N92-22465

COMPUTER INTERFACES FOR THE VISUALLY IMPAIRED

Gerry Higgins
NASA Marshall Space Flight Center
Mission Analysis Division EO41
Huntsville, AL 35812

ABSTRACT

Information access via computer terminals is an essential part of many jobs. This concept extends to blind and low-vision persons employed in many technical and nontechnical disciplines. This paper details information on two aspects of providing computer technology for persons with a vision related handicap. The first is research into the most effective means of integrating existing adaptive technologies into information systems. This will detail research that has been conducted to integrate off-the-shelf products with adaptive equipment for cohesive integrated information processing systems. Details are included that describe the type of functionality required in software to facilitate its incorporation into a speech and/or braille system. The second aspect is research into providing audible and tactile interfaces to graphics based interfaces. The paper includes parameters for the design and development of the Mercator Project. This project will develop a prototype system for audible access to graphics based interfaces. The system is being built within the public domain architecture of X-Windows to demonstrate that it is possible to provide access to text based applications within a graphical environment. this information will be valuable to suppliers of ADP equipment since new legislation requires manufacturers to provide electronic access to the visually impaired.

INTRODUCTION

Over the past several years, computer interfaces have become a major topic of discussion, research and disagreement. The interfaces that exist play a major role in defining how we use computing environments for engineering, analysis, business and numerous other tasks. These interfaces are most commonly discussed and defined in terms of "look-and-feel." This is to say that the appearance of the screen and the response to pointing devices and/or keyboard input determine how users judge the effectiveness of the interface.

The generic terminology, "look-and-feel," emphasizes a visual approach in defining the software user interface. Objects are displayed on the screen to represent textual and graphical information. The relationship of these objects to one another is used to focus the user's attention, define software status, depict options and of course convey meaning. The "feel" of the software is somewhat misleading since there is rarely a tangible tactile element to the user interface. The "feel" relates more to how the visual elements of the display are modified through the use of keyboard or pointing device input.

This emphasis on visual representation has dramatic consequences when a visually impaired individual requires access to computers. The information on the screen must be made available either through an audible output device or through a tactile device. This means a great deal more than just having the screen read aloud by a voice synthesizer. The audible or tactile interface must provide the same explicit and implicit information as is present in the original visual interface. There must be methods for the user to truly interact with the computer just as a sighted counterpart would do. This means audible or tactile responses to inputs, the ability to "look" around the screen, the ability to determine relationships between data that is grouped together and the ability to understand the current state of the software.

In the 1980's, many advances were made in making PC environments available to the visually impaired. However, with one notable exception, these interfaces rely on character based visual environments such as the one found in the early IBM PC. The character based nature of these computing platforms were key to the early success of these endeavors. However, modern technology has rapidly moved computing software and display methodologies into a graphics environment. New technologies must now be developed to insure that visually impaired individuals can continue to participate in tomorrow's information environments.

This paper will outline several of the existing technologies and explain how they can be integrated to make an audible and tactile interface. The latter half of the paper will define research funded through NASA to extend this type of interface into newer graphical environments. This research and prototype is called the Mercator Project and is being developed for the X-Windows Graphical User Interface (GUI) environment.

EXISTING TECHNOLOGIES

Speech and braille output from computers dates at least back to the 1970s. Early speech systems were developed on CP/M machines and HP computers. Soon after the arrival of both the Apple and IBM PC, speech products became available for those environments as well. These speech systems relied on two main components. The first is the speech synthesizer. The synthesizer has the capability of stringing together phonemes to create words. These devices respond to ASCII data that is sent to them from the computer. The second component is now termed a screen access program. This program operates to join the computer with the voice synthesizer in creating audible output from standard off-the-shelf software. This means that through this combination of speech synthesizer and screen access program, a visually impaired user can operate software such as word processors, spreadsheets, data base programs and communications packages.

The speech synthesizer is used to announce each input that the user makes through the keyboard. It also announces information that is displayed on the computer screen. This sounds like a relatively simple solution but the application of the technology proves to have many more elements than may be initially assumed. Some of these elements include:

- How often does the user want to hear an updated screen?

- Is the information being input by the user actually going into the location intended by the user? (Remember, the user cannot just look at the screen to answer this question.)

- As various parts of the screen are updated, what information should be immediately vocalized and what information should be available upon user request? For example, the user needs to know when a menu appears on the screen. However, the user of a word processor will not need to hear the line and column updates after each typed character.

- How does the user know what is in a dialogue box and what is outside the box?

- How does the program differentiate between actual cursor position, highlighted menu options, multiple text windows on the screen and error messages?

