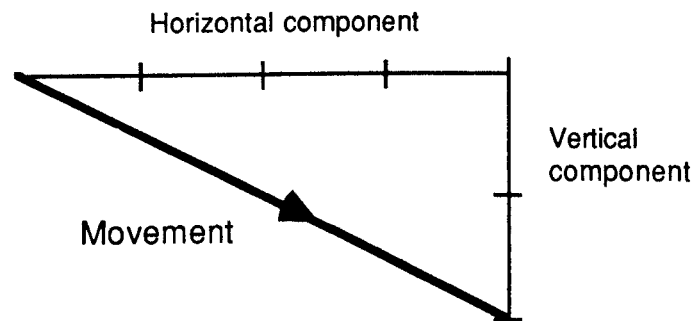


Figure 7.1. A diagonal mouse movement, resolved into vertical and horizontal components. The predominant direction is horizontal.

There would have to be design decisions made about the coarseness of the filter and whether diagonal movements should be allowed at all. For instance, how should a movement at exactly 45° be treated?

It was one of these other researchers (Stuart Card of Xerox) who pointed out that a fundamental difference between using a visual wimp interface and an auditory one is that the former is based on *recognition* whereas the latter relies on *recall*. That is the essence of the memory problem.

The question of whether it is possible to develop a generalized auditory interface to arbitrary wimp programs is discussed later. There is a suggestion that this is too complex with current technology. However, Tom Moran of Xerox made an analogy with the way modern systems have been developed so that they can be easily converted for use by speakers of different languages by changing the written contents of menus etc. A considerable effort had to go into the design of systems to make this possible. It was suggested that similar work would make adaptation of interfaces to different modes of interaction more feasible. In this way they might be made accessible to users with different disabilities.

7.6 Summary

Responses in the questionnaire were mixed. Criticisms and opinions varied widely, but the principal problem of using Soundtrack for the majority of subjects concerned remembering the layout of the screen and windows. The fact that subjects had different ideas about how Soundtrack might be improved does suggest that there is a lot of scope for 'customization' of such an interface. Ideas of how this might be implemented are discussed in Section 11.4 on possible developments of Soundtrack.

That most of the experienced word processor users would prefer to use their old system rather than Soundtrack is neither surprising nor alarming. Soundtrack is a prototype program with many recognized deficiencies. It seems most likely that conventional, non-wimp computers are to disappear, so that the common comparison - Wordstar on a Frank Audiodata - will not be available. It will then be *necessary* for visually disabled users to have some form of adaptation of wimp software. In any longer-term study it would be interesting to introduce Soundtrack to people who had not used any word processor previously. These users could be trained to a level of expertise and would not have any prejudices about the nature of word processors.

Some of the subjects mentioned the idea of giving would-be trainees free time to use Soundtrack on their own. This does seem to be a good idea. Such practice can be very important when using any computer program and has the advantage that the person can work in their own way without the anxiety induced by being observed by someone else. This would form an important part of any training of the complete novices mentioned above.

S2 appeared to have been confused by the number of tones which can be sounded. He was correct in suggesting that the tones on activation of a window were more complex than necessary. As explained in Section 5.3, when a window is activated its tone is sounded twice. However, now it is active the tone for the object within the window is also sounded. Thus three tones are heard. There seems no reason why the window's tone should not be

sounded just once. That would mean that the activation action could still be recognized - by the sounding of two tones.

There was one comment which was made by almost everyone who used Soundtrack during the exercises, which was recorded on the exercise record sheets - but not specifically covered by the questionnaire. It was the suggestion that there should be some kind of frame supplied within which the mouse would be moved. One advantage of the bitpad was that it did provide a physical reference, which could be supplemented by overlays, and this was covered by the questionnaire. The fact that this was such a frequent comment is significant. The practical problems of providing such a frame are quite serious and these are discussed later, in Section 10.5.

In fact opinions were divided on the use of the bitpad. Three of the five subjects who tested the bitpad did prefer it. One valid objection made by one of the subjects who preferred the mouse was that to use a bitpad implies adding a non-standard item of hardware, whereas it is desirable to provide adaptations based on software whenever possible. It so happens that there is a proprietary bitpad available for attachment to the Macintosh which will replace the mouse, but this is not true of all wimp-based computers and the aim of this project was to investigate the more general case. (The question of how adaptations should be implemented is discussed more fully in Chapter 10.)

Considering the fact that all the subjects felt a need for a physical frame of reference for the mouse, there is a question as to why they did not all prefer using the bitpad. It might be that the bitpad did not provide the sort of reference they envisaged. Alternatively it may be that any such frame would not be as useful as people supposed; that interacting with sounds is a novel and so disturbing experience, but that people can get used to it and no longer need the support of a physical frame.

The use of simple pitch variations to provide spatial information does not seem to have been very successful. Most of the subjects did not use the pitch of tones to aid their navigation

around the screen. However, none of them seemed to find the variation in pitch a disadvantage and one of them did find them useful. There is scope for using other patterns and modulations of sounds. More work ought to be carried out to find better forms of sound to use.

There is a possibility that there might have been some bias in the results of the structured interviews since the subjects knew that the tester who was filling in the forms was also the program's developer. However, this possibility is belied by the results of the free interview which was conducted by someone else, and which produced very similar responses. Indeed, the (unstructured) interviews produced no significantly different information apart from S1's suggestion of the addition of a pronunciation dictionary. This is a practical suggestion which might be taken up in future developments (see also Chapter 11).

The interview results reported in this chapter are the most important indicators of the ease-of-use of Soundtrack. Although there seems to have been general agreement that it is a usable program, a number of problems were clearly identified. Principal among these were difficulties in remembering the layouts of screen objects. One limitation of this evaluation is that it has effectively been confined to investigating the learning stage of use. This was due to practical considerations of getting volunteer test subjects. The results might be different if experienced users of Soundtrack were interviewed.

A number of practical suggestions about the nature of the problems identified by the evaluation subjects and of possible solutions to them have been made by other researchers. Some of these should be acted on in future developments, as suggested in Chapter 11.

The interviews were conducted after the subjects had completed their training in using Soundtrack. Other data was obtained while they were using it and this is described and analysed in the following chapters.

Chapter 8

Timing data

8.1 Modelling the interaction

As mentioned above, in Chapter 6, one of the aspects of using Soundtrack which should be measured is the time taken to perform actions with it. In the first instance timing data will give an indication of whether Soundtrack will be usable at all; whether the time taken to accomplish editing tasks is reasonable. Assuming it is usable, measurements of times give a basis for comparison with other word processing programs, at a detailed level. If regularities can be detected in the data obtained it will be possible to generalize the results, giving them a wider usefulness. A great deal of work has been carried out already on pointing behaviour of sighted computer users, and the work described in this chapter may form the basis for extending that to the use of auditory interfaces by blind people.

Since the novel aspect of the program is the use of the mouse it was decided that measurements should be made of the time taken by subjects to locate objects on the auditory screen. The first part of this chapter presents the derivation of a model of pointing behaviour. The following sections describe the method by which timing data was obtained. This data fits well with the proposed model, and on that basis it is possible to analyze in some detail the nature of the interaction.

To measure the time to locate an auditory object it is necessary to specify the distance between objects. For the purposes of these tests a generalized method of measuring distances on a grid was used. The distance between two objects is measured in terms of the

number of intervening objects. The distance is defined as the sum of the number of objects crossed horizontally and the number crossed vertically. So, for instance, referring to Figure 8.1, the distance between windows A and G is 3, that is 2 horizontally plus 1 vertically.

A	B	C	D
E	F	G	H

Figure 8.1. A grid, illustrating the method used to measure distances.

Locating a target involves performing a series of sub-actions. These are identified as follows:

1. Choose target This refers to the action of deciding where the next mouse action should occur. This can be quite a complex decision. Normally it will involve the following steps:

1. Deciding what the next task is (e.g. opening a file, cutting some text, responding to a dialogue etc);
2. recalling the method to achieve that task;
3. deciding what the first sub-task of that method is;
4. recalling which object corresponds to that action.

2. Plan route The user must recall the position of the chosen target relative to the current location of the mouse. This involves the subject remembering the screen layout so that she may decide how many objects to move over horizontally and how many vertically.

3. Move The mouse is moved to the next (adjacent) object. Inherent in this action is the sounding of that object's tone. The amount of information derived from that sound depends

on the individual user. At a minimum users will rely on it to signal that they have entered a new object. Most will also use it as an indication of their progress along their planned route, in terms of the *number* of beeps they have heard. Some may derive more information about their current location from the pitch of the tone.

4. Click mouse As mentioned above, this will often be done only when the user thinks she has reached the target. It will also be done if she gets lost and disorientated.

The diagram in Figure 8.2 illustrates a method which it is hypothesized that people would use to locate an object on the auditory screen. The boxes in the diagram represent the sub-actions. One cycle of the diagram represents the movement from one screen object to an adjacent one. That implies that movement over a distance d would involve (at least) d cycles. Notice that most of the sub-actions are optional. Typically the **Choose target** and **Plan route** actions will occur only in the *first* cycle of a movement, whereas the **Click mouse** will only occur once the target has been reached.

For example, for a movement of distance 2, a typical sequence would be:

Choose target	}	Motion / Cycle 1
Plan route		
Move		
Move	}	Motion / Cycle 2
Click mouse		

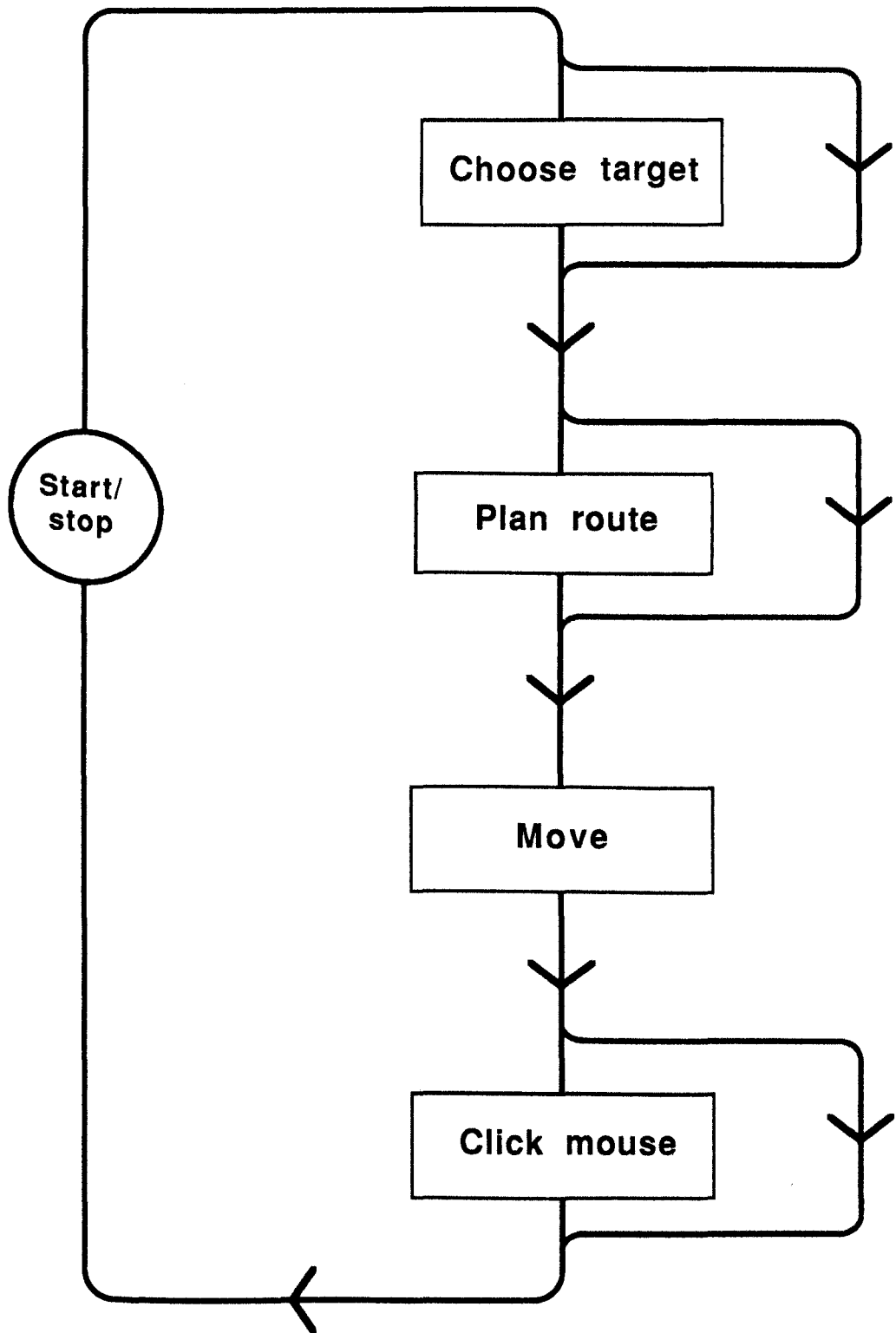
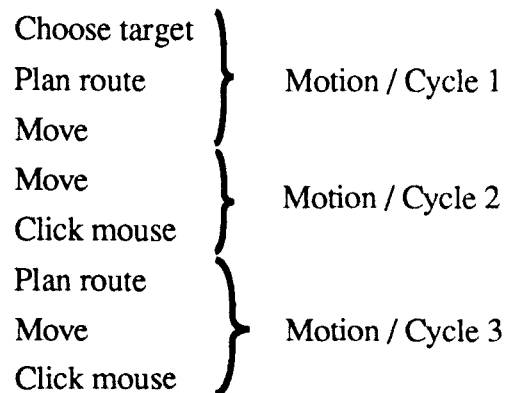


Figure 8.2 The method used to locate objects on the auditory screen.

The above description is quite general. It does, for instance, take account of errors such as the user 'missing' the target. Should she think she has hit it, and so press the mouse, she will recognize the error. She would then recycle around the method. So, in such a case the sequence might be:



Such a movement will be referred to below as an *error*.

Within the above model the edges of the screen can be considered to be another object. In practice it differs from other objects in that its tone is sounded continuously, as long as the mouse is within it (off the screen), rather than giving just a short-lived beep. This does affect users' behaviour, as is shown later.

The time taken to locate an object on the screen can be described in terms of the model embodied in the diagram in Figure 8.2. The time to locate an object will depend on the number of cycles around the diagram and the choices made in each cycle. Direct movement to the target by the shortest route implies executing one cycle which includes the **Choose target**, **Plan route** and **Move** operations, followed by zero or more cycles involving only **Move** and completed by one **Click mouse**. So, the time to position the mouse within a target, at a distance d , can be described thus:

$$\begin{aligned}
 T_{\text{position}} &= T_{\text{choose}} + T_{\text{plan}} + T_{\text{move}} && \text{[cycle 1]} \\
 &+ (d-2)T_{\text{move}} && \text{[cycle 2 to cycle d-1]} \\
 &+ T_{\text{move}} + T_{\text{click}} && \text{[cycle d]}
 \end{aligned}$$

Thus

$$T_{\text{position}} = T_{\text{choose}} + T_{\text{plan}} + dT_{\text{move}} + T_{\text{click}}$$

where

T_{choose} is the time to choose the target;
 T_{plan} is the time to plan the route;
 d is the distance to the target;
 T_{move} is the time to move to the next window;
 T_{click} is the time to click the mouse.

If simplifications are made, it is possible to calculate T_{choose} by applying Hick's Law (Welford, 1968). The simplifications amount to making the selection of the next target window a straight, one-from-eight choice. In other words there would be no need for the complex mental processing implied by steps 1 to 3 in the **Choose target** sub-action described above. As described below, this simplification was achieved in one exercise within the evaluation (Exercise 0.1), by the tester telling the subject which window to move to next. Hick's Law states that the time taken to make a choice from n equally probable alternatives is given by:

$$T_{\text{choose}} = I_c \log_2 (n + 1) \quad (1)$$

where I_c is a constant, found to be of the order of 0.15 s bit^{-1} (Welford, 1968).

T_{plan} and T_{click} can be considered to be constant, as is T_{choose} if n is fixed

Thus,

$$T_{\text{position}} = T_{\text{think}} + dT_{\text{move}} \quad (2)$$

where

$$T_{\text{think}} = T_{\text{choose}} + T_{\text{plan}} + T_{\text{click}} = I_c \log_2 (n+1) + T_{\text{plan}} + T_{\text{click}}$$

T_{think} is a constant, representing the time component during which the mouse is *not* moved. It includes both the time taken preparing to move and that spent in checking the final position - by way of a mouse click. If T_{position} is plotted against d a straight-line graph should be obtained, from which it should be possible to calculate values of T_{think} and T_{move} from the intercept and slope respectively.

8.2 Relationship to models of sighted targetting

A comprehensive explanation of the behaviour of sighted people locating screen objects with a mouse is embodied by Fitts' Law (Fitts, 1954). One derivation of this law is based upon a

psychological model of the processes involved, called the *Model Human Processor* (Card, Moran and Newell, 1983). The Model Human Processor is divided into three interacting sub-systems: the *perceptual system*, the *motor system* and the *cognitive system*. It is assumed that the movement of a mouse towards a target is not a continuous process, but consists of a series of micro-corrections with a certain accuracy. Thus the subject moves the mouse in the direction of the target, observes any deviation from the optimal trajectory and so makes a correction. These processes are repeated cyclically until the target is reached. To make a micro-correction takes a minimum of one cycle of the Perceptual Processor to observe the hand, one cycle of the Cognitive Processor to decide on the correction and one cycle of the Motor Processor to perform the correction. Hence the time taken to perform one micro-correction is given by:

$$T_{\text{cycle}} = T_P + T_C + T_M$$

where

T_P is the period of the Perceptual Processor;

T_C is the period of the Cognitive Processor;

T_M is the period of the Motor Processor.

It is found that T_{cycle} is of the order of 0.240 s (op. cit.).

On the basis of the model and assumptions given above it is possible to derive Fitts' Law:

$$T_{\text{position}} = I_M \log_2 (2D / s)$$

where

$$I_M = - T_{\text{cycle}} / \log_2 e$$

and

D is the (linear) distance to the target;

s is the size of the target;

e is the error in each cycle (assumed to be constant).

It is found (op. cit.) that e has a value of the order of 0.07, so that

$$I_M = 0.063 \text{ s bit}^{-1}$$

Fitts' Law has a useful generality. It can be assumed that all sighted people using a mouse interact in much the same way. Such universality cannot be applied to systems using an auditory form of interaction. In an auditory system, the information fed back to the user

depends on the auditory protocol implemented. For a sighted person, the feedback on their progress towards the target is essentially continuous. It would be possible to derive auditory protocols with this property also - for instance a continuous tone could be sounded whose pitch varies with the proximity of the mouse to the target. However, the protocol implemented in this project (the sounding of a beep as a new object is entered) does not have this property.

In consequence, there is very little similarity between the processes involved in a sighted person using a mouse and a blind person using the auditory version. It has been pointed out that both interactions involve a series of cyclic processes, rather than a continuous action. However, there the similarity ends. The cycles involved in a Fitts' Law interaction are on a much smaller scale in time and space than in the auditory case. According to the above-mentioned figures, the sighted person receives feedback on their progress towards the target approximately every 0.24 s. With the form of auditory protocol used in this project, however, the user receives audible feedback only after a gross movement. It is shown later in this thesis that this takes of the order of 0.7 s.

Another incompatibility has to do with the contribution of kinaesthetic information. Card, Moran and Newell (1983) take no account of any kinaesthetic contribution to the Perceptual System, but consider only sight and hearing. In aiming a mouse it is probably true that the visual feedback is so useful and precise that it overwhelms any contribution from the muscles; not only can a sighted person watch the progress of the cursor towards the target on the screen, but also they may be aware visually of the direction of motion of the mouse - out of the corner of their eye, as it were. For a blind person this is clearly not true. Information on the motion of the mouse from the non-visual senses is vital. If the auditory protocol employed provides a discontinuous form of feedback (as in this project) then kinaesthetic and tactile information must be relied on during the periods of silence. It will be shown later in this chapter that this reliance on this (imprecise) form of feedback was an aspect which the blind evaluation subjects found most disconcerting about using this system.

8.3 Measurements of location times

In order to obtain information on the time taken to locate objects one exercise was given repeatedly during the evaluation. Because it was not part of the graded sequence of exercises, it was numbered Exercise 0.1. In fact, usually from the second session onwards, most subjects completed Exercise 0.1 at least once in every session. In this way it was possible to generalize about timings by way of averaging measurements and to observe any trends in performance. The exercise was completed with tracing on at the end of each session. It was always ensured that the subject had already re-familiarized herself with the layout of the screen earlier in the session - either through other exercises, or by completing a practice Exercise 0.1 without tracing on.

The exercise was carried out as follows (see also the exercises sheets in Appendix B). All windows were de-activated. The tester spoke the name of a window which the subject then attempted to find. When they believed they had located the window they pressed the mouse button. If the speech elicited indicated that they were not in the correct window they would continue searching in this way until they were. As soon as the subject was in the correct window the tester would speak the name of another window, and the process would be repeated. This was repeated for *nine* windows. The order of the list of windows to visit was random, and different lists (identified by letters A to J) were used in each presentation of the exercise.

Document 1	Speech Menu	Interface Menu	Dialogues
Document 2	Edit Menu	File Menu	Alerts

Figure 8.3. The layout of the auditory screen (also shown in Figure 5.1).

Within this exercise, the distance between the mouse starting point and the target window was measured according to the general method outlined earlier. The distances between each of the windows - shown in Figure 8.3 - are defined as in Table 8.1.

	Doc. 1	Speech Menu	Interface Menu	Dia- logues	Doc. 2	Edit Menu	File Menu	Alerts
Document 1	0	1	2	3	1	2	3	4
Speech Menu	1	0	1	2	2	1	2	3
Interface Menu	2	1	0	1	3	2	1	2
Dialogues	3	2	1	0	4	3	2	1
Document 2	1	2	3	4	0	1	2	3
Edit Menu	2	1	2	3	1	0	1	2
File Menu	3	2	1	2	2	1	0	1
Alerts	4	3	2	1	3	2	1	0

Table 8.1. Distances between auditory windows.

Subjects do not always move directly to the target window. Two forms of deviation are recognized. If the subject clicked the mouse in any window other than the target it is deemed that they made an *error* as mentioned above, in Section 8.1. If they did get the target but passed through more than the minimal number of windows it was said to be *indirect*. For the purposes of these measurements both errors and indirect movements were ignored - they are considered separately, in Section 8.5. Also the timing of the first movement was discarded since the subject is likely to need time to orientate themselves.

The time taken to locate the target window, T_{position} , is plotted against distance, d , in Figure 8.4. It is a limitation imposed by the 4 x 2 window layout of the screen that only four points have been plotted.

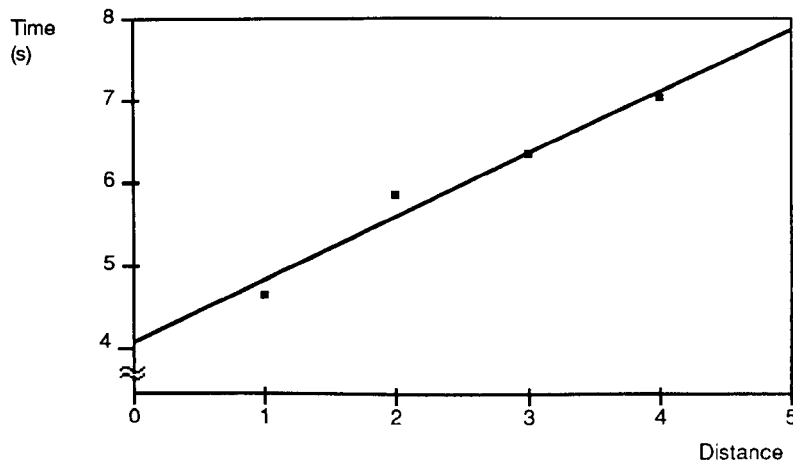


Figure 8.4. Plot of time to locate auditory windows against distance to the window.

From the graph it is possible to calculate T_{think} and T_{move} as the intercept and slope. This yields values of

$$T_{\text{think}} = 4.15 \text{ s}$$

$$T_{\text{move}} = 0.73 \text{ s.}$$

Exercise 0.1 does not precisely represent the sort of interaction which someone using Soundtrack to do real editing would be involved in. Within Exercise 0.1, T_{choose} (which is a component of T_{think}) will be smaller than in a realistic situation. In the exercise T_{choose} represents the time taken for the subject to hear the name spoken by the tester, but during practical use of the program the corresponding task would normally involve the recall of an algorithm to achieve the current (sub-) goal. Subjects in the evaluation did complete some less artificial exercises (see Chapter 9) but it is not possible to extract comparative data from these. Timings in these exercises are affected to too great an extent by irrelevant influences (the subject talking to the tester, for instance) - which is, after all, the reason for presenting artificial, more constrained exercises, such as Exercise 0.1.

However, given the manner in which Exercise 0.1 was presented, Hick's Law (Equation 1) can be applied to calculate T_{choose} . Given that

$$I_c = 0.15 \text{ s bit}^{-1} \text{ and } n = 8,$$

$$\text{then } T_{\text{choose}} = 0.48 \text{ s.}$$

$$\text{Recall } T_{\text{think}} = 4.15 \text{ s}$$

$$\text{and } T_{\text{think}} = T_{\text{choose}} + T_{\text{plan}} + T_{\text{click}}$$

From the data obtained in Exercise 0.1, a mean value for T_{click} was obtained.

$$T_{\text{click}} = 0.69 \text{ s}$$

$$\text{Thus } T_{\text{plan}} = T_{\text{think}} - T_{\text{choose}} - T_{\text{click}} = 4.15 - 0.48 - 0.69 = 2.98 \text{ s}$$

These component times are summarized in Table 8.2.

Description	Symbol	Value (s)
Move mouse to adjacent window	T_{move}	0.73
Click mouse	T_{click}	0.69
Choose target window	T_{choose}	0.48
Plan route to target window	T_{plan}	2.98
($T_{\text{choose}} + T_{\text{plan}} + T_{\text{click}}$)	T_{think}	4.15

Table 8.2 Summary of timing data. Note that all the times were obtained from experimental data *except* T_{choose} , which was calculated on the basis of Hick's Law.

8.4 Discussion of the model

The points plotted in Figure 8.4 lie close to a straight line, which suggests that a linear equation, such as Equation (2), *is* appropriate. Whether it represents an accurate model was ascertained through other measurements and analyses.

