Alistair Edwards

IEE

Conference Publication No. 258

International Conference on

Speech Input/Output; Techniques and Applications

INTEGRATING SYNTHETIC SPEECH WITH OTHER AUDITORY CUES IN GRAPHICAL COMPUTER PROGRAMS FOR BLIND USERS

Alistair D.N. Edwards

Open University, Walton Hall, Milton Keynes, MK7 6AA, UK

INTRODUCTION

For most computer users interacting with a computer involves using a terminal comprising a keyboard and a video screen. The communication is a very much text-based - words are typed on the keyboard and read from the computer screen. Although most users rely on their sense of sight in this interaction, a significant number of people who have a visual disability are able to use such terminals by supplementing them with special communications devices. These devices work by translating visible text into a form which can be sensed by a blind person. That form may be, for instance, synthetic speech (Vincent (1)) or tactile sensations (as for the Optacon - see Shirley and Schofield (2)).

However, there is currently a major trend in computer design towards replacing such text-based communication with a form of user interface which is much more graphical. So-called WIMPs - window, icon, mouse and pull-down menu systems - were developed within systems such as Xerox's Smalltalk (Goldberg and Robson (3)) and are now more generally available even on quite inexpensive microcomputers such as the Apple Macintosh (Duff (4)). The medium of communication is no longer simple comprehension of written words, but interpretation of visible and spatial properties of graphical objects including icons, windows and menus.

In addition to the screen and the keyboard there is an additional device which is used to *point* at objects on the screen. This commonly takes the form of a *mouse*, a box which can be moved around a flat horizontal surface. As it is moved the motions are detected and mimicked by the movement of a *cursor* on the screen. Most commands are communicated by the user by manipulating the cursor, via the mouse, to point at a particular object on the screen, and then pressing a button on the mouse. Guiding the cursor to the chosen object, is simply a question of coordinating hand movements with the visual input.

It might seem that such a user interface is too intensely visual to be made accessible to a user who cannot see. Even the addition of translating devices seems unlikely to help: how, for instance, can a speech synthesizer 'pronounce' an icon or a tactile sensor indicate that a portion of a window is overlapped by another window? This paper, however, summarizes the design for an interface which is intended to make Wimp-type programs accessible to visually disabled users. The aim of the project is to see whether wimp programs in general can be adapted for use by blind people.

The paper deals with the principles proposed for design of an alternative interface. It then illustrates the application of those principles to a test program - a word processor. The evaluation of the programs is then described. Finally there is a summary of work which might follow on from this project.

DESIGN

The design of the alternative interface is based upon two principles. Firstly all visual information is replaced by auditory information. Secondly, the resultant auditory interface is constrained with respect to the visual one. That is to say that objects which would normally be displayed on a screen are represented by *sounds*. Thus we can refer to the entity with which a blind user interacts as an *auditory screen*, a collection of objects which can only be heard, not seen.

Constraints - the screen layout

In adapting the interface a balance has to be reached between providing direct auditory analogues of the visual interaction and making a usable program. Some aspects of a visual Wimp program cannot be translated exactly in a form which can be used easily. 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. For a sighted user it is relatively easy to recognize when windows are overlapped and to deal with them. However, there is no sensible, simple way of representing overlapping objects as sounds. The simplest approach to such problems is to not allow them to arise - by constraining the design of the auditory interface. Therefore, the first constraint placed on the auditory interface is that the so-called *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. The *role* of any one window never changes - although its contents may do.

An auditory window can be *activated* by the user via the mouse. Activation causes the window to be divided into separate objects with which the user can then interact. The layout of those objects within the window varies between different windows, but is again based on a fixed grid. Through the mouse, the user can cause objects within a window to be *executed* - in a manner similar to the activation of a window. The effect of executing an object depends on its identity. At any time no more than one window can be active. The mouse can be moved out of the active window and into another one which can then be activated. Activating a new window automatically causes the previously active one to be de-activated. This concept of activation of individual windows means that the user can concentrate on one particular aspect of the interaction at any time, so reducing the short-term memory load.

Sounds

Two forms of sound are used in this design to communicate information from the computer to the user: musical tones and synthetic speech. Tones can be used when the information is simple and should be communicated quickly, whereas speech is used when the information is more complex. Thus, the user can hear the position of the mouse *relative* to objects on the screen as tones which are generated when it is moved from one object to another. However, if the user needs to know which object is currently being pointed at - the mouse's *absolute* position - the object's identity can be spoken in synthetic speech. In addition, any information which would be displayed as text in a visual program (such as the contents of

a document file) can be spoken.

