RC 19824 (04/28/94)
Computer Science/Mathematics

IBM Research Report

Screen Reader/2: Access to OS/2 and the Graphical User Interface

Jim Thatcher
IBM Research Division
T.J. Watson Research Center
Yorktown Heights, NY 10598

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties).

IBM Research Division
Almaden . T.J. Watson . Tokyo . Zurich

Screen Reader/2
Access to OS/2 and the Graphical User Interface

Jim Thatcher

Interaction Technology
Mathematical Sciences Department
IBM Research
Yorktown Heights, NY 19598

Abstract

Screen Reader/2 is IBM's access system for OS/2, providing blind users access to the graphical user interface (GUI) of Presentation Manager, to Windows programs running under OS/2, and to text mode DOS and OS/2 programs. Screen Reader/2 is a completely redesigned and rewritten follow-on to IBM's Screen Reader Version 1.2 for DOS.

There has been considerable discussion about the technical challenges, difficulties, and inherent obstacles presented by the GUI. Not enough time and energy has been devoted to the successes in GUI access, in part because the developers of GUI access software have had their hands full trying to solve very difficult problems.

This paper will describe how IBM Screen Reader makes the GUI accessible.

1. Introduction

IBM Screen Reader/2 has a history that extends back close to 10 years. During this past decade, blind users, both inside and outside of IBM, have determined how the product has evolved.

In 1988, IBM introduced Screen Reader/DOS. It was simple and powerful. It included two features that distinguished it from other screen access software. First, it had a separate 18-key keypad and second, it was completely programmable in the sense that all keypad functions and automatic speaking were determined by profiles written in Screen Reader's *Profile Access Language* or *PAL.*

More recently, users have needed to work with OS/2 and Windows applications in their work, education, and home environments. Therefore, IBM developed Screen Reader/2. It is the only screen access system to give blind computer users that access and incorporate access to DOS as well. Screen Reader/2 also uses the 18-key keypad and, like its DOS-based predecessor, it is totally programmable.

1.1 The Screen Reader Philosophy

The first and most important principle of the five-point Screen Reader philosophy is that blind users must have access to the same computing environment as their sighted colleagues. This principle has given rise to four other principles on which the Screen Reader products are based:

1. There is information that must be spoken automatically.
2. A user needs to be able to customize his system to accommodate personal preferences and application idiosyncrasies.
3. Interference between the access software on the one hand, and the operating system and application software on the other, must be minimized.
4. The GUI and the text-based DOS interface are different ways of doing the same thing. Familiar text-based access methods should be the basis of GUI access.

In 1978 at IBM Raleigh, Al Overby realized that all information on the IBM 3377 terminal display could be made available to blind users through synthesized speech. Then they would have main-frame access just like their sighted colleagues. He developed a modified 3377 terminal including a 12-key keypad and a Votrax ML-1 Multi-Lingual Voice System. He called this prototype SAID (Synthetic Audio Interface Driver). Its IBM internal cost was about $10,000! Al's SAID system was

the precursor of IBM's 3278 Talking Terminal product, and the inspiration for PC SAID, the PC version of SAID, which ultimately became IBM Screen Reader.

It is worth noting that in both the SAID and PC SAID efforts, there was no *product plan* involved. That is, the developers and researchers were tackling a problem and seeking an in-house solution at a time when very little was available commercially.

The origin of Screen Reader's powerful autospeak capability[1] is the fact that there was no such feature on the SAID system, even to announce system status or lock key status. This was first amongst improvements for SAID suggested by users of that prototype. With the 3377 being a truly dumb terminal (no processor) there was no way to change the SAID or Talking Terminal keypad mapping and there were as many opinions about that layout as there were users. This led us to the third principle – the user needs to be able to customize his system.

Conflicts between application's keyboard use and screen access keyboard use have plagued access developers and users alike since the advent of screen access systems. Screen Reader avoids this problem by using an independent keypad.

The last philosophical principle listed above is probably the most important for this paper. The fundamental idea is that the GUI and the text-based DOS interface are different ways of doing the same thing. The more the GUI components can be mapped to textual equivalents, the better.

The thesis is that most, if not all, of the GUI can be so mapped. The crucial questions are, "How convenient are the constructs when mapped to the textual world? Can the blind user be accurate and efficient compared with his sighted colleagues?"

2. Basic Screen Reader

This section describes the basic components of Screen Reader both for DOS and for OS/2. The important thing is that these are all common to both text-mode DOS and the graphical user interface. That is precisely what Screen Reader/2 tries to do – abstract away what is graphical about the graphical user interface.