Recalling that Equation (2) is derived from the method illustrated in Figure 8.2, it can be assumed that the first action after a click of the mouse includes **Choose target** and **Plan route**. So, by comparing the times of movements (cycles) which occur after a mouse click

with those which do not, it is possible to get some idea of the time difference for these alternative cycles. Averaging these times over all the subjects yields the following means:

Average movement time *following* a mouse click: 4.28 s

Average movement time *not* following a mouse click: 0.90 s

These figures compare quite closely with those obtained above for T_{think} and T_{move} , so confirming the accuracy of the model.

Equation (2) embodies the assumption that the subject moves at a constant speed across windows. This may be thought to be an invalid assumption. It might be, for instance, that a subject would be more likely to move slowly across the first window and then more quickly across any subsequent ones. In order to test this, movements over distances of 2 or greater were examined and split into the time taken to move into each of the windows in the path. The results of this analysis are shown in Figure 8.5.

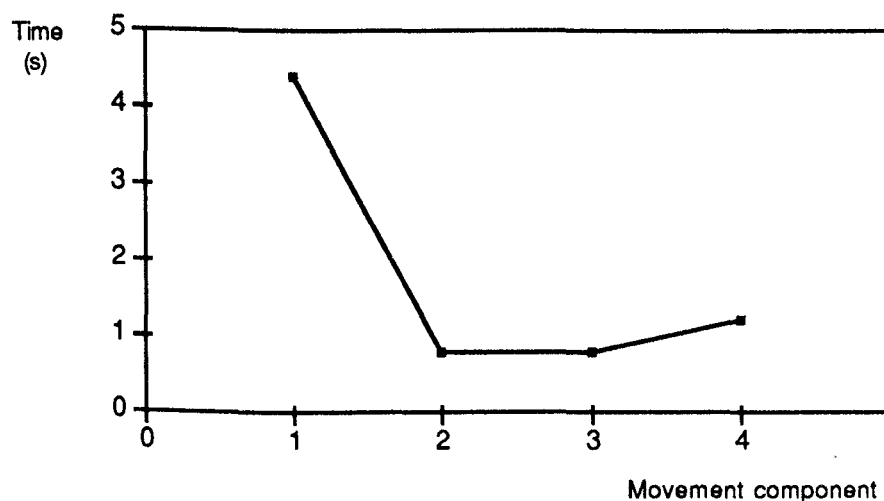


Figure 8.5 Time taken to move across each of the windows in movements in which $d \geq 2$.

As is clear from Figure 8.5, the constant speed assumption is reasonable. The time for the first component is longer, as would be expected since it includes T_{choose} and T_{plan} , and the time for subsequent moves is approximately equal. The actual mean times are given in Table 8.3. The times for movement components 2 - 4 are the same as those mentioned above as

occurring *not* following a mouse click. Their mean value is 0.93s, and the standard deviation is 0.24. In other words, measured T_{move} is approximately constant, and further support is lent to the proposition that Equation (2) is a good model of the interaction.

Component	Time
1	4.40
2	0.80
3	0.78
4	1.20

Table 8.3. Times (in seconds) taken to move across each of the windows in movements in which $d \geq 2$.

All the subjects moved the mouse towards the target in separate horizontal and vertical components. It might be thought that movements in the two directions would be separated by a pause. However, this was not found in the timings. In fact one problem which subjects had in using Soundtrack was that it is easy to *inadvertently* move the mouse diagonally. So-doing can be very confusing to a person who does not use the pitch of the window tones for orientation. For instance a user may have the mouse in **Document 1** (see Figure 8.3) and may wish to move to the **Speech** menu. They move in what they assume to be a horizontal direction, but instead of going into the target window they cross the corner of **Document 1** and enter the **Edit** menu.

The question of whether peoples' performance will improve with practice is discussed later, in Section 10.2.

8.5 Errors

The numbers of errors and indirect movements are expressed as a percentage of the total movements tested, since different subjects completed Exercise 0.1 a different number of times. The overall means were as follows:

Error movements 18%

Indirect movements 12%

Individual subjects' error rates are discussed below.

8.6 Individual data from Exercise 0.1

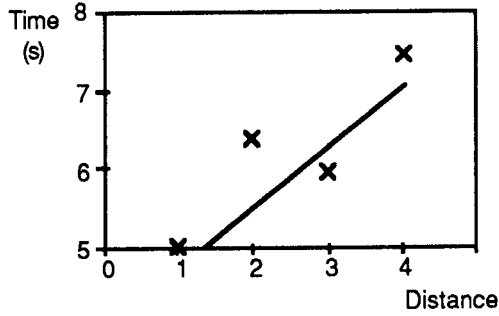
Individual subjects' mean location times are given in Table 8.4, along with the overall mean times. Little significance could be attached to the performance of individual subjects on Exercise 0.1 since they did not complete the exercise a sufficient number of times. All of them except Subjects S1 and S2 completed Exercise 0.1 with tracing on just three times. That represents 24 separate motions. Fewer measurements were made at distance 4, simply because there are fewer routes on the screen of that length.

		Subject							
		Mean	S 1	S 2	S 3	S 4	S 7	S 8	S 9
D i s t a n c e	1	4.67	5.05	3.95	5.33	6.47	5.06	4.44	4.76
	2	5.85	6.38	5.29	5.97	7.39	5.83	5.47	6.22
	3	6.35	5.97	5.78		8.19	9.10	5.90	6.92
	4	7.03	7.44	6.47	7.37	6.40	7.67	6.94	

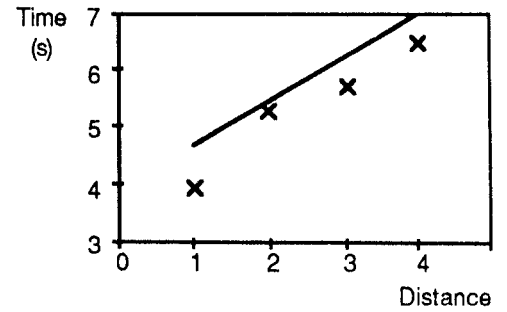
Table 8.4. Mean times (in seconds) taken by individual subjects to locate windows at different distances and the overall mean times. Blank entries signify distances for which the subject did not complete any direct locations.

The most useful information which can be gleaned from this data is the overall average location times, since these can be used to make hypotheses about people's likely behaviour in using Soundtrack, as was done above. The times recorded for individual subjects are compared with the overall means in Figure 8.6. Points are plotted for each subject of the time to locate windows at each of the distances (1 to 4). Also plotted, as a line is the mean time for all the subjects as a way of comparing performance. So, for instance, if a subject's

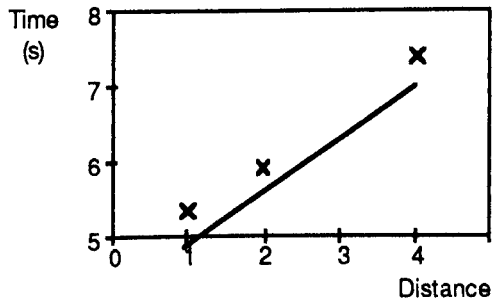
points tend to be above the line then that subject was generally slower than average, and vice versa.



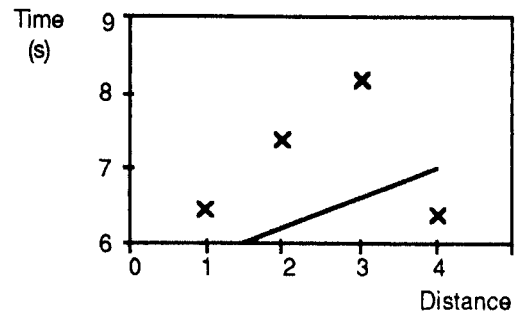
A Subject S1



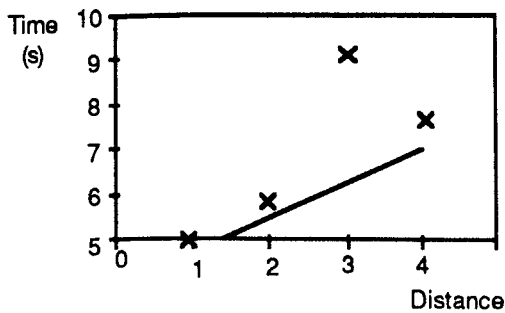
B Subject S2



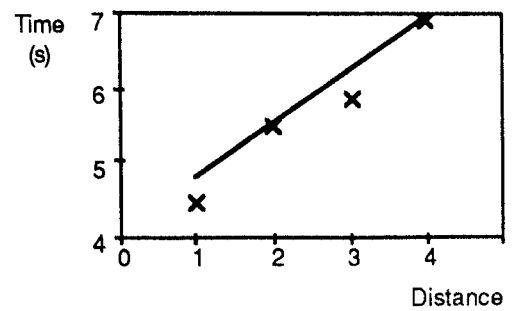
C Subject S3



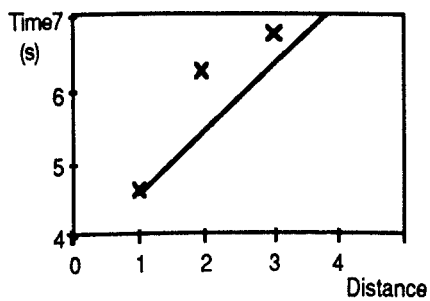
D Subject S4



E Subject S7



F Subject S8



G Subject S9

Figure 8.6

Individual times for locating windows. The subject's times are plotted as points, which can be compared with the lines, which represent the overall mean times.

Table 8.5 summarizes individual rates of error and indirect movements. These are given both as absolute numbers and rates expressed as a percentage. The rate expression takes account of the fact that individual subjects completed Exercise 0.1 different numbers of times. On the other hand, many of the numbers are so small that the rate does not have a lot of meaning. For instance, to say that S1 had an error rate of 3% might be misleading since that represents a single error.

	Errors		Indirect	
	Num-ber	Rate (%)	Num-ber	Rate (%)
S 1	1	3	3	8
S 2	0	0	1	2
S 3	10	42	3	12
S 4	4	17	6	25
S 7	3	12	5	21
S 8	5	21	1	4
S 9	7	29	3	12

Table 8.5. Summary of individuals' errors and indirect movements.

There several features of interest amongst these results, which are discussed below.

Subject S1 This subject was tested only *once* at distance 4, so that his time at that distance (Figure 8.6A) is not an average. All three of his indirect movements occurred in the same session (number 5).

Subject S2 Subject S2 was slightly quicker than average at locating windows, as shown in Figure 8.6B. His mean time to move after a mouse click (4.24s) was close to the overall

average, but movements not following a click were quicker than average (0.55s). This suggests that his manual dexterity was better than average.

Subject S3 There is no time for distance 3 in Figure 8.6C since S3 did not complete any direct movements at that distance. His error rate was very high. Some of those errors were quite large. Locating a window sometimes involved incorrect mouse clicks in more than one window as well as going off the edge of the screen. However, this subject *does* appear to show an improvement in terms of his error rate, as shown in Figure 8.7.

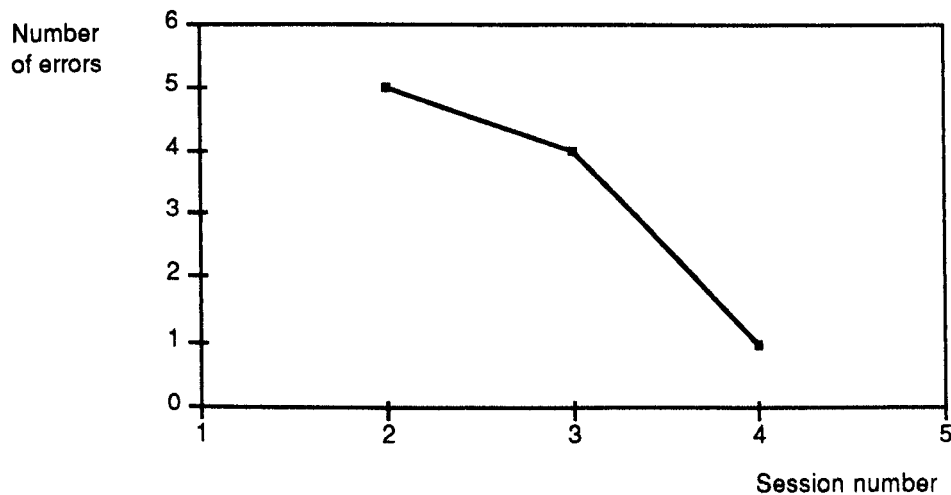


Figure 8.7. Number of errors made by subject S3 in each session. (Note that Exercise 0.1 was not given in session 1).

Subject S4 S4 was very interesting in that he was the only one of the subjects tested who used the absolute pitch of tones to locate windows. This means that he did not use the method described in Figure 8.2. Hence his results do not fit in with Equation 2, as is evident from Figure 8.6D. Using the data obtained there appears no obvious model describing the method he used. It seems that the subject would move the mouse back and forth between windows in order to force a window's tone to be repeated, so that he could hear how it matched the target tone he had in mind. From the records taken, it is impossible to distinguish such movements from those which were deliberately aimed in a particular direction. The implications of this variation of method are discussed further below, but it can be seen from Figure 8.6D that generally he was somewhat slower than average. This

was mainly due to the method he used to locate windows. The fact that he frequently used such a method is reflected in his high rate of indirect movements. Though slow, this method does not appear to have been any less accurate for this person, since his error rate is almost exactly average.

Subject S7 Looking at Figure 8.6E suggests that S7 was significantly slower at distance 3. However, he only completed a move of distance 4 once, so that the time given for that distance is not a mean. If one were to discount the time at that distance, the remaining points do appear to be almost linear, but significantly slower than the overall mean.

Subject S9 Subject S9 did not complete any tests at distance 4 (as shown in Figure 8.6G); he attempted two locations at that distance, but one was an error and the other was indirect.

8.7 Locating smaller objects

In order to measure any effect of the size of the auditory object to be located on the time taken a further exercise was devised. This was numbered Exercise 0.2. Because subjects S1 and S2 were able to give more time to the evaluation than the others this exercise was given only to those two. This exercise was similar to Exercise 0.1, except that instead of locating windows within the auditory screen, one of the windows was activated and the subject located objects within that window. In fact a document window was filled and activated to be used in this exercise. This was chosen because it is one of the more complex layouts but it was also a window used frequently in the other exercises so that subjects had a lot of practice in using it. The **Scroll bar** and **Thumb bar** objects (see Figure 8.8) were not used because they were not introduced to the subjects until late in the exercises. The procedure for the exercise was identical to that used in Exercise 0.1. Names of objects within the document window were read from a list in which the ordering was random.

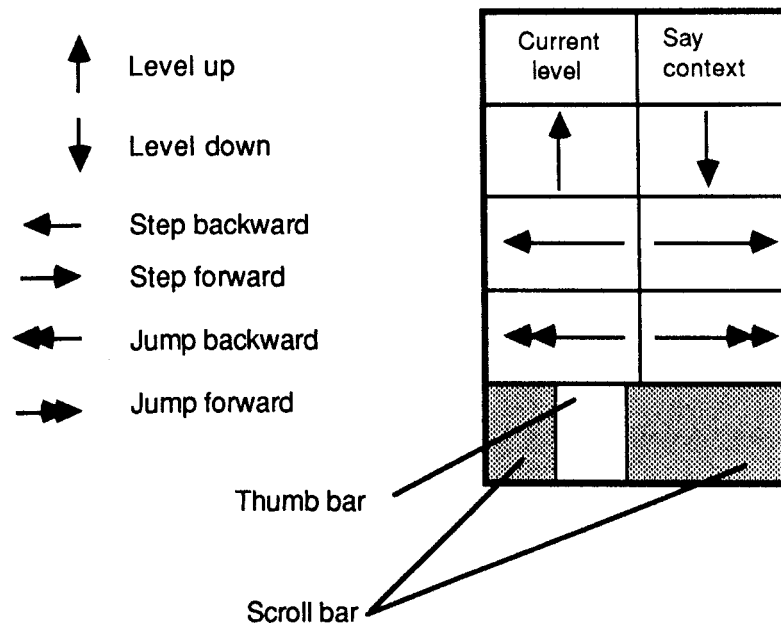


Figure 8.8 The Document window (Also shown in Figure 5.2).

It might be expected that the values of T_{choose} , T_{plan} and T_{click} - and hence of T_{think} - obtained in Exercise 0.2 would be approximately the same as in Exercise 0.1. On the other hand, one would expect T_{move} to be smaller, due to the shorter distances moved by the mouse.

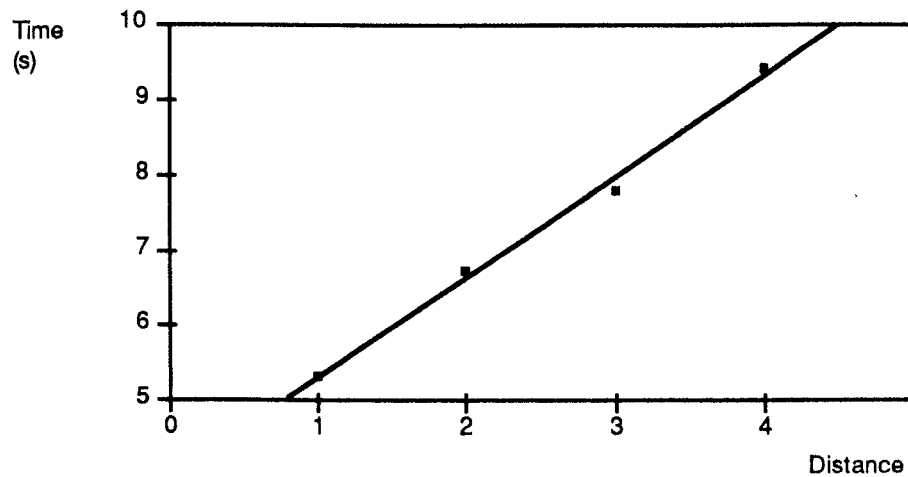


Figure 8.9. Mean time for subjects S1 and S2 to locate objects within a Document window - in Exercise 0.2.

However, when location time is plotted against distance, as in Figure 8.9, the following values are obtained:

$$T_{\text{move}} = 1.35 \text{ s}$$

$$T_{\text{think}} = 3.96 \text{ s}$$

$$T_{\text{plan}} = 2.79 \text{ s}$$

The value of T_{think} , and hence of T_{plan} , are almost the same as in Exercise 0.1, as expected. Yet the value of T_{move} is more than fifty-percent *larger* than in Exercise 0.1. It might be that the large value of T_{move} is due to subjects S1 and S2 being slower than average at moving the mouse. However, if the mean times for just those two subjects in Exercise 0.1 are plotted against time, the following values are obtained:

$$T_{\text{move}} = 0.64 \text{ s}$$

$$T_{\text{think}} = 4.16 \text{ s}$$

$$T_{\text{plan}} = 2.99 \text{ s}$$

In other words, on average, subjects S1 and S2 do not appear to be any slower at moving the mouse when locating larger objects. This result suggests that when locating smaller objects the subjects really did move the mouse more slowly. This may have been due to the need to move it more precisely. The error and indirect movement rates in Exercise 0.2 are shown in Table 8.6. S1's error rate is rather high. However, the indirect rates are not high, which suggests that subjects did achieve a degree of precision in moving between small objects and showed no tendency to 'overshoot' their targets.

	Errors		Indirect	
	Number	Rate (%)	Number	Rate (%)
S 1	10	25	2	5
S 2	2	5	3	7.5

Table 8.6. Errors and indirect movements in Exercise 0.2.

8.8 Conclusions from the timing results

Having obtained this timing data and formulated a widely-applicable model it is possible to make comparisons with other word processors. Fitts' Law can be used to compare with a visual program. The calculation can be carried out on the basis of a hypothetical program which would use the same screen layout as Soundtrack (as shown in Figure 8.3), but with *visible* windows of the same dimensions. According to Fitts' Law, the time taken to locate these windows would be as given in Table 8.7.

Distance	Time (s)
1	0.06
2	0.12
3	0.16
4	0.19

Table 8.7. Times to locate windows in a visual program, calculated from Fitts' Law.

These times are several orders of magnitude faster than the mean times obtained in the experiment above, as given in Table 8.4. It might be said that Fitts' Law does not take account of thinking time, but similarly T_{move} (the average time to move between two adjacent windows) is calculated on the assumption that it does not include thinking time. Recalling that $T_{\text{move}} = 0.73$ s, it is clear that the auditory program is slower than a visual counterpart.

As discussed in Section 8.1, the targetting task in Exercise 0.1 represented an approximation of the task in a real operational situation. The **Choose target** sub-action was reduced to the subject being told to which window to move next. In practical use of Soundtrack this task would involve the recall of an algorithm to produce some desired effect, a four-stage process as elucidated in Section 8.1. This implies that the time spent 'choosing' a window in this experiment is probably less than someone would spend in a less artificial situation. In other words, at 0.48 s, the value obtained for T_{choose} is on the low side. Unfortunately it is not possible to relate this to performance in the more realistic exercises (Exercises 31 and 33). Timings in those exercises were affected to too great a degree by irrelevant influences.

Eliminating such complications is, after all, the reason for setting up artificial exercises such as Exercise 0.1.

Having acknowledged the likely underestimation of T_{choose} , it is interesting to note that the time taken to plan the movement of the mouse, T_{plan} , comes out as by far the largest elementary component of the targetting time (see Table 8.2). The planning sub-action involves the user recalling the layout of the screen from long-term memory, and then using this representation to devise a route to the target window. The time taken to do this will depend most critically on the nature of the mental representation of the screen. As will be shown below, most of the evaluation subjects used a verbal aide memoire. It seems likely that this would be very slow to process, which would explain this result.

The observations concerning T_{plan} probably also explain one of the other results of the evaluation, the inconclusive timings obtained in Exercise 0.2. Most of the subjects appeared to have *no* concrete mental representation of the contents of any of the windows, as was shown in the structured interviews reported in Chapter 7. Planning is therefore difficult, and largely a hit-and-miss exercise. It would be based on partial representations of the window.

The slowness of using the mouse is exaggerated by the two-stage style of command execution. In other words, the execution of a command often involves firstly locating and double-clicking a window and then locating and double-clicking the appropriate object therein. The results of Exercise 0.2 were inconclusive, but it must be that the elapsed time to execute objects in this manner will be *of the order* of twice those in Table 8.2. Of course, locating some visual objects (particularly menu entries) is also a two-stage process, but both stages are very much faster.

An attempt was made to detect improvements in proficiency of using Soundtrack by calculating the *speed* of locating objects in different presentations of Exercise 0.1. However, on this measure no learning effects were apparent. There are several possible explanations for this. Most subjects completed Exercise 0.1 just three times with tracing on.

It is likely that over such a small number of tests no significant learning would occur. Also for most subjects the exercises were spaced out at weekly intervals and learning would have been improved if there had been less of an interval. Some of the subjects did make comments to that effect in answering the questionnaire. Added credence is given to this explanation by the fact that *short-term* learning effects were observed in Exercise 33 (see Section 9.2), which suggests that more intense teaching might be effective. Learning effects are discussed more fully in Section 10.2.

It should be borne in mind that subjects' behaviour in these (and other) exercises may have been adversely affected by anxiety. This was heightened by the fact that they were being observed by the experimenter. Despite repeated reassurances that the object under test was the program and not the user (see, for instance the exercise instructions in Appendix B) comments made by the subjects suggested that they were being examined. Such effects of anxiety degrading performance have been observed by other researchers, including Kennedy (1975).

The extra burden for visually disabled users of learning screen layouts was mentioned above, in the context of the extra time taken to accomplish tasks. Clearly from the results reported above this was a major problem for the subjects who tested Soundtrack. It seems that most users were able to learn the layout of the windows within the screen well, but found it too difficult to remember the layout within the windows. There are several factors which probably contribute to this. Firstly subjects in this evaluation received much more practice in using the screen. That is partly a result of the design of the interface, the screen being the first level of interaction, but also the format of the evaluation - whereby subjects performed Exercise 0.1 several times. Most subjects were able to develop methods to help them remember the screen layout (which were usually verbal), but none had such a strategy for the windows. Again this is partly a question of the amount of practice received. Also there may be a feeling that the windows were too complex - in terms of the number of them

as well as their design - for the subjects to believe it was worth the effort of devising aides memoire.

It would be interesting to investigate further how people learn the layouts. This would, however, represent a significant piece of research in itself because it would entail consideration of the spatial perception of the people. As described in Lowenfeld (1980), there is no consensus as to the nature of spatial perception in blind people. It seems unlikely that work in this field will become any more unified in the near future so that interface designers will have to continue to rely on a more empirical approach.

The pointing behaviour model devised is just that; it is not a law. One major exception was identified, proving that there is no one method of locating objects. Intuitively it might be expected that smaller objects would be quicker to locate, there being less distance to move between them. So it was perhaps surprising that this was not found to be. However, this result does not affect the status of the model, since the size of the target is not one of its parameters.

The purpose of gathering timing data was to see whether there were any regularities which could be recognized. Such regularities were found and these were explained in terms of a simple model. This model will make a valid basis for a wider generalization of these results in future work. For instance, it may be possible to extend the work of Roberts and Moran as a means of comparing word processors with auditory interfaces.

Chapter 9

Usage

9.1 Introduction to the traced exercises

Exercise 0.1 presented the subjects with an artificial task, so that measurements could be made of the time taken to perform basic (sub-)tasks. Another aspect of the evaluation examined the subjects performance in a more realistic task. The last (and most challenging) exercise completed by each of the subjects was recorded in two forms: as a trace of mouse and keyboard interactions and a tape-recorded dialogue between the tester and the subject. The two sources of information were combined in written transcripts. These exercises were the most realistic, most like the sort of task someone using Soundtrack in a job would have to complete. This means that they are more a test of its true usefulness.

It was possible to spend more time with subjects S1 and S2 and they completed *two* related exercises - 30 and 31. Details of these exercises are given in in Appendix B. Exercise 30 was given to these subjects twice in one session - to show up any learning effects. Both Exercises 30 and 31 involved the subject typing in text dictated by the tester. The difference was that in the former exercise the subject typed in the text and then saved the file without checking it for mistakes - although they were able to make any corrections which could be achieved by backspacing and re-typing. In the second exercise they were allowed to go back, having finished the typing, and corrected any mistakes they could find. Each presentation of the exercises involved typing a different piece of text, but each of them had approximately the same complexity.

