COMPUTING FOR THE BLIND USER

BY ARIES ARDITI AND ARTHUR E. GILLMAN

Some special human factors must be considered in assembling a workable system

INEXPENSIVE COMPUTERS and nonvisual communications hardware have, in theory, made personal computing as accessible to blind as to sighted persons. But in practice, personal computing has its own special set of problems for the blind user. In this article we'll present some of the human-factors issues specific to nonvisual personal computing. Our concern is to make computers more accessible and efficient for blind and visually impaired persons. We hope our suggestions will be useful to individuals and to designers of hardware and software. Many of the improvements we discuss below can be implemented in several ways, often in more than one component of the system. They are intended to illustrate human-factors issues rather than to critique specific products.

The system we use as a basis for this discussion is a popular one for blind and visually impaired users and is inexpensive enough for home use as well as employment settings. It consists of an Apple IIe microcomputer operating under DOS 3.3, a Votrax Personal Speech System for voice output, and Raised Dot Computing's Braille-Edit program version 2.44a. [*Editor's note: There is a more recent version of Braille-Edit with a number of new features and enhancements. See Henry Brugsch's review, "Braille-Edit," on page 251. Also, for an address list for manufacturers of products mentioned in this article, turn to page 208.*] Most blind users have a printer for producing sighted (conventional) hard copy. Another useful peripheral is a braille printer, since braille hard copy is easier to proofread than voice output. While we will not specifically discuss braille hard copy, many of the human-factors issues discussed here are relevant to the design of braille printers.

Braille-Edit is an integrated software package designed to satisfy most blind users' needs to process documents. It is intended for use with a low-cost artificial-voice system such as the Votrax Personal Speech System or Street Electronics' Echo series (including the Echo+ speech synthesizer) and various other peripherals. Braille-Edit is not intended to (and does not) make all programs that run on the Apple accessible to the blind user, nor is it particularly useful in programming the computer. But it has a number of desirable utilities for the blind user, such as a translator of text to and from grade II braille (a commonly used coding system similar to Speedwriting shorthand) that makes impressively few errors and a copy facility for copying files to and from a paperless brailler such as the Versabraille from Telesensory Systems Inc. (TSI).

The hardware and software designed to make a system accessible to the blind user can be viewed as an in-

(continued)

Aries Arditi, Ph.D., is a vision scientist and experimental psychologist in the Department of Research at the New York Association for the Blind (111 East 59th St., New York, NY 10022) and an adjunct associate professor in New York University's Department of Psychology.

Arthur E. Gillman, M.D., is a practicing psychiatrist and past director of research at the New York Association for the Blind. He is now clinical director of Central Westchester Community Services, Rockland Children's Psychiatric Center. He can be contacted at 34 Pryer Manor Rd., Larchmont, NY 10538.

Our concern is with the accuracy, speed, and generality of the blind-user interface.

terface between the user and hardware or software that presumes that the user is sighted. Our concern is with the accuracy, speed, and generality of the blind-user interface. Accuracy and speed are areas in which data transmission to and from the computer can be improved by better design of the individual components of the interface or by the coordination or compatibility of the components. Speed and accuracy need to be balanced to produce the greatest levels of efficiency. Generality has to do with the scope of programs not specifically written for blind users that can be run by the blind user as a result of interface design.

ACCURACY

Inaccuracies in data interchange between the blind user and the computer arise from imperfect translations of normal input and output into the special modes of the blind user. The normal text output of the screen must not merely be redirected to a voice device but must also be translated into speech that effectively conveys all of the information on the screen. Similarly, if you type in braille, the braille text input must be accurately translated to standard computer text prior to processing.

One of the main functions of any voice-based software interface is to send the text that would normally go to the screen to the voice-output device. The fact that text data is not essentially visual is what makes the accessibility of computers to blind people possible in the first place. The voice peripheral receives streams of characters and pronounces either full words or individual characters. Full-word pronunciation breaks up each word into likely phoneme or allophone strings based on its spelling. Even though there may be errors in pronunciation resulting from exceptions to these pronunciation rules, this scheme has the advantage of an unlimited vocabulary of pronounceable words. An excellent and readable discussion of issues relating to text-to-speech conversion can be found in reference 1.