Screen Reader (for both DOS and OS/2) consists of an 18-key keypad, a keypad cable, audio cassettes, printed and braille documents, and diskettes with on-line documentation and software.

The keypad is attached through the mouse port. If the mouse port is not available, you can install an adapter card that simulates it.

The software can be grouped into roughly five sections.

1. Installation utilities.
2. Configuration files and utilities.
3. The PAL compiler and profiles.
4. On-line documentation.
5. Executables and dynamic link libraries (Screen Reader/2 only).

Screen Reader speaks information from the display or changes settings in the speech environment as the result of a user request through the keypad (or because of some *autospeak*).

2.1 Basic Read Requests

Screen Reader includes literally hundreds of standard read requests that are executed using the keypad. The nature of the response is determined by the current Screen Reader *Mode* and *Format* setting.

Screen Reader has two reading "modes," called *cursor* and *pointer*. In cursor mode, all reading requests are relative to the application cursor. In pointer mode, all read requests are relative to Screen Reader's own "pointer" (called the review cursor by some screen reading packages).

Because other screen readers do it differently, it is important to emphasize that all Screen Reader reading requests work in either pointer or cursor mode. The request is the same key sequence on the Screen Reader keypad, (like {2} for current line). Depending on the current mode, that will be the cursor line, or the line containing the Screen Reader pointer.

The Screen Reader "format" also influences any read request. There are four reading formats called *text*, *pronounce*, *spell*, and *phonetic*.

In text format, Screen Reader tries to read as if someone was reading a book aloud. In pronounce format, the reading is the same but punctuation and blank lines are also announced. In spell format, all words are spelled. And in phonetic format, everything is spelled using the international phonetic alphabet.

For example, if a user pressed the {2} key on the Screen Reader keypad and he was in *pointer* mode and *spell* format, he would hear the line containing the Screen Reader pointer, spelled in its entirety.

Screen Reader includes shortcuts, too. For example, there is a key sequence that requests Screen Reader to spell the current word without changing the active format.

Here is a list of some basic reading requests:

- Read line number X.

[1] The autospeak concept is discussed in Jameson [4], Thatcher [7], and, of course, IBM [9]. *Autospeaks* are a key part of the Profile Access Language (PAL). They allow users to specify an expression (part of the display for example) to be monitored and when the value of the expression changes, the body of the autospeak is executed; that, in turn, can be arbitrary PAL code, i.e., any conceivable Screen Reader action.

- Read the entire view or read from current position to the end (refer to 3.4, “The Concept of View”).
- Read the current, previous, next sentence, line, word, or character.
- Move pointer to cursor, to the top left, to the bottom left, or to the right edge.
- Spell the current, previous, or next word.
- Announce the ASCII value of current character.
- Say the current character using the phonetic alphabet.
- Announce the color, font, style and size of the current character.
- Search for a string, search again, or search from the bottom.
- Ignore capitalization or not.
- Announce spaces or not.
- Treat entire screen as single line (wrap) or not.
- Announce line numbers while reading or not.
- Use the dictionary or not.

This is just a small listing of the hundreds of read requests that a user of Screen Reader/DOS or Screen Reader/2 can make.

2.2 Editing

The Edit Facility is one of Screen Reader’s most important features because it provides feedback as the user types or when he moves the cursor.

The five major feedback options in the edit facility are enumerated below:

- **Line Browse**.

  When line browse is on, every time the cursor moves one position to a new character, that character is spoken. When the cursor moves horizontally a word at a time, the word is spoken. And, when the cursor changes row or the line changes as in a scroll, the new line is spoken.
- **Flush**. When flush is on, each new edit facility response stops the previous response.

The Default edit settings are line browse and flush. In addition there are three ‘echo’ settings:

- **Character echo**. Echoes each character as typed.
- **Word echo**. Echoes words that have been entered or modified as the space is typed at the end of the word.
- **Line echo**. Echoes new or modified lines when the cursor moves to a new line.

The Edit Facility contains other options that can make life easier, including a margin indicator and a column browser.

2.3 Profiles

Everything discussed so far – read requests, modes and formats, the Edit Facility – all of it is accomplished through small Screen Reader software programs called *profiles*. These profiles are written in a special Pascal-style language called the Profile Access Language, or PAL. Profiles completely define the basic screen reading environment and they are used as add-ons to tailor that basic environment for specific applications.