The other subjects did not attain a level of practice which would have made it realistic for them to be given exercises as difficult as 30 or 31. Hence a special one was developed as their final test. This was numbered 33. (Exercise 32 was a planned one which was considered too difficult for any of the subjects to attempt in practice, see Appendix B). It was based upon the construction of a letter. There were three tasks planned. The first consisted of opening two documents copying some text from one and pasteing it into the other. The second involved the typing of some replacement text. The third intended task was the correction of a spelling error. In fact none of the subjects had sufficient time to undertake the third one (although none of them noticed the error, which was the mis-spelling of *thank* as *tank*).

Output from the trace program consists of an entry for each *event*. Events are assigned numbers, simply counting from 1. The time since the last event is reported. Otherwise the information on each event in the trace depends on the nature of the event. Events fall into six categories:

1. Timestamps. These are generated whenever tracing is started. They record the date and time and so can be used to identify traces.
2. Movements - from one object to another. That may be from one object within an active window to another, between windows or off the screen. Within an active window only the object is named in the trace, whereas when the mouse moves into another window both that window's name and that of the 'underlying' object is given.
3. Mouse-downs, or single-clicks. The mouse's coordinates in the screen are reported.
4. Double-clicks. The coordinates are again reported.

5. Key downs. These correspond to the subject pressing a key on the keyboard.
6. Auto keys. These are generated when a key is held down, so that it 'auto-repeats', printing multiple copies of the current character.

The following annotated fictional extract illustrates each of these types of record:

A timestamp, giving the date... ...and time	5/3/1986 18:47:05
Event number 1. Event 1's 'time' is number of seconds since the program started. A movement event. The window and object into which the mouse has moved.	1 905.45 Movement File : Open
Number of seconds since last event. A mouse down and its screen coordinates.	2 2.30 mouse down : 331, 249
A double-click and its coordinates.	3 1.50 double click : 332, 246
After double-click the event number is incremented by 2 (one per click). For a movement within an active window, only the destination object is given.	5 5.05 Movement New
	6 2.34 double click : 329, 229
	8 8.02 Movement Edit: Paste
	9 3.05 Movement Document window 2 Jump forward
	10 4.07 Movement Document window 1 Step forward
	11 2.41 double click : 30, 153

A key is pressed. Some key down events below are not labelled as such, but only the key's character is given.	13 2.56 key down a
	14 0.27 auto key a

There were two bugs in the trace program. If the mouse was moved out of the active window then two Movement events were recorded - one as soon as the mouse returned to that window and another when it came to rest again. This does not have any significant effect on results. The second one was that as far as counting events was concerned a double-click was counted as two events (one per click). Hence there is a gap of one in the numbering following a double-click.

The transcripts also include the spoken dialogue and their layout attempts to reflect the temporal sequence of events and dialogue. For instance, the beginning of a dialogue fragment is positioned to the right of its preceding event. Naturally, events and dialogue often occur in parallel and the transcripts attempt to reflect this. The speakers in the dialogue are identified as the subject, S, and the tester, T.

Below there is a collection of general points which arose in these exercises. This is followed by summaries of the notable features of the transcripts for individual subjects, supported by excerpts from the transcripts where appropriate.

9.2 General features

One objective of tape recording the interactions between the subject and the experimenter was to gauge the level of help that the subject received. The aim of the tester was to give the minimum amount of help. He asked the subject to perform a particular task and then only prompted with more detailed assistance if he judged that the subject needed it. It is possible only to give general indications about the variations in assistance which individuals required, but it certainly varied widely.

One interesting characteristic was whether subjects clicked the mouse when they thought they had located a target object. Clicking would elicit speech by which they could check whether they had indeed found their objective. This action is referred to below as *click-checking*. Subjects were divided into those who almost invariably did click-check and those who often did not bother. There were varying degrees of accuracy amongst those who did not.

All of the subjects appeared to have a similar level of familiarity with the screen and window layouts. That is to say that they all seemed able to move quite directly to different windows, but were much more indirect and click-checked more frequently when looking for objects within active windows. In fact three of the subjects (S3, S4 and S7) used tactile diagrams during the exercise, to remind themselves of the layouts of particular windows. The diagrams were always on hand during sessions and if the subject asked to see one of them, or the tester thought they were lost, they were given the diagram to feel while the tester explained the layout.

Two subjects (S3 and S4) seemed to assume that all windows had a strict grid layout. For instance, when using the **Select file** dialogue they treated the **File name** object as if it occupied only the left half of the window. When moving between it and the **Up** button they moved via the **Down** button, in separate vertical and horizontal movements.

Among the five subjects who completed Exercise 33, three made the same serious error of opening the wrong file. This was as a result of double-clicking on the **Open** button in the **Select file** dialogue before having moved the required file name into the file name frame. The root cause seems to be confusion and unfamiliarity with the procedure for opening files, compounded by there being two objects in different windows which have the same name. The subjects were asked to open particular files. The first stage of this procedure is to execute the **Open** entry in the **File** menu. That causes the **Select file** dialogue to be put into the dialogue window. Unfortunately, one of the buttons within that dialogue is also

called **Open**. It seems that some of the subjects assumed that the next stage in the process was to execute this.

Exercise 33 was deliberately designed to involve the repetition of particular tasks - such as the opening of *two* files. In fact, because of errors made in opening files, several subjects had to perform file opening three times. This gave some opportunity to see whether subjects learned in the short term. In other words it was possible to observe whether they needed less prompting the second time they performed a task. Again individuals varied. Three of the five subjects who did Exercise 33 did indeed seem to remember the process and needed less help.

A number of errors were made in the dictation exercises which appear to be mis-spellings. It is difficult to classify these. They might be genuine mis-spellings, where the subject did not know how to spell the word, but did not ask for clarification from the dictator, but they might also have been caused by the subject hitting the wrong key or by her mis-hearing the word dictated. Typing is *not* a novel feature of Soundtrack, so that it was not felt necessary to investigate these errors more deeply, although it might be possible to distinguish genuine mis-spellings from other errors by applying the sort of analysis described in Pain (1981).

In Exercise 33 subjects proof-read and corrected their text. Some spelling errors were corrected in this process, but others remained uncorrected. The remaining ones were almost invariably errors which could not be heard in the synthetic speech; the pronunciation could not be distinguished from that of the word correctly spelt.

As described earlier, the edges of the auditory screen were marked by a continuous 'buzzing' noise. Subjects were quick to recognize that and understood its significance, as was shown in a number of the traces; on hearing the buzz they would move the mouse quickly back into the screen to silence the buzz.

The time taken to complete these exercises gives a rough indication of how productive Soundtrack is as a word processor. There are a number of limitations on this measurement. It was mentioned above (Section 6.3) that it was not possible to apply Roberts and Moran's (1983) evaluation process because none of the volunteer subjects attained an expert level of proficiency. More broadly, that fact implies that it is not possible to give a realistic indication of the program's potential productivity. The person who is most expert in using Soundtrack is its developer. Therefore, to gauge the potential efficiency of a user he completed Exercises 31 and 33. He is not visually disabled, however, as described earlier, his sight is not likely to have been of any material advantage in using Soundtrack. The developer completed the exercises using Soundtrack both in its auditory and visual modes. Thereby it was possible both to see what was the potential level of performance of the auditory version and to see roughly how it compares with visual word processing.

Table 9.1 gives the total times taken for Exercise 31 and Table 9.2 gives the totals for Exercise 33. It was not practical to break these times down into a smaller grain size (such as the taxonomy of editing tasks used by Roberts and Moran - op.cit.) because the times taken to complete the sub-tasks vary greatly, often due to extraneous circumstances. For example, some subjects needed more coaching and assistance from the tester on particular sub-tasks, which implied that they spent time not interacting with the program but listening to instructions.

ADNE (visual)	ADNE (auditory)	S 1	S 2
3:46	8:45	49:18	25:45

Table 9.1 Total times (in minute and seconds) taken by the developer (ADNE) and two of the test subjects to complete Exercise 31. ADNE completed the exercise using both the visual and the auditory interfaces of Soundtrack.

The times given in Tables 9.2 should be treated as somewhat approximate. Time spent by the evaluation subjects dealing with gross errors (such as opening the wrong file and then closing it again) was *not* included. Also a complete trace was not available for subject S8 due to technical problems, so his time appears lower than it should be. Similarly S9's time will be an underestimate since he missed out one of the tasks in Exercise 33 (replacing the 'X' in *Dear X* by a name).

ADNE (visual)	ADNE (auditory)	S 3	S 4	S 7	S 8	S 9
1:08	5:03	18:55	16:18	15:11	12:22	12:21

Table 9.2 Total times (in minute and seconds) taken by the developer (ADNE) and five of the test subjects to complete Exercise 33. ADNE completed the exercise using both the visual and the auditory interfaces of Soundtrack.

All the same, the times measured for ADNE are an indication of what is possible, and they do represent something of an expert level of usage. They suggest that it is possible to use Soundtrack quite efficiently - but that no one using Soundtrack is likely to attain the same speed as a sighted user of a visual word processor.

All the subjects needed a high degree of assistance to complete these exercises. That assistance was generally at quite a low level, being told a step at a time what to do next. This would suggest that they did not have an adequate understanding of the program. All the subjects were able to locate windows by name, but sometimes people needed assistance with navigating within windows (e.g. "Move one more beep to the right.") Errors sometimes occurred when the subjects acted on their own initiative, notably when they were opening a file (for the reasons discussed above). That this error occurred confirms the suggestion that their understanding of the system was incomplete. What is more, their

reactions suggest that they could not quickly have recovered from the error without assistance.

However, recall that most subjects stated in their interviews that they felt they could have used Soundtrack without assistance. It is probable that they found the presence of the tester inhibiting them by making them fear the making of any mistakes in front of an audience. It might be that left on their own they would have learned more. They might have made more mistakes initially, but would have got a better understanding of the system by using it unaided.

9.3 Individual subjects' results

Subject S1

S1 seemed well able to navigate around the screen. He generally moved directly to windows, although he *always* did a click-check. This coincides with S1's behaviour in Exercise 0.1 in which his rate of indirect movements was below average. Within menus he usually found entries directly, but he seemed less sure about the layout of documents and dialogues. This confirms his opinion, expressed in the questionnaire, that the layout of active windows is a difficult aspect of using Soundtrack.

Exercise 30 The dictated text produced by S1 in this exercise is reproduced below:

As usual, at five o'clock thaaaaaat morning reveille was sounded bey the bvlows of a hammer on a length of rail hanging up near the staff quarters. The intermittant sound bearly penetrated the window panes on which the frost lay two fingers thick, and they ended almost as soon as they'd begun. It was cold outside, and the camp gaurd was eluctant to go on beating out the reveille for ling.

There are nine identifiable errors. Three of them - the missing apostrophe in *o'clock* and the mis-spellings of *intermittent* and *barely* may have nothing to do with Soundtrack, but with the subject's English. Others are all typing errors and are examples of Soundtrack's speech was not being sufficiently clear to show up the errors. It will be remembered that

Soundtrack allows the user to hear text either letter-by-letter or word-by-word. The default method is the latter, and it was this mode that all the subjects used in these dictation exercises - including S1. Hearing the kind of typing errors illustrated above would be dependent on the speech synthesizer's quality of pronunciation. The following extract illustrates problems S1 had hearing errors. He did not hear some of the typing errors (for example the misspelling of *guard*). He was aware of some other errors - either because he knew the keys he had pressed or because he could hear them. One which he heard was the mis-spelling of *long*, but since he was not sure what the actual error was he could not have corrected it with simple backspacing and re-typing and so he was encouraged to ignore it. (Note that repeated key-down event, as shown below, are reported by the tracing program by the identity of the key, the 'key down' label is omitted).

350 0.25 o	351 0.18 u	352 0.35 t	353 0.55 s	354 0.28 i
355 0.20 d	356 0.22 e			
357 0.23 space		T: Sorry, I forgot the comma.		
358 5.00 backspace				
359 3.10 ,	360 0.40 space	361 1.85 a	362 0.33 n	363 0.33 d
364 0.25 space	365 0.20 t	366 0.20 h	367 0.18 e	368 0.22 space
369 0.42 c	370 0.25 a	371 0.45 m	372 0.33 p	373 0.27 space
374 1.18 g	375 1.12 a	376 0.32 u	377 1.37 r	378 0.53 d
379 0.28 space	380 2.60 w	381 0.22 a	382 0.28 space	383 2.65 backspace

384
0.97
s

385
0.47
space

S: Oh, I see.

T: It just says the end of what you typed.

S: I can't see where I am now. I dunno where I am, re... Ah, well.

386
0.55
e

387
10.95
l

388
0.77
u

389
0.28
c

390
0.58
t

391
0.27
a

392
0.27
n

393
0.35
t

394
0.52
space

S: That's probably spelt wrong.

T: It sounded alright.

S: It did sound as though it had got the 'E' in.

395
5.87
t

396
0.25
o

397
0.20
space

398
0.32
g

399
0.23
o

400
0.22
space

401
0.53
o

402
0.23
n

403
0.25
space

404
0.40
b

405
0.25
e

406
0.42
a

407
0.28
t

408
0.25
i

409
0.20
n

410
0.58
g

411
0.23
space

412
2.45
o

413
0.23
u

414
0.25
t

415
0.23
space

416
0.28
t

417
0.20
h

418
0.25
e

419
0.28
space

420
1.32
r

421
1.32
e

422
2.05
v

423
0.53
e

424
0.28
i

425
0.40
l

426
0.27
l

427
0.87
e

428
0.40
space

429
0.53
f

430
0.20
o

431

432

433

434

435

0.28 r	0.27 space	1.98 l	0.67 i	0.25 n
436 1.13 g	437 0.42 .	438 0.53 space		
439 0.22 space				

S: That didn't sound right. It didn't sound like 'long'.

T: We won't worry about that. The next exercise will be the same sort of thing, but you'll go back and check things like that.

The extra *a*'s in *that* were generated by auto-repeating of the key. It is interesting, but puzzling that subject S2 made similar errors with the letter *a* on the same day (see below), but there did not appear to be any fault in the key. It has been used without further problems ever since.

Unfortunately technical problems with the tape recorder meant that the spoken dialogue accompanying S1's second attempt at Exercise 30 was lost. Nevertheless the text produced is reproduced below:

Mr Speaker rose and surveyed the Commons. He tugged at his long black silk gown, then nervously tweeked the full bottomed wig that covered his balding head. The House had almost got out of control during a particularly roudy session of Prime Minisster's questions, and he was delighted to see the clock reach three thirty. Time to pass on to the next bbusiness of the day.

This time S1 made eight errors. Again, however, three of them (the spelling of *rowdy* and the missing hyphens in *full-bottomed* and *three-thirty*) could be attributed to S1's English; without the tape recording of the spoken dialogue it is not possible to know whether the subject was given any more guidance on these points. Three errors are repeated letters (in *particularly*, *Minister's* and *business*), however two of those mistakes were *not* caused by the key auto-repeating and so must have been due to separate presses of the key. The number of errors shows no significant improvement over the first attempt at this exercise. However, given the nature of these errors - mis-typings - this is to be expected; it is unlikely that the subject's ear for the synthetic speech would become sharper during the session.

Exercise 31 The contents of the file produced in this exercise are reproduced below:

The clanging ceased, but everything outside looked like the middle of the night when\////////////////////////////////////\ ivanDenisovich Shukhov got up[to go to the bucket. It was pitch dark except for the yeellow light cast on the window by three lamps, two in the outer zone one inside the camp itself. And noone came to unbolt the bbarrack hut door; there was nop sound of the barack orderelies pushing a pole into placce to lift the barrel of light soil and carry it out.

There are a total of eleven errors in this text. It is perhaps surprising that there should be more than in the earlier exercise which did not allow the making of corrections. However, there are a number of points to note about the errors which are shown. The repeated oblique strokes were caused by the subject pressing the wrong key instead of the shift. It should be noted that this error would not be heard unless the **Proof read** speech option was on. Similarly the extraneous square bracket between *up* and *to* would not normally have been spoken. In this exercise only two of the errors may be due to misunderstanding by the subject. He may have mis-spelt *orderlies* and mis-heard or mis-understood the term *night soil*. A number of mistakes were caused by the subject's hands getting displaced from the home keys, so making consistent errors. The use of the mouse is probably the cause of such errors, which would not have occurred in a system on which the user would normally have her hands constantly on the keyboard. This must be seen as a disadvantage of a mouse-based system. Because of lack of time the last sentence was not proof-read or corrected. The subject did spend approximately 45 minutes on this exercise. It is noticeable that a lot of the other errors are ones which would be difficult to hear in synthetic speech.

S1 was able to complete the exercise with quite a low level of prompting from the tester. Manipulation of the mouse was good. There were very few problems with double-clicking. Generally he was able to locate objects quickly although he did occasionally get very disorientated. However he always click-checked.

The following extract illustrates a number of features of S1's behaviour. At one point in this exercise, when attempting to cut an incorrect character from the document, S1 appeared

to get confused about the order in which sub-tasks must be done. That is to say, he started to look for **Cut** in the **Edit** menu, before he had selected the offending letter.

S: Ah, so we've got an extra 'V' there that we don't want.

T: Yes.

S: Now either you've got to cut that out, or over-type from the 'V' forward.

T: Err, yes, I guess so, yeah.

645

25.00

Movement

Jump forward

S: So, what's the best thing to do with that?
Cut it out, I think.

T: Yes.

646

4.55

Movement

Scroll bar

647

0.62

Movement

Document window 2

Say context

648

0.90

Movement

Edit : Cut

649

1.73

mouse down : 204, 196

650

2.83

double click : 204, 194

T: You'll have to have the 'V' selected before you can cut it, though.

S: Ah. So I've got to go back to it.

T: Yes.

From these records there is some ambiguity as to whether S1 understood the concept of selection levels well. He did not mention this as a problem in the interviews, but from the transcripts it appears that in some instances he handled level manipulations well but not in others. He knew the sequence of levels, but was not always aware of what effect changing the level would have on the current selection. However, the following extract shows him manipulating selections quite skilfully. The extract occurred when he had finished typing in

the text and it had been suggested that he should proof read it. Without any prompting, he went ahead and altered the selection appropriately:

543
2.70
Movement
Current level

544
1.88
mouse down : 35, 45

545
4.10
Movement
Level up

546
2.07
mouse down : 28, 68

547
2.33
double click : 27, 68 S: Point, character,...

549
4.17
double click : 27, 68 S: ...word,...

551
3.83
double click : 27, 68

S: ...sentence That's probably too much. Do I need to read it all through first, do you think, or just go through to stop it at its first mistake?
T: Whichever way you prefer.
S: I think probably then I've gone up... I think I've gone up to sentence level.

553
21.60
Movement
Current level

554
0.97
mouse down : 24, 39 S: Yeah. I think it'd probably be best at word level, wouldn't it.

S1 does not appear to have understood that any input replaces the current selection - which may be a range of text . Several times he seemed to expect Soundtrack to behave similarly to Wordstar, thinking in terms of over-typing. (In Wordstar - and many other older word processors - input text can either be inserted within existing text or typed 'over' it, replacing it character-for-character). Take the following two extracts, for instance:

T: So there's that. Now there's presumably some rubbish at the end of that - because it's not over-typed what was there before, it's moved it along.

S: Oh, it'll have moved it along. Oh, I see. Once you've got that... The first one it'll overwrite, and the next...

T: Well, it overtypes whatever the current selection is, so 'r'... the letter 'r' was the selection, I think, wasn't it?

S: What, if you put a... Can you delete those with the space bar, like you can on word processing?

T: You can actually...

S: If I pressed the space bar now what would happen? Would that 'i' go?

T: It would go, and there would be a space put in its place.

S: And so you would get a spacing error rather than a... So, really you've got to go to the Edit and cut this.

T: Yes

Subject S2

Exercise 30 The output from S2's first attempt at this exercise is as below:

As usual, aaat five o'clock that morning reville was sounded by the blows of a hammer on a length of rail hanging up near the staff quarters. The intermitent sound barely penetrated the window panes on which the frost lay two fingers thick, and they ended almost as soon as they'd begun. It was cold outside, and the camp guard was reluctant to go on beating out the rev ille for long.

There are just five distinct errors. The tester must be blamed for one of them - the mis-spelling of *reveille* - which he did spell out, but incorrectly. The error in *intermittent* - was probably a mis-spelling by S2, rather than a typing mistake. In this exercise S2 did not always click-check and made no errors on the occasions on which he did not check. The multiple *a*'s in *at* were produced by auto-repeating of the keys, as mentioned above.

In the second attempt at the exercise S2 was able to perform the task with much less prompting from the tester.

Mr Speaker rose and surveyed the Commons. He tugged at his long black silk gown, then nervously tweaked the full bottomed wig that covered his balding head. The House had almost got out of control during a particularly session of Prime Minister's questions, and he was delighted to see the clock reach 3.30. Time to pass on to the next business of the day.,

There are six errors in this text. One was probably a lack in the subject's English - the lack of a hyphen in *full-bottomed*. The mis-typing of *during* appears to have gone unnoticed by the subject. Why he missed the word *rowdy* ("*...a particularly rowdy session...*") seems curious, although he was aware that he had done so:

224 0.22 g	225 0.20 space	226 0.30 a	227 0.27 space	228 0.35 p
229 0.15 a	230 0.22 r	231 0.27 t	232 0.35 i	233 0.20 c
234 0.35 u	235 0.22 l	236 0.15 a	237 0.27 r	238 0.57 l
239 0.22 y	240 0.25 space	241 0.50 s	242 0.20 e	243 0.27 s
244 0.17 s	245 0.18 i	246 0.17 o	247 0.37 n	
248 1.02 space				

S: Oh...damn. Oh well, good practice putting a word in.

Another error should probably not be counted. The time three-thirty was spelled out in the original text, whereas the subject used figures, this not having been specified by the tester. That leaves *four* significant errors.

Exercise 31

He stood shifting from foot to foot waiting for the 5j00 odd mmbers present to settle down before he intoned solemnly, "Members desiring to take the oath." The packed assembly switchded its gaze from the speaker to the far end of the Chamber, like a crowd watching a tennis match. There, standing at the bar of the Commons was the victor of the first by-election since the Labour party had taken office some two monmths before.

In this exercise S2 made just five errors. One may have been a spelling error - *intoned*. During the dictation the subject was rather more careful to ask for clarification of points of grammar. He noticed one error - the mis-typed *members* in the first sentence - while he was typing in, but forgot subsequently to go back to fix it.

Below the text as it was after it had been typed in, but not proof read or corrected. At this stage it contained eight distinct errors:

He stood shifting from foot to foot waiting for the 5j00 odd mmbers present to settle down before he inntoned solkemnly, "Members desiring to take the oath." The packed assembly switchcded its gaze from the speaker to the far end of the Chamber, like a crowd watching a tennis match. There, standing at the bar of the Commons was the victor of the first by-election sincde teh Labour party had taken office some two monmths before.

So, the subject managed to find and correct four errors.

In this exercise S2 quite often did not click-check. However, he did once make an error of double-clicking the wrong object as a result.

At one point the subject got quite lost:

S: Can you see what's happening on the screen?
T: No.
S: Great!

He got into quite deep confusion, which was largely due to having set the selection level to **point** level and then attempting to do a **Cut** - which had no effect. He had deliberately gone to **point** level, but was clearly unaware of the full implications of so-doing:

S: That 'h' wasn't cut out, then.
T: I thought it was. You must have cut something else. Did you get the space after it, by mistake?
S: [unintelligible]...point...
T: Had you gone down to point level?
S: Yes, so I could get it in between...
T: Ah, yeah. If you're at point level, then if you cut something... well you can't cut it.

S: Can't cut it?

T: Hmm, you need the character 'h' to be selected before you could cut it. I thought.. When you said you were going to point level, I thought you were going to type in the other 'h' - between the 't' and the 'e'.

Subject S3: Exercise 33

The second time S3 opened a file he still needed quite detailed instructions. However, because of an error he made he had to actually open a file a third time, and this time he needed less help. So practice did seem to be improving his use of the program. S3 seemed very unsure about the layout of active windows. He consulted the tactile diagram of the **Document** window during this exercise and there were times when he appeared to be moving the mouse aimlessly, clicking it to try to orientate himself. He always click-checked.

He seemed unsure about the activation concept. Almost every time he had to activate a window he firstly checked with the tester that it was the right thing to do. The following extracts also illustrate his lack of confidence in this aspect. In the first one he seems to confuse activation of a window with the execution of an object within it.

T: Right, now it's asking if you want to save that file that we just opened, so we have to go up to the Dialogue box and activate it.

58
9.68
Movement
Interface : Auditory

59
3.65
Movement
Off screen at the TOP

60
0.23
Movement
Interface : Hidden

61
1.52
Movement
Dialogue : Save question
In content

62
0.78
mouse down : 397, 58

T: So, activate that.
S: I don't want to save it, though, do I?
T: No, but you've got to activate the dialogue first. Then you can find the 'No' within it.

In the next extract he seems to think that moving out of an active window will cause it to become deactivated.

86
0.43
Movement
Edit : Find

S: Does this mean I've got to activate it again? I've come out of it.
T: No, no, it's still active.

S3 was one of the subjects who appeared to assume that the **Select file** dialogue had a strict grid layout.

S3 seemed unsure about how new typing was incorporated within an existing document - that it replaced the current selection. This is not surprising because he was familiar with word processors which allow either insertion within the existing text or letter-for-letter over-typing.

Subject S4: Exercise 33

Many subjects had trouble remembering the layout of windows. However, S4, for one, appeared to know the layout of at least one window - the **File** menu - well. He found entries within it very directly. He used the pitch of objects to locate them (and was the only one of the test subjects who did so). The following extract illustrates the technique he used, whereby he moved the mouse back and forth between objects, listening for the tone he wanted.

10
3.97
double click : 306, 237

T: Okay, so move up to the dialogue... and activate it.

12
11.10
Movement
Interface : Auditory

13
0.35
Movement
Dialogue : File select
Yes

14
3.95
mouse down : 428, 110

15
4.25
Movement
Alert

16
0.53
Movement
Dialogue : File select
Cancel

17
0.72
mouse down : 457, 167

18
3.25
Movement
New

T: You were in it, then you moved again.
S: That is...
T: It that it now?

19
5.23
Movement
Interface : Auditory

20
0.35
Movement
New

21
0.62
Movement
Alert

22
0.25
Movement
Open

23
0.80
Movement
Alert

24
0.57
Movement
Open

25
0.97
Movement
Interface : Auditory

26
0.68
Movement
Dialogue : File select
Yes

S: Oh, that is it.