Full-word pronunciation is desirable when the computer is prompting the blind user or conveying the results of its operations. For example, it is quicker and more natural for the system to pronounce "Enter selection" than to spell it out. Full-word pronunciation is also useful in editing documents, primarily as a means of finding your place in a document. But you cannot rely on artificial full-word text for other aspects of editing because text-to-speech conversion pronounces neither nonprinting characters such as space or tab characters nor punctuation symbols that are embedded in a text file. Typographical errors are also difficult to detect since they may result in only subtle if detectable pronunciation errors. For example, a voice device would pronounce "really" the same way it would pronounce "reelly" because it would analyze the two character strings into the same component phonemes (or allophones).

Spelled-pronunciation mode is the only viable way to know precisely which text characters have been typed and is therefore preferred for most editing and all proofreading. But even with spelled-pronunciation mode it may be difficult to distinguish between similar sounding letters such as "m" and "n" or "t" and "p."

The particular voice device you choose determines both the intelligibility of the artificial speech and usually the quality of the text-to-speech (full-word pronunciation) conversion. The "voice synthesizer" chip within the device is responsible for generating the sounds corresponding to each phoneme or allophone element, and the discriminability of such sounds is largely determined by this chip. A microprocessor and memory, also parts of the voice device, typically perform the text-to-speech conversion. The time and memory taken by the algorithm that implements the conversion are important factors influencing the accuracy of the conversion. Although most speech devices allow you to control a number of speech parameters, such as speech rate, inflection, and pitch, through special commands or controls, none of the less-expensive speech devices has as yet incorporated these parameters into their text-to-speech algorithm. This would excessively reduce the speed and/or increase the complexity of the algorithm. The result is a usually intelligible but monotonous parade of pronounced phoneme strings separated by short pauses for word separation. The near future will undoubtedly bring us more natural sounding artificial speech at low cost.

Thus far, we have discussed problems translating text to voice. Translating braille to text poses additional accuracy problems. One way for a blind user to enter text into the computer is with a braille keypad or keyboard. The braille keypad can be simulated with an ordinary keyboard by using the space bar and the six keys on the lower tier of the keyboard. These six keys correspond to the dots of a braille character cell. With Braille-Edit you can choose to have whatever is typed show up on the screen as ordinary print-style text (through braille-to-ASCII conversion) or as visual braille, which is of some use to the sighted transcriber.

However, the lack of a one-to-one correspondence between ASCII codes and braille symbols causes significant problems. If the software doesn't take this into account, you will get garbage on the screen and gibberish through the voice device. For example, the "dot 6" character in braille signifies that the next character should be interpreted as uppercase. To have the text spoken out properly, the software that takes input

(continued)

Rates of information exchange between the user and the machine will be inherently worse for a blind user.

from braille-keypad mode and sends it to the screen or speech synthesizer must delete dot 6 characters and capitalize the first character of the subsequent word. This is fine for final copies of documents, but since this translation takes time, it is not really practical for writing and editing. To make the braille-keypad mode usable for editing documents, the voice-output device must be made to say "dot 6" or "capital" at the moment a dot 6 is encountered. This can either be done within programs like Braille-Edit, or it can be incorporated into intelligent voice-output-device firmware, whereby they can be made to run in braille mode, filtering out all special braille characters and making appropriate translations when the text is received.

A specific difficulty for systems using braille input resulting from the lack of one-to-one character correspondence is that if you are using the braille-keypad mode, two different systems for coding numbers may have to be remembered. Standard braille uses the convention of coding numbers as strings of characters whose ordinal position in the alphabet (for "a" through "i") is equal to the value of each digit in the number; each such string is preceded by a braille "#" character. For example, the two-character string "34" corresponds to the braille symbols for "#cd," a three-character string, and the print and braille translation facilities assume that numbers have been typed in the latter format.

But user commands that include numerical arguments (e.g., the command for "advance the cursor 34 characters") are best entered as Nemeth code, a system of braille used for mathematics, in which braille numbers are represented by a string of characters just as in ordinary braille. However, in Nemeth code, the number is not preceded by "#"; instead, each character in the number is shifted down one row in the braille cell matrix to distinguish the character as a digit. Because Nemeth code omits the "#," a Nemeth number requires the same number of characters to represent as a print-style digit string and therefore is much more easily translated into a string of ASCII digits.