- How does the screen access program provide "look-around" capability to the speech user without interfering with the normal operation of the program?

- If colors and video attributes are an important part of the display, how will this information be represented to the speech user?

These issues have been adequately addressed in numerous commercial products that work with character based software running in the MS/DOS environment. Many of the same issues have been addressed to create a speaking interface for the Macintosh operating system. This screen access program, developed by Berkeley Systems, was the first to provide the visually impaired access to a graphical user interface. However, this program does not work with all Macintosh software and only works with text based applications. IBM is working on a version of their Screen Reader program that will provide access to Presentation Manager and OS/2. This is the current IBM graphical user interface.

Tactile devices for producing braille representations of the computer screen are available for MS/DOS computing environments. These devices provide a small window of braille characters, usually one line at a time. This window can be adjusted and moved around the display to show various elements of any display. These devices are limited to functioning only in character based environments.

Both the speech based screen access program and the braille devices rely on the character based nature of the PC platform. In standard DOS applications, a programmer causes a text string to be placed on the video display by either making calls to the Basic Input/Output System or by putting the characters directly into the display memory map. The current adaptive technology can either intercept these calls to BIOS or look directly at the memory map for display information. For each character on the screen, there are 2 bytes in memory that represent the displayed value. One of these bytes is the actual character. The other byte represents information on foreground color, background color and other video attributes such as blinking video. In text mode, it becomes rather easy to obtain information about what is displayed at any given time. Some programs have also

become quite adept at analyzing the memory map and informing users about the changes that are important.

This type of technology solved many problems for the visually impaired computer user until software developers began developing more complex and unique forms of conveying interface information. Many developers stopped using cursor positioning as a method to draw the user's attention to a particular part of the screen. This is now routinely done with alternate symbols from the upper part of the ASCII symbol set or by changing colors and video attributes. Screen access programs have become much more sophisticated in providing audible representations by allowing tracking of these newer forms of representations. In research done at NASA on these types of character based interfaces, it was determined that careful planning of a few interface elements of standard software could make access easier for the visually impaired. These elements include:

- Consistent use of colors or video attributes.
  This means that the developer should use a unique combination of colors or attributes to indicate points of interest within a given program. These points of interest might be highlighted menu items, current field names or error conditions.

- Consistent differentiation between elements of the display.
  This implies that there should be variants in color or attributes for different elements of the display. For example, the developer should not use the same video scheme for an error message and a menu selection item.

- Boxes or windows that are formed around text groupings should be complete.
  The current screen access programs are capable of tracing boxes that pop onto the screen but the lines must be complete and drawn with the extended ASCII symbol set.

- Designs that preclude multiple writes of the same information.
  Some DOS based programs and many mainframe programs tend to update the same information multiple times. This is usually because the design of the program doesn't adequately control the order of displayed information. The effect is that the audible output devices speak certain lines multiple times.

- Consistent layout of menus and dialogue boxes.
  Some programs pop up menus in different locations based on information that may not be readily apparent to the casual software user. A menu may appear in the top left corner of the screen while the next menu appears in the middle of the screen. It is much more straightforward for the user of the audible interface if menus consistently appear in the same region of the display.

- Keyboard shortcuts should be available for all actions.
  Visually impaired users of software are still not able to use pointing devices. A programmer should provide keyboard shortcuts for the non-mouse user in order to avoid numerous keystrokes to position an on-screen pointer.

- Consistent usage of cursor positioning.
  The cursor is often locked off screen during the entirety of some applications software. In other applications, the cursor's positioning on the screen has little to do with the action or input that is expected from the user. Many screen access program and especially braille displays rely on cursor positioning for basic tracking information. When possible, the cursor or a clearly defined alternate should appear at the display item of highest interest.

Transition

Designers and users of software are becoming more interested in the interface characteristics offered in graphical environments. This means that fewer programs will be introduced with character based displays. When there is no character based display, there is no memory map for the screen access program to use in providing an audible or braille interface. In these graphical environments, text is drawn on the screen by selectively turning on and off display pixels. This allows a lot of flexibility to the sighted user in creating "better looking" displays. However, it presents a tremendous access barrier to non-sighted computer users. This is because there is not an exact representation of the screen image kept in memory. In order to obtain access to the displayed information, software must be developed that intercepts data before it is changed to a graphic image. Other solutions involve performing optical character recognition processes on the display image. This is a costly and slow process at best.

The transition to these types of graphical displays has already begun. It is possible that the computer technology that has been so revolutionary in providing independence to many non-sighted individuals will change to environments that are not suitable for nonvisual usage. The effect of this will be loss of jobs, loss of information access and several steps backward in efforts to integrate the visually impaired into mainstream society.