27
0.75
mouse down : 404, 161

S: Yes, yes.
T: So, if you double-click that...
S: So...

28
5.27
double click : 399, 159

Notice how, at event 26, S4 recognized that he had found his target object before he clicked it; he had located its tone.

S4 was one of the subjects who consulted a tactile diagram during this exercise. He examined the **Select file** dialogue. However, despite this, he did appear to think that the dialogue had a strict grid layout. As is shown in the extract below, he moved from the **Name** item to the **Down button** via the **Up button** (see Figure 5.7).

92
34.03
Movement
File select
Name

93
0.82
mouse down : 389, 44

T: Right so that's the current umm...
S: What was that? Oh, 'Crime', yes.
T: 'Crime', yeah. So, we don't want to open that one. It's further down the list, the one we want...

- 94
14.15
Movement
File select
Up button
T: ...It's called 'insert'.
- 95
0.95
mouse down : 385, 80
S: Right?
T: You want the Down Button, yeah.
S: Down Button, yes, which is...?
T: To the right.
- 96
5.97
Movement
File select
Down button
- 97
0.37
mouse down : 454, 108
- 98
1.97
double click : 455, 109
T: Right, you can move back up and see what the current file name is.
- 100
2.67
Movement
File select
Up button
- 101
0.23
Movement
File select
Name
- 102
0.83
mouse down : 412, 49
T: "Errors".

However, S4 did appear to learn from the repetition of the file-opening task. Having gone through the above trials, he completed the second one with much less prompting.

Subject S7: Exercise 33

Several times during this exercise subject S7 had problems double-clicking the mouse. He commented that it seemed to help if he relaxed more. He appears to use the edges of the

screen to orientate himself. For instance, the following extract occurred when he was trying to locate the top right-hand window (the **Dialogue** window).

T: ...The first one we want to open is called 'insert'. So you go up to the dialogue...

38
3.60
Movement
Interface : Auditory

39
0.23
Movement
Dialogue : File select
Cancel

40
0.45
Movement
Off screen to RIGHT

41
0.23
Movement
Dialogue : File select
Down button

42
1.10
Movement
Off screen at the TOP

43
0.23
Movement
Dialogue : File select
In content

44
0.53
mouse down : 423, 69 T: Okay, now you activate it...

45
5.97
double click : 410, 113

Subject S7 did not always click-check before double-clicking. Despite that he made no errors of double-clicking the wrong object.

Subject S8: Exercise 33

This subject was able to open a file with a minimal amount of prompting from the tester - even the first time. However, she did also apparently display confusion about the **Open** object within the **Select file** dialogue, although she asked the tester to avoid potential errors, as shown in the following extract:

T: Okay, so you go up to the dialogue, and the one you want is called 'insert', so you have to move down the list to get to 'insert'.

S: Don't want Open do we?

T: No.

She did always click-check. She seemed to know the layout of the screen well, but did get quite lost in some of the windows - particularly the **Document** window and **Select file** dialogue.

Subject S9: Exercise 33

Judging by the low level of help he received, S9 was the most competent of the subjects in this exercise. Also, he seems to have had no problems in manipulating selection levels. For instance, the following extract shows how he needed only to be given a suggestion about setting the selection level and he was then able to go ahead and so do without any more prompting:

112

3.37

mouse down : 95, 118

T: Then we want to move... Might as well move the level up to the Whole Text level.

113

7.50

Movement

Step forward

114

0.50

mouse down : 88, 93

115

2.43

Movement

Level down

116

0.50

mouse down : 87, 60

117
2.92
Movement
Speech : Say characters

118
0.48
Movement
Level down

119
0.73
Movement
Level down

120
0.23
mouse down : 117, 63

121
3.00
Movement
Level up

122
0.80
mouse down : 50, 61

123
3.00
double click : 50, 61 S: So...

125
1.87
double click : 52, 62

127
2.07
double click : 52, 61

129
1.57
double click : 52, 59

The selection has now reached **paragraph** level, but the subject sensibly checks this by using the **Current level** control.

131
1.20
Movement
Current level

132
0.88
Movement
Off screen at the TOP

133
0.45
Movement
Current level

134
0.73
Movement
Current level

135
0.80
mouse down : 49, 39

Now he knows which level he has reached, he correctly goes back and lifts it one more level.

136
2.83
Movement
Level up

137
0.70
mouse down : 43, 62

138
2.33
double click : 44, 61

This subject, however, did seem to have some trouble remembering the layout of the screen. He did almost invariably click-check.

9.4 Discussion of the traced exercises

One concept which caused subjects difficulty was that of text selection and its associated levels. This is less of a fundamental problem since it is specific to programs involving the processing of text; it is not general to all wimp programs. There are a number of factors which contribute to this difficulty and these are discussed more broadly in Chapter 10.

As was shown above, subjects S1 and S2 did make quite a large number of errors in their dictation exercises, within rather short pieces of text. It is difficult to say whether this shows that Soundtrack is inherently difficult to use or whether this was due to the subjects' inexperience. In an attempt to answer this question, the developer of Soundtrack - the

person who understands the program best - also performed Exercises 30, 31 and 33. Although he is not visually disabled, he performed the exercise with the windows *hidden* so that his sight was of little advantage. The only possible assistance he would have got from his sight during the exercise was that he would be getting visual information about the location of the mouse (*not* the mouse pointer, which was invisible). There is also a related question as to whether he would have had a better internal representation of the layout of the screen, having *seen* it. Certainly he must have had a complete conceptual model of the program. Also he had extensive prior experience of using (visual) wimp-based systems. He was able to type in and proof read a paragraph as in Exercise 31 without any errors. This does suggest that with sufficient training it should be possible for people to become quite proficient at using Soundtrack.

The times for the developer to complete Exercises 31 and 33 suggests two conclusions. It confirms that the evaluation subjects can indeed be classed as novice users, that it is possible not only to use it more precisely, as mentioned above, but also that it is possible to be much quicker at using it. However, it also shows that Soundtrack is slower than a visual word processor.

Difficulty in proof-reading is common to most audio-enhanced word processors. As was illustrated above, many of the typing errors remaining in documents could not have been heard - unless the document was read letter-by-letter. Obviously this would be an impossibly tedious exercise for a document of any size. This fact suggests that it is very important that spelling checkers and correctors should be made available to visually disabled word processor users. Since there are a large number of such programs available for sighted users, this ought to be quite straight-forward - as long as it is possible to adapt visual software.

Chapter 10

Evaluation outcomes

10.1 Introduction

The four previous chapters have described the evaluation and its results in some detail. This chapter firstly summarizes the results of measurements of learning which were made in different parts of the evaluation. Some conclusions which arise directly from the evaluation are then discussed. Other conclusions are less direct and these are included under separate headings. These relate to problems of adaptation, the possible provision of a physical frame to constrain mouse movements and the text-selection mechanism.

10.2 Learning

Several of the responses in the interview related to the question of learning. A rough measure of competence was defined as being able to use Soundtrack without assistance. The questionnaire separately asked whether the subjects thought they had attained that level and how long they thought it would take some other novice to reach it. Recall that there were two groups of subjects: those who had received about 2¹/₂ hours of training and those who received over six hours. However, nearly all the subjects - from both groups - believed that they had reached the level of competence.

Estimates of how long it would take to train a new user varied widely, between two hours and over six hours. There was, however, more agreement to the effect that training should be organized in longer sessions, more closely spaced. There was a common feeling that the intervals of up to one week between the evaluation training sessions was too long. This may have been one of the causes of the variation in the estimates; some subjects may have felt that a trainee would learn in less time than they had if the training was better scheduled.

The training times - both actual and estimated - can also be compared with the amount of training given to students learning to use other word processors. For instance, the RNIB Commercial Training College runs some intensive, full-time training courses on the Frank Audiodata, which take a minimum of one week for able students. The Royal National College for the Blind, Hereford, also runs courses on using the Audiodata. These consist of two 1¹/₂-hour lessons per week and last about ten weeks. Tutors in both colleges say (in personal communications) that some students can reach a very high level of proficiency within those courses. However, they stress that the level attained varies widely, according to individual students. This accords with the fact that the subjects' estimates of training time varied widely.

An attempt was made to measure learning effects within the timing data obtained in Exercise 0.1. In order to do this the quantity *speed* is defined. Within any execution of Exercise 0.1, the average speed is defined as the total time taken to locate each of the requested targets divided by the total distance to those targets. Two aspects of performance will influence the speed: manual dexterity and the directness of movements to the target. For this reason, the speeds were calculated in two ways: firstly including direct movements, errors and indirect movements and secondly including only direct movements. Improvements in either skill should lead to an increase in speed.

Using this measure it is *not* possible to detect any apparent improvement in the performance of any of the subjects; there was no significant trend in speeds - whether or not errors and indirect movements are taken into account. Again, the scheduling of the presentation of the exercises may have been the cause of this in that there may have been too great a gap between presentations of Exercise 0.1. This data is only measuring one aspect of using the program - the pointing ability - and another feasible explanation of this result is that subjects had quickly attained optimal pointing efficiency for blind users of Soundtrack.

That learning had occurred in other aspects of using the program is obvious. Subjects had got from being introduced to a completely new piece of equipment to a point where most of them felt they could use it unaided. The records of Exercises 31 and 33 show that learning did occur *within* sessions. For instance, Exercise 33 involved repeating the action of opening a file and, as noted earlier, several of the subjects did this more competently the second or third time.

10.3 Immediate conclusions

Soundtrack is the basis of a way of making wimp software accessible to visually disabled people. However it is not an ideal word processor. This proposition can be discussed on the basis of the findings of the above evaluation.

Dealing with the positive results first. In a comparatively short time, most subjects attained a fairly high level of competence in using Soundtrack. In fact the majority of the subjects considered that they would now be able to use Soundtrack without assistance. So, it does seem feasible that many visual wimp programs could be adapted using the same principles. This would give access to software which has previously been denied to visually disabled people.

Having made that broad statement about the positive results of this project, it must be admitted that the burden of the results of the evaluation relate to shortcomings in the design of Soundtrack. The remainder of this chapter concerns those results. Specific modifications to the design which may overcome some of these deficiencies are presented in the next chapter, in the section on possible future developments.

In Chapter 8 timings of Soundtrack were compared with those which might be obtained for a similar visual program. It is not possible to make any other direct temporal comparisons with other word processors in the absence of any data on such programs but it is possible to speculate further. For instance, it seems likely that thinking time for the auditory program

will be longer than for a visual one. The power of visual menus derives from the fact that they give the user ever-present prompting of the command categories available (i.e. recognition). In the auditory program the same prompts do exist but they are not immediately accessible in the same manner, since the user must find and click on an object to hear its identity (involving extensive recall). This is an example to the phenomenon discussed earlier whereby sight can process input information so quickly that it can essentially receive several items of information in parallel, whereas hearing is much more a serial sense. Click-checking for prompting is slow so that most users make the effort to remember the layout of windows. The task of recalling this spatial information will add to the thinking time. Direct comparisons with conventional word processors which use commands typed on the keyboard (such as Wordstar) are also not possible since no comparative data is available.

The two-level design of windows within a screen is intended to relieve memory loads. The user needs firstly to choose one of eight windows. Having activated the chosen window they must select from the (small) number of objects contained therein. Ideally, therefore, the user should need to hold information about a small number of objects in working memory at any time. However, there seems to be a problem switching between windows. It seems that the problem is of users not knowing *what* to load into memory, since they lack any explicit internal representation of windows.

To fix the number of component screen objects at no more than nine was probably an oversimplification (see Section 4.4). It is possible that such a number would be more applicable to a visual program, but an auditory program like Soundtrack involves a high degree of recall - of spatial information, for instance. A greater insight into the nature of the memory problems encountered in using Soundtrack might be gained if its operations were analyzed more formally, using the sort of goal analysis techniques suggested by Payne (1985), Kieras and Polson (1985) or Card Moran and Newell (1983).

10.4 Adaptation

The problem which was discussed earlier of Soundtrack users having to click-check to access prompts illustrates a major problem of *adapting* software which was originally designed for sighted people: that an interface which makes it easier for a sighted person to use may actually result in an adapted design which is *more difficult* for the visually disabled person to use. A user's proficiency at using a computer system and the speed with which they learn to use it depends critically on the mental model they have of the system. It is only recently that designers have thought seriously about the user's model and have attempted to explicitly develop such a model within their design. Prior to that, system developers relied to a large extent on users having an understanding of how the underlying computer worked to be able to use a program. Thus, a quite detailed knowledge of the nature of computers served as the user model, and the use of most software was restricted to those with a high level of expertise, such as programmers.

As was mentioned in Chapter 2, the design of most wimp systems has been influenced by that of the Xerox Star, which introduced the desk-top metaphor. However, this interface is presented in a form which is entirely visual. If we just think of three of the constituents of wimps: windows, icons and menus, each one of those concepts is an inherently visual analogy; a window is something through which one *sees*, an icon is a *picture* and a menu is something one *reads* in a restaurant. For sighted people, vision is the principal means of interaction with the inanimate physical world. It is thus simple, for example, to make the connection between a paper document and an icon representing such a document. The icon gives the document a more concrete representation than the abstract, technical description of a disc file which might be given by a non-wimp-based system. Another example is that the action of making a selection from a pull-down menu is similar to that of ordering food in a French restaurant when one is too shy to attempt to pronounce one's choice.

So much of the power of the interface is lost when the user can no longer see it. Representation of objects in a non-visual manner is much less easy. Presumably a blind

person associates a paper document with a certain tactile sensation and perhaps a rustling sound. That would be very difficult to approximate on a computer. Similarly, the idea of pointing to an item on a food menu must be quite alien to a congenitally blind person. In order to adapt a visual interface, it has therefore been necessary to take a step away from the carefully designed model which makes wimp-style interfaces so attractive. We have, for example, something which we still call a menu, which bleeps and talks - quite unlike anything in a restaurant or on any desk top! As long as we are endeavouring to *adapt* software designed for sighted people for use by visually disabled people this is likely to remain a problem. Chapter 11 includes some suggestions as to how interfaces might be made more accessible by the application of tactile output and sounds which are less symbolic.

The alternative is not to adapt software but to design programs specifically for use by visually disabled people. In that case it would be possible to develop metaphors specifically suited to people whose experience of the world is non-visual. To a certain extent the design of the document window was a step in this direction. A direct analogy of the visual window was abandoned. Instead the design based on the grammatical rather than spatial structure of the document.

10.5 Providing a mouse reference frame

A related problem is that the auditory screen actually has no physical manifestation, it exists purely as sounds. However, it seems important to people to have a physical frame of reference - as evidenced by the number of people who wanted some kind of frame within which to move the mouse. Recall also that one subject said he found it disconcerting that he found himself pointing to the desk when referring to the auditory screen. Such reactions are entirely understandable because we are accustomed to most objects being detectable by all the senses. Objects which 'escape' one of the senses are often sources of fear: ghosts we can see but cannot hear, or conversely noises 'in the dark'.

The bitpad went some way to providing such a tangible frame of reference. The tablet of the bitpad itself gave the user a tactile outline which corresponded approximately to the boundaries of the screen (in both its visual and auditory incarnations). To relate it more closely to the auditory screen an overlay was added which consisted of tactile lines corresponding to the outlines of the windows. The bitpad also made orientation more simple because the positioning of the stylus was *absolute* whereas the mouse works *relatively*. That is to say that positioning the stylus in (say) the top left-hand corner of the bitpad always moves the cursor to the top left-hand corner of the screen. This is true whether the stylus is kept in contact with the bitpad or picked up and put down there. On the other hand, the position of the mouse on the desk and the cursor on the screen are not fixed in this way. For instance, if the mouse is picked up, any movement of it 'through the air' is not detected and the cursor remains in the same position on the screen. This means that users cannot relate the position of the cursor to anything fixed - in the way the subject who pointed to the desk was trying to do.

Although nearly all the subjects requested the provision of a frame, there are a number of obstacles to so doing - beyond the supplying of a bitpad. The relative positioning mechanism of the mouse is one such factor; it is possible for the position of mouse to become 'out of step' with that of the cursor. This can occur, for instance, if the mouse should be lifted from the surface of the desk (as described above), or if the ball under it should 'skid' at all. A similar effect to skidding can occur when the processor is busy. For instance, during a disc transfer, the processor may 'miss' interrupts from the mouse, so that the mouse pointer moves more slowly than the mouse. There would then be a problem of recognizing that this had occurred and then resetting the mouse, to get it back 'in step' with the cursor.

Some subjects suggested that any frame should not only reflect the layout of the screen, but should also give guidance as to the internal structure of windows. There is a fundamental reason why this could not be practical. Auditory windows are essentially 'soft' and

reconfigurable. Thus, for example, the **Dialogue** window may contain any one of a number of dialogues, each with a different layout.

Finally, an objection to the provision of the frame is that it represents additional, adapted hardware, whereas it is generally cheaper and easier to incorporate all adaptations in software. In the case of the Macintosh computer, there is a bitpad commercially available which can be used as a substitute for the mouse - without modification of application software. This may therefore be felt to be an acceptable addition for some users. In the more general case, it may be that it is possible to provide a simple frame which defines the outline of the screen to the mouse. This could be used as an aid to the early training of users, but ought not to be adopted for general use. In this way it might ease the process of learning to use the software, while an instructor would be available to help should the mouse and frame need to be 're-set'.

A very different approach to providing a tactile frame, by adding active control of mouse movements is suggested in Chapter 11.

10.6 The text selection mechanism

Many of the subjects had difficulty in manipulating selections in documents. There are a number of reasons for this. One is probably due to apparent inconsistencies which can occur between the current selection level and the segment of text which is actually selected. For instance, it is possible for the level to be **sentence**, but the selection to be a number of words and not a whole sentence. This is due to the fact that in Soundtrack selections are extended to the right of their current position. To illustrate how this comes about, consider the following scenario. There is a document with one sentence in, the current level is **word** and one word is selected:

The quick brown  jumps over the lazy dog.

If the user now moves the level up to **sentence** level, the selection extends to the end of the sentence to the right:

The quick brown fox jumps over the lazy dog.

Now there is the situation described above, in which the selection does not reflect the level. To extend the selection to the whole sentence it would be necessary to do a **Step backward**.

It could be even more confusing if the original selection had been the last word of the sentence:

The quick brown fox jumps over the lazy dog.

In this case raising the level would have no effect on the selection, and the user would then find that they were at **sentence** level, but with one word selected. Corresponding anomalies can occur at other levels, whenever the level is raised - **character** to **word**, **sentence** to **paragraph** and so on. Clearly there is a likelihood of confusion in cases such as these.

Consistency could be maintained if the selection was made to extend in both directions. That is to say that in the above example, on moving to **sentence** level the whole of the sentence would become selected. The problem with this, and the reason Soundtrack was implemented in the way it was, was that users will tend to work *forwards* through a piece of text. Suppose that in the above example the user had been proof-reading the text. She has checked the first three words in the sentence and then performs some correction of the fourth, *fox*. It is assumed that she will then want to work on through the text, and not have to hear the earlier words again. This effect would be greater at higher levels, when the repetition of several sentences might occur. There would appear to be some scope for further investigation as to which of the two approaches - selection extension to the right or in both directions - is more convenient.

It has been mentioned that some subjects did not like the fact that the level automatically became **point** whenever anything is typed on the keyboard. The reason selection was implemented in this way was to try to reduce the possibility of the kind of anomaly mentioned above. Any characters which are typed are inserted into a document at the *point* following the previous character. So, by definition, the selection must be a point between two characters at that time. Indeed, that is the reason it is necessary to have a **point** selection level. As implemented, therefore, whenever a user has typed anything the selection will be a point and the level will (consistently) be **point**. However, some subjects found this inconvenient because they frequently had to move through several levels as a result of this. For instance, the level might be **word**, the user types a correction, making the level **point**, but then needs to return to **word** level. To achieve that means finding the **Level up** control and double-clicking it three times.

If the mechanism of selection extension to the right is retained, then users will have to cope with inconsistencies between level and selection. In this case, it may be appropriate for the level not to be altered on typing. That would mean, for example, suppose the user is correcting the following sentence with the level at **word**:

The quick brown jumps over the lazy dog.

The user then types in a word. The selection will still become the point following the last character typed, but the level will still be **word**:

The quick brown fox jumps over the lazy dog.

Now there are a number of possibilities as to what the user might do, altering the selection and/or level. For instance, stepping forward would select to the end of the nearest word to the right:

The quick brown fox jumps over the lazy dog.

Alternatively, stepping back would have selected the previous word:

The quick brown fox jumps over the lazy dog.

Moving the level down (to **character**) would select the next character:

The quick brown fox **J**umps over the lazy dog.

The whole mechanism could be modified to operate in a manner consistent with this. There is again scope for a further investigation as to whether this is preferable to the way it was implemented in Soundtrack.

As mentioned earlier, the design of the **Document** window deviated from the principle of implementing direct analogues of visual interactions. The success of this design can be judged from the subjects' performance in the traced exercises (30, 31 and 33) and from their responses to the questionnaire. The layout of this window is illustrated earlier, in Figure 5.2.

The operation of the **Current level** control was inconsistent from all others, in that a single-click on it did not evoke the object's name, but the state of the current selection level in the window. None of the subjects seems to have noticed this inconsistency as they did not comment upon it. Many of them did use this facility to confirm the current level. Double-clicking caused the speaking of the current selection. This was a facility which was used much less frequently. Most subjects appeared to be able to remember what was the last selection spoken and relied on them being spoken as the selection was altered.

The **Say context** control was used quite frequently and to good effect, particularly when subjects were doing operations involving the typing in of individual characters.

Problems which many subjects had with the concept of selection levels have already been discussed above. However, most of them seemed to have little difficulty manipulating the associated controls. The majority seemed well able to remember the sequence of levels as they moved up and down. One subject (S1) did remark that he found the positioning of the two controls to be counter-intuitive, that he expected **Level up** to be on the right and vice versa. However, once he had had the designer's logic explained - that documents are read from the top left-hand corner down to the bottom right-hand - he became less confused.

Some subjects remarked that they would like a quicker mechanism to skip over several levels - to get rapidly from, say **point** level to **word** level. This problem was probably exacerbated by the fact that the level goes down to **point** when characters are typed (see above).

The **Step forward** and **Step backward** controls did not appear to present any problems. Most of the subjects were introduced to the jump controls only in the last exercise which they completed in the series but they did seem to be able to use them quite proficiently.

Referring to the **Scroll bar** and **Thumb bar**, the obvious explanation for the problems subjects had in manipulating these was that their design was completely contrary to the fixed grid concept. The position of the **Thumb bar** is not fixed and may be moved without the direct control of the user. Within Soundtrack this was the only object which had to be dragged and, as such, the subjects seemed to find the interaction clumsy. This might not have been a problem if they had received more practice. The whole operation relied heavily on spatial understanding. The position of the **Thumb bar** within the **Scroll bar** reflects the current position of the selection within the document. The former is a one-dimensional horizontal quantity while the latter is a more complex, two-dimensional concept. Thus the parallel is likely to be a difficult one to grasp for a person who can see neither side of it.

10.7 Summary

This chapter has shown how Soundtrack can be used, by showing how a number of people coped with learning to use it. From this it has been possible to conclude that Soundtrack does represent a viable approach to the adaptation of visual software. There were a number of problems in using the word processor, and these have been discussed. A major observation is that interface developments which make software easier for a sighted person to use, may actually make it *more difficult* for a visually disabled user. In the next chapter possible follow-up work is discussed, including suggestions as to how some of the limitations described above might be overcome.

Chapter 11

Conclusions

11.1 Introduction

This chapter concludes the thesis by firstly looking back and reviewing the project and then looking forward to the future. The retrospective view covers both the achievements of the research and criticisms of its limitations. The discussion of possible future work falls into two categories: further development of Soundtrack and more-general suggestions. The latter discussion raises suggestions for a number of projects which might be undertaken to follow up this work. The chapter concludes with a brief summary of the whole project.

11.2 Claims and achievements

It has been shown that it is possible to adapt visual wimp interfaces for use by blind people, by applying design principles of visual-to-auditory translation and constraining of the interface. The word processor, Soundtrack, which was developed as part of this project is the first wimp-based program which can be used by people who are totally blind.

The nature of the hand-ear coordination required to interact with a spatially-oriented auditory interface has been investigated. A methodology has been devised which could be the basis for extending existing well-established work on visual human-computer interfaces into an auditory dimension.

A multi-faceted evaluation has been carried out which means that there is evidence to back up the above assertions and which could form the basis of future comparisons of similar products.

Less major achievements include: exploration of some of the specific features of the Apple Macintosh and the adaptability of its interface and the development of an extension to keystroke level modelling.

11.3 Criticisms of the research

The word processor implemented did not have the full functionality of a commercial visual program. It would be rash to extrapolate the results obtained to a full-scale program. One element of visual wimp interfaces, the icon, was not implemented in the auditory adaptation.

A set of design principles was devised and they were tested by all being incorporated into the word processor. This means that it is difficult to abstract the features which relate to a particular design decision. For instance, the evaluation concluded that subjects had difficulty understanding the auditorily encoded spatial information. Had this been tested separately, in an abstract experiment it would have been possible to make more definite conclusions about the nature of the problem. As it was there was the probability of interference from other aspects of the word processing system.

The evaluation effectively measured only the ease-of-learning; it did not encompass the question of the ease-of-use for expert users. Indeed, none of the evaluation subjects spent enough time using the program to get beyond the early stages of learning it. This makes it difficult to make comparisons with other word processors (both for visually disabled and sighted people). At the same time, using only experienced computer users as subjects may have biased the results.

11.4 Enhancements to Soundtrack

Having collected data and opinions as to how it might be made more usable, it would seem reasonable to take heed of those ideas and see whether they do indeed improve the interface. Suggestions arose in the interviews and also have come from other people who have seen the program demonstrated.

Incorporating alternative interactions. As has been mentioned previously, two features of visual programs were deliberately omitted from Soundtrack despite the fact that they might make the program easier to use. These were the use of control-key equivalent commands and the option of typing in file names instead of selecting them from a list. These are both mechanisms which are alternatives to point-and-select interactions. They were omitted from Soundtrack in order to force the user to use the novel mechanism. The evaluation of Soundtrack described above shows that the point-and-select mechanisms are workable within an auditory interface. However, the typing-based interactions may be easier to use and so could be included in a production version of Soundtrack.