These translation problems should not necessarily be solved with software. It would not be difficult, for example, to design a braille keyboard that emulates and could replace a standard keyboard. Likewise, a voice device could easily have a braille-to-voice conversion mode built into its firmware.

Another solution to braille-to-text translation problems is for the blind user to learn to type with a standard computer keyboard as Braille-Edit users do. This circumvents the lack of one-to-one correspondence between braille and ASCII codes entirely. By using spelled-pronunciation mode, you get immediate feedback as to which keys you have pressed. (Incidentally, small raised dots on the D and K keys of the Apple keyboard aid in finding the home keys.)

SPEED

Rates of information exchange between the user and the machine will be inherently worse for a blind user than a sighted user because vision conveys text information more quickly than other sense modalities. The sighted user viewing a display screen can receive a screenful of information all at once, whereas the blind user working with artificial-voice output or a braille computer terminal receives information from the computer in serial fashion only and much more slowly than the sighted user. Because of this, increasing speed is especially important for the blind user.

A general design issue that greatly affects the speed of the blind-user interface is the choice of driving the program by menus or by command language. A menu lists on the screen all the options currently available and prompts you to select one. This has the advantage that commands need not be memorized, and thus the user does not need experience to run the program. In contrast, a command language generally provides only a prompting character such as "*" to signal that the program is ready to receive a command. It takes time to learn a command language, but once you learn it, you can specify desired actions more tersely and more quickly than with a menu.

For the blind user, the menu format is especially slow because the menu must be listened to rather than seen. Also, since it is easy to forget options listed early in the menu, lengthy menus must often be repeated. For these reasons, we think command-language format is better for the blind user. Command languages often provide help-command supplements as teaching and mnemonic aids. The command language with help facilities combines the advantages of a menu with those of the command language because you can use a help menu while learning the program and avoid it when it would otherwise be unnecessary clutter.

Another bottleneck that reduces the speed of the blind-user interface is the difficulty in translating screen format to voice format. Redirection of the screen contents to the artificial-voice device makes the primary output of the computer intelligible, but a good deal more effort must go into making the output easily accessible to the blind user. Since so much software is screen-oriented, it is desirable to buffer the output to a "virtual screen." This virtual screen is an area of memory that contains one or more screens full of text and a virtual "voice" cursor. The virtual cursor may be moved about the virtual screen without affecting the contents of the ordinary terminal cursor. With key-

(continued)

stroke commands such as "Speak phrase at cursor," the blind person can quickly read text anywhere on the screen. This feature has been incorporated in several "talking terminals," in TSI's VERT, and in other recent software packages for the IBM PC.

It is important for the user to know the locations of both the actual and the virtual cursors, and there should be a command serving this function. Locations should be spoken out as line and column number rather than serial character position because it is easier to imagine location in two dimensions than in one. For example, it is of limited use to know that you are 323 characters into a page, especially when that page may contain 100 space characters.

Speed of use also depends greatly on the user's ability to adjust voice parameters of the system. Because intelligibility depends on many factors, including speech rate and sound-frequency content (or "pitch"), such controls should be accessible to the user in some way, whether by command or by knobs on the voice device itself. Your ability to understand the artificial voice improves the more you listen to it, and you can maximize efficiency by using the fastest speech you can understand.

There should also be some simple way to interrupt the voice device. A problem with our example system is that during output of long menus, the only way to terminate voice output is to turn off the power switch on the voice unit. Otherwise, you must listen to boring, lengthy menus you have memorized so thoroughly that it is irritating to have to wait for them to finish. Turning off the power on the voice device to clear its buffer and force it to silence also resets the unit. Then commands to set parameters such as speech rate must be sent all over again. A special key or button on the voice device could be dedicated to the "Abort voice output" function.

The choice of word processors versus text-file processors also affects user speed. These are two fundamentally different approaches in programs designed to aid document writing. With what we call the "text-file processor" approach, you create a text file with commands embedded within the text to be executed later by the text-file-processor program. These commands do nothing to the text on the screen during an editing session but give the text a more pleasing spatial layout later, when the chapter is printed through the text-file processor. The embedded commands are, of course, filtered out and do not appear in the output.