THE MERCATOR PROJECT

Over the past 2 years it has become especially clear that blind employees at the Marshall Space Flight Center will need to be able to access graphical user interfaces to perform their jobs. Many of the software and hardware systems that are being upgraded for use in space Station support will be based on graphical displays. Analysis shows that the majority of these systems will be hosted in environments that rely on the X-Windows protocol for display of textual and graphics information. The impact of this type of graphical user interface is not limited to work at MSFC. X-Windows is being employed throughout the government, academia, and commercial software implementations.

As a part of research into this problem, MSFC began working with software developers at the Georgia Institute of Technology. The research team headed by Elizabeth Mynatt and including David Burgess, Keith Edwards, John Goldthwaite, Bill Putnam, Tom Rodriguez and Enian Smith has developed a concept and system design for an audible interface to GUIs. This environment is called "The Mercator Project: A Nonvisual Interface for X Windows and UNIX Workstations."[1] The

1. Elizabeth Mynatt and W. Keith Edwards, The Mercator Project, unpublished, 1991.

name refers to "Gerhardus Mercator, a cartographer who devised a way of projecting the spherical Earth's surface onto a flat surface with straight—line bearings."[2] The relationship between Gerhardus Mercator and the Mercator environment is in the mapping of a visual interface to an audible interface. In addition, all interfaces can be said to aid in navigation through the underlying software. The Mercator environment will be prototyped during the winter and spring of 91-92 at the Georgia Institute of Technology. NASA personnel including Gerry Higgins and Craig Moore will be providing technical expertise, requirements and design concepts. The initial prototype will be hosted on a Sun Workstation. However, the intent from the outset of the project is to make the Mercator environment platform independent. The goal is to make a final system that will work in any standard UNIX implementation on a workstation that supports X-Windows.

The MOTIF interface standards are being employed to help insure that Mercator will work with a wide variety of applications. MOTIF is an interface standard that is expected to be used by a majority of software developers who program X-Windows applications. In the Computer Glossary, Motif is defined as: "(Open Software Foundation/Motif) A graphical user interface (GUI), developed by OSF, that offers a PC-style behavior and appearance for applications running on any system that supports X Window, Version 11. It conforms to POSIX, ANSI C and X/Open's XPG3 standards."[3] By concentrating on MOTIF applications, it is possible to define specific interaction characteristics that can be expected within the Mercator/MOTIF environment. These include:

- Predefined keyboard shortcuts.
- Standard interface objects.
- A limited number of menu types, 3 to 4 basic menus.
- Predefined method of displaying visual queues.
- Eighteen different types of cursors.

The predefined keyboard shortcuts within MOTIF will facilitate easier use of the environment to the nonvisual user because the user will not need to rely on a pointing device. X-Windows applications generally rely heavily on navigation through movement of a mouse. This type of mouse movement works well for the visual user but is especially tricky for the visually impaired user because there is no fixed point of reference. This means that there is no relationship between the physical position of a mouse on the desktop and the pointing cursor that is on the screen. There are two approaches to solving this problem within Mercator. The first is to rely on the keyboard shortcuts as a way of activating menu items and/or scroll bars. The second method is to employee a pointing device with a fixed point of reference. For this approach, the research team is investigating the use of a touchpad. The touchpad provides a physical relationship between a position on a tactile surface and the display

---

2. Elisabeth Mynatt and W. Keith Edwards, The Mercator Project, unpublished, 1991.

3. Alan Freedman, The Computer Glossary (The Electronic Version), 1991.

objects on the screen. It should be possible to lay a raised grid on top of a commercially available touchpad in order to give the visually impaired user basic positional information.

In X-Windows, display is accomplished through a set of widgets that handle items such as buttons, scroll bars, text fields, and other icons. MOTIF offers a standard set of widgets to be used in applications. These widgets are used as display objects that must be interpreted into the Mercator environment. Mercator will be designed to provide audible access to the standard display objects available through MOTIF. Mechanisms will be available to provide audible differentiation between these widgets. There will be descriptive methods to let the Mercator user know the type of display widget that is being utilized by the application. This will be done through a combination of voice queues and representational sounds. This ability to indicate the type of display widget will extend into areas such as display cursors and visual queues. The Mercator user will need the same ability as the sighted user to manipulate these display objects and to know the effect that the manipulation is producing. The Mercator environment will provide this type of feedback through speech output, audible tones, and responses to input from the touchpad.

Another feature that is being developed to help in orientation to manipulation of these display objects is a 360 degree sound environment. To use this feature, the user will ware a set of stereo headphones. Sound objects will be audibly arrayed in a 360 degree spectrum around the user's head. These objects will move within the sound space as the user manipulates them via the keyboard or touchpad. Additionally, it will be possible to differentiate between sound objects through pitch and tone of voices or audible queues.