The user would still have the point-and-select mechanisms available and would be forced to use them in certain circumstances. For instance, visual programs generally provide control-key equivalents only of the more frequently-used commands. For the less-used ones the menu-entry selection would still be necessary. Similarly, typing of file names (into the **Select file** dialogue) is only feasible if the user can remember the name of the file she requires. Otherwise the list mechanism does allow her to hear the name of all the candidate files, and hopefully she will then be reminded which one she wanted.

The fact that a menu entry has a control-key equivalent must be communicated to the user. In visual menus this is done by adding it to the text of the entry. Figure 11.1 shows an example menu, in which some entries do have control-key equivalents. The **Open** command, for instance has the equivalent *control-O* (represented by $\wedge O$). In Figure 11.1 **New** and **Open** have control-key equivalents, but **Close** and **Quit** do not.

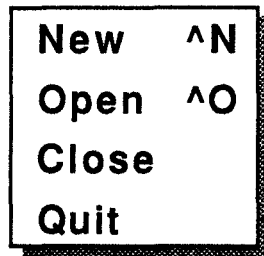


Figure 11.1. A visual menu which has control-key equivalents for the entries **New** and **Open**.

It is debatable how this communication of control-key equivalents could be implemented in an auditory interface. Should the control-key equivalent be pronounced when the menu entry name is spoken (e.g. "*New, control-N*")? That would mean that the user would be aware of the equivalent, and would be reminded of it each time he clicked on that entry. That should aid learning. However, it would slow the interaction down and might become tedious. The word *control* is perhaps too verbose and it might be replaced by a tone, which would be more succinct. While that might speed communication, it would increase the load on the user distinguishing different tones.

Whichever method of signalling is used, it would probably be best if this could be switched on and off, via an entry in the **Speech** menu. A novice user would not then have to be bothered with this further complication. Once the novice had become accustomed to using the program, she might turn the option on, and so learn and use the control-key equivalents. Once she has learned them, she might turn the option off once again.

There are also detailed questions as to how file name typing might be implemented. Should the user be forced to type the whole name? That would be unambiguous, but may be unnecessarily tedious. The best way to reduce the typing is for the program to analyze the list of names and allow the user to stop typing as soon as she had entered a string which

uniquely identified a name within the list. For instance, should the user wish to select the file *myprogram* from the following list:

mydata

myprogram

she could stop typing once she had entered the substring *myp*. The fact that a selection had effectively been made could be signalled by the sounding of a tone. The user would then have the option of acting on that selection (cause it to be spoken, or open it by executing the **Open** button, for instance) or (redundantly) typing the rest of the name.

Alerts. A proposed design for alerts was described in Chapter 4, but it was not implemented because of practical limitations. However, they could now be incorporated in a production version of Soundtrack.

Multiple clicks. When using Soundtrack, it is often necessary to activate one object repeatedly. For instance, if the user of Soundtrack wishes to move the document level from **point** all the way up to **whole text** she must double-click on the **level up** button five times. This could be speeded up by allowing the user to hold down the mouse button to signal that the object's action should be repeated (rather as a letter on the keyboard will be auto-repeated if its key is held down).

This might be implemented as follows. The user would still have to initially click the button twice, which would cause the object to be executed once, but instead of releasing the button after the second click it would be held down. After a suitable pause (to ensure that this is an intentional holding-down of the button), the object would be executed a second time. Thereafter, as long as the button remained down the object would be re-executed.

Re-entering the active window. On re-entering a window which is already active three tones are sounded in Soundtrack. The window's tone is sounded twice, to signal that this is the active one, followed by the tone of the underlying object. As mentioned earlier, the initial

double tone is redundant. It would be sufficient, and less confusing to sound only two tones: the window's once, followed by the internal object's.

Level adjustment on typing. Whenever the user types into a document the level becomes **point**. The motivation behind this design decision was that of maintaining consistency between the selection and the level, as discussed above in Section 10.6. However, the responses of some of the evaluation subjects suggest that this was in fact a nuisance. In fact, as discussed earlier, complete consistency of level and selection is not possible to achieve. So it would seem that the level should not be altered on typing. This would also mean that the level can only be altered by a deliberate action of the user.

Definition of syntactic units. Recall that, as far as the selection mechanism is concerned, a word is an unbroken string of letters and/or digits, not including the delimiting non-alphanumeric characters. That definition can make manipulation of words more intricate than necessary, since it means that the user must also rearrange spaces. It would be more convenient to allow a trailing space to be included as if it is part of the word. Notice that this applies only to spaces which terminate the word, not to other characters such as punctuation marks. Similarly, a paragraph was defined as being terminated by a newline character, but not including the newline. In fact, manipulation would again be simpler if the newline was to be included as part of the paragraph.

Scroll bar. The scroll bar, as implemented proved to be an impractical device. This appears to have been mainly because of the lack of feedback to the user as to the current horizontal position of the thumb bar as it moved. This might be overcome by the addition of an auditory signal. The two extreme positions of the thumb bar are well defined so it would be necessary to provide feedback which clearly marks these limits. Recalling the discussions in Chapter 3, an appropriate signal would be a repeating short tone. The position of the thumb bar would be signalled by the gap between the tones. At one extreme the tones would

become so frequent that they would merge into a single sound. At the other end, the gap could become essentially infinite, so that in fact no tone would be sounded.

Extending selections. Soundtrack allowed selections to be made only within the units of the current level. In other words, if the level is **word** then it is only possible to select single words. However there are times when it would be useful to be able to select a number of words - but not a whole sentence (the next level up). It would be quite simple to introduce a mechanism whereby the user could *extend* the current selection. For example, suppose a word is selected as shown:

Adapting user **interfaces** for visually disabled users.

The user could execute a **Step forwards** but by simultaneously holding (say) the shift key down the end of the selection would be extended to the end of the following word

Adapting user **interfaces for** visually disabled users.

The selection start would not be altered. Two words would then be selected, and the selection could be further extended as desired. The same mechanism would work analogously at all levels except **point** and **whole text**.

This facility combined with the multiple clicking mechanism proposed above would result in a means of selecting text rapidly.

Renaming the **Open** button. Some confusion seemed to occur in opening files during the evaluation. This may have been partly due to the fact that there are two objects within Soundtrack with the same name. Both the **File** menu and the **Select file** dialogue contain objects named **Open**. Clearly confusion could be avoided by renaming one of these. The button in the dialogue might be renamed **Open it**, for instance.

Filtering diagonal movements. As discussed in Section 7.5, it would be possible to reduce the problem of inadvertent diagonal mouse movements by 'filtering' the signals from the mouse. As mentioned earlier, it would be necessary to define the coarseness of the filter, what range of angles of movement will be counted as purely horizontal or vertical. Once a

movement has been processed the mouse's coordinate system will be reset around the new position. There would have to be some testing to assess the viability of this mechanism and the parameters to set for the ideal coarseness.

Auto-repeat warning. During the evaluation, subjects occasionally made errors caused by keys auto-repeating. This is not a problem for sighted users who will immediately see the effect of an inadvertent auto-repeat on the screen, but for a user relying on speech, repeated letters within words are usually very difficult to hear. It would, therefore, be a good idea to incorporate explicit auditory feedback whenever a key auto-repeats, which should ensure that a user will only ever use this feature deliberately.

Phonetic alphabet. Although Soundtrack allows text to be spoken letter-by-letter, mistakes can still occur because it is often difficult to hear which letter is being spoken, to distinguish between similar-sounding letters, such as *M* and *N*, *F* and *S* etc. This problem is tackled in a very different environment - that of vocal radio communication - by the use of the phonetic alphabet (*alpha, bravo, charlie, delta* etc). This could be incorporated as an alternative mode of speech.

Silencing speaking of input. As implemented, it is possible to choose whether input from the keyboard will be spoken by way of a menu command. The Vincent Workstation and Frank Audiodata use another mechanism which may be more convenient. On those systems, speech is automatically switched off when the user attains a set rate of typing. This is a simpler mechanism for the user to control. Speech output is slow, and cannot generally keep up with a quick typist (this is certainly true on Soundtrack), but this mechanism would mean that an efficient touch typist is not held back by the speech. On the other hand, if she should want to hear what she is typing she is likely to want to enter it more slowly anyway. The user would still have the option of switching it off all together (i.e. regardless of her typing speed) through a menu command. Also it should be possible to allow the user to adjust the trigger speed at which speech is switched on and off.

Customization. There are elements of the design of Soundtrack which need not be pre-defined by the designer, but can be left for the users to adjust, and so to 'customize' the interface to their own requirements and tastes. Some of the evaluation subjects advocated quite wide-ranging customization facilities, such as allowing users to define their own screen and window layouts. That would seem to be too great a degree of freedom, but there are a number of other possibilities. As a minimum, it would be desirable for the user to be able to adjust the volume of the auditory output. It would probably be best to allow these to be adjusted separately. The sort of mechanism described above for re-implementing the scroll bar might be used to provide the volume controls.

Adding features. The volume controls would amount to adding features to the auditory word processor. It would also be desirable to add further word processing facilities to the underlying program. Section 5.2 included a list of features of the visual word processor, *MacWrite*, which were omitted from Soundtrack, such as typeface selection and paragraph formatting. More of these might be incorporated in a later version. Some thought would have to go into incorporating the resultant increased complexity. For instance, it would be necessary to increase the number of windows. It would be possible to simply increase the number on the screen from eight. Alternatively it might be better to introduce another level of interaction, so that extra windows could be put onto a secondary screen, which would replace the primary one when an appropriate command is executed.

If visual features such as typeface selection and paragraph formatting are to be introduced, some thought will have to be applied into the question of how they are to be communicated through the auditory interface. This would presumably be achieved through some combination of modulation of tones and the synthetic speech.

Visual feedback. As was explained in Chapter 5, the prototype version of Soundtrack was implemented such that, in normal operation no visual feedback was given. This restriction was applied merely for the sake of the evaluation, to make it reasonable to use sighted and

partially sighted subjects. However, a production version could have visual feedback added. This would probably take the form of a grid representing the auditory screen. The internal layout of windows would be displayed as they are activated. This visual information would be of use to sighted people who might be helping a visually disabled user to learn or to use the system. Displaying the content of active documents as is already possible with Soundtrack is also useful. This might be retained by arranging the display of the grid such that it overlays the document, without significantly obscuring it.

Re-evaluation. It would be useful to evaluate the next version of Soundtrack. This could be done in such a way as to also fill in some of the gaps in the original evaluation. It would be useful to involve a subject or subjects who had no prior experience of using word processors and to train them to the level of expert. This would give a more realistic impression of the power and usefulness of the program. It was not possible in the original evaluation because of limitations on the amount of commitment which could be expected of the voluntary subjects. Subjects in the new evaluation would therefore have to be paid for their contribution.

Programming techniques. The prototype version of Soundtrack was implemented in such a way that it was, as far as possible, independent of the hardware on which it runs (i.e. the Macintosh). However, the production version need not be restrained thus. In particular, many Macintosh Toolbox routines expect by default that the user interacts in a standard way, but have facilities whereby this may be over-ridden. For instance, when a dialogue is opened a Toolbox procedure fields all mouse events. This would, for instance, interpret a click within the coordinates of a button in the dialogue as an execution of that button. However, by passing an appropriate procedure parameter to the Toolbox procedure this could be altered. In this way a click at some other coordinate could be passed on to the Toolbox procedure as if it had occurred within the button. This approach would make the modification of the program cleaner and more consistent. It would probably also reduce the size of the additional code. It would, however, be quite specific to the Macintosh.

Another facility specific to the Macintosh which might be used is that of *resources*. These are items of data attached to programs which can be altered without access to the source code of the program. Items such as menus, strings and windows are usually defined as resources. One of the features of resources is that it makes it comparatively easy to modify a program for use in different countries, so that text displayed by the program can be changed to other (natural) languages. In Soundtrack resources might be used to define the strings to be spoken by the speech synthesizer. This would facilitate the use of alternative synthesizers - which might use different phonetic encodings. The resource mechanism is explained fully in Apple Computer Inc. (1986) and Chernicoff (1985).

There is a more general discussion on programming techniques below, in Section 11.10. This advocates an investigation of the utility of using object-based programming languages. This is another technique which might be tried out on a new version of Soundtrack. It might be particularly convenient since Apple's Macintosh Programmers' Workshop Pascal does have object-based extensions.

11.5 Evaluation of pointing devices

Subjects in the evaluation of Soundtrack experienced some problems in using the mouse pointing device. An alternative device, the bitpad tablet and stylus, was also tested informally, and was preferred by a number of the subjects. This suggests that there is scope for further investigation of suitable pointing devices for use by visually disabled people. There has been quite extensive testing of the available visual pointing devices (see, for example, Card, English and Burr, 1978), but no corresponding evaluation for visually disabled users. This is, of course, not surprising since Soundtrack is the first wimp program designed to be used by blind people, which has special emphasis on hand-ear coordination. A number of suggestions of alternative pointing devices have been made by researchers who have examined Soundtrack, such as roller balls and joysticks, but without proper testing it is impossible to judge the practicality of any of these suggestions.

Controlled tests should be carried out to compare the efficiency of all the currently available devices for visually disabled users.

11.6 Experiments with auditory modulations

The results of the evaluation strongly suggest that using tones of different pitches is not a very useful means of communication. There must be more effective auditory means of communication. Bly (1982) has shown that even complex, multi-dimensional data can be communicated effectively in musical tones. Other researchers are beginning to investigate the possibility of enhancing computer interfaces by adding a sound dimension to the output. The sounds used might be purely symbolic. Sumikawa, Blattner and Greenberg (1986) propose the construction of *earcons* by linking together short sequences of rhythms and pitches. These would be built in hierarchical structures, reflecting the structures of the entities they represent. A different approach is advocated by Gaver (1987) whereby less symbolic sounds might be used which in some way resemble or represent the associated entity. Such suggestions should be pursued, both in the context of enhancing visual interfaces for sighted users and in the development of better purely auditory interfaces for blind people. A first step would be to carry out a pilot project in which a simple interface, based on Gaver's ideas, would be implemented and tested. This would give an indication as to the viability of this approach.

The model of pointing behaviour developed in Chapter 8 applies specifically to the auditory protocol implemented in Soundtrack. There is, therefore, scope for further investigation of the nature of 'hand-ear' coordination in a more general way. This could also contribute to the extension of Roberts and Moran's (1982) evaluation procedure to include auditory interfaces.

11.7 Graphics programs

Soundtrack is limited in its power. Basically, the design principles applied are not sufficient to cope with the adaptation of any programs which are graphical in nature. Bit-mapped computer displays have led to the development of a very large number of drawing programs for sighted users. None of these would be easy to adapt for access through an auditory interface for blind users. However, this does suggest an interesting challenge. Not only that, but it does also represent a real need. Blind people do often lack the facility to produce even the simplest of pictures. Yet this would often be very desirable. For instance, learning geometry is very much more difficult for a student who cannot sketch diagrams. It would be worthwhile to extend the ideas behind Soundtrack to this kind of problem.

Drawing programs use the mouse to define graphical objects on the screen. In general the user selects a graphical figure, such as a line, circle or rectangle from a menu and then defines a point on the screen at which the item will start to be created, by pressing the mouse. Then as the mouse is dragged the item grows, following the mouse pointer. Releasing the mouse button defines the limit of the item which then remains static on the screen. The approach to doing this through an auditory interface would be based upon the same ideas implemented in Soundtrack, whereby the user gets auditory feedback about her movement of the mouse. The user would have to have access to more complex information also, however. She would need to be able to get information about the graphical figures and their inter-relationships. This could be presented in a combination of symbolic sounds and speech.

11.8 Visual programming languages

In the introduction to this thesis it was pointed out that the wimp interface represents a stage in a trend towards more visual forms of interaction with computers. This is a continuing trend. One direction is in the development of visual programming languages. Within such languages programs are defined by the positioning of graphical objects relative to each other

on a screen. Examples of early graphical programming languages are *Interface Builder* (marketed by Expertelligence), the *Pinball Construction Set* (written by Electronic Arts and distributed by MacSerious Software) and *Ark* (the Alternate Reality Kit, see Smith, 1986).

Interface Builder is not (yet) a purely visual language. It enables the user to interactively design screen objects on a Macintosh. These objects can have actions attached to them, but those actions must be defined in a non-visual language (Common Lisp). However, the program's developer, Jean-Marie Hullot, has said that his ultimate aim is to develop a programming system on which the programmer would only use the mouse and never touch the keyboard. Obviously such a development could have a devastating effect on programmers who are visually disabled. Interface Builder is a genuine general-purpose programming system, as is demonstrated by the fact that it was built by being bootstrapped through itself.

Ark is a conceptually similar system, which runs on more powerful Xerox machines. It also is not purely visual in that detailed levels of the program must be written in Smalltalk. One application of Ark has been to create simulations of a wide range of physics experiments. One feature which both Ark and the Pinball Construction Set have in common is that they can control objects which move spontaneously on the screen (a ball rolling down a pinball table or a subatomic particle in a bubble chamber, for instance).

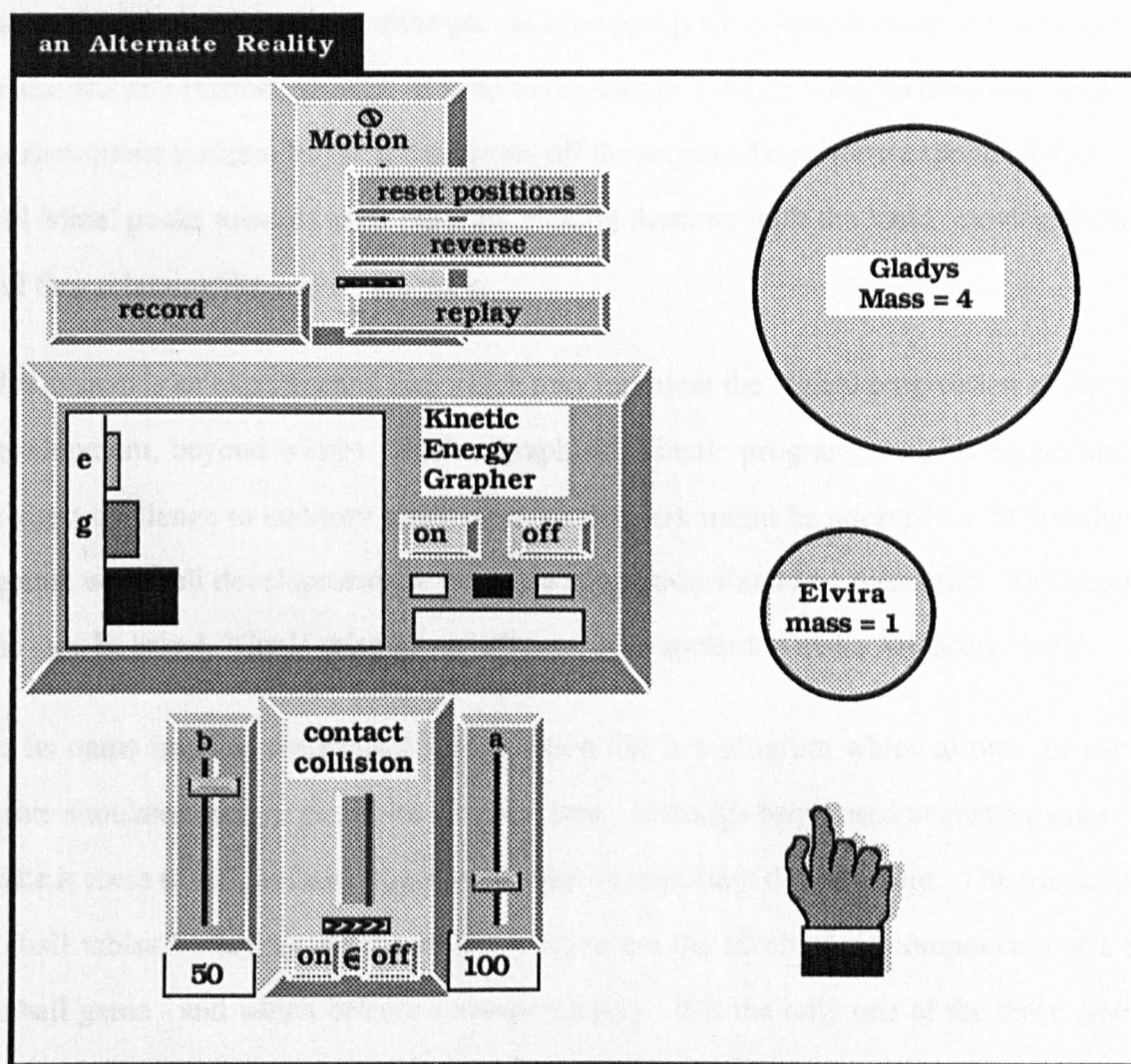


Figure 11.2. A typical Ark screen, showing a physics simulation.

Ark programs are built from visual components on a screen. (See Figure 11.2). The mouse pointer on Ark has the shape of a hand. To create a new program component the user uses the hand to 'lift' the component out of a 'factory'. The factory is represented by another picture. A compound program object can be built by the hand connecting 'wires' between objects. For instance, a user may create a simulation of pucks sliding on ice and colliding with each other. She might create a switch object - which looks like a light switch - which will control friction in the simulation. If the user switches friction off, by means of the hand and the switch, the pucks will behave in an idealized, friction-free manner. Clearly,

constructing an Ark program is an intensely visual task. At the same time, using the program is also. In the above example, as soon as a puck is introduced it will start to slide on the 'ice' in a random manner. It is up to the user to 'catch' it with the hand and place it in an appropriate position before it disappears off the screen. To perform experiments, the user will 'slide' pucks towards each other by picking them up with the hand, moving the hand and then releasing the grip on the puck.

This system is an experimental one which may represent the logical progression of interface development, beyond wimps. Such a graphical, kinetic program probably represents the greatest challenge to auditory translation, so that Ark might be adopted as the benchmark against which all developments in interface adaptation should be measured. The question can then be asked, "Could this form of adaptation be applied to make Ark accessible?"

As its name implies, the Pinball Construction Set is a program which allows the user to create simulated pinball games on the computer. Although being used to create games may make it seem trivial, in fact it represents quite an important development. The user creates pinball tables by laying out icons which represent the mechanical components of a real pinball game - and which behave correspondingly. It is the only one of the three systems mentioned which is purely visual; the programmer never has to write conventional code - nor does she have the facility to so do. However, being purely visual it is not as general purpose as the other systems, and can only be used to create pinball games.

11.9 Integration with other senses

For the purposes of investigating the utility of visual-to-auditory translation Soundtrack was deliberately designed not to give any visual feedback. This meant that all users were forced to rely on the auditory output. However, the interface could be extended so that it could be used by partially sighted people who prefer to use their residual vision as much as possible. An obvious approach would be to combine the best features of Soundtrack with those of a program like Inlarge (see Section 1.5).

A third sense - touch - might also be brought into play. There is research which is at an early stage at the moment at the Western Blind Rehabilitation Centre in the United States into the possibility of linking the mouse pointer on a wimp screen with the tactile output of an Optacon (Steele et al., 1986). There is scope for experimentation as to how this might best be implemented. The obvious approach is for the mouse pointer to act as if it were the Optacon camera. This would allow the user to sense exactly what is on the visual screen and to read any text thereon. However, it is likely that some of the visual items would be too complex to interpret, such as icons. In that case it might be better to simplify the screen along the same lines as was done in Soundtrack. The tactile output would then give simple information such as when a window boundary is being crossed. A powerful interface might be developed by combining these tactile features with auditory output.

An alternative approach to provision of tactile information would be to give the computer control of the movement of the mouse in such a way as to give the user the impression that the mouse is moving under physical restraints. 'Brakes' could be added to the ball under the mouse which would be applied in such a way as to provide physical resistance to its movement. Thus, if the mouse pointer is constrained to moving within the screen, then the mouse should be physically prevented, by application of a brake, from moving beyond those limits. The brakes could also be partly applied to provide resistance short of stopping the mouse. This could be used to mark boundaries - such as those between Soundtrack's auditory windows - or to simulate physical devices, such as a thumb bar moving along a notched guide.

Such broadening of interfaces to use different senses, in different ways would support the argument for greater flexibility of interface design, as discussed in the following section.

11.10 A generalized interface

One of the objectives behind this research was to investigate the feasibility of developing a generalized auditory interface which could be applied transparently to a wide number of

visual application programs. Ideally such an interface would be implemented as a program which would be run in conjunction with programs and have the effect of transforming the interface to the program into one based on sounds. It has been shown in this project that such a change would necessarily involve transforming both the output from the interface and the form of the input to it. The idea is that if such a generalized interface program were run in conjunction with, say MacWrite, then it would appear to the user that they were using a program similar to Soundtrack, but in fact behind the interface MacWrite would be doing all the processing. Ideally that same interface program would be capable of running with a number of word processors and other text-based applications.

The different forms of adaptation were discussed in Chapter 1. Three forms of adaptation were identified:

1. custom-build a device;
2. carry out relatively minor modifications to an existing device;
3. develop a device which can be used to make a range of existing devices accessible.

The ideal generalized interface described above would come into the third category. It would be a program which could be used with a variety of applications, in the way that the software screen enlarger *Inlarge* (Berkeley System Design) runs transparently with any applications programs. Such a generalized auditory interface would, however, be very difficult to realize in practice.

This kind of interface would have to work by intercepting communications between the user and the application. For instance, assuming the interface used a different (Soundtrack-style) screen layout, the interface would detect a mouse click and calculate the mouse's position in terms of the auditory screen's layout. Suppose the click occurred in an auditory menu, the interface would then pass a signal on to the application as if a click had occurred in its corresponding (visual) menu. Similarly, if the application writes any text to the screen, the

interface must catch that information and divert it to the speech synthesizer. This is the way that programs like Inlarge work.

The basic problem of applying such an approach to an auditory interface is the need for the interface to have access to a high-level description of the application with which it is running. It might be said that such an interface needs to be intelligent. An example which illustrates the difference between such a program and a non-intelligent adaptation such as Inlarge is that the display on a wimp-based system is *bit-mapped*. That is to say that internally a letter is effectively represented as a set of dots which are either black or white. At that level of representation the information that a particular set of black dots form a letter has been lost. In an auditory interface that would make it impossible for a speech synthesizer to pronounce the letter. The loss of the letter identity is not a problem in a program like Inlarge. In effect it merely enlarges the dots displayed which will retain the same outline. For an auditory interface to work, it must be capable of intercepting information at a higher level, that is it would recognize that the application was sending a letter to be displayed on the screen before the display hardware received the letter and turned it into a dot pattern.