Thus, different tones are generated when the mouse is moved as follows:

from one window into another;
from one object to another in the active window;
out of the active window;

TESTING THE DESIGN PRINCIPLES - A CASE STUDY

In order to test the principles outlined above an experiment is being carried out whereby a typical Wimp program - a word processor - has been modified to use an auditory interface. The program has been implemented on an Apple Macintosh microcomputer. It has been written in the Macadvantage (Edwards and Woodman (5)) implementation of UCSD Pascal (Clark and Koehler (6)) and comprises around 5000 lines of source code. The choice of hardware does have some influence on the final product. The Macintosh has several built-in sound generators. Tones are thus easily generated, using its square-wave generator. Also it has free-form generators, and using *smoothtalker* software it is possible to generate synthetic speech. This obivates the need for additional hardware. The mouse on the Macintosh has but one button, which restricts the possible button protocols.

In the particular case of this program, the auditory screen grid consists of eight windows. These are arranged in 4 x 2 grid, as Figure 1. The actual number of auditory windows required for different applications is likely to vary, but it is encouraging that it can be as few as eight. The roles of the eight windows are as follows:

1. four **menu** windows,
2. two **document** windows,
3. one **dialogue** window,
4. one **alert** window.

As the mouse is moved around the auditory screen a tone is generated whenever it crosses the border between any two windows. A simple pattern of tones is used, whereby the pitch of the tone increases as the mouse is moved to the right or downwards. The interval in pitch between the notes for two successive objects is two whole tones. Movement into the active window is signalled by its tone being sounded twice. A third tone is then also generated, corresponding to whichever object within the active window is being pointed at.

Spoken help is requested by pressing (or *clicking*) the mouse button once. For instance, pressing the button while the mouse is in the second window in the top row will cause the program to say "Speech menu". Clicking it twice in quick succession - so-called *double-clicking* - will cause the object currently being pointed at to be activated or executed. Thus, double-clicking in an inactive window will activate it (which is confirmed by that window's tone being sounded twice), and double-clicking an object within an active window will cause the object's tone to be generated - and its corresponding action executed.

Menus

These are direct analogues of the pull-down menus of a visual program Related commands are grouped into one menu with an appropriate heading. The user locates the desired menu window and then activates, it as described above. Thereafter tones are generated within the active menu window corresponding to a number of entries, arranged vertically. The user executes a command by locating its entry in the menu and double-clicking it.

The contents of the four menu windows are shown in Figure 2.

Documents

Text files to be edited are 'displayed' in document windows. The limitation of including just two such windows represents a further constraint of the interface. In a visual program editing operations are applied to a portion of the text which is said to be *selected*. The section to be selected is chosen by the user marking its extremities directly on the screen using the mouse, and is displayed in a highlighted manner. That is to say that making a selection is a matter of applying hand-eye coordination skills, and the selection can be verified by reading from the screen.

This form of interaction is too difficult to translate into a direct auditory analogue. Therefore, the auditory version of a document window is rather different. It still uses the concept of an area of selection, but instead of interacting directly with the text, the user can select text in fixed, grammatical units. Those units are:

the whole document

paragraphs

sentences

words

characters

the point between two characters.

Every document has a current *level* associated with it - corresponding to one of the above six units. In other words, if the current level in a document is *sentence* level, then all operations will act on sentences. The setting of the current level is controlled by two objects in the document window: the **level up** and **level down** controls. The layout of the document window is illustrated in Figure 3. So, for example, suppose there is a document open, whose current level is sentence and the user moves the mouse to the **level up** object. If she clicks the mouse button once, the program will say "Level up". If she then double-clicks the level will become sentence level.

Clicking the **current level** control will produce speech telling the user what is the current level in that document. Double-clicking it will cause the current selection to be spoken. The position of the selection can be altered using the other controls. **Stop forward** moves the selection one unit forward. That is to say at word level, for example, it would move the selection to the next word. **Step backward** has the same effect, but in the opposite direction. The **jump forward** and **jump backward** controls similarly move the selection, except that they move it in units one level up. So, for instance, if the current level is *word* level the user double-clicks **jump backward** the selection will move back one *sentence*, to the first word of the previous sentence.

The **scroll bar** and **thumb bar** represent a direct analogue of a visual control. The **thumb bar** is a movable object whose position within the **scroll bar** reflects the relative position of the current selection in the document. So, for instance, if the selection is in the middle of the text, the **thumb bar** will be in the middle of the **scroll bar**; if the selection is at the end of the document the **thumb bar** will be at the right-hand end. The user can alter the position of the selection using the thumb bar. By double-clicking the thumb bar, the user *picks it up* and can then move it to some other position within the scroll bar. The selection will then be readjusted to reflect that new position.