A more popular approach with personal microcomputers is that of the word processor, in which editing and formatting the page are combined in the same process. Writing with a word processor makes the screen appear similar to the printed page that will eventually be printed out. As you type, the words are automatically arranged on the "page" to fit within the margins with the currently selected line spacing, etc. Commands such as "Center next line" are performed on the screen as soon as you issue them. For example, whereas the file to be formatted with a text-file processor might contain only $$c to signal that the next line should be centered when formatted, the word processor would insert the number of spaces needed for the line to be centered during the editing session.

Each approach has its advantages. Files to be text-formatted are much shorter because blank spaces and lines are added only when the file is processed. Also, you can make sweeping changes to the entire file with single commands such as "Set double spacing." For the blind user working with voice output, blank space and lines are abbreviated as commands such as $a20 rather than being counted one by one as the voice device repeats "space, space, space. . . ." A disadvantage of text-file processors is that it is more difficult to imagine how the page will appear when printed out. Although this might not seem to be a drawback for blind users, in fact most blind people make use of spatial imagery in similar ways as sighted people (reference 2) and *do* imagine the spatial structure of the printed page. Another disadvantage is that the command strings may be somewhat distracting when spoken out by the voice device in full-word pronunciation mode.

Advantages to the word processor are that the text need not be cluttered with the embedded commands, and you see the page in a form that is more similar to its printed appearance. A disadvantage is that since the text is being rearranged continuously during editing, words move about more. It may be difficult for the blind user to find words, since they rely more on the constancy of word positions in each line. Most word processors, however, allow you to enable and disable the automatic formatting facilities at will.

The choice of word processor or text-file processor generally depends on how the location of the cursor is represented and on how the voice system represents blank space. If cursor location is represented in character units, then the text-file processor is preferred because it is difficult to interpret a cursor location number due to the possibility of significant amounts of blank space. However, if line and column representation is used, and blank space can be spoken out in abbreviated form such as "20 spaces" or as nonspeech sounds representing spaces or linefeeds, then word-processor format is preferred because it is easier to imagine the final output.

GENERALITY

Finally, generality must be considered in designing a system for the blind user. Although a system designed from the bottom up to serve only blind users would have obvious advantages for the blind person, a more realistic goal, and one that fosters better communication with sighted users and produces more employment opportunities, is to design a system that gives the blind user access to all general-audience software that is not intrinsically visual.

A flexible and general system should allow the user to customize

(continued)

the workings of the blind-user interface. You should be able to set all parameters of that interface while an applications program is running. Generally, this involves having either a separate keypad or switches on the voice device, or a special switch or command that redirects normal keyboard input to the voice device. This approach would avoid the possible conflicts between keys (e.g., function keys) that are used both to operate the applications program and to control the voice device.

Some voice devices now allow considerable flexibility in speech style, including options such as whether numbers ought to be pronounced by individual digits or as full words, how to handle abbreviations and acronyms, whether or not to signal punctuation symbols, etc. Some allow you to redefine pronunciations of words and thus correct deficiencies of the text-to-speech algorithm. Such options can be useful, but they must be easily accessible through hard controls or software. Our Votrax speech synthesizer, for example, has codes to set the speech rate and to abort speech, but Braille-Edit has no facility to send such commands. If the Votrax had separate knobs or buttons to control them, the system would be completely software-independent in regard to these functions.

Another important feature of the virtual screen with the virtual cursor described above is that it allows blind users to read portions of the screen that they would not be able to read with an ordinary cursor. An ordinary cursor is generally prohibited from moving into certain areas of the screen (protected fields, e.g., an area that displays current margin settings, tabs, etc.) where information is displayed constantly rather than being pushed off the screen (from scrolling) after the screen is full. With the ordinary cursor, the blind user has constant access only to those portions of the screen that have been explicitly directed by the program to voice output. But a virtual cursor can move within the protected fields, and you can read what would otherwise be hidden without affecting the actual cursor that must obey the boundaries set by the applications program.

Perhaps the major disadvantage of our example system is that it is not possible for us to run any applications programs that are not specifically designed for the blind user. Braille-Edit is a useful program that performs all the functions featured on its menus and probably makes the Apple II computers more accessible to blind people than any other system, but greater flexibility can be achieved by systems that modify or extend the operating-system facilities.

Most applications programs send

(continued)

You already own a computer that can talk. Now let it.

Now you can upgrade almost any personal computer and make it more powerful than ever, by giving it the power of speech.