Development Approach

A two tiered approach will be taken in development of the Mercator environment. The first tier is the applications level. This development will deal with making standard X-Windows and MOTIF applications available in the Mercator environment. The development will concentrate on only text applications. The second tier of the development is to create an audible workspace that is analogous to the visual desktop seen in many graphical interfaces. This element of the Mercator environment will not be required if the user is only going to run standard X-Windows applications.

Basic X-Windows access

X-Windows is a graphical environment that offers a lot of possibilities in creating a nonvisual interface. This is because of the client/server nature of the X-Windows protocol. The client, a piece of software running on either the workstation or a networked computer, must communicate with a server about the information to be displayed and the input provided by the user. This is called interprocess communications.[4] This requires messages to be sent between the client and the server. This message passing is one of the primary means that will be employed to obtain information for the Mercator audible display. As data is past back and forth between the client and server, the

4. Robert W. Scheifler and J. Gettys, The x window system, ACM Transactions on Graphics, (2), April 1986.

Mercator software will track displays as they are being built and manipulated.[5] This type of additional processing will not interfere with the message passing between the client and server that was intended by the software developer. It will only act to intercede on behalf of the user of the nonvisual interface. This "tap" into the flow of information between client and server will allow a partial off-screen model of the display to be built. This off-screen model will be used by the Mercator environment to produce the audible element of the nonvisual display. This model will also be used to track input by the user such that the Mercator environment will know how to provide audible queues as to the changes on the display and actions that might be taken. This client/server relationship is very beneficial because it restricts developers from directly manipulating display elements on the screen. All displays must occur through the pipeline between the client and X-Windows server. This circumstance makes it much more likely that the Mercator can provide full access to a wide variety of software.

The latest release of X-Windows, X11R5, provides new functionality that will be very useful in building this access tool. This release allows clients to be queried about the state of display objects that relate to a given application. Through this query process, Mercator will be better able to determine how display images change, how pointers and cursors have been altered, and the actions that are available from pull down menus or other hidden elements of the display.

Mercator extensions

The desktop metaphor is used to provide sighted users of graphical interfaces with a familiar working environment. These environments provide methods for displaying objects, moving objects around, transferring information between objects and launching applications. These types of activities are also performed by visually impaired individuals but the activities are usually accompanied by tactile information. For example, picking up a physical document (reinforced by the touch of the document) and then locating the physical trash can for document deposit. Since there is no immediate economical way of reproducing this tactile information, the Mercator environment will provide feedback to these types of activities via voice output. It will be necessary to vocalize information about what the objects are and the types of actions that can be associated with the objects.

The Mercator environment will provide an extension to the desktop metaphor based on research done at Xerox PARC.[6] The concept derived by this research is termed Rooms. In the Rooms metaphor, the user can collect like objects or applications to perform similar tasks. This helps to avoid the clutter that can be found on the proverbial desktop. These Rooms or workspaces can be connected via doors to form a network of workspaces that can be arranged to suit the users needs. This is an especially appropriate metaphor for visually impaired users. Most non-sighted computer users are very capable of visualizing themselves in the midst of this workspace. It is appropriate to think of oneself in the middle of an environment with the tools needed to do a particular job arrayed around a room. Mercator will provide appropriate audible feedback to represent moving around this workspace and from one workspace to another as in audible doorways. It should be possible for the user to learn to navigate through this network in much the same way that mobility skills are taught

5. Elizabeth Mynatt and W. Keith Edwards, The Mercator Project, unpublished, 1991.

6. D. Austin Henderson and Stuart K. Card, Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface, ACM Transactions on Graphics, pages 211-243, July 1986.

to non-sighted individuals. The 360 degree sound capability, mentioned earlier, will be a key ingredient to producing these Room effects. This technology will allow for positional representations that can be acted on by the listener. Navigation effects that produce pathway information can be simulated for passing from one room to another within the 360 degree sound space.

Prototype Completion

The Mercator prototype is scheduled to be completed by March of 1992. It is hoped that additional funding can be found to produce the full Mercator environment. This of course depends on results obtained from the initial work. Georgia Tech has been assembling a candidate list of visually impaired computer users to view the system after prototype completion. Their evaluation of the work will influence future development of the system. Additionally, many individuals who participate in providing adaptive solutions are being asked to make input to the design and development of the system. It is the goal of this project to ensure that visually impaired individuals continue to have access to work place information and that the users of Mercator be able to continue to work along side their sighted counterparts.