This aspect of Screen Reader is often misunderstood. The application-specific profiles are developed for making applications easier to use. Everything that is done in application-specific profiles could be done without them (probably), but with more effort (probably a lot more effort).

The simplest example of why an application-specific profile is necessary is when the application has a status area in which error messages are displayed. Using the keypad and standard read requests, the Screen Reader user can go to that area and find out if a messages is there, and read it. With an application-specific profile, however, that area can be monitored and automatically read. In addition, a simple application-specific key sequence could be defined to read the status area.

The Screen Reader products support over 30 programs with application-specific profiles. These profiles are supplied for OS/2, DOS, and Windows applications.

3. The Graphical User Interface

As was stated earlier, the graphical user interface is an alternative to the text-based DOS environment for doing the same kinds of things.

3.1 Text Mode Compared to Graphics Mode

Let’s take a look at the two different computing environments.

The model of the display for text-mode computing is a twenty five by eighty array of pairs of numbers. (The size of this array can vary.) The first number in the pair is the ASCII value of the character that appears in that position of the screen and the second number is the attribute - it gives color information for foreground and background colors and tells whether or not the character is blinking.

For example, if the background is blue and there is a white ‘I’ in the upper corner of your text-mode display, then the first pair of numbers in the array will be <73, 31>. 73 is the ASCII value of capital I, and 31 is the number that represents white on blue.

In text-mode computing everything is stored in display memory and it is stored there in a useful form. The display hardware uses only display memory to form the image on the screen. Display memory actually contains this array of pairs of numbers. This is what gets displayed, nothing more, nothing less.

The ASCII numbers that represent characters are exactly what a screen reader sends to the synthesizer for it to translate ASCII text into speech. When 73 is sent to the text-to-speech device, it says 'I'.

For graphics-based computing, there still is display memory, but now the numbers in that display memory only represent pixels, which are just dots of color. (Pixels are also referred to as picture elements or PELs.)

For example, a white on blue 'I' is made up of about 128 pixels - some are the color blue, others - comprising the I - are white. The display memory holds only pixels. It contains no ASCII values.

3.2 Access to Graphics-Mode Computing

As you can gather from the preceding explanation of the difference between text-mode and graphics-mode computing, getting speech access to the GUI is difficult.

Some people suggest that character recognition could be used to determine the contents, especially text, on the graphics display.

It is fairly certain that character recognition is feasible for a single static screen, but not for changing screens as we find in a computing environment.

The alternative to doing character recognition is to 'catch' the textual data before it is turned into dots of color, into pixels. Berkeley Systems was the first to use what they called an *off-screen model* (OSM) when they introduced outSpoken for the Macintosh in November 1989, the first screen reader for a graphical user interface. (Refer to Boyd et. al. [3].)

To the best of this author's knowledge, all screen readers for graphical user interfaces use some form of OSM. The idea of the off-screen model is to intercept everything that is going to the display before it becomes pictures, and record relevant information in a separate data structure or data base, called the off-screen model. The information recorded there will include the text, its position, color, font, and window handle. (A window handle is a tag that identifies the window.)

That is the minimum the OSM will contain. The OSM for different screen readers may contain more and different information depending, in part, on the level at which the drawing calls are intercepted.

Once the off-screen model is available, a screen reader can go into the OSM instead of the display buffer, to get the text and/or icons that are on the display. As we shall see, Screen Reader/2 organizes the OSM data *as if* it were looking at a text-mode display.

3.3 Main Windows

Visually the most obvious graphical objects on the display running OS/2 are windows. Each running program displays its output in its own *main window*. (There may be several main windows open since OS/2 is a multitasking operating system.)

In general, the main window is made up of a border, titlebar, some icons at the corners and a central work area. Each component of the main window is itself a window in a technical sense.

The main windows are usually overlapping, being spread out like papers on a desk, thus the name Desktop used in OS/2. Whether or not a main window is the foreground window is, of course, also indicated graphically. The colors of the border and title bar of the foreground window are different from those of background windows. The actual colors are not important and the OS/2 user can change them at will.

3.4 The Concept of View

Screen Reader/2 is designed to direct the users access to the current foreground main window. The rectangle of data on the display contained within the main window (remember it usually consists of technically several windows) is called a *view*. As the user switches between applications, or between main windows, Screen Reader/2 automatically adjusts the view to the current main window.