Dialogues

Many commands require extra information from the user which is obtained by them interacting in a special window known as the **dialogue** window. This is again a direct analogue of the visual interaction. For instance, if the user closes a document window, the system must know whether they want the document to be saved onto disc or not. If so, it will need to know the name by which it is to be saved. This information is obtained via dialogues, which contain control objects, similar to those in the document windows. There are four different dialogues:

1. **save files**, which ascertains whether the user wants a file saved on disc;
2. **get file name**, which obtains the name for a new file which is being saved;
3. **select file**, through which the user can specify the name of an existing file which he wants to open;
4. **get target**, through which the user can specify the target for a string search in a document.

Any one of the four can occupy the dialogue window at a given time.

Alerts

Errors are signalled to the user by special tones. The user would be able to get further information on the nautre of the error through the **alert** window. In the prototype program, the **alert** window has not yet been implemented. This would be of most use in explaining errors which are external to the program, such as the system being unable to save a file because of lack of space on a disc.

EVALUATION

The prototype version of the word processing program is currently being evaluated. It is being tested by visually disabled people, but some sighted subjects are also being used by way of comparison. The objective of the evaluation is to assess how easy the program is to use. From its results it should be possible to conclude whether this is a viable form of interaction for blind computer users.

The evaluation takes the form of training the subjects to use the word processor. This is done through a graded set of exercises. During this training the users interactions for some of the exercises are monitored by the computer and a record saved on a file for later analysis. This consists of data on the movement of the mouse, clicking of the mouse button, typing on the keyboard and the timings of these events. A substantial amount of work has been done on monitoring the interactions of sighted computer users of Wimp programs. From these analyses it has been possible to construct so-called keystroke models of interaction (Card, Moran and Newell (7)). However, these are not applicable to the form of interaction of a blind user using an auditory screen. The major difference is that a sighted user receives constant feedback as to the position of the cursor in relation to other objects on the screen, whereas the user of this form of audible screen is involved in a more complex interaction involving their spatial memory, cues in the form of tones and spoken help. It seems unlikely, therefore, that it will be possible to construct a correspondingly simple model for auditory interaction.

Nevertheless, it is possible to observe how users interact with the auditory screen. For instance, a lot can be learned from measuring the time different users take to locate different objects on the screen. Also, it is possible to analyze the methods which users employ for locating objects, for instance whether they use spoken help to orientate themselves, or do so by finding the edges of the screen. This data, when complete should give some measure of how difficult users find it to interact with this style of program, and how this compares with other forms of interaction.

In addition, more subjective assessment is being made by administering a questionnaire and by an interview with the subjects.

FUTURE WORK

The objective of the evaluation is to assess whether the auditory screen provides a viable form of computer interface, which could be applied to a variety of Wimp programs. If the results appear to be positive then further work will be done to ascertain whether the same approach can be applied to programs other than word processors and to other forms of interaction not included in this particular example. It is expected that it will be possible to elaborate the general principles for this kind of interface, such as auditory analogues of visual representations and constrained layouts of auditory windows on geometric grids. At the same time it is likely to establish the limits of validity of these principles - visual interactions which cannot be transformed into auditory ones. It is encouraging that our approach can be applied to the word processor requiring only a small number of windows which are easy to discriminate.

If the approach seems viable, then an obvious next step would be to develop the word processor into a usable product. The prototype has been designed to text the idea of an auditory screen and therefore has been restricted to using only auditory interaction. A production word processor might be extended to include some visual interaction which would be of use to users with partial sight, probably through the added facility of text enlargement.

Beyond that there is scope for a great deal of development. A variety of existing visual commercial programs should be adapted and evaluated. It is likely that there are some applications for which the above design will not be suitable. For instance, there are a number of Wimp programs which are used to create graphics. The design described above is not powerful enough to make such picutres into audible objects. A means of making drawings is a serious requirement of many blind people. So it would be interesting and useful to extend these ideas to the manipulation of graphical objects which the user can create.

In a more general area, it might be interesting to apply some of the emerging ideas of object-based programming to auditory objects. This could lead to the development of tools which would make the production of the kind of auditory program described above very much more straight-forward.

REFERENCES

1. Vincent, T., 1983, "Microcomputers and synthetic speech: some experiences", in New Beacon, RNIB Braille Publication No. 797, Royal National Institute for the Blind, London, England.