The Votrax Personal Speech System is the least expensive sophisticated voice synthesizer available today. The PSS's text-to-speech vocabulary is virtually unlimited, and you can define an exception word table and customize your translations. So the PSS can say just about anything!

It's a speech and sound specialist.
The PSS can also mix speech and sound effects or speech and music. It contains its own speaker, a programmable master clock, 256 programmable frequencies, a programmable speech rate for a more natural rhythm, and 16 programmable amplitude levels for incredible control of word emphasis. You can control the volume. Plus, it doesn't use any of your computer's valuable memory.

It's computer friendly.
The PSS is unbelievably easy to use. It doesn't need an interface card for most computers. It comes with standard serial and parallel ports. Speech, music, and sound effects are as simple as printing out a document.

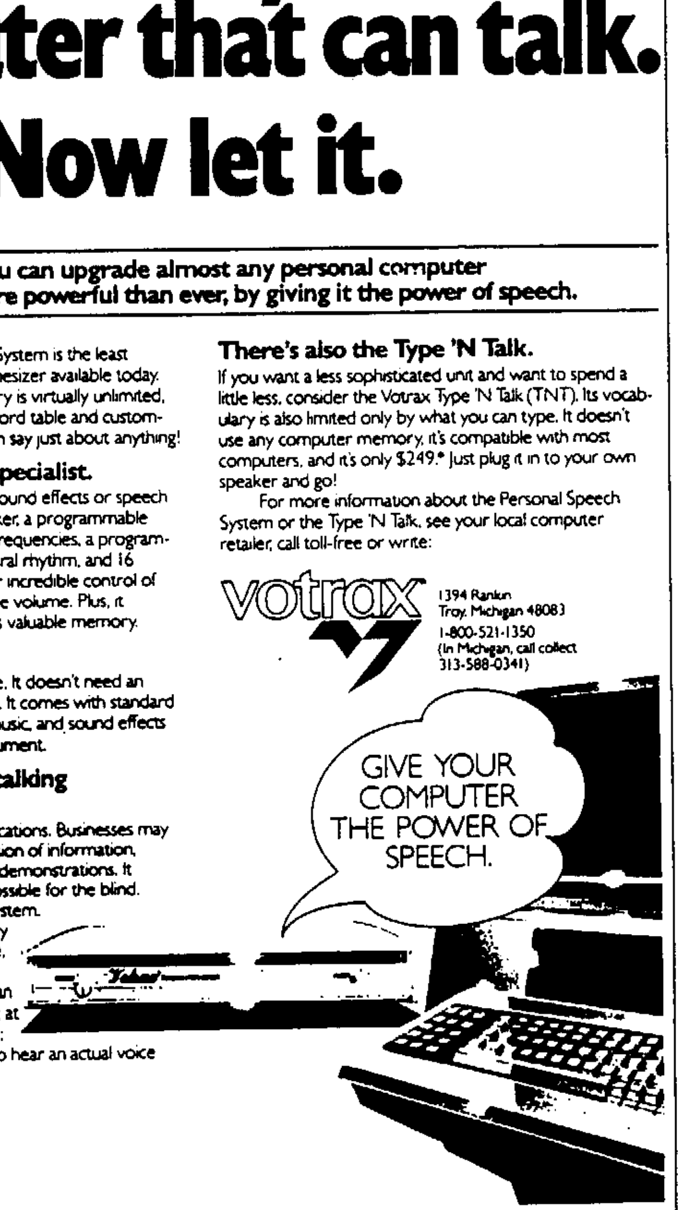
What do you do with a talking computer?
There are countless practical applications. Businesses may want the PSS for spoken transmission of information, narration of displays, and product demonstrations. It makes verification of data input possible for the blind. It can be part of a burglar alarm system. Children can use the PSS as a study aid. And it helps games come alive, speaking while you play.

Whatever your computer can do, the PSS can help it do it better, at a cost that makes it all worthwhile: only $395.* Call (313) 583-9884 to hear an actual voice demonstration of the PSS.

There's also the Type 'N Talk.
If you want a less sophisticated unit and want to spend a little less, consider the Votrax Type 'N Talk (TNT). Its vocabulary is also limited only by what you can type. It doesn't use any computer memory, it's compatible with most computers, and it's only $249.* Just plug it in to your own speaker and go!