The reason the view is set as it is by Screen Reader/2 is that conventional read requests would be utterly confusing without it. Successive lines or blocks of text might come from different main windows and make no sense at all. The user can have it both ways. With standard keypad requests, the user can make the whole display (or *desktop*) the view, or only the contents of a single pushbutton. In both these cases, the view would be 'formatted' in the same way as the main window, sorting all text strings by baseline.

Views are determined by the windows. A view always includes all child windows contained within a specified window. Thus the view determined by the desktop includes everything displayed; the view determined by a pushbutton is just the button text or icon.

Besides adjusting the view, Screen Reader/2 automatically announces the window title when the main window or application changes.

Screen Reader/2 interacts with the operating system to recognize main windows and figure out which main window is in the foreground. This information is *not* stored in the off-screen model. Once the view has been established by Screen Reader/2, then the OSM comes into play.

Unlike other screen readers, IBM Screen Reader/2 treats the current view as if it were a text window. To do this Screen Reader/2 must make decisions as to how to present the graphically displayed text in a familiar text-mode character array.

This simplifies access to the graphical view, though it is not always a totally accurate reflection of what is displayed. It is accurate enough, however, since as argued above, *most* of what one is doing with his computer under the GUI was being done before in text mode. With many users of Screen Reader/2, the experience has been that this filtering of the displayed data is a help for review and that no significant information is lost.

3.5 Transforming the Graphical View to Text

The algorithm for the transformation is simple. The main window will technically consist of several windows, like titlebars, menus and the like.

All the text contained in the main foreground window and windows it contains is sorted by baseline, the bottom line of the text strings. The resulting strings are the rows of the view. If a full screen of text from a text-mode word processor were displayed in a window under OS/2, then all those rows of text would be rows in the Screen Reader/2 view. Columns are a bit more artificial than rows, because most fonts under the graphical user interface are proportional, i.e. different characters have different widths. But just the same, each row is assumed to have the same number of 'columns,' namely the maximum character length of all the rows of the view. So, in the full screen of text mentioned above, instead of 80 columns in text mode, the number of columns would be the same as the length of the longest line of text. Shorter lines are just filled out with spaces (just as they are in text-mode computing).

The main window is, in this way, organized just as if it were a text-mode application. Let's say there are 25 rows and 80 columns in the current main window. That means that 25 distinct baselines were found in the main window and the maximum length of those lines is 80 characters. If this character array were displayed on a text-mode display, then all the characters in column 1 would be visually aligned in a vertical column (as the word suggests) as would those in column 2, column 3, and so on. This is absolutely not true with the graphical user interface. In fact, we should not use the word 'column'. Instead we should say 'character position'.

There is no left-to-right relationship between the position of a character in a line and its position on the display except relative to the other characters on the line.

Even if two lines (or rows) start out at the same place (left-to-right, the same x-coordinate), even if they are vertically aligned to start, the next character is probably not lined up because the GUI usually uses proportional fonts, not the mono-spaced fonts of text-mode computing.

Two blocks of text could be the same font, size and style and have the *same* baseline and still be on different 'rows.' This happens when the text occurs in separate windows. Text in distinct windows is certainly intended to be read separately, so Screen Reader/2 puts that text on distinct 'rows.'

All the information that is ignored by this row-column transformation of the view is available to the Screen Reader/2 user. This includes:

- The pixel position of each character and its size.
- The font, pitch, style and color of each character.
- The pixel size of the view and its position on the desktop.
- The identity of the window containing the text.
- Whether one row or column truly lines up below another.
- The amount of white space between rows.
- The amount of white space at the beginning and end of rows.

3.6 Menus

Menus are basically textual objects and the main screen reading requirement is to set the view correctly.

A serious problem caused by menus in text-mode computing was exactly the one addressed by views above. For text-mode computing, when a pull-down menu appears, it does so on the top of the main display and reading becomes a nightmare unless some accommodation is made to restrict reading to the box containing the menu. This is just as true for the GUI.

A menu is just another window that drops down on top of the general work area. When it does, Screen Reader/2 changes its view to the menu so that reading requests will make sense.

Menu items are announced as the user moves up and down the menu. In addition, certain graphical properties are reflected conveniently in sound:

1. If the menu item is *greyed* (not usable in the current state) a beep is heard. The beep is the same frequency and duration as that heard by a sighted user when trying to actually execute a greyed item by either pressing Enter or by clicking with the mouse.
2. A menu item may have an accelerator (or fast path) letter, indicated by that letter being underlined. Screen Reader/2 announces this to the user when such an item is encountered.
3. Similarly, some menu items have an icon indicating a sub-menu and this too is announced.