The problem of transforming a bit-mapped display is just one example of the complexity involved. There are others, such as how to represent graphical items like scroll bars and icons.

It would seem that, given the design of current software, this would be very difficult, and probably impractical. However, it might be more feasible if basic applications software were to be designed with greater flexibility in its interface.

The design and implementation of Soundtrack has highlighted the difference in the degree of adaptation. It is not possible to conclude that it is infeasible to adapt all wimp software, however this research suggests that it would be impractical. One of the results to come directly from the implementation of Soundtrack was that it was *not* possible to provide

accurate auditory analogues of all the visual interactions. In particular the auditory document window differed radically from that in the visual program. In fact, the auditory version incorporates entities which are not part of the visual one, such as the **Level** controls, and indeed the concept of selection level. It would be possible to include such entities in a program modified for auditory operation, but with the amount of extra code involved it would be arguable as to whether this any longer represented an adaptation.

The second approach to adaptation is to modify a particular program. This has a disadvantage in cost since it implies duplication of effort, possibly to the extent that as much effort would go into the adaptation as was put into the original implementation. In referring to computer software this effort would be spent on programming and it may be that the use of modern programming techniques could reduce the effort required.

One such possible technique is that of object-oriented programming (Goldberg and Robson, 1983 and Pascoe, 1986). Programs are constructed of *objects* which interact by sending each other *messages*. As an example, objects often receive messages such as **draw**, which tells the object to display an image which represents itself in some way, on the screen. Such an object might be modified such that when it receives the **draw** message it actually displays itself on an auditory screen, rather than a visual one. Furthermore, the technique of *inheritance* applied in most object-oriented languages means that many different objects may share the same **draw** method, so minimizing the amount of modification required.

Another new program construction technique enables the specification of the (visual) interface of a program to be separated from the specification of its behaviour. This means that an application can be run with different interfaces. At the simplest level, this means that programs can be tailored for use by people who speak different languages. So, for instance, menu entries can be translated into other languages. Clicking on (say) the third entry in a menu will have the same effect on the application regardless of whether the entry is labelled *Close* or *Fermez*. The basis of this technique is that the description of the interface is held as

data in a file separate from the application code. That data is interpreted when the application is run so that the desired appearance and behaviour of the interface is achieved. It is possible to modify the interface by editing the data file. If the application is then run again, without it having been modified or re-compiled, it will be presented to the user through the new interface, although the application's behaviour will not have changed at all. The *resource* facility on the Apple Macintosh (Apple Computer Inc., 1986) is one example of this technique, and Apollo's Domain/Dialogue (Apollo Computer Inc, 1985) is another, more sophisticated one.

As described above, this technique has thus far been applied only to the appearance of visual interfaces. Given a suitably powerful interface-description mechanism (such as Domain/Dialogue), it could be extended to auditory interfaces, and might provide a suitable mechanism for transforming visual programs into auditory ones. This would be similar to the sort of transformation described above in object-oriented programs, whereby the specification of the interaction between the mouse and a visible item would be transformed into an interaction with an auditory item.

In fact interfaces need not be exclusively visual or auditory. Combining several of the ideas discussed above could result in interfaces which exploit both. What is more, such interfaces could be very flexible, so that users could specify their own preferred style of interface. For a sighted user that might involve a combination of visual and auditory, whereas a blind user would be able to specify an entirely auditory interface.

11.11 Representations of text

This project made available to a blind person the same kind of word processing facilities as available on visual wimp-based systems. Yet such word processors do have limitations, as was suggested earlier, in that they are not designed with regard to the user's internal representation of the text on which they operate; operations are defined in terms of the spatial structure of the text. The limitations of this approach are highlighted by the difficulty blind

word processor users have who have to 'read' text via other senses which lack the flexibility of vision. That was why the manipulation of documents in Soundtrack was based on syntactical structure. However there is a great deal of scope for applying more powerful representations and displays of text.

One example is the concept of the fisheye view (Furnas, 1986). This is based upon an analogy with photographic fisheye lenses which show a very broad, 180° picture within which the central portion is shown clearly and in detail, but the outer edges are distorted and show less detail. Furnas applies this approach to the display of hierarchically structured information, such as legal texts and computer programs. For each structure a Degree of Interest function is defined. Given the current point of focus, this gives a value for the degree of interest of all other points in the structure. The function incorporates a term describing the distance between the two points and a term which reflects the a priori level of interest of each point. The fisheye view is then constructed by cutting out all points with a degree of interest lower than a chosen threshold value. Having such a view enables the user to view and work in detail on parts of the structure, but to always have visible the current context.

Given the linear nature of speech output it is difficult for a blind person to navigate through a large document. Some form of auditory fisheye view might be a significant aid. The same sorts of controls as employed to control levels, selections and contexts in Soundtrack might be applied to control a fisheye 'lens' on a text. Whereas the Say context control allows users to hear the context within fixed bounds, controls could be added which would allow the user to hear parts of the text at a controlled level of interest.

As this goes beyond the facilities provided currently by visual word processors, such a facility could not be included in Soundtrack. However, it would be well worth investigating within the context of developing better tools which are custom-made for use by visually disabled people.

11.12 Summary

Developments in the design of computer interfaces have led to interactions becoming increasingly visual. These developments have succeeded in making computer-based equipment easier to use for the majority of people, so it is inevitable that such developments will be adopted widely. However, this trend is likely to seriously further disadvantage people who are visually disabled - unless means can be found whereby such obstacles can be overcome.

This thesis describes the successful attempt to provide a practical solution of the problem posed to visually disabled people by modern visual interfaces. A set of design principles were proposed as the basis of a means of adapting such visual interfaces for use by people who are totally blind. A word processor was implemented by applying those principles. The word processor was evaluated and it was concluded that it was a usable program. However, there was still a lot of scope to improve its ease-of-use. In fact, the results of this investigation into auditory interfaces suggest that there is potential to greatly enhance human-machine interfaces through the addition of an auditory component. This project produced the first system which makes wimps accessible to blind people.

References

- Adams, J., (ed.), *Learning to Cope - computers in Special Education*, an Educational Computing Special, 1982.
- Anandan, P., *Formal analysis of program editing*, Master's Thesis, Department of Computer Science, University of Nebraska, Lincoln, Nebraska, USA, (1979).
- Apollo Computer Inc., *Domain/Dialogue User's Guide*, Apollo Computer Inc, (Order No 004299), (1985).
- Apple Computer Inc., *MacWrite*, Apple Computer Inc., Cupertino, California, (1984).
- Apple Computer Inc, *Inside Macintosh, Volumes I, II and III*, Addison-Wesley, Wokingham, 1986.
- Bilofsky, W., *The CRT text editor Ned - introduction and reference manual*, R-2176-ARPA, Rand Corporation, Santa Monica, California, USA, 1977.
- Bly, S., *Sound and computer information presentation*, PhD Thesis, University of California, Davis, California, USA, (1982).
- Card, S. K., English, W. K. and Burr B. J., *Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT*, Ergonomics, 21, 8, pp601-613, (1978).
- Card, S. K, Moran, T. P. and Newell, A., *The psychology of human-computer interaction*, Lawrence Erlbaum Associates, London, (1983).
- Chapman, E. K., *Visually handicapped children and young people*, Routledge and Kegan Paul, London, (1978).
- Chernicoff, S., *Macintosh revealed*, Hayden Book Company, New Jersey, USA, 1985.
- Clark, R. and Koehler, S., *The UCSD Pascal Handbook*, Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1982.
- Dickinson, J., *Distance and location cues in retention of movements by a congenitally blind subject*, Journal of Psychology, 97, pp215-219, (1977).
- Doorlag, D. M. and Doorlag, D. H., *Cassette braille: a new communication tool for blind people*, Journal of Visual Impairment and Blindness, April 1983.
- Edwards. A. D. N., *Prospects for the use of computer-aided learning in special education*, Open University Computer-Assisted Learning Research Group, Technical Report, No. 46, (1984).
- Edwards, A. D. N., *Integrating synthetic speech with other auditory cues in graphical computer programs for blind users*, Proceedings of the IEE International Conference on Speech Input and Output, London, (1986).
- Edwards, A. D. N. and O'Shea T., *Courseware authoring systems for special education*, CAL85, Nottingham, (1985).

- Edwards, A. D. N. and O'Shea T., *Making graphics-based programming systems usable by blind people*, Interactive Learning International, 2, 3, (1986).
- Edwards, A. D. N. and Woodman, M., *UCSD Pascal on the Macintosh*, USUS (UK) Newsletter, Number 15, (September 1985).
- Edworthy J. and Patterson, R. D., *Ergonomic factors in auditory systems*, (in) Brown I. D. (ed), Proceedings of Ergonomics International 85, Taylor and Frances, (1985).
- Embley, D. W. and Nagey, G., *Behavioural aspects of text editors*, Computing Surveys, 13, 1, pp33-70, (1981).
- Fitts, P. M., *The information capacity of the human motor system in controlling the amplitude of movement*, Journal of Experimental Psychology, 47, 6, pp381-391, (1954).
- Freedman, J. L. and Landauer, T. K., *Retrieval of long-term memory: "Tip-of-the-tongue" phenomenon*, Psychological Science, 4, 8, pp309-310, (1966).
- Furnas, G. W., *Generalized fisheye views*, Proceedings CHI '86, Boston Massachusetts, March 1986.
- Gaver, W. W., *Auditory icons: using sound in computer interfaces*, (to appear in) Human-Computer Interaction, (1987).
- Goldberg, A. and Robson, D., *Smalltalk-80: the language and its implementation*, Addison-Wesley, London, 1983.
- Goldenberg E. P., Russell, S. J. and Carter, C. J., *Computers, education and special needs*, Addison-Wesley, London, (1984).
- Good, M., *Etude and the folklore of user interface design*, Sigplan Notices, 16, pp 34-43, June 1981.
- Goodrich, G. L., Jaffe, D. L., Schrier, E., Stevens, M. and Wung, P. C. F, *Tactile graphic braille display*, Rehabilitation Research and Development Center Veterans Administration Medical Center Annual Report, 1986.
- Hammer, J. M. and Rouse, W. B., *Analysis and modeling of freeform text editing behavior*, Proceedings International Conference on Cybernetics and Society, Denver, Colorado, USA, October 1979.
- Hawkrige, D., Vincent T. and Hales G., *New information technology in the education of disabled children and adults*, Croom Helm, London, (1985).
- Hermelin, B and O'Connor, N., *Spatial modality coding in children with and without impairments*, in *Spatial Abilities: Development and Psychological Foundation*, Academic Press, New Jersey, USA, 1982.
- Hofmann, M. A. and Heimstra, N. W., *Tracking performance with visual, auditory, or electrocutaneous displays*, Human Factors, 14, 2, pp131-138, 1972.
- Irons, I. T. and Djourup, F. M., *A CRT editing system*, Communications of the ACM, 15, 1, (1972).

- Jennings, P, *Spreadsheet for the visually handicapped*, Final-year project report, Computer Studies Department, Sheffield City Polytechnic, 1985.
- Jensen, K. and Wirth, N., *Pascal user manual and report*, Springer-Verlag, Berlin, 1975.
- Jernigan, K., *Blindness - concepts and misconceptions*, Braille Monitor, (1965).
- Joy, W. and Horton, M., *An introduction to display editing with Vi*, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1980.
- Kemmis, S., *Nomothetic and idiographic approaches to the evaluation of computer-assisted learning*, in Kemmis, S., Atkin, R and Wright, E., *How do Students Learn? working papers on computer-assisted learning*, (Occasional Paper no 5), Centre for Applied research in Education, University of East Anglia, (1977).
- Kennedy, T. C. S., *Some behavioural factors affecting the training of naive users of an interactive computer system*, International Journal of Man-Machine Studies, 7, pp817-835, (1975).
- Kernighan, B. W, *A tutorial introduction to the Unix editor*, Bell Laboratories, Murray Hill, New Jersey, USA, 1978.
- Kieras, D. E. and Polson, P. G., *An approach to the formal analysis of user complexity*, International Journal of Man-Machine Studies, 1985.
- Krasner, D., *Smalltalk-80: Bits of History, Words of Advice*, Addison-Wesley, London, (1983).
- Kurzweil, R., *The technology of the Kurzweil voice writer*, Byte, pp177-186, March 1986.
- Lakoff, G. and Johnson, M., *The metaphorical structure of the human conceptual system*, Cognitive Science, 4, pp195-208, (1980).
- Ledgard, H., Whiteside, J. A., Singer, A. and Seymour, W., *The natural language of interactive systems*, Communications of the ACM, 23, 10, pp556-563, (1980).
- Lowenfeld, B., *The changing status of the blind: from separation to integration*, Charles C. Thomas, Springfield, Illinois, USA, (1975).
- Lowenfeld, B., *Psychological problems of children with severely impaired vision*, in Cruikshank, W. M. (ed.) *Psychology of exceptional children and youth*, Prentice-Hall, Englewood Cliffs, New Jersey, USA, (1980).
- Lunney, D. and Morrison, R. C., *High technology laboratory aids for visually Handicapped chemistry Students*, Journal of Chemical Education, 58, 3, pp228-231, (1981).
- McGillivray, R., (ed.), *Voice output for computer access by the blind and visually impaired*, Aids and Appliances Review, The Carroll Center for the Blind, Newton, Massachusetts, USA, Volume 9, Number 10, 1983.
- Meyrowitz, N. and van Dam, A., *Interactive editing systems: part I*, Computing Surveys, 14, 3, pp321-352, (1982a).

- Meyrowitz, N. and van Dam, A., *Interactive editing systems: part II*, Computing Surveys, 14, 3, pp353-415, (1982b).
- MicroPro International, *Wordstar Reference Manual*, MicroPro International Corporation, San Rafael, California, USA, 1981.
- Miller, G. A., *The magical number seven, plus or minus two: some limits on our capacity for processing information*, Psychological Review, 63, 2, pp 81-97, 1956.
- Mills, A. W., *On the minimum audible angle*, Journal of the Acoustical Society of America, 30, 4, pp 237-246, (1958).
- Pain, H., *A computer aid for spelling error classification in remedial teaching*, in Lewis, R. and Tagg, D., (eds.) *Computers in Education*, Noth-Holland, Amsterdam, (1981).
- Pascoe, G. A., *Elements of object-oriented programming*, Byte, August 1986.
- Patterson, R. D., *Guidelines for auditory warning systems on civil aircraft*, Civil Aviation Authority, CAA Paper 82017, (1982).
- Payne, S. J., *Task-action grammars*, in Shackel (ed.) *Human-Computer Interaction - Interact'84*, North-Holland, Amsterdam, (1985).
- Pearce, B. G. (ed.), *Health hazards of VDTs?*, Wiley, Chichester, (1984).
- Plastics Design Forum, *A computer terminal for blind people*, March/April 1983.
- Reisner, P., *Formal grammar and human factors design of an interactive graphics system*, IEEE Transactions on Software Engineering, Vol. SE-7, 2, (1981).
- Roberts, T. L., *Evaluation of computer text editors*, PhD Thesis, Stanford University, (1979).
- Roberts, T. L. and Moran, T. P., *A methodology for evaluating text editors*, Proceedings of the Conference on Human Factors in Computer Systems, Gaithersburg, Maryland, USA, (1982).
- Rostron, A. and Sewell, D., *Microtechnology in Special Education*, Croom Helm, London, (1984).
- Salvadori Paleotti, A., Kilroy, J. and Edwards, A. D. N., *Providing computer-aided learning for mentally handicapped children*, Computer Education, February 1985.
- Schmucker, K. J., *Object-oriented Programming for the Macintosh*, Hayden, Hasbrouck Heights, New Jersey, USA, (1986).
- Smith, R. B., *The Alternate Reality Kit: an animated environment for creating interactive simulations*, Proceedings 1986 IEEE Computer Society Workshop on Visual Languages, Dallas, Texas, USA, (1986).
- Smith, D.C., Irby, C., Kimball, R., Verplank, B., and Harslem, E., *Designing the Star user interface*, Byte, April 1982, pp242-282.
- Smith, D., Irby, C., Kimball, R. and Harslem, E., *The star user interface: an overview*, Proceedings of the National Computer Conference, 1982.

- Smith-Kettlewell Institute, *Catalogue of Vocational and Educational Aids*, Smith-Kettlewell Institute, San Francisco, (1983).
- SofTech, *The MacAdvantage: UCSD Pascal*, SofTech Microsystems, San Diego, California, USA, (1984).
- Spielberger, C. D., *Anxiety as an emotional state*, in Spielberger, C. D. (ed), *Anxiety: current trends in theory and research, Volume 1*, Academic Press, New York, (pp23-49), 1972.
- Stallman, R. M., *Emacs manual for Twenex users*, Massachusetts Institute of Technology, AI Memo 555, 1981.
- Steele, R. D., Goodrich, G., Hennies, D., M^cKenzie, J. and Duluk, J., *Establishing design/operational features for portable blind reading aids*, Rehabilitation Research and Development Center Veterans Administration Medical Center Annual Report, 1986.
- Stevens, M. D., *Tactile computer terminal*, Proceedings of the 2nd International Conference on Rehabilitation, Ottawa, 1984.
- Stevens, S. S. and Newman, E. B., *The localization of actual sources of sound*, American Journal of Psychology, 48, pp297-306, (1936).
- Sumikawa D. A, Blattner, M. M., Joy, K. I. and Greenberg, R. M., *Guidelines for the syntactic design of audio cues in computer interfaces*, Proceedings of the Nineteenth Annual Hawaii International Conference on System Sciences, Honolulu, Hawaii, USA, May 1986.
- Sumikawa D. A, Blattner, M. M. and Greenberg, R. M., *Earcons: structured audio messages*, (submitted for publication 1986).
- Surridge, R., *Creme de la KRM: the Kurzweil Reading Machine*, Public Library Journal, 1, 3, pp31-37, (1986).
- Sutherland, A. T., *Disabled we Stand*, Souvenir Press (Educational and Academic), London, 1981.
- Teitelbaum, T., *The Cornell program synthesizer: a syntax-directed programming environment*, Communications of the ACM, 24, 9, page 563 (1981).
- Thacker, C. P., M^cCriecht, E. M., Lampson, B. W., Sproull, R. F. and Boggs, D. R., *Alto: a personal computer*, in Sieworek, D., Bell, C. G. and Newell, A, (eds.), *Computer Structures: Principles and Examples*, M^cGraw-Hill, New York, 1982.
- University of Cambridge, *Zed reference manual*, University of Cambridge, Computing Service, November 1984.
- Vincent, T., *Computing and the blind*, Open University, Computer-Assisted Learning Research Group Technical Report No. 52, (1986).
- Vinje, E. W. and Pitkin, E. T., *Human operator dynamics for aural compensatory tracking*, IEEE Transactions on Systems, Man, and Cybernetics, SMC-2, 4, pp504-512, 1972.

- Walther, G. H. and O'Neil, H. F., *On-line user-computer interface - the effect of interface flexibility, terminal type, and experience on performance*, Proceedings 1974 National Computer Conference, Volume 43, pp379-384, AFIPS Press, Arlington, Virginia, USA (1974).
- Ward, W. D., *Absolute pitch*, *Sound*, 2, 3, pp14-21, 1963.
- Welford, A. T., *Fundamentals of skill*, Methuen and Co., London, (1968).

INDEX

activation 33, 34, 35, 58, 59, 64, 79, 80, 119, 125, 130, 131, 176, 177, 181, 203
 acuity, visual 11
 adaptation 6, 12, 114, 187, 200, 218
 aircraft 42-43, 48
 alert 35, 63, 64, 94, 204
 Alternate Reality Kit (Ark) 212-215
 (See also Visual programming languages)
 Alto (Xerox) 36-37
 Announce dialogues 87
 anxiety 102, 130, 157
 Apollo 38, 72, 219
 Apple Computer Inc. 8, 18, 38, 70, 73-75, 200, 209, 219
 Ark (Alternate Reality Kit) 212-215 (See also Visual programming languages)
 Association of Visually Handicapped Office Workers (Avhow) 105, 106, 107
 Audiodata (Frank) 13, 15, 106, 117, 118, 125, 127, 130, 188, 206
 audio recording 109
 Audiocalc 16
 auditory 3, 6, 14, 16, 19, 26, 54-60, 62-64, 67-70, 72, 74-77, 79-81, 85, 86, 90, 95, 97, 98, 101, 108, 110, 113-115, 126, 129, 133, 135, 136, 139, 140, 142-144, 152, 155, 158, 164-166, 189-190, 192-193, 199-202, 204, 206-208, 210-211, 214-220
 auditory icon 69
 auditory object 54, 55, 75, 79, 113, 133, 152
 auditory screen 56-59, 76, 77, 90, 108, 133, 135, 136, 142, 152, 164, 192-193, 208
 auditory window 57, 58, 114, 143, 144, 216
 Edit menu 89, 147, 172
 File menu 80, 88, 91, 92, 116, 163, 178, 205
 document window 57, 63, 64, 66, 67, 80, 117, 152, 192, 218
 dialogue 34-36, 62, 63, 79, 87, 88, 90-94, 109, 126, 134, 159, 162, 163, 170, 178-179, 181-183, 201, 205, 208
 alert 35, 63, 64, 94, 203
 auto key 161, 162
 auto-repeat warning 206
 Avhow (Association of Visually Handicapped Office Workers) 105, 107
 Berkeley System Design 18, 216
 bit-mapped display 37, 217
 bitpad 28, 71, 115, 121, 123-125, 131, 193-194, 209
 blindness 10, 11
 Brailink 18, 106, 123
 braille 12, 17, 18, 21, 70, 97, 108, 113
 button 29, 32-36, 56, 58, 62, 63, 67, 71, 72, 79, 85, 86, 91, 121, 125, 141, 163, 180-182, 203, 205, 208, 211
 Cancel 62, 91, 92, 179, 182
 Current level 85, 86, 173, 185, 197
 Down button 91, 163, 181
 Level down 80, 81, 117, 153, 184
 Level up 80, 81, 117, 153, 173, 184-185, 196-197
 Name 163
 No button 92, 93
 Up button 91, 163, 180, 181
 Yes button 93
 Cancel 62, 91, 92, 179, 182
 Character level 67, 81
 chemical spectra 45
 click 29, 34, 80, 85, 91-94, 119, 135-138, 145-147, 150, 161, 162, 172, 173, 177-178, 180-182, 185, 190, 203, 205, 208, 216
 Close 88, 89, 92, 201-202, 218
 colour vision 11

- communication aid 21
- Computer-Aided Learning 4, 5
- Concept Keyboard 16
- constraint 2, 36, 54, 55, 57, 69, 70, 102, 199
- control-key alternatives 30, 86, 95, 201-202
- controlled experiments 99
- Copy 89
- Current level 85, 86, 173, 185, 197
- cursor 14, 24, 25, 27, 32, 33, 35, 54, 65, 66, 71, 76, 78, 79, 122, 140, 193
- customization 208
- Cut 89, 172, 176
- design principles 10, 199-200, 211, 220
- desktop 37
- diagonal movements 95, 115, 116, 128, 129
- dialogue 34-36, 62, 63, 79, 87, 88, 90-94, 109, 126, 134, 159, 162, 163, 170, 178, 180, 182-184, 201, 205, 208
 - Get target string 93, 134-138, 145, 155
 - Get file name 93
 - modal 34, 35, 62-64
 - modeless 34, 62-64, 94
 - Save file dialogue 92
 - Select file 79, 88, 91, 163, 178, 180, 183, 184, 201, 205
- disability 2, 3, 4, 5, 9-12, 19, 96, 104, 129
- disc 17, 22, 31, 34, 35, 62, 63, 80, 89, 91, 92, 94, 191-192,
- display editor 25
- document window 57, 63, 64, 66, 67, 80, 117, 152, 193, 219
 - Current level 85, 86, 173, 185, 197
 - Jump backward 84
 - Jump forward 84
 - Level down 80, 81, 117, 153, 184
 - Level up 80, 81, 117, 153, 173, 185, 186, 196-197
- double-click 29, 72, 79, 85, 87, 91, 94, 121, 125, 156, 161, 162, 163, 171, 176, 180, 182, 196, 203
- Down button 91, 163, 181
- dragging 29, 33, 85, 198, 211
- drawing program 211
- earcon 44
- ease-of-learning 97, 200
- ease-of-use 5, 97, 102, 112, 115, 132, 200, 220
- Edit menu 89, 147, 172
 - Cut 89, 172, 176
 - Copy 89
 - Paste 89, 161
 - Find 89, 90, 94, 178
- editable text 35
- editor 22-26, 37, 65, 98-101
- Electronic Arts 212
- Emacs 25
- Equation (2) 145-147
- error 16, 28, 34-36, 62-64, 81, 85, 89, 94, 100, 101, 118, 137, 139, 143, 147, 148, 150-152, 154, 160, 163, 164, 166-168, 171, 174-177, 182, 183, 186, 187, 189, 206
- Etude 102
- evaluation 2, 3, 20, 53, 75, 91, 94-98, 101-113, 115, 126, 128, 132, 138, 140, 141, 144, 152, 156, 157, 159, 165, 166, 186-187, 189, 199-201, 204, 206-210
- event 63, 160-162, 168, 180
- execute 56, 58, 64, 81, 89, 156, 163, 164, 205
- Exercise 0.1 138, 141, 145, 147, 148, 150-157, 159, 167, 188
- Exercise 0.2 152-154, 156
- Exercise 30 159, 167, 170, 174, 185
- Exercise 31 118, 165, 171, 175, 186