For more information about the Personal Speech System or the Type 'N Talk, see your local computer retailer, call toll-free or write:

votrax
1394 Rankin
Troy, Michigan 48083
1-800-521-1350
(In Michigan, call collect 313-588-0341)



*Suggested retail price

Inquiry 371

output and receive input through these facilities. Almost all, for example, request the operating system to type characters on the screen rather than writing a new section of program to perform this task. They do this both because it is easier and because it maintains the generality of the program; i.e., the program will run on all machines that use the same operating system. If the operating system is modified so that all screen output also automatically goes to a voice synthesizer, then virtually any program that is run will send whatever ordinarily goes to the screen to the voice output as well. Operating-system modifications would have the generality that allows programs not written for blind users to be accessible to them nevertheless. In this way, blind and sighted users can work together, using the same software. Braille-Edit is not this kind of modification—it is simply a good applications program written for the blind user. The alternative to operating-system modification is that each commercial program that is developed will require parallel development of a blind-users version of that program.

In recent months several new systems have been offered commercially that do work by these principles. One is TSI's Professional VERT, a voice-output hardware and software configuration for the IBM PC that we have not evaluated. Another is an operating-system modification and addition called The Enhanced PC Talking Program (from Computer Conversations). This program also runs on the IBM PC and will work with most inexpensive voice-output peripherals, although the Votrax Type-'N-Talk is recommended. Another similar, more recent package for the IBM PC from Computer Aids Corporation is called Screen Talk. Both VERT (with an additional plug-in board) and The Enhanced PC Talking Program can emulate several popular computer terminals, making them especially attractive in employment settings. Finally, Maryland Computer Services' Total Talk PC system is a talking version of the Hewlett-Packard HP 150 Touch-screen personal computer. Yet another system, a passive screen reader card for the IBM PC, currently being developed by Tim Cranmer and others at the National Federation of the Blind, will not modify or add onto the operating system but will receive and interpret all data sent to the video screen buffer, so that even programs that bypass operating-system calls to write to the screen can be run.

The rapid introduction of so many new systems illustrates how rapidly the field of making computers accessible to blind persons is growing and becoming more competitive. Undoubtedly the capabilities of computer hardware and software designed for blind users will continue to grow in parallel with the capabilities of systems designed for sighted users.

For a microcomputer to be as accessible as possible to blind people, designers must interface the system to the blind user at as primitive a level as possible. If the computer's operating system "thinks" it is receiving standard keyboard input but is in fact receiving braille that is translated before it is received by the operating-system software, then any program that uses the keyboard ought to be able to use braille entry as well. Similarly, if the operating system "thinks" it is sending output to a screen but actually sends it to a buffer that holds as much as a screenful or more, the blind user can peruse desired sections of it much the same way as a sighted person, for whom the program was originally designed. We hope that our observations have provided some structure to the various problems in designing a blind-user interface. We feel that solutions to the problems we have described will contribute to the more efficient and broader use of computers by blind people. ■

PRODUCTS MENTIONED

SCREEN TALK
Computer Aids Corp.
124 West Washington, Lower Arcade
Fort Wayne, IN 46802
(219) 422-2424

THE ENHANCED PC TALKING PROGRAM
Computer Conversations
2350 North Fourth St.
Columbus, OH 43202
(614) 263-4324

TOTAL TALK PC
Maryland Computer Services
2010 Rock Spring Rd.
Forest Hill, MD 21050
(301) 879-3366

BRAILLE-EDIT
Raised Dot Computing Inc.
408 South Baldwin St.
Madison, WI 53703
(608) 257-9595

ECHO+
Street Electronics Corp.
1140 Mark Ave.
Carpinteria, CA 93013
(805) 684-4593

VERT
VERSABRAILLE
Telesensory Systems Inc.
455 North Bernardo Ave.
Mountain View, CA 94043
(415) 960-0920

PERSONAL SPEECH SYSTEM
TYPE-'N-TALK
Votrax Inc.
1394 Rankin Dr.
Troy, MI 48083
(313) 588-2050

REFERENCES

1. Stoffel, David. "Talking Terminals." BYTE, September 1982, page 218.
2. Kennedy, J. M. "Haptic Pictures." *Tactual Perception*, W. Schiff and E. Foulke, eds. New Rochelle, NY: Cambridge University Press, 1982.