3.7 Dialogs and Controls

A *dialog* is one kind of main window that, at first consideration, may not seem to fit well into the row and column format that Screen Reader/2 adopts.

Like the main windows described above, dialogs have a border, title, icons in the corners and a main work area. In that work area, however, there are typically many other windows (in the technical sense, again) which allow the user to enter data and make choices. The individual windows (buttons, entry fields, static text fields, check boxes) are called *controls*. The user makes choices with the buttons, entry fields and check boxes while prompts or headings are provided by static text windows.

In general, this data is laid out in what is considered to be a visually pleasing way. Controls relating to each other will be grouped together, hopefully within a static text window called a *group box*. Except for this grouping, and the position of the static text relative to related controls, the layout does not convey information.

For example, in the OS/2 Enhanced Editor, a search dialog appears when you want to search for and possibly replace text in a file you are editing. At the top of the dialog are two entry fields. The first has the prompt 'search' and the second has the prompt 'replace.' You type the string you want to search for in the first entry field and press Enter.

These two entry fields and their prompts actually comprise four windows in this view, and four rows as well. As you move to them, however, with the standard Tab-key movement, you hear 'search, entry, blank' and 'replace, entry, blank' respectively. This is the prompt ('search' and 'replace'), followed by the kind of window ('entry', short for entry field) followed by the current state of that entry field ('blank').

Because of the feedback from Screen Reader/2 to the user, there is little interest in what rows and columns are involved, or the fact that the sighted user might say that the prompt and the entry field data appear to be on the same 'row.'

Continuing with this example, below the entry fields is a box (looks like a window and is called a 'group box') with a heading 'Options.' This box contains five check box controls which consist of small squares (about one-quarter inch) which may or may not have check marks in them. Each has text immediately to the right of the small square. These are used to select options such as 'ignore case,' or 'change all occurrences.'

As the user tabs into this area and moves the selection cursor from check box to check box with the arrow keys, Screen Reader/2 responds with:

'Group box, options, check box, ignore case, checked'

or

'Group box, options, check box, change all occurrences, not checked'

'Group box' tells the user that the controls (check boxes) are grouped together with the heading 'options'. Next 'check box' announces the type of control encountered. Associated with the check box is text indicating the option to be chosen. In the two examples above, 'ignore case' and 'change all occurrences' are those options. Finally, 'checked' and 'not checked' indicate that the status of the check boxes, checked or enabled in the first case, and not checked or disabled in the second.

It is standard in a graphical user interface to toggle the check state of a check box with the Spacebar. Pressing the Spacebar when the 'ignore case' check box is selected, results in the announcement:

'Group box, options, check box, ignore case, not checked'

Finally, at the bottom of this dialog is a group of six pushbuttons, things like 'Find,' 'Change,' and 'Cancel.' When the user tabs to this group of buttons, and moves around the push buttons with the arrow keys, the information is presented in a form similar to that for check boxes:

'pushbutton, find'

'pushbutton, cancel'

In this case, the type of control, 'pushbutton' is announced together with the text on the button.

3.8 Mouse Actions and Other Tricks in Screen Reader/2

There is no doubt that not using a mouse with the graphical user interface is a disadvantage. It is quicker to do some things with a mouse than with the keyboard equivalents (when those keyboard equivalents are known).

Mouse actions (left and right button, click and double click) can be performed with Screen Reader/2 at any time using the Screen Reader keypad.

Think of the Search Dialog for the Enhanced Editor described in the previous section. As we discussed there, the normal use of that dialog is to use the Tab and or use arrow keys to move the selection cursor to the required spot, enter data, or change options with the Spacebar, and then press Enter to carry out the action. An alternative is available for any application.

Using a couple of basic Screen Reader/2 requests from the keypad, the blind user can perform a mouse single click or double click at the current position. So, when the Screen Reader/2 user reviews the contents of dialog and is reading, for example, the 'ignore case' check box, a single click (using the keypad) will change the check state of the check box from checked to not-checked or conversely. A single click (using the keypad) when

reading the pushbutton ‘OK’ will initiate the search, as will pressing the Enter key.

A mouse user brings a window to the foreground by moving the mouse to any part of that window and clicking with the left mouse button. If that window is totally obscured, either other windows must be moved, resized, minimized, or closed, or the window list must be used.