Exercise 33 157, 163-166, 177, 178,
 181, 183, 184, 189
 Expertelligence 212
 extending selections 205
 field observations 98
 field of vision 11, 12
 File menu 80, 88, 91, 92, 116, 163, 178,
 205
 Close 88, 89, 92, 201-202, 218
 New 80, 88, 92, 161, 179, 201-202
 Open 30, 62, 63, 80, 88, 91, 92,
 116, 161, 163, 164, 179, 180,
 183, 201-203, 205
 Quit 88, 89, 92, 201-202
 First Byte 71
 Fitts' Law 138-140, 155
 formal analysis 99
 Frank Audiodata 13, 15, 106, 117, 118,
 125, 127, 130, 187, 206
 generalized interface 6, 215-216
 Get file name 93
 Get target string 93, 134-138, 145, 155
 Goms 99
 grid 55-58, 60, 62, 76, 77, 91, 92, 115,
 133, 134, 163, 178, 180, 198, 208
 hand-ear coordination 199, 209
 hardware 7, 13, 15, 21, 22, 36, 37, 56,
 70-72, 115, 120, 123, 131, 183,
 208, 217
 Hick's Law 138, 145
 Hidden 90, 177
 highlighting 31, 32, 36, 217
 human-machine interface 1, 26, 220
 IBM 13, 15, 18, 107
 icon 8, 29, 31, 37, 68, 69, 73, 191, 200,
 214-215, 217
 idiographic evaluation 110
 information technology 1, 5, 9, 12
 Inlarge 18, 216-217
 Interface Builder 212
 Interface menu 90
 Hidden 90, 177
 interview 3, 105, 109, 110, 112, 126,
 127, 132, 187, 200
 introspection 98
 investment 96
 joystick 28, 209
 Jump backward 84
 Jump forward 84
 Key down 161
 keyboard 8, 15-18, 25, 27, 67, 70, 71,
 88, 120, 127, 159, 161, 171, 190,
 203, 206, 212
 keystroke-level model 100
 kinaesthetic sense 140
 Kurzweil Reading Machine 42
 learning 4, 5, 10, 30, 74, 97, 100-102,
 108, 113, 117, 118, 120, 123, 124,
 132, 156, 157, 159, 187-189, 194,
 198, 200, 202, 211
 level 3, 7, 9, 10, 19
 Level down 80, 81, 117, 153, 184
 Level up 80, 81, 117, 153, 173, 185,
 196-197
 lightpen 28
 Lisp 213
 list 12, 29, 35, 36, 62, 91, 95, 126, 141,
 152, 180, 183, 201-203, 207
 Living Videotext Inc. 65
 MacAdvantage 72-73
 Macintosh 8, 18, 31, 38, 70-78, 131,
 183, 194, 200, 208-209, 212, 219
 MacSerious Software 212
 MacTablet 71
 MacWrite 74, 207, 216
 magnification 12, 13, 18
 memory 22, 59, 68, 73, 74, 76, 77, 116,
 118, 129, 156, 190

menu 8, 29, 30, 33, 34, 58, 60, 61, 73, 75, 79, 80, 81, 85-92, 95, 107, 116, 117, 119, 127, 129, 147, 156, 163, 167, 172, 178, 190-192, 201-202, 205-206, 209, 211, 216, 218
 Edit 89, 147, 172
 File 80, 88, 91, 92, 116, 163, 178, 205
 Interface 90
 pop-up 29
 pull-down 29, 192
 Speech 81, 85
 ticked entry 87, 88, 90
 modal dialogue 34, 35, 62-64
 Model Human Processor 99, 139
 modeless dialogue 34, 62-64, 94
 More 25, 65, 71, 165, 207
 mouse 8, 27-29, 32-38, 54, 56-59, 62, 64-66, 70-72, 75, 76, 78, 79, 85, 86, 94, 95, 109, 114-116, 118-121, 123-125, 127-129, 131, 133-143, 145-147, 150, 151, 153, 154, 156, 159-164, 171-173, 177-186, 187, 192-194, 203, 205, 208-209, 211-213, 215-216, 219
 mouse pointer 27-29, 186, 194, 211, 213, 215
 mouse-downs 160
 Move 134-137, 145, 166
 Name 163
 New 80, 88, 92, 161, 179, 201-202
 No button 92, 93
 nomothetic evaluation 110
 object-oriented programming 73, 218-219
 Open 30, 62, 63, 80, 88, 91, 92, 116, 161, 163, 164, 179, 180, 183, 201-203, 205
 Optacon 6, 107, 123, 215
 paragraph 65, 67, 80-82, 84-86, 100, 196, 204, 207
 paragraph level 85, 185
 PARC (Xerox Palo Alto Research Center) 36, 37
 partially hearing 4
 Pascal 72, 73, 94, 209
 Paste 89, 161
 perimeter 11
 Perq 38
 personal workstation 22
 Phonetic alphabet 206
 Pinball Construction Set 213, 215
 (See also Visual Programming Language)
 pitch 77-79, 114, 117, 121, 122, 124, 125, 128, 131, 132, 135, 140, 147, 151, 171, 178
 Plan route 134-137, 145
 point 1, 4, 8, 15, 19, 24, 26, 27, 32, 33, 54, 56, 58, 67, 68, 71, 81, 86, 89, 122, 142, 171, 176, 177, 189, 195-196, 198, 203-205, 211
 point level 176, 177, 198
 pointing devices 209
 pop-up menu 29
 programming techniques 73, 209, 218
 Proof read 87, 88, 171
 Psychological models 99
 pull-down menu 29, 191
 questionnaire 101, 102, 106, 107, 109, 112, 113, 115, 116, 121-122, 130-131, 157, 167, 187, 197
 (See also Structured interview)
 Quit 88, 89, 92, 201-202
 resource 75, 209, 219
 RNIB (Royal National Institute for the Blind) 15, 105, 106, 126, 188
 Royal National College for the Blind 15, 188
 Royal National Institute for the Blind (RNIB) 15, 105, 106, 126, 188
 Save file dialogue 92
 Say characters 87, 88, 184
 Say context 86, 172, 197

- scotoma 11
- scroll bar 33, 114, 204, 207, 217
- selection 29, 30, 33, 35, 99, 126, 127, 179-182, 190, 196, 201, 203, 205
- Select file 79, 88, 91, 163, 178, 180, 183, 184, 201, 205
- sentence level 173, 195
- Smalltalk 36, 37, 73, 213
- Smith-Kettlewell Institute 6
- Smoothtalker 71, 72, 86, 118
- Snellen Chart 11
- software 2, 3, 5, 7-10, 15, 18-21, 30, 37, 38, 53, 66, 70, 71, 73, 74, 96-98, 105, 115, 128, 130, 131, 186, 189, 191-192, 198, 216-217
- Soundtrack 2, 3, 53, 54, 70-75, 77, 81, 85, 87, 90, 91, 94, 95, 97, 98, 101-104, 107, 109, 110, 113, 115, 117-119, 121-128, 130-133, 144, 147, 148, 155-157, 159, 164-168, 173, 182, 186, 187-191, 194-195, 197-201, 203, 205-211, 214-217
- Special Education 19
- spectroscopy 45
- Speech menu 81, 85
 - Announce dialogues 87
 - Proof read 87, 88, 171
 - Say characters 87, 88, 184
 - Say typing 87
- speech synthesis 7, 15, 16, 72, 86, 209, 217
- Star (Xerox) 31, 37, 68, 192
- Star Microterminals 16
- structured interview 110, 127, 132, 156
(See also questionnaire)
- Subject S1 105-107, 114-118, 126, 127, 148, 150, 152-154, 159, 165, 167, 168, 170-173, 186, 197
- Subject S2 105-107, 114, 115, 118, 119, 126, 127, 130, 148, 150, 152-154, 159, 165, 170, 174, 176, 186
- Subject S3 106, 107, 120, 121, 148, 150, 151, 163, 166, 177, 178
- Subject S4 106, 121, 122, 148, 150, 151, 163, 166, 178, 180, 181
- Subject S5 105
- Subject S6 105
- Subject S7 106, 122, 123, 148, 150, 152, 163, 166, 181, 182
- Subject S8 105-107, 123, 124, 148, 150, 166, 183
- Subject S9 106, 124-126, 148, 150, 152, 166, 184
- Summagraphics MacTablet 71
- Sun 38
- tactile 6, 12, 13, 16, 18, 26, 71, 108, 113, 140, 163, 177, 180, 192-194, 215
- tactile screen 16
- taso* 13-14, 127
- T_{choose} 137, 138, 144-146, 153, 155, 156
- T_{click} 137, 138, 145, 153
- T_{move} 137, 138, 144-147, 153-155
- T_{plan} 137, 138, 145, 146, 153, 154, 156
- T_{position} 137-139, 143
- T_{think} 138, 144-146, 153, 154
- Teco 99
- TeleSensory Inc. 6, 18
- text 3, 6-8, 11-15, 17, 18, 21-24, 26-28, 30-33, 35, 58, 64-68, 72, 74, 80-82, 86, 89, 90, 98-102, 107, 122, 127, 134, 159, 160, 164, 167, 168, 170, 171, 173, 175, 176, 178, 186, 194, 201, 203, 205-206, 209, 215-216
- Thumb bar 80, 84, 85, 114, 152, 153, 199
- ticked 87, 88, 90
- tiling 25
- timbre 77
- Timestamps 160

tone 15, 34, 36, 55-59, 62-64, 76-79,
87, 95, 117-122, 130, 134, 135,
137, 140, 151, 178, 180, 183, 202-
205

Toolbox (Macintosh) 73, 75, 209

tracing 22, 108, 109, 119, 141, 148,
157, 159, 160, 162, 166, 168, 186,
197

training 77, 96, 102, 103, 105, 108,
112, 113, 116, 118-120, 124, 130,
132, 187-189, 194

tunnel vision 11, 12

UCSD 72, 73

Up button 91, 163, 180, 181

Versabraille 6, 17, 18, 106

Vert 15, 16

Vincent Workstation 16, 206

Vista 18

visual disability 1, 2, 10, 12, 19, 165,
186, 212

visual feedback 58, 75, 140, 207-208,
214

visual programming languages 211

waveform 44

whole text 80, 81, 203

wimp 8, 20, 29, 36, 38, 57, 63, 66, 73,
75, 96, 97, 129, 130, 186, 189,
191, 199-200, 209, 211, 215, 217

window 8, 25-27, 29, 32-35, 57-59, 62-
64, 66, 67, 73, 76-81, 87, 86, 88,
90, 91, 92, 94, 108, 114, 116, 117,
119-121, 123, 124, 128, 130, 138,
134, 141-148, 151-153, 155-157,
160-163, 166, 167, 171, 172, 174,
177, 178, 182-184, 186, 190-194,
197, 203, 207-209, 215, 218

word level 173, 197, 199

word processing 2, 3, 5, 9, 15-17, 19,
30, 34, 53, 64-66, 69, 70, 74, 75,
76, 95, 98, 102, 103, 105-107,
112, 115, 121-124, 126, 130, 133,
155, 158, 165, 166, 173-174, 178,
187, 188-190, 200, 207-208, 216,
220

Wordstar 15, 106, 107, 122, 127, 130,
173, 190

wysiwyg (*what-you-see-is-what-you-get*)
26

Xerox 37, 129, 191, 212

Yes button 93

Appendices

- A** Evaluation Session Record Sheet
- B** The Evaluation Exercises
- C** The Structured Interview Questionnaire
- D** Transcript of a sample session of using Soundtrack
- E** Useful Addresses

Appendix A

Evaluation Session Record Sheet

One sheet was filled out for each session undertaken by each subject. For exercises which were traced the time of day was recorded at the start of the exercise, so that it was possible to identify the exercise in the trace print-out. The section on *dribble files* was intended simply as a reminder to the tester to ensure that the trace was correctly saved at the end of the session. The table referring to *0 exercise* was not used in practice. It was an experimental record which was discarded in the actual evaluation. A typical completed sheet is also included.

Evaluation session record

Subject number

Session number

Date: / /

Start time :

End time :

Time Exercise

: :	
: :	
: :	
: :	
: :	
: :	

Subject's comments:

Observations:

0 exercise

	found without speech correct?	
1		
2		
3		
4		
5		
6		
7		
8		
9		

Exercises completed:

Dribble file saved ?

Dribble file name

Appendix B

The Evaluation Exercises

This Appendix contains a copy of the instructions for the evaluation exercises. None of the subjects completed all the exercises. Subjects S1 and S2 did go so far as to complete Exercise 31, except that they *were* given verbal assistance - which was recorded on tape (see Chapter 9). The other subjects spent less time on the evaluation (see Chapter 6), so a special final exercise was devised for them. This was numbered 33. No one attempted Exercise 32.

Training Exercises

Exercises which are underlined will be monitored.

0. Progress exercises

Exercises 0 should be given several times during the evaluation, to plot progress. Each one should be given firstly, unmonitored, at the beginning of a session, and then monitored towards the end of that session.

0.1 Location of large objects

This exercise should first be given during the first session following that in which the subject completed exercise 3.

Locate a number of (almost) randomly chosen windows

- reset the windows

"I am going to ask you to find a number of windows. They have all been reset, so none of them is active at the moment. I will ask you to locate one window. Tell me when you think you have found it - you can check whether you have found the right one, by pressing the mouse and getting speech *if you want* if you're pretty sure you have found it then you need not, that's up to you. Do *not* activate (double-click) the window. As soon as you say you have found then one I want I'll give you the name of another one to find in the same way. I will ask you to find a number of windows in this way, in a random order. I will give you help to locate windows if you want it."

- find the windows in one of the lists

A	B	C	D
Speech menu	Edit menu	Document 1	Document 2
Dialogue	Interface menu	File menu	Edit menu
Document 2	Document 1	Document 1	Speech menu
File menu	Speech menu	Alert	Alert
Document 1	Alert	Interface menu	Dialogue
Edit menu	Document 2	Document 2	Document 1
Speech menu	Edit menu	Speech menu	File menu
Alert	Dialogue	Dialogue	Interface menu
Interface menu	File menu	Edit menu	Document 2
E	E	G	H
Document 1	Alert	Interface menu	Dialogue
Interface menu	File menu	Edit menu	Document 2
Edit menu	Document 2	Document 2	Document 1
Speech menu	Edit menu	Speech menu	File menu
Dialogue	Speech menu	Dialogue	Alert
File menu	Document 1	Alert	Document 2
Document 2	Dialogue	File menu	Speech menu
Speech menu	Interface menu	Edit menu	Edit menu
Alert	Edit menu	Document 1	Interface menu

I	J
Alert	Dialogue
File menu	Document 2
Speech menu	Interface menu
Dialogue	Document 1
Document 2	Alert
File menu	Speech menu
Interface menu	Edit menu
Speech menu	Dialogue
Dialogue	File menu

"I will ask you to do a similar exercise again later on in this session. That time it will be monitored. "

... Reset the screen

"I am going to repeat the exercise we did earlier, where I asked you to find certain windows, but this time I will monitor you. I'll just remind you of the procedure. Remember none of the windows is active at the moment. I will ask you to locate one window. Tell me when you think you have found it - you can check whether you have found the right one, by pressing the mouse and getting speech if you want if you're pretty sure you have found it then you need not, that's up to you. Do not activate (double-click) the window. As soon as you say you have found then one I want I'll give you the name of another one to find in the same way. I will again ask you to find a number of windows in this way, but this will be in a different order from last time. I will not give you any help in locating the windows this time."

- find the windows in one of the lists

0.2 Location of small objects

This exercise should be presented for the first time during the first session after the subject has completed exercise 17.

Locate a number of (almost) randomly chosen objects

- activate a document

"I am going to ask you to find a number of objects within a document window, in a similar way to the exercise in which you located a number of windows. Document n's window is already active. I will ask you to locate a control object in that window. Tell me when you think you have found it - you can check whether you have found the right one, by pressing the mouse and getting speech *if you want* if you're pretty sure you have found it then you need not, that's up to you. Do *not* activate (double-click) the control. As soon as you say have found then one I want I'll give you the name of another one to find in the same way. I will ask you to find a number of control objects in this way, in a random order. I will give you help to locate objects if you want it - and the tactile diagram is available."

- find the objects in one of the lists.

A

Level up
 Current level
 Jump forward
 Step backward
 Say context
 Level down
 Step forward
 Jump backward
 Step backward

B

Level down
 Say context
 Level up
 Jump forward
 Current level
 Step backward
 Jump backward
 Step forward
 Current level

C

Jump forward
 Level down
 Step backward
 Step forward
 Say context
 Jump backward
 Step backward
 Level up
 Current level

D

Say context
 Level up
 Jump backward
 Current level
 Step forward
 Say context
 Level down
 Jump forward
 Step backward

E

Say context
 Level up
 Jump forward
 Level down
 Step backward
 Current level
 Jump backward
 Step backward
 Step forward

E

Current level
 Level down
 Say context
 Step backward
 Jump forward
 Level up
 Step forward
 Current level
 Jump backward

G

Say context
 Jump forward
 Step backward
 Jump backward
 Step forward
 Level down
 Current level
 Level up
 Step backward

H

Step forward
 Say context
 Jump backward
 Say context
 Current level
 Level up
 Jump forward
 Step backward
 Level down

I	↓
Jump forward	Jump backward
Current level	Say context
Jump backward	Level down
Say context	Jump forward
Step backward	Current level
Level down	Step forward
Jump backward	Level up
Level up	Step backward
Step Forward	Say context

I will ask you to repeat that exercise later in this session with monitoring on.

...Locate a number of (almost) randomly chosen objects

- activate document 1

"I am going to repeat the exercise you did earlier when I asked you to find a number of objects within a document window. Let me just remind you of the procedure. Document 1's window is already active. I will ask you to locate a control object in that window. Tell me when you think you have found it - you can check whether you have found the right one by pressing the mouse and getting speech if you want if you're pretty sure you have found it then you need not. that's up to you. Do not activate (double-click) the control. As soon as you say have found then one I want I'll give you the name of another one to find in the same way. I will ask you to find a number of control objects in this way, but in a different order from last time. I will not give you any help in locating the objects, but you can use the tactile diagram if you want it."

- find objects from one of the lists

1. Introduction to the project

Objectives: To familiarize the subjects with the objectives of the

project and its evaluation, so that they can see the point of it all. This will hopefully aid their motivation. This is presented in the form of a script so that I can be sure that I have covered all the important points.

"A lot of modern computer systems incorporate a new form of interaction. As well as the keyboard they have a device with which you can point at objects on the screen, and a lot of commands to the system are communicated using the pointer: you point at an object on the computer screen and then press a button. Obviously this is a very visual form of interaction and the objective of this project is to see whether this form of computer can be made accessible to people who cannot see the screen.

The way I have tried to do this is to replace the things you would normally see with things you can hear.

That is, you hear beeps and synthetic speech. To test the ideas I have implemented a word processor, but the idea of the project is to see whether the principles could be applied to *any* such program, not just word processors.

I am using your help - and that of a number of other people - to evaluate this program. I need to see how easy it is to use. What I want you to do is to complete a number of exercises, to learn how to use the system. The important objective of the exercises is for you to learn how to use the program. So, take your time and feel free to ask questions. Remember it is the *program* I am assessing, not you. If you find anything difficult I want to know about that - because it means there is room for improvement of the program.

As part of the program assessment, some of the exercises will be monitored by the computer. That is, it will keep a record of everything you do on a file, which I can take away and analyze later. I will tell you before you start an exercise if it is going to be monitored. (In fact you'll know about it because when I set the computer to do monitoring it will say "Trace on"). Again, the monitoring is not an examination of your learning but of the performance of the program. Since timing is important, I would ask you *not* to ask questions while you are doing exercises which are being monitored, but to do those as well as you can but without help. Also, at the end of the project I will go through a questionnaire with you on tape, asking you about your views of the system, and there will also be a taped interview so that you can give your opinions on it. At the end of each session I will also ask you for your impressions. I'll keep these and then remind you

of them during the interview.

This is the main part of my research for a PhD. It will all be written up in my thesis. I will not use your name in anything I write - so you can say what you like! Whether the project goes any further will depend on how successful it is."

2. Introduction to the hardware

Objectives: To familiarize the subject with the computer. Although most of the subjects will have used computers before they will not have used a Macintosh - or anything like it probably. This will help them to relax and feel more comfortable with the task. It is most important to introduce them to the mouse.

Allow the subject to look at and feel the computer:

- the keyboard (the home keys);
- the screen/processor box;
- the internal disc drive;
- the external disc drive;
- a floppy disc;
- the mouse - shape, connection, button.

Do this *without* any program loaded - so that anything the subject does (pressing the mouse button etc.) will have no effect. Explain the way that the mouse is connected to the cursor on the screen, which moves in step with the mouse.

- warn that the discs may start to whirr occasionally, sometimes unpredictably

3. Introduction to the audible screen

Objectives: To introduce the subject to the concept of audible objects - the screen and the grid of audible windows.

Load the program.

Move the mouse around and hear the tones generated

- buzz marks the limits of the audible screen
- 8 different tones, corresponding to eight auditory *windows*, arranged in a 4x2 grid
- tone marks the mouse *crossing the boundary* of two windows
- tones increase in pitch to the right and downwards
- present tactile diagram of the screen

Find each of the eight windows.

4. Introduction to spoken help

Objectives: To introduce the subject to the protocol of pressing the button once to get spoken help. To familiarize them with the names of each of the auditory windows. To let them hear the synthetic speech.

Press the mouse button once will produce speech

- each window has a different role
- the windows:

Empty document (two of them)

Speech menu

Interface menu

Edit menu

File menu

Alert

Dialogue

Pressing the button while the mouse is off the screen will also produce speech.

Find each of the windows and to hear the name of each one.

5. Introduction to the activation concept

Objectives: To teach the subject how to perform a double-click. To introduce them to the concept of activation of windows.

Explain activation concept

- caused by a double-click

Activate the File menu window

- different tones are now generated within that window (do not yet explain those tones)
- double beep of window's note on activation - hence three tones (see below)

Move out of active window

- short buzz is heard
- double beep when on return to the active one
- third beep - the tone for the object within the active window at which you are currently pointing

At any time precisely *one* window is active

- inactive windows are still accessible and can be activated in the usual way - making the previously active one inactive

6. Introduction to a menu

Objectives: To introduce the menu concept. Specifically, introduce the File menu and the New command.

Menu is the principal means for communicating commands

- related commands are grouped under a heading in different menus

Activate the File menu

- all the entries therein are arranged vertically
- 4 of them
- top-most entry has same note as the window, again increase downwards
- show tactile diagram
- locate each of them by tone

- single-click protocol works within the active window

Entries within the File menu are:

New

Open

Close

Quit

- locate each of these
- say briefly what the effect of each entry is

Four windows: Speech, Interface, Edit and File are all menus

- diagrams of these are available
- activate each of them and find the names of each of the entries therein

Reset the active window.

Repeat the above task with monitoring on. (This is to be done twice so that the second time the subject will be familiar with the words which will be spoken).

Double-click protocol applies to *execute* a menu-entry command

- find New entry in the File menu and double-click it

7. Text entry

Objectives: To introduce the document window and the manner of entering text into a document.

New command has opened a new document file

- in the top left-hand window - formerly "empty document", now "unnamed file"
- activate that window
- type text into it - any, freely chosen text, form is irrelevant

[Exercise 26 might be given here]

8. Introduction to dialogues

Objectives: To introduce the dialogue window and to allow the user to save their typing. Introduce the Save file dialogue.

Close the current document

- system will want to know whether you want to save the file's contents on the disc
- via the Dialogue window
- dialogues are used by all commands which need more information before they can be carried out
- execute the Close command
- dialogue 'announced'
- Dialogue window is now *filled*
- since the program can progress no further until it knows whether you want the file saved, you cannot activate (double-click) any other windows at this point, you can still single-click them to hear what they are, though - so-called *modal* dialogue - *non-modal* dialogues later

Use the Dialogue window

- activate it
- tactile diagram available
- usual tone pattern - i.e. increasing pitch downwards and to right
- also usual click protocol

- explore it, look for, and explain:

Create a new file?

Yes

No

Cancel

- execute No

Since the program can do nothing more until the Close command has been completed, no other window can be activated at the moment.

9. Opening a file

Objectives: To allow the subject to use an existing file in the subsequent exercises. Introduce the Open command and the File Open dialogue.

Execute the Open command

- dialogue opens

Explore the dialogue

"One control contains a file name. In fact there is a list of file names, but only one is shown at a time.

To get at the others you move them, one at a time into the control. It is as if they are in a list on a piece of paper, each name on a line on its own in alphabetical order. You can move a different name into the control by *scrolling* the names up and down - like moving the piece of paper up and down.

To scroll you use two of the other controls: Up button and Down button."

- to hear current name, (single-) click in that control
- to open current name, either double-click name control or Yes
- Cancel will do just that
- scroll through all the names
- select one

10. Introduction to the text selection concept

Objectives: Sections of text which are selected form the basis of text editing in this system. It is necessary that the subject understands this concept before they can do any editing. To introduce the following document window controls: Current Level, Level up, Level down, Step forward, Step backward.

"In this type of system, all editing operations operate on a portion of the text within a document which is said to be *selected*. In a visual program, the currently selected portion of the text is displayed in a highlighted manner, different from how the rest of the text is displayed, so that the user can see which part of the text it is. A sighted user specifies which portion of the text is to be selected by pointing directly at it on the screen. This is too difficult to implement for someone who cannot see the text. Therefore, this system uses a slightly different approach. Selections can be made of:

the whole document,
 a paragraph,
 a sentence,
 a word,
 a character,
 the point between two characters.

The current selection of a document is communicated by being spoken - and this will occur every time the selection is altered - or if the user requests it. Each document has a selection *level* associated with it. All selection operations occur at that level. For instance if the current level is **sentence** then you can move the selection forward one sentence, backwards on sentence etc. Selection is altered by way of control objects in the document window."

Activate the open file window

- explore document window - tactile diagram
- some of the controls will not be explained until later.

- available levels: inter-character to whole text.

Initially selection is at end of the document and at inter-character level

- inter-character level is used to insert within existing words to be demonstrated later

Explore Current level control

- single click causes speaking of the current level
- double-click will say current selection

Explore Level up and Level down controls

- adjust the selection level to whole text level - click the Current level control to check on progress
- new selections are generally extended to the *right* of of the current selection start - which is why nothing is said up to whole text level
- listen to the document

Read document a paragraph at a time

Move down to sentence level and to move around

- words are spoken more slowly, but the speech can be interrupted at any word boundary
- experiment with this: select a sentence, interrupt its being spoken and then request the current selection to be spoken
- speech can be interrupted at any level, but will only stop between units - at then end of the current word at sentence level, the current sentence at paragraph level etc.

Move to word level and move around

Experiment at character level

Experiment at will

11. Introduction to editing

Objectives: To introduce the editing operations. To introduce the Edit menu and the Cut command.

Activate Edit menu

- explore the entries therein
- tactile diagram
- some of the entries will be explained later

Make a selection

- move to **sentence level** - listen to it
- move down to **word level** and to one of the words therein (preferably not the first word)
- emphasize that that word is *selected*, and to remember its context

Cut it

- **activate the Edit menu**
- **locate the Cut command** - explain the effect of Cut and reason for its name
- **execute Cut**

Check the effect

- **re-activate the document window**
- **move to sentence level** - listen to new sentence

12. Cut and paste

Objectives: To reinforce earlier lessons. To introduce Paste.