2. Shirley, S. and Schofield, J., 1982, "Survey of microelectronic aids and devices", Learning to Cope - Computers in Special Education, (ed.) Adams, J., Educational Computing, London, Englands.

3. Goldberg, A. and Robson, D., 1983, "Smalltalk-80: the Language and its Implementation", Addison-Wesley, London, England.

4. Duff, C.R., 1984, "Introducing the Macintosh", McGraw Hill, London, England.

5. Edwards, A.D.N. and Woodman, M., 1984, "UCSD Pascal on the Macintosh", UCSD p-Systems Users' Group (USUS) Newsletter, Autumn issue, London, England.

6. Clark, R. and Koehler S., 1982, "UCSD Pascal Handbook", Prentice-Hall, New Jersey, USA.

7. Card, S.K., Moran, T.P. and Newell, A., 1983, "The Psychology of Human-Computer Interaction", Lawrence Erlbaum Associates, New Jersey, USA.

Acknowledgements

I would particularly like to acknowledge the invaluable assistance given to me by the staff and students of the Royal National Institute for the Blind Commercial Training College in the evaluation of this project. I would also like to thank staff and students at the Royal National College for the Blind, Hereford, for their help and comments. Others who have contributed, and whom I thank, include David Calderwood, Mark Elsom-Cook, Tim O'Shea and Tom Vincent. This research is supported by a Postgraduate Studentship from the Science and Engineering Research Council.

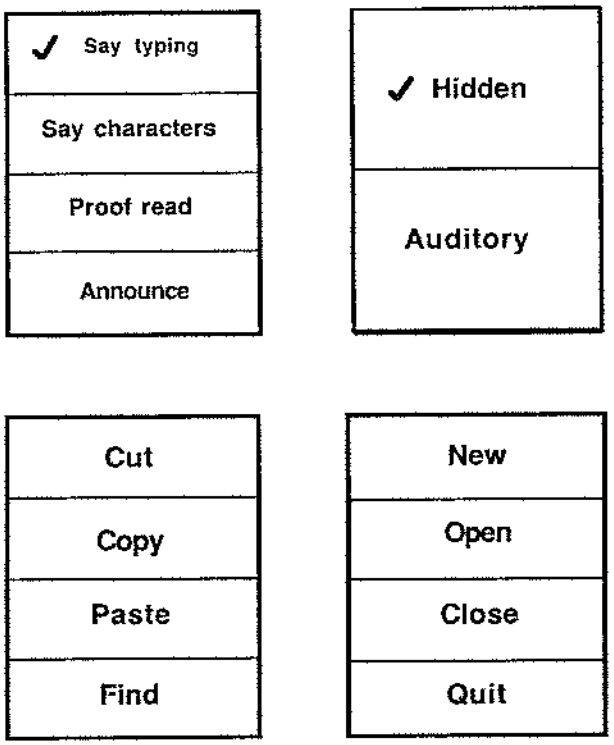

Figure 2. The four menu windows.

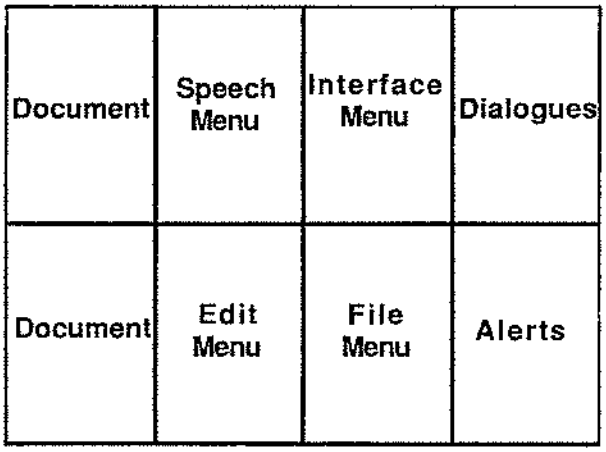

Figure 1. The auditory screen: a grid of 8 windows.

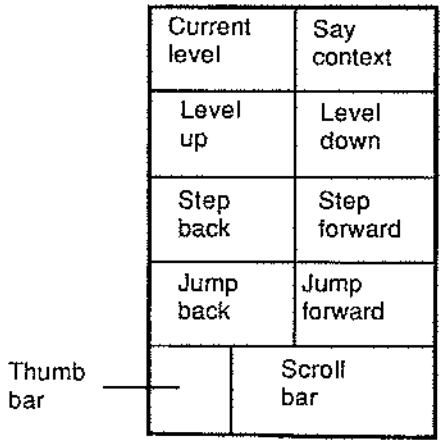

Figure 3. The document window layout.