From the keyboard, Ctrl+Esc brings up the list of open programs, called the *window list*. Then, the Screen Reader/2 user can scan that list one at a time, or press the first letter of the title, until the item is found and then open the folder or start the program by pressing Enter. This is a significantly longer activity than moving the mouse and clicking.

Screen Reader/2 users can identify up to 15 windows (in the default case) and when they want to switch to one of their favorite windows (whether obscured or not) they just enter the number of the window preceded by a simple chord on the keypad. Movement between a collection of standard windows is, with the Screen Reader/2 Switch List, at least as easy, if not easier, than the same task for the mouse user.

The OS/2 2.1 Desktop covers the entire display. On it are icons for programs and program folders and other objects like printers. To open (start) one of these objects the mouse user must find it, and double click. Other windows that are open are very likely to obscure one or more of these icons, and the only option here for the mouse user is to close, move, resize, or minimize some of the open windows. This is often a problem for the sighted user, but it would be a disaster for the blind user.

Screen Reader/2 solves this problem with an alternative desktop which is a normal folder object and can be brought to the foreground just like other folders. In particular, the Screen Reader/2 Switch List can be used to make the desktop the current view.

4. Conclusion

PC SAID, Screen Reader/DOS and Screen Reader/2 evolved based on the requirements and preferences of IBM employees doing their jobs. The internal electronic discussions on Screen Reader for DOS and Screen Reader/2 contain (currently and archived) over 140,000 lines of discussion.

That evolution has led to the form and character described in this paper. The basic reading and editing with Screen Reader/2 is carried over from its DOS counterpart. The profilability (programmability) derives from the fact that the user population is so diverse. ‘Hard-coded’ decisions were not acceptable.

It was only after submitting this paper that I came to realize that the Screen Reader/2 concept of view is unexpected and ‘controversial’, at least amongst those who have not used the product. This is the default configuration that Screen Reader users demanded as they moved their work from text-based computing to graphics based computing.

There are many capabilities in Screen Reader/2 (and of the Profile Access Langauge) which we have not discussed here. It remains to be seen whether or not more pixel based approaches could be simulated with Screen Reader/2 and PAL. This author believes that is possible.

The important point, however, is that the concepts discussed here work for a significant number users in their home and work environments.

Acknowledgement

Screen Reader is the result of the efforts of many people. The author has had a project in the Mathematical Science Department of IBM Research for almost 10 years and several people have worked on that project. The Special Needs Organization of the IBM PC Company is the organization responsible for the Screen Reader/2 product. They took PC SAID and made IBM Screen Reader. Many people have been involved there as well – developers, planners, and writers. Fran Hayden is one of those writers and she has been invaluable in making this paper more readable. The author is also grateful for the referees’ suggestions both on exposition and content. Finally there are users. Those individuals both inside and outside of IBM have provided both the direction and motivation for the Screen Reader effort.

Bibliography

[1] Adams, F.R., Crepy, H., Jameson, D.H. and Thatcher, J.W., “IBM Products for Persons with Disabilities,” IEEE GlobeCom '89. November, 1989. (1990) 496-502.

[2] Adams, F.R. and McTyre, J.H., “Screen Reader: An Audio Access System,” ICAART Conference Proceedings, Montreal June, 1988.

[3] Boyd, L.H., Boyd, W.L., and Vanderheiden, G.C., “The Graphical User Interface: Crisis, Danger and Opportunity,” **Journal of Visual Impairment and Blindness 84** (1990) 496-502.

[4] Jameson, D.H., “PC SAID: Computer Access for the Vsually Disabled,” M.Sc. Thesis, Trinity College, Dublin, Ireland, 1986.

[5] Emerson, M.R., Jameson, D.H., Pike, G.G., Schwerdtfeger, R.S., and Thatcher, J.W., "Screen Reader/PM - GUI Access by blind computer users," Proceedings, World Congress on Technology, December, 1991.

[6] Schwerdtfeger, R.S., "Making the GUI Talk," **BYTE**, 118-128 (1991).

[7] Thatcher, J.W., "Screen Reader/2 - Programmed Access to the GUI," To appear, International Conference on Computers for Handicapped Persons, 1994.

[8] Thatcher, J.W., "How to make OS/2 Talk and Why. Access to OS/2 with Screen Reader/2," Proceedings OS/2 World Conference, Volume 2, pp 331-346. July, 1994, Santa Clara, CA.

[9] "IBM Independence Series Screen Reader Reference," The IBM Corporation, Amonk, New York, 1989.