Introduce the idea of pasting

- the last section Cut is held in a *buffer*
- it can be **Pasted** back
- is inserted at *current selection* - can be back where it came from or elsewhere
- *replaces current selection* - if that is a point it will be inserted, if a section it will replace that
- note that once you Cut current selection becomes a *point* (it closes the gap) - but current level unaffected

Cut a word from a document and Paste it back in place

- Open a document
- Select a word within it
- Cut it
- Listen to Context
- Paste it back in place
- Listen again to Context

Cut a word from a document and Paste it back elsewhere

- Select another word
- Cut it
- Listen to Context
- Select another word
- Paste
- Listen to Context
- point out that second word has been *replaced*

Repeat, but Copy instead of Cut

Experiment

Close, but do not save the file

13. Introduction to proof-reading

Objectives: To get further practice at entering text. To introduce working at character level.

Correct a typing error in a given document - *Needleman* (error is the word 'bought' where it should be 'brought').

- Open a file
- move to paragraph level and listen to it - try to hear any errors
- move to word locate the offending word
- move down to character level
- locate the appropriate point
- type in the extra letter
- listen to the word again

14. Word-rearrangement exercise

Objectives: To get further practice of Cutting and Pasting.

Correct a jumbled-up sentence in a document - *week*

- Open the document
- move up to sentence level
- step back, to hear the sentence
- work out the correct sentence (with help as necessary)
- move down to word level and Cut and Paste it as necessary
- check the new sentence
- Close, but do not save it

15. Text entry practice

Objectives: To get further practice at entering text . To introduce the

Get filename dialogue.

Create file, and save its contents

- open a New file
- type dictated text into it
- Close it
- in response to Save file dialogue enter Yes
- Get filename dialogue opens
- explore it
- choose and enter a name
- double-click Yes

16. Practice at text insertion

Objectives: To learn about the manner of inserting text within a document.

Replace text in an existing document. *Westland*

- Open the document
- listen to it
- select the name
- replace it with the subject's name
- emphasize that whatever is typed *replaces* the current selection
- point out that the level automatically becomes point level
- check

17. Introduction to document jump controls

Objectives: To introduce the document controls: jump forward and jump backward.

Experiment with the jump controls

- go to whole text level to hear the whole document
- go to sentence level
- explain that jump controls allow you to move around in big chunks
- they do *not* affect the current level, but allow you to move in units one above the current level i.e. if you're at sentence level you move a paragraph at a time, at word level you move a sentence at a time etc.
- experiment with jumping around the document
- go to word level and experiment again

18. Introduction to the speech menu

Objectives: To introduce the Speech menu and the Say typing entry. Also introduce the concept of ticked menu entries.

- activate the Speech menu
- explore the entries in it

Introduce *ticked* menu entries

- point out that two of the entries are said to be *ticked*
- explain the origin of the term
- explain that an entry which is ticked is in effect at that time, e.g. Say typing being ticked means that whatever is typed into a document will be spoken, Announce dialogues means that a dialogue's name will be spoken when it is opened
- double-click Say typing
- listen to it again, and hear that it is no longer ticked
- activate the document window and type into it - note that nothing is spoken
- point out that selections *are* still spoken, Say typing does not affect this, double-click Current level to show this

19. Introduction to character-level interaction

Objectives: To introduce methods of working at character level in a document. To introduce the Say characters , proof read and announce dialogues entries.

Remind the subject of the use of character level in a document window

- that all operations work on characters - i.e. characters are spoken, stepping forward and backward occur one character at a time

Introduce character speech

- find the **say characters** entry in the speech menu
- activate it
- listen to it and check that it is now *ticked*
- re-activate the document window
- point out that the level has not changed - check this
- modify the selection so that it is spoken - character-by-character
- point out that punctuation is *not* spoken
- alter the current level and hear that the selection is still spoken character-by-character

Typing is also now spoken character-by-character

- ensure that **Say typing** is ticked
- type into the document

Introduce proof-reading

- execute **Proof read**
- check that it is ticked
- listen to current selection again (with **Say characters** still ticked)
- point out that punctuation is spoken
- ensure that some capital letters are heard
- untick **Say characters** and ensure that level is not character level
- listen to the current selection

20. Introduction to the interface menu

Objectives: To introduce the Interface menu and the entries hidden and auditory.

- activate the interface menu
- explore it - tactile diagram available

Explain that the word processor can be used by a sighted person, not using sounds, but using their visual counterparts, like any visual program

- executing the auditory entry will make the program switch to the visual version

Explain that there is an intermediate mode, in which the contents of documents are shown on the screen

- this is so that sighted people can see what is going on
- it is initiated by executing the hidden entry

21. Introduction to string finding

Objectives: To introduce the facility for searching for strings in a document. This involves introducing the find entry in the edit menu and the get target string dialogue (non-modal).

Find a string within a document

- move the selection in a document to the beginning
- double-click Find
- the get target string dialogue opens
- explore the dialogue
- 2 objects: target string and find
- single-click target will cause it to be spoken, if it buzzes there's no target
- double-click target will clear the target
- you can type in a string as soon as the dialogue opens - like a document

- double-click find
- go to the document window and listen to the current selection
- level is unaltered
- point out that find will only match literally - no folding of cases
- point out that the dialogue stays there until another is opened - with the same target
- double-clicking find will search for the *next* occurrence
- if no occurrence found, there's a buzz and the selection is unaltered

22. Final entry in the file menu - quit

Objectives: *To introduce the final menu entry: quit.*

If the subject has not already done this in the course of the exercises

- double-click quit
- point out that if there are any documents open, the program must know whether they are to be saved on disc before the program can exit, so a save file dialogue is opened for each open document
- program terminates

23. Introduction to alerts

Objectives: *To explain the alert window*

When an error occurs the program buzzes. If the user wanted more information about the nature of the error they would do that through the alert window, which would work in a way similar to the dialogues. However, it has not yet been implemented, through lack of time.

24. Introduction to the scroll bar

Objectives: *To introduce the operation of the scroll bar.*

- open a document, if necessary
- find the scroll bar and the thumb bar

- point out that the thumb bar moves (horizontally) within the scroll bar
- its position within the scroll bar reflects the current position of the selection within the document, i.e. if the selection is at the end of the document, the thumb bar will be at the right-hand edge, if the selection is at the beginning, the thumb bar will be at the left edge and so-on in the middle
- the thumb bar not only shows the current position of the selection, but can also control it
- it is operated by double-clicking the thumb bar, but on the second click the button is held down and then it follows the horizontal motions of the mouse, until the button is released again
- point out that this is included as a direct analogue of a visual control

25. Introduction to the operation of multiple windows

Objectives: To ensure that the subjects are aware that it is possible to have two documents open at one time and that operations can be carried out across them.

Cut from one document and paste into another

- ensure that two documents are open
- select a sentence from one
- cut it
- activate the other document
- select a suitable point within it
- paste
- listen to the modified document

26. Exploring the keyboard

Objectives: To familiarize the subject with the layout of the non-alphanumeric keys on the keyboard to get further practice in operating at character level and to familiarize the subject with the pronunciation of characters.

- create a new document
- tick say characters and proof read
- press each of the typing keys in turn
- hold shift key and repeat
- put the caps lock on and repeat

27. Introduction to cancelling dialogues

Objectives: To familiarize the subject with the facility to cancel a dialogue's action.

The subject will know of the cancel entries in some of the dialogues, they should get practice at using them.

- double-click open
- get filename dialogue opens
- double-click cancel
- dialogues closes, no document is opened, system unchanged

28. Correction practice

Objectives: To give the subjects practice at correcting errors in a document, in preparation for later, realistic exercises.

Open a given file (*crime*) with 16 names in it, each of which does not start with a capital letter. (*Fixing the number of errors will help the subject to locate them*).

Crime:

In 1921, Thomas (The Butcher) Covello and Ciro (The Tailor) Santucci attempted to organize disparate ethnic groups of the underworld and thus take over Chicago. This was foiled when Albert (The Logical Positivist) Corillo assassinated Kid Lipsky by locking him in a closet and sucking all the air out through a straw.

- switch on proof-reading
- replace the initial letters, as appropriate
- close and save the file

29. Spelling correction practice

Objectives: To give the subjects practice at correcting spelling errors in a document, in further preparation for later, realistic exercises.

Open the file *errors* with 8 spelling errors in it (*Fixing the number of errors will help the subject to locate them*).

- correct them
- close and save the file

errors:

It is always useful, sooner or later, to remember the real world, because the reality of that world has a habit of asserting itself. In the real world, for example, Sir Raymond Lygo and Sir Austin Pearce (though choked and puce) have to go on doing business with Mr Leon Brittan, or at least the Government to which he currently belongs. What was thus on Wednesday night a gulf of testimony beyond understanding becomes on Friday a mutually agreed "misunderstanding."

It is always useful, sooner or later, to rememember the real world, because hte reality of that wprld has a habit of asserting itsef. In the real world, for example, Sir Raymond Lygo and Sir Austin Pearce (though choked and puce) have to go on doing bussiness with Mr Leon Brittan, or at least the Government to which he currently belong. Wat was thus on Wednesday night a gulf of testimony beyond understanding becomes on Friday a mutually agreed "misunderstanding."

30. Text-entry practice - without checking

Objectives: To practise operation of the word processor - entering text. To provide an exercise for monitoring and comparison with

text-entry with checking.

Enter text into a new file and save it

- double-click new

- activate the document

- point out that on this word processor, paragraphs are marked by two carriage-returns

- explain to the subject that they are going to type in some dictated text

- tell them that punctuation will be given, and spellings if they want

- if they make an error which can be corrected by backspacing they may do so, however, there will be no proof-reading and correction

- point out that this is *not* a test of their typing skill, and speed is unimportant

- when monitoring is on, minimal help will be given

- they can type with speech on or off, as desired and in say character mode or not, as desired

- type in the following dictated text:

A

As usual, at five o'clock that morning reveille was sounded by the blows of a hammer on a length of rail hanging up near the staff quarters. The intermittent sound barely penetrated the window-panes on which the frost lay two fingers thick, and they ended almost as soon as they'd begun. It was cold outside, and the camp-guard was reluctant to go on beating out the reveille for long.

The clanging ceased, but everything outside looked like the middle of the night when Ivan Denisovich Shukhov got up to go to the bucket. It was pitch dark except for the yellow light cast on the window by three lamps - two in the outer zone, one inside the camp itself.

And no one came to unbolt the barrack-hut door; there was no sound of the barrack-orderlies pushing a pole into place to lift the barrel of nightsoil and carry it out.

(Opening paragraphs of *One Day in the Life of Ivan Denisovich*, by Alexander Solzhenitsyn)

B

Mr Speaker rose and surveyed the Commons. He tugged at his long black silk gown, then nervously tweaked the full-bottomed wig that covered his balding head. The House had almost got out of control during a particularly rowdy session of Prime Minister's questions, and he was delighted to see the clock reach three-thirty. Time to pass on to the next business of the day.

He stood shifting from foot to foot waiting for the 500-odd members present to settle down before he intoned solemnly, "Members desiring to take the oath." The packed assembly switched its gaze from the speaker to the far end of the Chamber, like a crowd watching a tennis match. There, standing at the bar of the Commons was the victor of the first by-election since the Labour party had taken office some two months before.

(From *First Among Equals*, by Jeffrey Archer)

31. Text-entry practice - with checking

Objectives: To practise operation of the word processor - entering text. To provide an exercise for monitoring and comparison with text-entry without checking.

Enter text into a new file and save it

- double-click new

- activate the document

- explain to the subject that they are going to type in some dictated text

- tell them that punctuation will be given, and spellings if they want

- if they make an error which can be corrected by backspacing they may do so

- this time there will be proof-reading and correction

- re-iterate that this is *not* a test of their typing skill, and speed is unimportant

- when monitoring is on, no other help will be given
- they can type with speech on or off, as desired and in say character mode or not, as desired
- type in dictated text
- check and correct it as necessary.

32 The ultimate test

Objectives: To give a realistic, difficult test.

Cut and paste between documents

- open two existing documents
- listen to one of them
- cut a specified sentence from it
- activate the other document
- listen to it
- locate the appropriate point
- paste the sentence in
- check, as necessary
- quit the editor and save both documents

33. Alternative final exercise

Objectives: To give a realistic, more difficult test. To demonstrate the power of Soundtrack - cutting and pasting between two documents. To practise locating and correcting a spelling error. To demonstrate the use of the quit command.

Copy a paragraph across two documents

- open the file letter
- open the file insert
- activate letter
- listen to it

- activate *insert*
- listen to the two paragraph of *insert*
- select one of the paragraphs
- copy it
- activate *letter*
- locate the insertion point
- paste

Insert a name

- locate and select *X* in *Dear X*
- cut
- type in the replacement

Correct spelling error

- locate the word *tank* in *letter* (a mis-spelling of *thank*)
- adjust the selection to the point between the *t* and the *a*
- type in an *h*
- check

Quit

- locate and execute *quit* in the *file* menu
- respond to the *save file* dialogues
- save *letter*
- do not save *insert*

Appendix C

The Structured Interview Questionnaire

Auditory Wimp Evaluation

Questionnaire

Date / /

Subject number

This is the second part of the evaluation of the 'auditory Wimp' word processor program. Remember, the aim of the evaluation is to see how easy it is to use. I have monitored some of the exercises you did to measure how easy you found it. Now I want to gather your opinions about the program. Please remember that I want to know about good and bad aspects of the program. If I am aware of problems I will be able to try to correct them later.

The third part of the evaluation will be an interview, conducted by a colleague of mine. That will give you the opportunity to express your own opinions without the constraints imposed by the questions here.

I will remind you that your name will not be used in the reporting of the results.

1. Background

1.1 Degree of sight: normal sight / partially sighted / totally blind

1.2 If blind, is this since birth ? yes / no

1.2.1 If no, when were you first registered as disabled ?

1.3 Familiarity with equipment

typewriter none / untrained / trained / regular user times /

word processor none / untrained / trained / regular user times /

Which one ?

other computers none / untrained / trained / regular user times /

Wimp programs none / untrained / trained / regular user times /
(sighted subjects)

Which program(s) ?

1.4 Musical ability

none / listening / some training / plays instrument

1.5 Degree of familiarity with synthetic speech

none / occasional / regular: times /

2. Training

2.1 What is the minimum level of experience you think someone would need to learn to use this program:

- a) regular experience of using a computer or word processor
- b) some experience in using a word processor or computer
- c) training in touch-typing, but without use of a computer or word processor
- d) no previous experience necessary at all

2.2 If you had to teach someone in the above category to use the program, what do you think is the aspect of it they would find most difficult to learn ?

2.3 What additional aids would you use to train another user (e.g. Braille notes, audio tapes etc. - not modifications to the program or equipment) ?

2.4 Would your approach to teaching another user be any different from the way you have been taught, and if so why ?

2.3 How long do you think it would take the person to learn enough about using the program such that they could be left on their own to practise ?

3. Auditory objects

3.1 What method did you use to remember the location of objects ?

3.2 Is there any way you think the layout of objects could be made easier to remember ?

3.3 If you were teaching someone else to use the program, would you suggest they use any particular method to help them remember the location of objects ?

3.4 Were any of the objects too small ?

3.5 Tone patterns

3.5.1 Can you describe the pattern of tones ?

3.5.2 Do you think the pattern of tones helped you to locate objects ?

3.5.3 Is there a different pattern which you think might be more helpful ?

4. Overall view of the program

4.1 Were there aspects of this program which you liked ?

4.2 Were there aspects of this program which you did not like ?

4.3 What was the most difficult aspect of using this program ?

4.4 Do you think you could use this program now without help?

4.5 What level of encouragement would you need to use this word processor in a job ?

- a) None. You would be happy to work with it.
- b) You would want to test other talking word processors for comparison before you would choose between this one and the alternatives.
- c) You would use it in preference to a typewriter.
- d) You would use it if it was a compulsory part of the job.
- e) You would not be prepared to use it in a job.

5. Added features

5.1 What was your opinion of the experimental added features ?

5.1.1 Diagonal movement alarm

5.1.2 Horizontal and vertical movement feedback

5.1.3 Others

5.2 What modifications or additions would you suggest making to this program to make it easier to use ?

6. Other comments

Appendix D

Transcript of a sample session of using Soundtrack

This appendix is a textual and graphical representation of a typical session of using Soundtrack. This session is *not* one of the evaluation exercises. The bottom line is a spoken commentary by the user. Above that, appears any spoken output from the program. Other auditory output from the program is represented in a simple musical notation. Tones are represented by bars, where the vertical location of a bar indicates its musical pitch, and its length represents its duration. Clicks of the mouse button are represented by black circles: one for a single-click and two overlapping circles mean a double-click. An audio cassette of this session is available.

This is a short demonstration of the Soundtrack word processor. What I am going to do is to create a document, type into it and do some editing of the text.

The program has been started up and is waiting for me to do something. If I just move the mouse around different tones are sounded, corresponding to the different windows.




First I need to create the file. That means executing the New command in the File menu.

Edit menu

I am not sure where the mouse is at the moment, so I'll press the mouse button to find out.

It said "Edit menu". So that's the wrong window.






Bottom

The buzz means that I've moved off the screen. I suspect I'm off the bottom, but I can check that by clicking the mouse.

Yes, "bottom".



File menu

So, back on the screen again.

I think that's the File menu, but I'll check.

"File menu", yes. Now I need to activate that menu. I do that by





clicking the mouse twice quickly. That's a double-click



Those three beeps mean that the window, the File menu, is now activated.



The first two beeps were just the window's tone repeated. The window is now split into separate entries, each with its own tone. The third beep was the tone for whichever

of those entries the mouse is currently in. If I move it around within that window I can hear the tones of all the entries. If I should move out of the active window I'll

██████████

██████

hear a short buzz, followed by the tone of the window I move into...

...like that. I'll move back into the File menu, the active window, and I'll hear the three



tones again, which tell me I'm entering the active window.

There, the objects within the active window work in the same way as the windows



themselves. Each has a tone...

...and I can press the mouse to get some speech. I'll find each of the entries in this menu.

Open

That's 'open'...





Close

...close...

Quit

and quit, and there's one more at the top...

New

...new. And that's the one I want. If I



double-click on there, a new file will be created.



Two tones were sounded there. The first tells me the object was double-clicked, the second is



the tone of the top left-hand window, which is where the new document is effectively 'displayed'. I'm going to move to that window now.



The file menu window is still active, so I got the short buzz again as I moved out of it. It'll automatically become de-activated when I activate another window. I'm now in the

Unnamed document

window that I think the document's displayed in, but I'll just click to check.

"Unnamed document". That's it. That is the new document



I have created. It has no name yet. Before I can do anything with it I must activate the window.

Right, now I can type into this new, empty



document. It's set up automatically so that my typing will be spoken word-by-word. That means nothing is spoken until I've finished a word - by typing a space or a

Mary had a little lamb. Its fleece was white as snow.

punctuation mark. I'll just type a couple of sentences from a nursery rhyme.

I can manipulate that

text now. First I think I'll listen to it all again. I can specify which section of text I want to work on by selecting it. I'll know which part is selected because it will be

spoken. So I'll start by selecting the whole thing I just typed, so that I can hear it all. Because I have just been typing, the selection will be a point just beyond the end of

the text. The document has a level associated with it. That will currently be point level, where a point can be in between any two characters. I can check the current level



Point level

by finding the object in the top left-hand corner of the window and clicking on it.

"Point level". I can adjust the level by moving the mouse down





Level up

one object

This is the Level Up control. I can check that by clicking.

Right, now if I double-click the level goes up.

That will now be



character level. I can check that by going back up to the Current Level control and clicking.

Character level

I'll keep moving the level up until I reach



██████████ ██████████

██████████

paragraph level, because the two sentences I typed are a paragraph, although a short one.

Word level

That's word level



██████████ ██████████

██████████

██████████ ██████████

██████████

Sentence level

Sentence level

Paragraph level

Paragraph level



Right, but the selection is still at the end of the text. To get the paragraph selected I must move the selection back. I can do that with the Step Backward control. That is



Step backward

the third one down on the left-hand side. I move to the second down...

...and to the third.

and click to check



Mary had a little lamb. Its fleece was white as snow.

"Step backward". Now I'll double-click, which will select the whole paragraph and cause it to be spoken.



Now I'll do some editing. I think I'll delete the word 'little'. The whole paragraph is selected at the moment. If I move the level down, just the first sentence will be



Level down

selected. I can do that with the Level Down control, which is to the right of the Level Up.

"Level down". So if I double-click



Mary had a little lamb

on there the level will become sentence, and just the first sentence will be selected and spoken.

I will move the level down again, to





Mary

word level.

"Mary", that's the first word of that sentence. I want to move the selection to "little". I do that with the Step Forward control. That is to



Step forward

the right of the Step Backward, or in other words, it's one down from where the mouse is at the moment.

I can double-click there to move forward



had

a little

one word

"had", and so on until I get to "little".

Right, "little" is now selected and I can delete it by cutting it. That's an editing command



Edit menu



and it's in the Edit menu. So, I'll move the mouse out of this Document window and find the Edit menu.

"Edit menu".



Copy



I activate that.



And now I need to find Cut in that menu

That's Copy.



Cut

"Cut", so I double-click that.



Right, "little" should have been deleted from the document. We can go back and check. I activated the Edit menu, so





Unnamed document

it's still active, and the document is deactivated. So, I'll move back to the document.

There it is.



:Unnamed document". So, I'll re-activate it.



When I moved out of this document, the level was Word, and "little" was selected.

Now "little" has been cut, but the level remains the same. There is a way of hearing text surrounding the current selection, without altering the selection.



Say context

I need to find the Say context control, which is in the top right-hand corner of the window.

"Say context", that's it.



Mary had a lamb.

Now if I double-click I'll hear the sentence which contained the word I just deleted.

So, Mary's lamb is no longer a little one.



File menu

Now I'll close the document. Close is a file command and is in the File menu, so I'll try to find that.



"File menu". Activate it.



Now I must find Close.



Open

"Open"



Close

Right, that's Close.

When I close a file, the program will need to know what to do with its contents, whether to save them on disc or not. Extra information like that is always obtained



Save file dialogue

via a 'dialogue'. So, if I double-click now, a dialogue will be opened, and the fact that it has been will be announced



"Save file dialogue". That's been opened in the Dialogue window, which is in the top right-hand corner of the screen. I cannot do anything else until I've dealt with



Save file dialogue

the dialogue. So, I move to it.

That's the Save file dialogue, and it must be activated.



Create a new file?

If I look at the top left-hand object



it's asking me if I want to create a new file. There are three options, "Yes", "No" and "Cancel".



Cancel

No

Yes

I'll just find the three object corresponding to these choices:

"Cancel"

"No"

"Yes". Now I don't think



No

I will save it, after all the rhyme is incorrect. So I need to find the No object again.

"No", and I double-click that.



Right, that's closed the file, but without writing it onto the disc. So now I'll close down the program. I do that with the Quit command, which is back in the File menu.



File menu

So, back to that menu, and activate it

"File menu", activate it,

and find Quit





Close

Quit

that's Close

"Quit" and double-click on there. And that's closed the whole thing down.

Appendix E

Useful Addresses

Below is a number of addresses of institutions whose work is relevant to this project.

Addresses

Apple Computer Inc.,
20525 Mariani Avenue, Cupertino, California, USA, 95014.
Manufacturer of the Macintosh.

Association of Visually Handicapped Office Workers,
8 Basterfield House, Golden Lane, London EC1Y 0TN.
An organisation *of* visually disabled people who work in office jobs, usually with the aid of Information Technology.

Berkeley System Design,
1708 Shattuck Avenue, Berkeley, California, USA.
Manufacturer of the *InLarge* screen magnifier for the Apple Macintosh.

Bit 32 Ltd.
32 North John Street, Liverpool, L2 9QJ
Developer of software for Macintosh users with physical disabilities, in particular the *Head Start Workstation*.

Expertelligence
559 San Ysidro, Santa Barbara, California, USA, 93108.
Marketer of *Interface Builder*, object-oriented visual programming language for the Macintosh.

Frank Audiodata
PO Box 1161, D6839 Oberhausen, West Germany.
Manufacturer of the Frank Audiodata.

Living Videotext Inc.,
2432 Charleston Road, Mountainview, California, USA 94043.
Manufacturer of the *More* ideas processor.

MacSerious Software
36 Queen Street, Helensburgh, G84 9PU.
Distributers of a range of Macintosh software, including the *Pinball Construction Set*.

Research Centre for the Education of the Visually Handicapped,
Selly Wick House, 59 Selly Wick Road, Selly Oak, Birmingham B29 7JE
Developer and supplier of a wide range of low-cost software for use in the education of visually disabled people.

Royal National College for the Blind,
College Road, Hereford HR1 1EB.
Runs full-time Further Education courses for visually disabled people, including courses in word processing and computer programming.

Royal National Institution for the Blind,
Braille House, 206 Great Portland Street, London.

Principle British charity concerned with the welfare of visually disabled people.
Among other activities, it runs courses on word processing at its commercial college.

Sensory Information Systems
26 England's Lane, London NW3 4TG.

Suppliers of *Vincent Workstation*, other word-processing hardware, including being the UK distributor for the *Frank Audiodata*. UK distributor of *Optacon*. Also supply braille-transcription software.

Smith-Kettlewell Institute of Visual Sciences,
2232 Webster Street, San Francisco, California, USA 94115.

Research institute which concentrates on the development of low-cost, do-it-yourself aids for visually disabled people. Publishes a regular bulletin containing instructions for constructing devices, available in small print, large print, and audio cassette.

Star Microterminals Ltd,
22 Hyde Street, Winchester, Hampshire S023 7DR.
Manufacturers of the *Concept Keyboard* input device.

TeleSensory Inc,
455 North Bernardo Avenue, PO Box 7455, Mountain View, California, USA, 94039.
Manufacturers of the *Optacon* reading device, *Versabrilie* electronic braille computer and *Vert* synthetic speech screen reader.

Western Blind Rehabilitation Center,
Veterans' Administration Medical Center, 3801 Miranda Avenue, Palo Alto, California, USA 94304

A centre of research and evaluation of aids for use by visually disabled people. Currently involved in investigation of tactile output from computers (see Goodrich et al, 1986 and Steele et al 1986).